What is Appium?

APPIUM is a freely distributed open-source mobile application UI Testing framework. Appium allows native, hybrid and web application testing and supports automation test on physical devices as well as an emulator or simulator both. It offers cross-platform application testing, i.e., single API works for both Android and iOS platform test scripts.

Appium is an 'HTTP Server' written using a Node.js platform and drives iOS and an Android session using Web driver JSON wire protocol. Hence, before initializing the Appium Server, Node.js must be pre-installed on the system

What does Android Debug Bridge (ADB) mean?

The Android Debug Bridge (ADB) is a client-server program used in Android application development. The Android Debug-Bridge is part of the Android SDK and is made up of three components: a client, a daemon, and a server. It is used to manage either an emulator instance or an actual Android device.

adb devices (lists connected devices)
adb root (restarts adbd with root permissions)
adb start-server (starts the adb server)
adb kill-server (kills the adb server)
adb reboot (reboots the device)
adb devices -l (list of devices by product/model)
adb shell (starts the backround terminal)
exit (exits the background terminal)
adb help (list all commands)
adb -s <deviceName> <command> (redirect command to specific device)
adb –d <command> (directs command to only attached USB device)
adb –e <command> (directs command to only attached emulator)

Package Installation

adb shell install <apk> (install app)
adb shell install <path> (install app from phone path)
adb shell install -r <path> (install app from phone path)
adb shell uninstall <name> (remove the app)

File Operations

adb push <local> <remote> (copy file/dir to device)
adb pull <remote> <local> (copy file/dir from device)

```
import io.appium.java_client.AppiumDriver;
import org.openqa.selenium.remote.DesiredCapabilities;
{
        DesiredCapabilities capabilities = new DesiredCapabilities();
        capabilities.setCapability("deviceName","Android Emulator");
        capabilities.setCapability("platformVersion", "4.4");
}
```

Appium Desired Capabilities

Desired Capabilities are keys and values encoded in a JSON object, sent by Appium clients to the server when a new automation session is requested
{ "platformName": "iOS", "platformVersion": "11.0", "deviceName": "iPhone 7", "automationName": "XCUITest", "app": "/path/to/my.app" }

Capabilities	Description	Values/Uses
appPackage	Call desired Java package in android that user want to run	Value= com.example.myapp/ Obj.setCapability("appPackage", "com.whatsapp");
appActivity	Application Activity that user wants to launch from the package.	Value= MainActivity, .Settings Obj.setCapability("appActivity", "com.whatsapp.Main");
appWaitPackage	Package from which application needs to wait for	Value=com.example.android.myapp
appWaitActivity	Any Android activity that user need wait time	Value= SplashActivity capabilities.setCapability("appWaitActivity", "com.example.game.SplashActivity")
LaunchTimeout	Total time (in ms) to wait for instrumentation	2000
UDID	To identify unique device number for connected physical device	166aestu4
androidInstallPath	The name of the directory on the device in which the apk will be push before install. Defaults to /data/local/tmp	e.g. /sdcard/Downloads/

Capabilities	Description	Values/Uses
adbPort	Port used to connect to the ADB server (default 5037)	5037
systemPort	systemPort used to connect to appium-uiautomator2-server or appium-espresso-driver. The default is 8200 in general and selects one port from 8200 to 8299 for *appium-uiautomator2-server*, it is 8300 from 8300 to 8399 for *appium-espresso-driver*. When you run tests in parallel, you must adjust the port to avoid conflicts. Read Parallel Testing Setup Guide for more details.	e.g., 8201
remoteAdbHost	Optional remote ADB server host	e.g.: 192.168.0.101
androidDeviceSocket	Devtools socket name. Needed only when tested app is a Chromium embedding browser. The socket is open by the browser and Chromedriver connects to it as a devtools client.	e.g., chrome_devtools_remote

iOS Only

These Capabilities are available only on the XCUITest Driver and the deprecated UIAutomation Driver.

Capability	Description	Values
calendarFormat	(Sim-only) Calendar format to set for the iOS Simulator	e.g. gregorian
bundleId	Bundle ID of the app under test. Useful for starting an app on a real device or for using other caps which require the bundle ID during test startup. To run a test on a real device using the bundle ID, you may omit the 'app' capability, but you must provide 'udid'.	e.g. io.appium.TestApp

Capability	Description	Values
Udid	Unique device identifier of the connected physical device	e.g. 1ae203187fc012g
launchTimeout	Amount of time in ms to wait for instruments before assuming it hung and failing the session	e.g. 20000
locationServicesEnabled	(Sim-only) Force location services to be either on or off. Default is to keep current sim setting.	true or false
locationServicesAuthorized	(Sim-only) Set location services to be authorized or not authorized for app via plist, so that location services alert doesn't pop up. Default is to keep current sim setting. Note that if you use this setting you MUST also use the bundleId capability to send in your app's bundle ID.	true or false
autoAcceptAlerts	Accept all iOS alerts automatically if they pop up. This includes privacy access permission alerts (e.g., location, contacts, photos). Default is false.	true or false
autoDismissAlerts	Dismiss all iOS alerts automatically if they pop up. This includes privacy access permission alerts (e.g., location, contacts, photos). Default is false.	true or false
nativeInstrumentsLib	Use native intruments lib (ie disable instruments-without-delay).	true or false
nativeWebTap	Enable "real", non-javascript-based web taps in Safari. Default: false. Warning: depending on viewport size/ratio; this might not accurately tap an element	true or false

Capability	Description	Values
safariInitialUrl	Initial safari url, default is a local welcome page	e.g. https://www.github.com
safariAllowPopups	(Sim-only) Allow javascript to open new windows in Safari. Default keeps current sim setting	true or false
safariIgnoreFraudWarning	(Sim-only) Prevent Safari from showing a fraudulent website warning. Default keeps current sim setting.	true or false
safariOpenLinksInBackground	(Sim-only) Whether Safari should allow links to open in new windows. Default keeps current sim setting.	true or false
keepKeyChains	(Sim-only) Whether to keep keychains (Library/Keychains) when appium session is started/finished	true or false
localizableStringsDir	Where to look for localizable strings. Default en.lproj	en.lproj
processArguments	Arguments to pass to the AUT using instruments	e.g., -myflag
interKeyDelay	The delay, in ms, between keystrokes sent to an element when typing.	e.g., 100
showIOSLog	Whether to show any logs captured from a device in the appium logs. Default false	true or false
sendKeyStrategy	strategy to use to type test into a test field. Simulator default: oneByOne. Real device default: grouped	oneByOne, grouped or setValue

Capability	Description	Values
screenshotWaitTimeout	Max timeout in sec to wait for a screenshot to be generated. default: 10	e.g., 5
waitForAppScript	The ios automation script used to determined if the app has been launched, by default the system wait for the page source not to be empty. The result must be a Boolean	e.g. true;, target.elements().length > 0;, $.delay(5000); true;
webviewConnectRetries	Number of times to send connection message to remote debugger, to get webview. Default: 8	e.g., 12
appName	The display name of the application under test. Used to automate backgrounding the app in iOS 9+.	e.g., UICatalog
customSSLCert	(Sim only) Add an SSL certificate to IOS Simulator.	e.g. -----BEGIN CERTIFICATE----- MIIFWjCCBEKg... -----END CERTIFICATE-----
webkitResponseTimeout	(Real device only) Set the time, in ms, to wait for a response from WebKit in a Safari session. Defaults to 5000	e.g., 10000
remoteDebugProxy	(Sim only, <= 11.2) If set, Appium sends and receives remote debugging messages through a proxy on either the local port (Sim only, <= 11.2) or a proxy on this unix socket (Sim only >= 11.3) instead of communicating with the iOS remote debugger directly.	e.g. 12000 or "/tmp/my.proxy.socket"

Capability	Description	Values
enableAsyncExecuteFromHttps	capability to allow simulators to execute asynchronous JavaScript on pages using HTTPS. Defaults to false	true or false
skipLogCapture	Skips to start capturing logs such as crash, system, safari console and safari network. It might improve performance such as network. Log related commands will not work. Defaults to false.	true or false
webkitDebugProxyPort	(Real device only) Port to which ios-webkit-debug-proxy is connected, during real device tests. Default is 27753.	12021
fullContextList	Returns the detailed information on contexts for the get available context command. If this capability is enabled, then each item in the returned contexts list would additionally include WebView title, full URL and the bundle identifier. Defaults to false.	true or false

```
adb devices' & press Enter
<$ adb devices>
emulator-5554 device
emulator-5556 device
emulator-5558 device
```

COMMAND DETECTING A SINGLE DEVICE FROM MULTIPLE CONNECTED DEVICES-

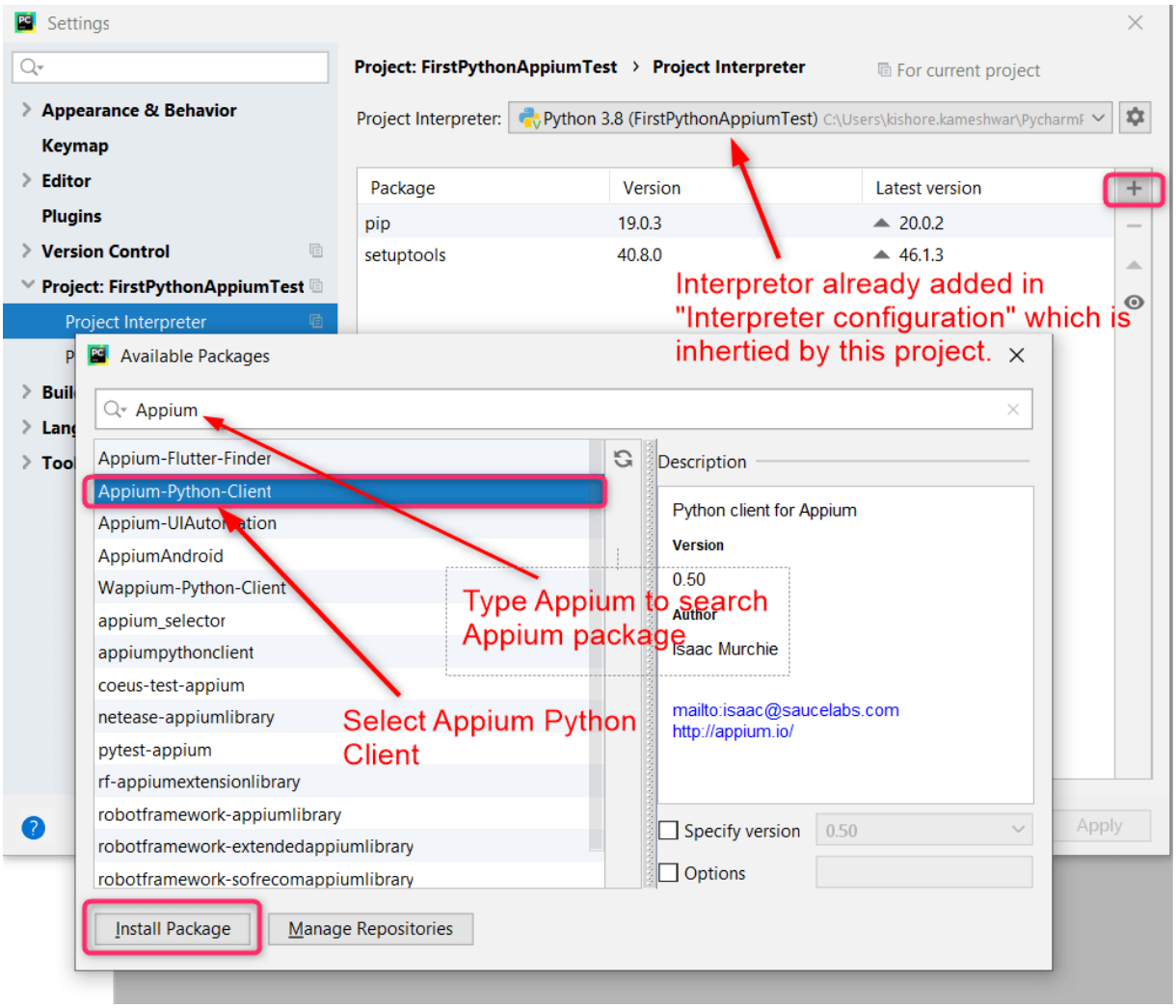
```
<$ adb –s emulator-5554 install Guru99.apk>
```

```
<!-- https://mvnrepository.com/artifact/io.appium/java-client -->
<dependency>
   <groupId>io.appium</groupId>
   <artifactId>java-client</artifactId>
   <version>7.0.0</version>
```

</dependency>

PYTHON

pip install Appium-Python-Client (1.0.2)



UIAUTOMATORVIEWER

"Uiautomatorviewer" is a part of the Android SDK manager and will be accessible once you install the SDK manager. Download and install Android SDK manager

```
c:\users\<username>\AppData\Local\Android\sdk\tools
```

You'll notice a batch file with name

```
uiautomatorviewer.bat
```

EXPLAIN WHAT IS APPIUM INSPECTOR?

Similar to Selenium IDE record and Playback tool, Appium has an "Inspector" to record and playback. It records and plays native application behavior by inspecting DOM and generates the test scripts in any desired language. However, Appium Inspector does not support Windows and use UIAutomator viewer in its option.

Driver-Specific Setup

- The XCUITest Driver (for iOS and tvOS apps)
- The Espresso Driver (for Android apps)
- The UiAutomator2 Driver (for Android apps)
- The Windows Driver (for Windows Desktop apps)
- The Mac Driver (for Mac Desktop apps)

Verifying the Installation

To verify that all of Appium's dependencies are met you can use appium-doctor. Install it with npm install -g appium-doctor, then run the appium-doctor command, supplying the --ios or --android flags to verify that all of the dependencies are set up correctly.

```
json { "automationName": "XCUITest", "platformName": "iOS", "platformVersion": "12.2", "deviceName": "iPhone 8", ... }
```

Status

Retrieve the server's current status

```
Java - driver.getStatus();
pyhon - selenium.webdriver.common.utils.is_url_connectable(port)
```

Execute Mobile Command

```
java- driver.executeScript("mobile: scroll", ImmutableMap.of("direction", "down"));
python - self.driver.execute_script("mobile: scroll", {'direction': 'down'})
```

Create New Session

Java	`DesiredCapabilities desiredCapabilities = new DesiredCapabilities();`
	
	`desiredCapabilities.setCapability(MobileCapabilityType.PLATFORM_VERSION, "10.3");`
	
	`desiredCapabilities.setCapability(MobileCapabilityType.DEVICE_NAME, "iPhone Simulator");`
	
	`desiredCapabilities.setCapability(MobileCapabilityType.AUTOMATION_NAME, "XCUITest");`
	
	`desiredCapabilities.setCapability(MobileCapabilityType.APP, "/path/to/ios/app.zip");`

	URL url = new URL("http://127.0.0.1:4723/wd/hub"); IOSDriver driver = new IOSDriver(url, desiredCapabilities);

python	desired_caps = {
	'platformName': 'Android',
	'platformVersion': '7.0',
	'deviceName': 'Android Emulator',
	'automationName': 'UiAutomator2',
	'app': PATH('/path/to/app')
	}
	self.driver = webdriver.Remote('http://127.0.0.1:4723/wd/hub', desired_caps)

Get/Retrieve Session Capabilities

Java	String sessionId = driver.getSessionId().toString();
python	Map<String, Object> caps = driver.getSessionDetails(); desired_caps = self.driver.session

Navigate backwards in the browser history, if possible (Web context only)

Java	driver.back();
python	self.driver.back()

Screen shot

Java	File file = ((TakesScreenshot)driver).getScreenshotAs(OutputType.FILE); FileUtils.copyFile(file, new File("C:/temp/Screenshot.jpg"));
python	directory = '%s/' % os.getcwd() file_name = 'screenshot.png' driver.save_screenshot(directory + file_name)

Set Timeouts

Java	driver.manage().timeouts().pageLoadTimeout(30, TimeUnit.SECONDS);
python	self.driver.set_page_load_timeout(5000)

Implicit Wait Timeout

Java	driver.manage().timeouts().implicitlyWait(30, TimeUnit.SECONDS);
python	self.driver.implicitly_wait(5) # waits 5 seconds

Set Script Timeout

Java	driver.manage().timeouts().setScriptTimeout(30, TimeUnit.SECONDS);
python	self.driver.set_script_timeout(5000)

EXPLICIT

Java	WebDriverWait wait = new WebDriverWait((AppiumDriver)driver , 10); wait.until(ExpectedConditions.visibilityOfElementLocated(By.id("text1")));
python	from selenium.webdriver.support.ui import WebDriverWait wait = WebDriverWait(driver, 10) webdriver.wait.until(driver.find_element_by_id('com.cloudapper.android:id/item_bg'))

Get Orientation

Java	ScreenOrientation orientation = driver.getOrientation();
python	orientation = self.driver.orientation

Set Orientation

Java	driver.rotate(ScreenOrientation.LANDSCAPE);
python	driver.orientation = "LANDSCAPE"

GET Location

Java	Location location = driver.location();
	
	//location.getAltitude();
	
	//location.getLatitude();

	//location.getLongitude();
python	location = self.driver.location()

SET GEOLOCATION

Java	driver.setLocation(new Location(49, 123, 10));
python	self.driver.set_location(latitude, longitude,altitude)

GET AVAILABLE LOG TYPES

Java	Get the log for a given log type. Log buffer is reset after each request Set<String> logTypes = driver.manage().logs().getAvailableLogTypes();
python	log_types = driver.log_types

GET LOGS

Java	LogEntries logEntries = driver.manage().logs().get("driver");
python	logs = driver.get_log('driver');

LOG EVENT (STORE A CUSTOM EVENT)

Java	CustomEvent evt = new CustomEvent();
	
	evt.setEventName("funEvent");
	
	evt.setVendor("appium");
	
	driver.logEvent(evt);

python	driver.log_event('appium', 'funEvent')

GET EVENTS STORED IN APPIUM SERVER

Java	driver.getEvents();
	
	Update Device Settings
	
	driver.setSetting(Setting.WAIT_FOR_IDLE_TIMEOUT, 5000);

python	self.driver.update_settings({"sample": "value"}))

Retrieve Device Settings

Java	`Map<String, Object> settings = driver.getSettings();`
python	`self.driver.get_settings`

Start Activity

Java	Start an Android activity by providing package name and activity name `driver.startActivity(new Activity("com.example", "ActivityName"));`
python	`self.driver.start_activity("com.example", "ActivityName");`

Toggle Location Services

Java	`driver.toggleLocationServices();`
python	`self.driver.toggle_location_services();`

Send SMS

Simulate an SMS message (Emulator only)

Java	`driver.sendSMS("555-123-4567", "Hey lol");`
python	`self.driver.send_sms('555-123-4567', 'Hey lol')`

GSM Call

Java	Make GSM call (Emulator only) driver.makeGsmCall("5551234567", GsmCallActions.CALL);
python	self.driver.make_gsm_call('5551234567', GsmCallActions.CALL)

GSM Signal

Java	Set GSM signal strength (Emulator only) driver.setGsmSignalStrength(GsmSignalStrength.GOOD);
python	self.driver.set_gsm_signal(GsmSignalStrength.GOOD)

Network speed

Java	Set network speed (Emulator only)

	driver.setNetworkSpeed(NetworkSpeed.LTE);
python	self.driver.set_network_speed(NetSpeed.LTE)

Start Recording Screen

Java	driver.startRecordingScreen();
	
	driver.startRecordingScreen(new BaseStartScreenRecordingOptions(....));

python	self.driver.start_recording_screen()

Stop Recording Screen

Java	driver.stopRecordingScreen();
python	self.driver.stop_recording_screen()

Perform Touch ID

Simulate a touch id event (iOS Simulator only)

Java	driver.performTouchID(false); // Simulates a failed touch
	
	driver.performTouchID(true); // Simulates a passing touch

python	self.driver.touch_id(false); # Simulates a failed touch self.driver.touch_id(true); # Simulates a passing touch

Toggle Touch ID Enrollment

Toggle the simulator being enrolled to accept touchId (iOS Simulator only)

Java	driver.toggleTouchIDEnrollment(true);
python	self.driver.toggle_touch_id_enrollment()

Open Notifications

Open Android notifications (Emulator only)

Java	driver.openNotifications();
python	self.driver.open_notifications();

Get System Bars

Retrieve visibility and bounds information of the status and navigation bars

Java	Map<String, String> systemBars = driver.getSystemBars();
python	self.driver.get_system_bars()

Get System Time

Get the time on the device

Java	String time = driver.getDeviceTime();
python	time = self.driver.device_time time = self.driver.get_device_time() time = self.driver.get_device_time("YYYY-MM-DD")

Get Display Density

Retrieve display density(dpi) of the Android device

Java	driver.getDeviceDensity();
python	self.driver.get_display_density()

Finger print

Authenticate users by using their finger print scans on supported emulators.

Java	driver.fingerPrint(1);
python	self.driver.finger_print(1)

Get Active Element

Gets the active element of the current session

Java	WebElement currentElement = driver.switchTo().activeElement();
python	

Get Current Context

Java	String context = driver.getContext();
python	context = driver.current_context

Get All Contexts

Get all the contexts available to automate

Java	Set<String> contextNames = driver.getContextHandles();
python	contexts = driver.contexts

SET CURRENT CONTEXT

Java	Set<String> contextNames = driver.getContextHandles();
	
	driver.context(contextNames.toArray()[1]);
	
	// ...
	
	driver.context("NATIVE_APP");

python	webview = driver.contexts[1] driver.switch_to.context(webview) # ... driver.switch_to.context('NATIVE_APP')

MOVE MOUSE TO

Java	Actions action = new Actions(driver);
	
	action.moveTo(element, 10, 10);
	
	action.perform();

python	actions = ActionChains(driver) actions.move_to(element, 10, 10) actions.perform()

DOUBLE CLICK

Java	Actions action = new Actions(driver);
	
	action.moveTo(element);
	
	action.doubleClick();
	
	action.perform();

python	actions = ActionChains(driver) actions.move_to_element(element) actions.double_click() actions.perform()

Button Down

Click and hold the left mouse button at the current mouse coordinates

Java	Actions action = new Actions(driver);
	
	action.moveTo(element);
	
	action.clickAndHold();
	
	action.perform();

python	actions = ActionChains(driver) actions.move_to_element(element) actions.click_and_hold() actions.perform()

Button Up

Releases the mouse button previously held

Java	Actions action = new Actions(driver);
	
	action.moveTo(element);
	
	action.clickAndHold();
	
	action.moveTo(element, 10, 10);
	
	action.release();
	
	action.perform();

python	actions = ActionChains(driver) actions.move_to_element(element) actions.click_and_hold() actions.move_to_element(element, 10, 10)

	action.release(); actions.perform()

TAP

Single tap on the touch enabled device

Java	TouchActions action = new TouchActions(driver);
	
	action.singleTap(element);
	
	action.perform();

python	from appium.webdriver.common.touch_action import TouchAction # ... actions = TouchAction(driver) actions.tap(element) actions.perform()

DOUBLE TAP

Java	TouchActions action = new TouchActions(driver);
	
	action.doubleTap(element);
	
	action.perform();

python	from appium.webdriver.common.touch_action import TouchAction # ... actions = TouchAction(driver) actions.double_tap(element) actions.perform()

TOUCH DOWN

Java	TouchActions action = new TouchActions(driver); action.down(10, 10);

	action.move(50, 50);
	
	action.perform();

python	

SCROLL

Scroll on the touch screen using finger based motion events

Java	`TouchActions action = new TouchActions(driver);`
	
	`action.scroll(element, 10, 100);`
	
	`action.perform();`

python	`from appium.webdriver.common.touch_action import TouchAction`
	
	`# ...`
	
	`actions = TouchAction(driver)`
	
	`actions.scroll_from_element(element, 10, 100)`
	
	`actions.scroll(10, 100)`
	
	`actions.perform()`

FLICK

Flick on the touch screen using finger motion events

Java	`TouchActions action = new TouchActions(driver);`
	
	`action.flick(element, 1, 10, 10);`
	
	`action.perform();`

python	`from appium.webdriver.common.touch_action import TouchAction`
	
	`# ...`
	
	`actions = TouchAction(driver)`
	
	`actions.flick_element(element, 1, 10, 10)`
	
	`actions.perform()`

MULTI TOUCH PERFORM

Java	`TouchActions actionOne = new TouchAction();`
	
	`actionOne.press(10, 10);`

	`actionOne.moveTo(10, 100);`
	`actionOne.release();`
	`TouchActions actionTwo = new TouchAction();`
	`actionTwo.press(20, 20);`
	`actionTwo.moveTo(20, 200);`
	`actionTwo.release();`
	`MultiTouchAction action = new MultiTouchAction();`
	`action.add(actionOne);`
	`action.add(actionTwo);`
	`action.perform();`

python	`from appium.webdriver.common.touch_action import TouchAction`
	`from appium.webdriver.common.multi_action import MultiAction`
	`# ...`
	`a1 = TouchAction()`
	`a1.press(10, 20)`
	`a1.move_to(10, 200)`
	`a1.release()`
	`a2 = TouchAction()`
	`a2.press(10, 10)`
	`a2.move_to(10, 100)`
	`a2.release()`
	`ma = MultiAction(self.driver)`
	`ma.add(a1, a2)`
	`ma.perform()`

SCROLLING

Java	`JavascriptExecutor js = (JavascriptExecutor) driver;`
	`HashMap<String, String> scrollObject = new HashMap<String, String>();`
	`scrollObject.put("direction", "down");`

	`js.executeScript("mobile: scroll", scrollObject);`
python	`driver.execute_script("mobile: scroll", {"direction": "down"})`

Is Device Locked

boolean isLocked = driver.isDeviceLocked();
self.driver.is_locked()

Unlock

driver.lockDevice();
driver.unlockDevice();

self.driver.lock();
self.driver.unlock();

Shake

Perform a shake action on the device
driver.shake();
self.driver.shake();

Is Keyboard Shown

boolean isKeyboardShown = driver.isKeyboardShown();
driver.is_keyboard_shown()

Hide Keyboard

driver.hideKeyboard();
self.driver.hide_keyboard()

Rotate

driver.rotate(new DeviceRotation(10, 10, 10));
name type description
hix number x offset to use for the center of the rotate gesture
y number y offset to use for the center of the rotate gesture
radius number The distance in points from the center to the edge of the circular path
rotation number The length of rotation in radians
touchCount number The number of touches to use in the specified gesture. (Effectively, the number of fingers a user would use to make the specified gesture.) Valid values are 1 to 5.
duration number The length of hold time for the specified gesture, in seconds.

Press Key Code

driver.pressKeyCode(AndroidKeyCode.SPACE, AndroidKeyMetastate.META_SHIFT_ON);
self.driver.press_keycode(10);

Long Press Key Code

driver.longPressKeyCode(AndroidKeyCode.HOME);

```
self.driver.long_press_keycode(10);
```

Push File

Place a file onto the device in a particular place

```
driver.pushFile("/data/local/tmp/foo.bar", new File("/Users/johndoe/files/foo.bar"));

dest_path = '/data/local/tmp/test_push_file.txt'
data = bytes('This is the contents of the file to push to the device.', 'utf-8')
self.driver.push_file(dest_path, base64.b64encode(data).decode('utf-8'))
```

Pull File

```
byte[] fileBase64 = driver.pullFile("/path/to/device/foo.bar");
file_base64 = self.driver.pull_file('/path/to/device/foo.bar');
```

Pull Folder

Retrieve a folder from the device's file system

```
byte[] folder = driver.pullFolder("/path/to/device/foo.bar");
folder_base64 = self.driver.pull_folder('/path/to/device/foo.bar');
```

Emulate power state

Emulate power state change on the connected emulator.

```
driver.setPowerAC(PowerACState.OFF);
self.driver.set_power_ac(Power.AC_OFF)
```

Emulate power capacity

Emulate power capacity change on the connected emulator

```
driver.setPowerCapacity(100);
self.driver.set_power_capacity(50)
```

Get Clipboard

```
driver.getClipboard(ClipboardContentType.PLAINTEXT); // get plaintext
driver.getClipboardText();

self.driver.get_clipboard()
self.driver.get_clipboard_text()
```

Install App

Install the given app onto the device

```
driver.installApp("/Users/johndoe/path/to/app.apk");
self.driver.install_app('/Users/johndoe/path/to/app.apk');
```

Is App Installed

Check whether the specified app is installed on the device

```
driver.isAppInstalled("com.example.AppName");
self.driver.is_app_installed('com.example.AppName');
```

Launch App

Launch the app-under-test on the device

```
driver.launchApp();
self.driver.launch_app()
```

Background App

```
driver.runAppInBackground(Duration.ofSeconds(10));
self.driver.background_app(10)
```

Close an App

```
driver.closeApp();
self.driver.close_app()
```

Reset App

Reset the currently running app for this session

```
driver.resetApp();
self.driver.reset()
```

Remove App

```
driver.removeApp("com.example.AppName");
self.driver.remove_app('com.example.AppName');
```

Activate App

Activate the given app onto the device

```
driver.activate_app('com.apple.Preferences')
driver.activate_app('io.appium.android.apis')

driver.activateApp('com.apple.Preferences');
driver.activateApp('io.appium.android.apis');
```

Terminate App

Terminate the given app on the device

```
driver.terminate_app('com.apple.Preferences')
driver.terminate_app('io.appium.android.apis')

driver.terminateApp('com.apple.Preferences');
driver.terminateApp('io.appium.android.apis');
```

Get app state

```
driver.queryAppState('com.apple.Preferences');
driver.queryAppState('io.appium.android.apis');

driver.query_app_state('com.apple.Preferences')
driver.query_app_state('io.appium.android.apis')
```

0 is not installed. 1 is not running. 2 is running in background or suspended. 3 is running in background. 4 is running in foreground. (number)

GET CURRENT ACTIVITY

Get the name of the current Android activity
String activity = driver.currentActivity();
activity = self.driver.current_activity;

GET CURRENT PACKAGE

Get the name of the current Android package
String package = driver.getCurrentPackage();
package = self.driver.current_package;

SAMPLE CODE FOR APK

```
private AndroidDriver<WebElement> driver;
File classpathRoot = new File(System.getProperty("user.dir"));
File appDir = new File(classpathRoot, "../apps");
File app = new File(appDir.getCanonicalPath(), "ApiDemos-debug.apk");

DesiredCapabilities capabilities = new DesiredCapabilities();
capabilities.setCapability("deviceName", "Android Emulator");
capabilities.setCapability("app", app.getAbsolutePath());
capabilities.setCapability("appPackage", "io.appium.android.apis");
capabilities.setCapability("appActivity", ".ApiDemos");
capabilities.setCapability("platformVersion", " platformVersion");
capabilities.setCapability("model", deviceName);
capabilities.setCapability("allocation", "wait");
capabilities.setCapability("platformName", "wait");
capabilities.setCapability("fullReset", "wait");
capabilities.setCapability("autoInstrument", "wait");
capabilities.setCapability("sensorInstrument", "wait");

driver = new AndroidDriver<WebElement>(getServiceUrl(), capabilities);
```

ANDROID WEB BROWSER

```
private AndDriver<WebElement> driver;

@BeforeClass
public void setUp() {

DesiredCapabilities capabilities = new DesiredCapabilities();
capabilities.setCapability("platformVersion", " platformVersion");
capabilities.setCapability("model", deviceName);
```

```
capabilities.setCapability("allocation", "wait");
capabilities.setCapability("scriptName", testcaseName);
capabilities.setCapability("fullReset", "wait");
capabilities.setCapability("autoInstrument", "wait");
capabilities.setCapability("sensorInstrument", "wait");

driver = new AndroidDriver<WebElement>(getServiceUrl(), capabilities);
}
```

IOS IPA TEST

```
private AppiumDriver driver;

@BeforeClass
public void setUp() {

DesiredCapabilities capabilities = new DesiredCapabilities();
capabilities.setCapability("deviceName", "iphone");
capabilities.setCapability("platformName", "iOS");
capabilities.setCapability("platformVersion", " platformVersion");
capabilities.setCapability("model", deviceName);
capabilities.setCapability("scriptName", testcaseName);
capabilities.setCapability("newCommandTimeout", "120");
capabilities.setCapability("bundleId", " bundleId");
capabilities.setCapability("app", "app path");

driver = new IOSDriver<WebElement>(getServiceUrl(), capabilities);
}
```

LAUNCH ANDROID NATIVE APP

```
RemoteWebDriver driver;
@Test
public void addTest() throws Exception {
//Connect to an emulator and open calculator app. Note: appium server and
simulator should be running
DesiredCapabilities capabilities = new DesiredCapabilities();
capabilities.setCapability("BROWSER_NAME", "Android");
capabilities.setCapability("VERSION", "8.0");
capabilities.setCapability("deviceName","Emulator");
capabilities.setCapability("platformName","Android");
capabilities.setCapability("appPackage", "com.android.settings");
capabilities.setCapability("appActivity",
"com.android.settings.Settings");
try {
driver = new AndroidDriver<>(new URL("http://127.0.0.1:4723/wd/hub"),
capabilities);
driver.manage().timeouts().implicitlyWait(15, TimeUnit.SECONDS);
}catch(SessionNotCreatedException e){
```

```
throw new RuntimeException("Driver not created with capabilities: " +
capabilities.toString());
}
}
```

PERFECTO

```
RemoteWebDriver driver;

@Test
public void appiumTest() throws Exception {

String cloudName = "<<cloud name>>";
String securityToken = "<<security token>>";

String browserName = "mobileOS";
DesiredCapabilities capabilities = new DesiredCapabilities(browserName,
"", Platform.ANY);
capabilities.setCapability("securityToken", securityToken);
capabilities.setCapability("model", "Galaxy S.*");
capabilities.setCapability("enableAppiumBehavior", true);
capabilities.setCapability("openDeviceTimeout", 2);
capabilities.setCapability("appPackage", "com.android.settings");
capabilities.setCapability("appActivity",
"com.android.settings.Settings");

try{
driver = (RemoteWebDriver)(new AppiumDriver<>(new URL("https://" +
Utils.fetchCloudName(cloudName)  +
".perfectomobile.com/nexperience/perfectomobile/wd/hub"), capabilities));

driver.manage().timeouts().implicitlyWait(15, TimeUnit.SECONDS);
}catch(SessionNotCreatedException e){
throw new RuntimeException("Driver not created with capabilities: " +
capabilities.toString());
                      }
}
```

```
desired_caps = {}
    desired_caps['platformName'] = 'iOS'
    desired_caps['platformVersion'] = '10.1'
    desired_caps['deviceName'] = 'DEVICE_NAME'
    desired_caps['browserName'] = ''
    desired_caps['automationName'] = 'XCUITest' # Possible without.
    desired_caps['app'] = 'https://applitools.bintray.com/Examples/eyes-
ios-hello-world.zip'
```

```
    wd = webdriver.Remote('http://127.0.0.1:4723/wd/hub', desired_caps)
    wd.implicitly_wait(60)
```

RUNNING YOUR TEST

```
DesiredCapabilities desiredCapabilities = new DesiredCapabilities();
desiredCapabilities.setCapability(MobileCapabilityType.BROWSER_NAME, "Safari");
desiredCapabilities.setCapability(MobileCapabilityType.AUTOMATION_NAME, "XCUITest");
URL url = new URL("http://127.0.0.1:4723/wd/hub");
AppiumDriver driver = new AppiumDriver(url, desiredCapabilities);

capabilities = { 'browserName': 'Safari', 'automationName': 'XCUITest' }
driver = webdriver.Remote('http://localhost:4723/wd/hub', capabilities)

DesiredCapabilities capabilities = new DesiredCapabilities();
capabilities.setCapability(MobileCapabilityType.PLATFORM_NAME, "Android");
capabilities.setCapability(MobileCapabilityType.PLATFORM_VERSION, "9.0");
capabilities.setCapability(MobileCapabilityType.DEVICE_NAME, "Android Emulator");
capabilities.setCapability(MobileCapabilityType.AUTOMATION_NAME, "UIAutomator2");
capabilities.setCapability(MobileCapabilityType.BROWSER_NAME, "Chrome");
```

FIREFOX IN APPIUM ANDROID

```
def generate_caps():
    common_caps = {
        # It does not really matter what to put there, although setting
'Firefox' might cause a failure
        # depending on the particular client library
        'browserName': 'MozillaFirefox',
        # automationName capability presence is mandatory for this Gecko
Driver to be selected
        'automationName': 'Gecko',
        # Should have the name of the host platform, where the geckodriver
binary is deployed
        'platformName': 'mac',
    }
    android_caps = {
        **common_caps,
        'moz:firefoxOptions': {
            'androidDeviceSerial': '<device/emulator serial>',
            # These capabilities depend on what you are going to automate
            # Refer Mozilla documentation at
https://developer.mozilla.org/en-
US/docs/Web/WebDriver/Capabilities/firefoxOptions for more details
            'androidPackage': 'org.mozilla.firefox',
        },
```

```
    }
    desktop_browser_caps = {
        **common_caps,
    }
    return [android_caps, desktop_browser_caps]

@pytest.fixture(params=generate_caps())
def driver(request):
    drv = webdriver.Remote('http://localhost:4723/wd/hub', request.param)
    yield drv
    drv.quit()

class TimeoutError(Exception):
    pass
```

Start and Stop Appium Server programmatically

```
AppiumDriverLocalService appiumService;
String appiumServiceUrl;

@BeforeSuite
public void setUp() throws MalformedURLException {

 appiumService = AppiumDriverLocalService.buildDefaultService();
appiumService.start();
appiumServiceUrl = appiumService.getUrl().toString();
System.out.println("Appium Service Address : - "+ appiumServiceUrl);
}

@Aftersuite
public void End() {
System.out.println("Stop appium service");
appiumService.stop();
}
```

LOCATORS

```
List<WebElement> linearLayoutElements = (List<WebElement>)
driver.findElementsByClassName("android.widget.FrameLayout");

driver.findElementsByXPath
```

driver.**findElementsById**

driver.**findElementsByAccessibilityId**

ID
Class Name(iOS, Class Name is represented as the full name of the XCUI element)
Xpath
Accessibility ID(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”.)
Android UI Automator
Android View Tag (Espresso Only)
iOS UI Automation
driver.findElementByImage()

```
File file = new File("/Users/saikrisv/Desktop/login.png");
    File refImgFile = Paths.get(file.toURI()).toFile();
    String s = Base64.getEncoder().encodeToString(Files.readAllBytes(refImgFile.toPath()));
    driver.findElementByImage(s).click();

String selector = "new UiSelector().text(“Cancel”))

.className(“android.widget.Button”))";

MobileElement element =
(MobileElement) driver.findElement(MobileBy.AndroidUIAutomator(selector));
driver.findElement(MobileBy.iosNs.PredicateString(selector))
driver.findElement(MobileBy.iosclasschain(selector))
```

Q7) What is the default port number of Appium Server?

Answer: By default Appium runs on port no 4723, just type appium in cmd to find

How to start Appium at a different port instead of default port?

Answer: Launch Command Prompt (Windows) or Terminal (Mac) and type either appium –port <port number> or appium –p <port number>

What OS’s version does Appium support?
> Android OS should be >= 4.2
> iOS version should be >= 10.0

Q7) How many ways we can install Appium?

Answer: We can install Appium in two ways:

> Appium Installer (.exe for Windows OS and .dmg for Mac OS)
> We can also install Appium CLI using npm

20) Explain how you can install SD card in emulator?

To install SD card in emulator, you have to use the command

MKsdcrd –I mySDCard 1024M mySdCardFile.img

```
driver.context("NATIVE_APP")
```

Syntax : - To move to Chrome browser context

```
driver.context("CHROMIUM");
```

```
File upload in Appium

public class AppiumTest {

    AndroidDriver driver;
    WebDriverWait wait;
    String AppURL = "http://cgi-lib.berkeley.edu/ex/fup.html";

    @BeforeTest
    public void setup() throws MalformedURLException {

        // Create an object for Desired Capabilities
        DesiredCapabilities capabilities = new DesiredCapabilities();
        capabilities.setCapability(MobileCapabilityType.PLATFORM_NAME,
"Android");
        capabilities.setCapability(MobileCapabilityType.PLATFORM_VERSION,
"10.0");
        capabilities.setCapability(MobileCapabilityType.DEVICE_NAME,
"Pixel");
        capabilities.setCapability(MobileCapabilityType.AUTOMATION_NAME,
"UIAutomator2");
        capabilities.setCapability(MobileCapabilityType.BROWSER_NAME,
"Chrome");

        // Initialize the driver object with the URL to Appium Server and
        // passing the capabilities
        driver = new AndroidDriver(new
URL("http://127.0.0.1:4723/wd/hub"), capabilities);
        wait = new WebDriverWait(driver, 5);
        driver.setFileDetector(new LocalFileDetector());
    }

    @Test
    public void testSearchAppium() throws IOException {
```

```
        //Navigate to app url
        driver.get(AppURL);

        //Click on upload button
        By uploadBtn = By.name("upfile");

wait.until(ExpectedConditions.visibilityOfElementLocated(uploadBtn));
        driver.findElement(uploadBtn).click();

        //Push file to device
        driver.pushFile("/sdcard/download/test.pdf", new
File("C:\\Users\\HarryDev\\Downloads\\cmp_html_page_size.pdf"));

       //Switch to Native_App
        Set<String> contextNames = driver.getContextHandles();
        for (String strContextName : contextNames) {
            if (strContextName.contains("NATIVE_APP")) {
                driver.context("NATIVE_APP");
                break;
            }
        }

        //Click on 'Allow' - permission
        By elementView =
By.id("com.android.permissioncontroller:id/permission_allow_button");

wait.until(ExpectedConditions.visibilityOfElementLocated(elementView));
        driver.findElement(elementView).click();

        //Click on files
        By eleFile = By.xpath("//*[@text="Files"]");

wait.until(ExpectedConditions.visibilityOfElementLocated(eleFile));
        driver.findElement(eleFile).click();

        //select pdf file from downloads (location of pdf file)
        By eleDoc = By.id("com.android.documentsui:id/thumbnail");
        wait.until(ExpectedConditions.visibilityOfElementLocated(eleDoc));
        driver.findElement(eleDoc).click();

        //Switch to Chrome browser
        Set<String> contextNames1 = driver.getContextHandles();
        for (String strContextName : contextNames1) {
```

```
            if (strContextName.contains("CHROMIUM")) {
                driver.context("CHROMIUM");
                break;
            }
        }

        //Click on submit button
        WebElement btnElement =
driver.findElement(By.cssSelector("input[type=submit]"));
        wait.until(ExpectedConditions.visibilityOf(btnElement));
        btnElement.click();

        //Add a simple assertion
        By nextPageHeader = By.cssSelector("h1");

wait.until(ExpectedConditions.visibilityOfElementLocated(nextPageHeader));

Assert.assertTrue(driver.findElement(nextPageHeader).getText().equals("Fil
e Upload Results"));
    }

    @AfterTest
    public void tearDown() {
        if(driver !=null)
            driver.quit();
    }
}
```

Run your appium Tests on Real Devices over wifi

Step 1: Make sure both your host computer and Android device are on the same Wifi network.

Step 2: Connect the Android device to the computer using your USB cable. As soon as you do that, your host computer will detect

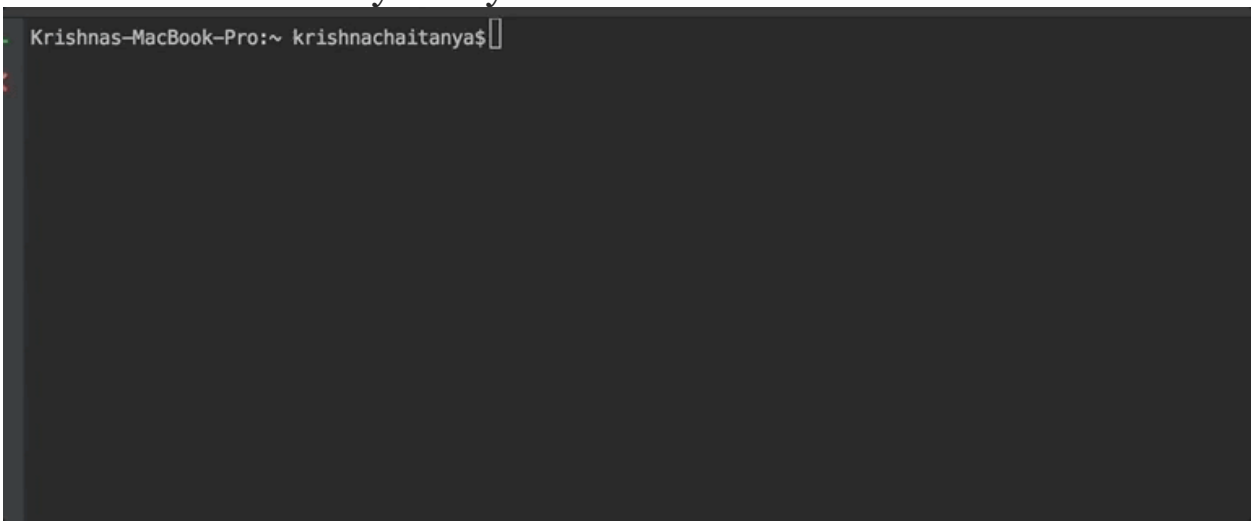
your device and adb will start running in the USB mode on the computer. You can check the attached devices with `adb devices.`

Step 3: Restart `adb` in tcpip mode with this command: `adb tcpip 5555`

Step 4: Find out the IP address of the Android device. There are several ways to do that:

- Go to Settings -> About phone/tablet -> Status -> IP address.
- Go to the list of Wi-fi networks available. The one to which you're connected, tap on that and get to know your IP.
- Try `$ adb shell netcfg`.

Step 5: Remove the USB cable and you should be connected to your device. Try using `adb devices` and you will see that the device is still connected to your system.

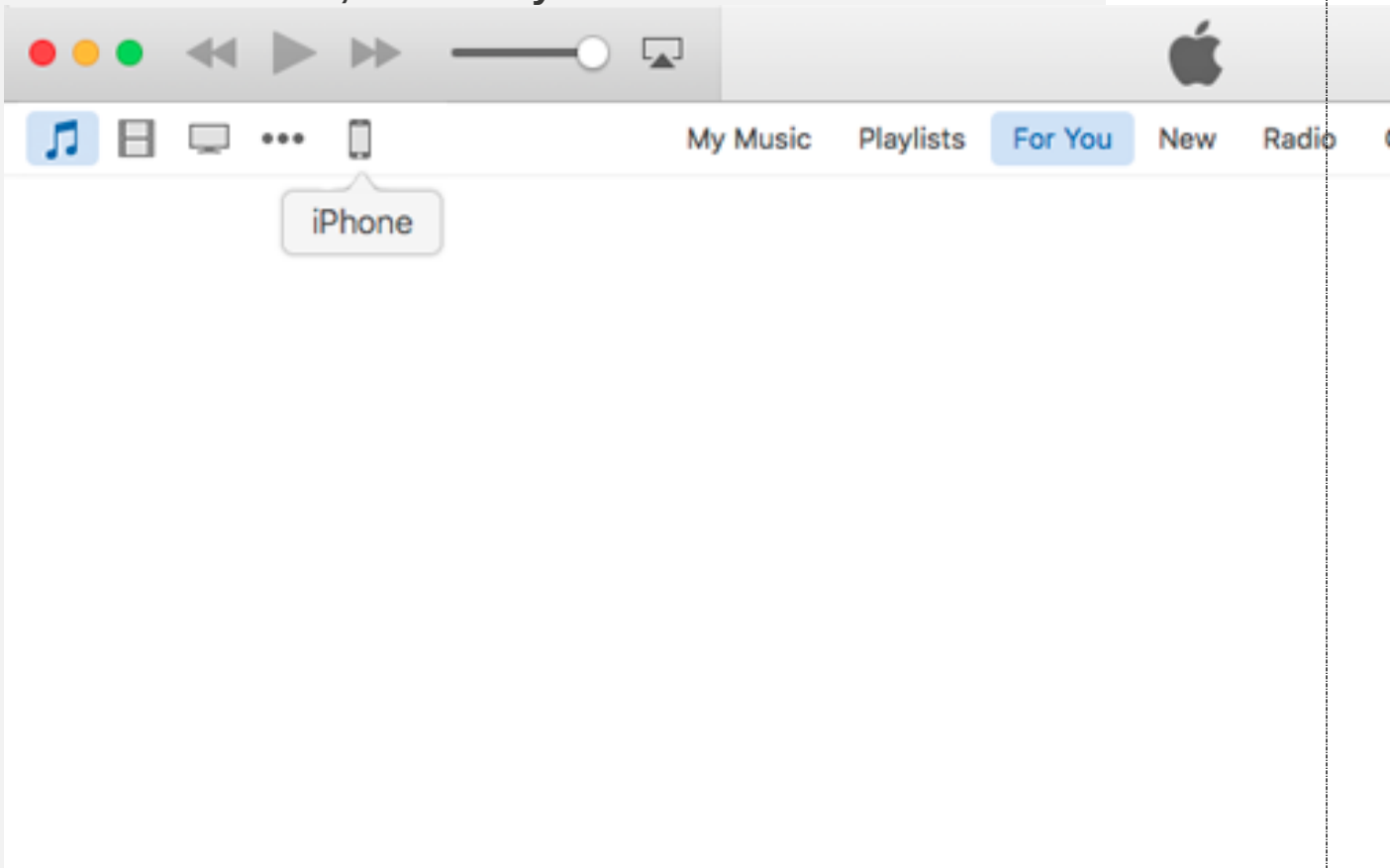


Step 6: Now open you test class and change/add the “deviceId” to the desired capabilities.

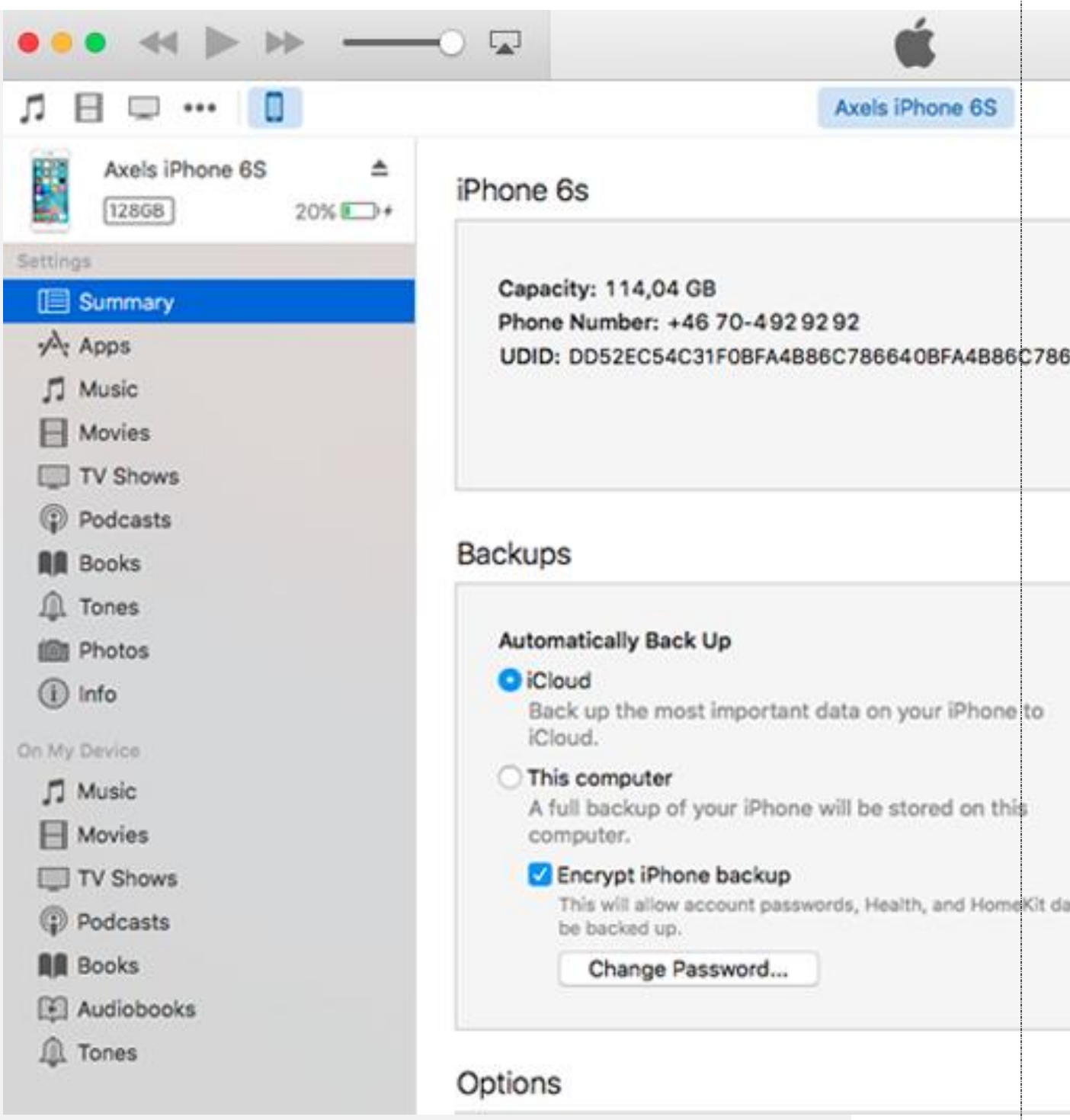
```
capabilities.setCapability("deviceId", "IP Address:5555");//In this
case:
capabilities.setCapability("deviceId", "192.168.31.219:5555");
```

Find Unique Device Identifier (UDID) on the iPhone

1.Launch iTunes & connect your iPhone, iPad or iPod (device).
2. Under Devices, click on your device



3. Next click on the ‘Serial Number’



4. This should change the Serial Number into the UDID
5. Choose ‘Edit’ and then ‘Copy’ from the iTunes menu
6. Paste into your Email, and you should see the UDID in your email message.

Finding the App Bundle ID

1. Use a tool like iExplorer, which allows you to browse your device storage directly.
2. Connect your iPhone/iPad to your Mac via USB and open iExplorer or a similar utility.
3. Open the Apps folder on your device and locate the app you're interested in.
4. Locate the `iTunesMetadata.plist` file and follow the steps above that you used to unpack the `.ipa` file.

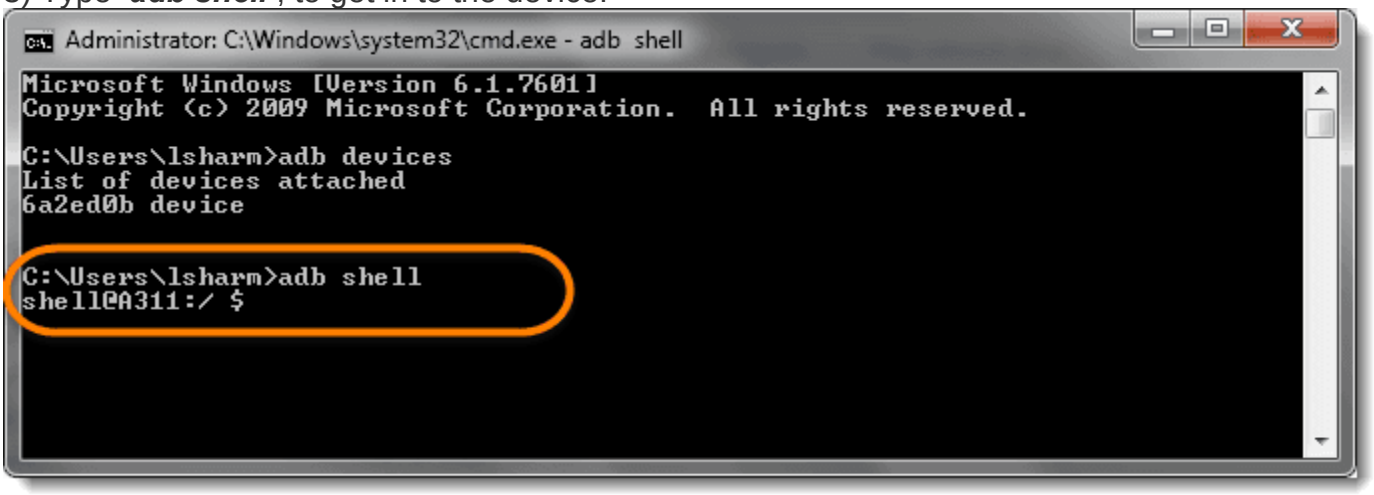
find appPackage and appActivity name of your App

APK Info is an app which you can download from Play Store, and **it will provide the appPackage and appActivity name of any app which is installed on your mobile device.**

Using 'mCurrentFocus' or 'mFocusedApp' in Command Prompt

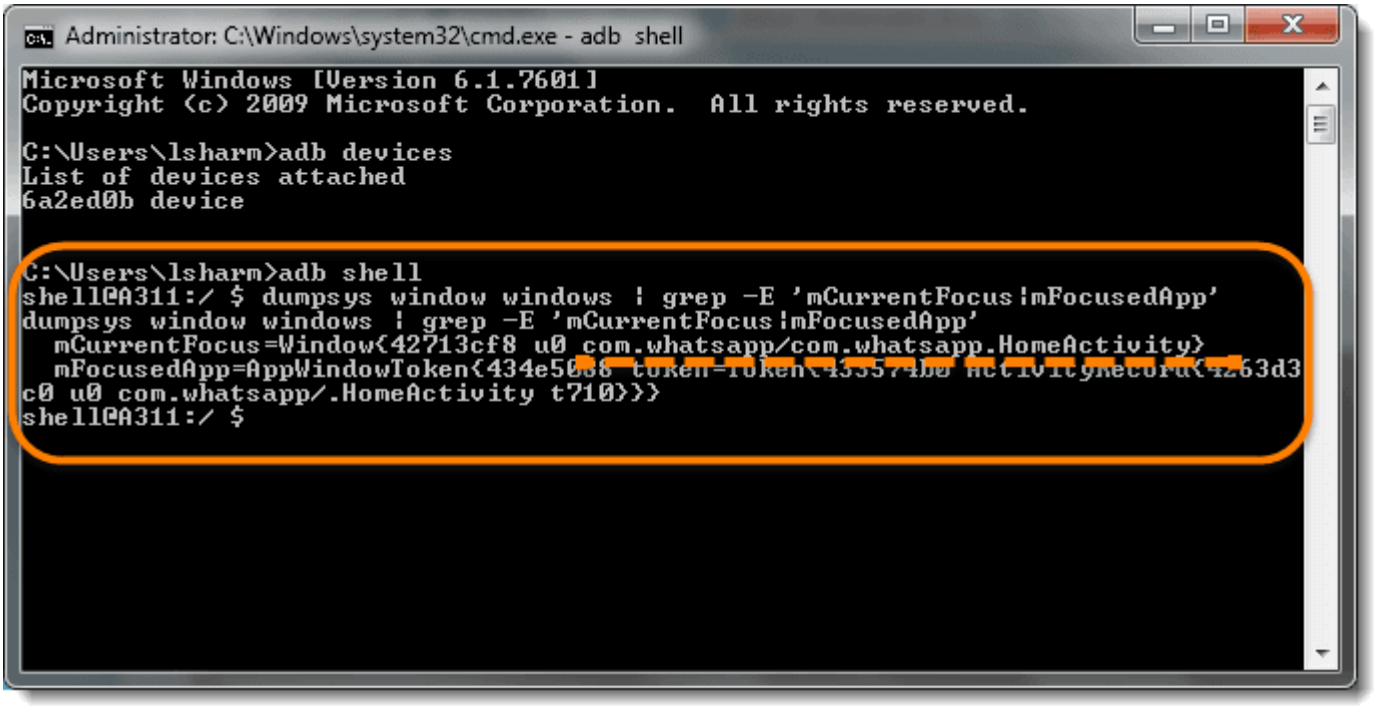
Steps to follow:
1) Open command prompt.

1) Go to ***Run*** *and type* ‘***cmd’*** for opening the ***Command Prompt*** interface.
2) Type ‘***adb devices***‘ in the window.
3) Type ***‘adb shell’***, to get in to the device.



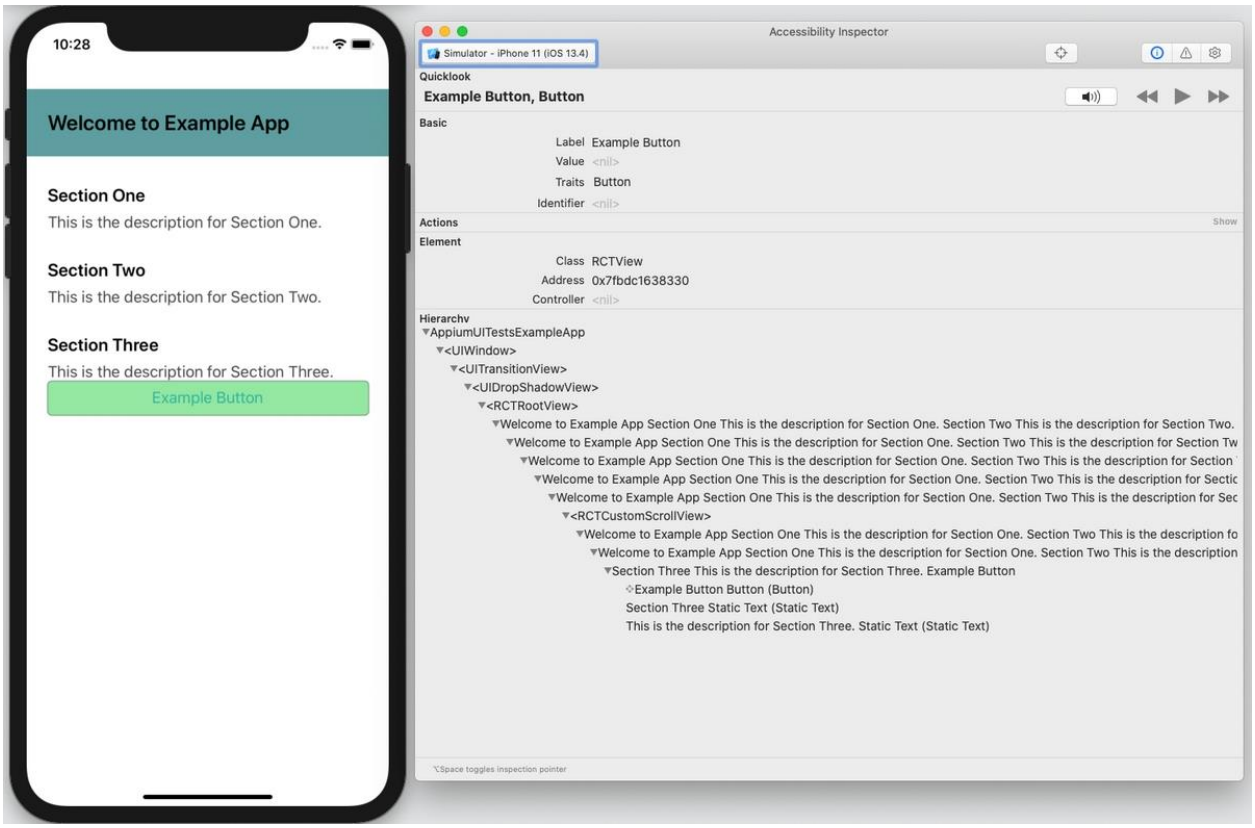
4) Now type the below mentioned command to get the information of the ***WhatsApp apk***:

dumpsys window windows | grep -E ‘mCurrentFocus|mFocusedApp’



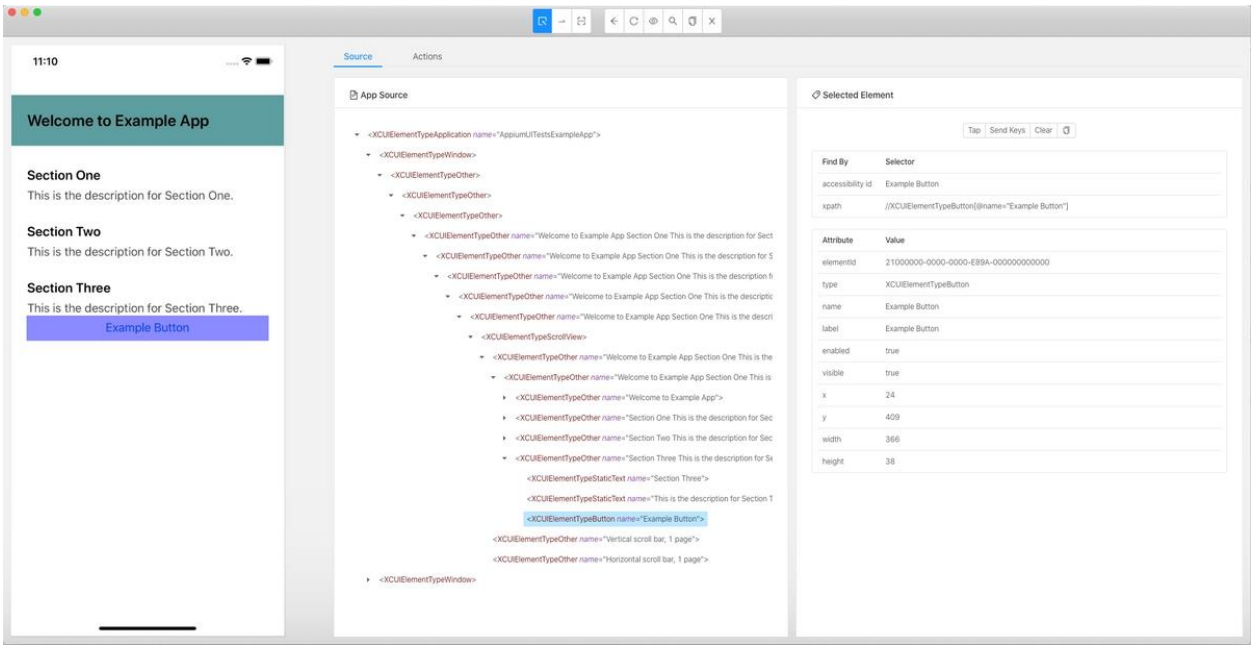
Find Element Selector for iOS App with Appium

`Accessibility Inspector` is a less well known but very useful tool developed by Apple to help developers improve the accessibility of their iOS apps



Appium Desktop

`Appium Desktop` is a desktop app developed by Appium team for Mac, Windows, and Linux. It's a popular tool used by many Appium developers to inspect app elements for iOS app as well as Android app in their automated mobile UI tests.


11:10
Welcome to Example App
Section One
This is the description for Section One.
Section Two
This is the description for Section Two.
Section Three
This is the description for Section Three.
Example Button
Source
Actions
App Source
<XCUIElementTypeApplication name="AppiumUITestsExampleApp">
<XCUIElementTypeWindow>
<XCUIElementTypeOther>
<XCUIElementTypeOther>
<XCUIElementTypeOther>
<XCUIElementTypeOther name="Welcome to Example App Section One This is the description for Sect
<XCUIElementTypeOther name="Welcome to Example App Section One This is the description for S
<XCUIElementTypeOther name="Welcome to Example App Section One This is the description f
<XCUIElementTypeOther name="Welcome to Example App Section One This is the descriptio
<XCUIElementTypeOther name="Welcome to Example App Section One This is the descri
<XCUIElementTypeScrollView>
<XCUIElementTypeOther name="Welcome to Example App Section One This is the
<XCUIElementTypeOther name="Welcome to Example App Section One This is
<XCUIElementTypeOther name="Welcome to Example App">
<XCUIElementTypeOther name="Section One This is the description for Sec
<XCUIElementTypeOther name="Section Two This is the description for Sec
<XCUIElementTypeOther name="Section Three This is the description for Se
<XCUIElementTypeStaticText name="Section Three">
<XCUIElementTypeStaticText name="This is the description for Section T
<XCUIElementTypeButton name="Example Button">
<XCUIElementTypeOther name="Vertical scroll bar, 1 page">
<XCUIElementTypeOther name="Horizontal scroll bar, 1 page">
<XCUIElementTypeWindow>
Selected Element
Tap
Send Keys
Clear
Find By
Selector
accessibility id
Example Button
xpath
//XCUIElementTypeButton[@name="Example Button"]
Attribute
Value
elementId
21000000-0000-0000-E89A-000000000000
type
XCUIElementTypeButton
name
Example Button
label
Example Button
enabled
true
visible
true
x
24
y
409
width
366
height
38