AUTOMATE THE PLANET

Most Complete Appium Java Cheat Sheet

Initialize

```
// maven: io.appium:java-client
appiumLocalService = new AppiumServiceBuilder().usingAnyFreePort().build(); // Creates local Appium server instance
appiumLocalService.start(); // Starts local Appium server instance
var desiredCapabilities = new DesiredCapabilities();
// Android capabilities
desiredCapabilities.setCapability(MobileCapabilityType.AUTOMATION_NAME, "UiAutomator2");
desiredCapabilities.setCapability(MobileCapabilityType.DEVICE_NAME, "Android Virtual Device");
desiredCapabilities.setCapability(MobileCapabilityType.PLATFORM_NAME, "Android");
desiredCapabilities.setCapability(MobileCapabilityType.PLATFORM_VERSION, "7.1");
// iOS capabilities
desiredCapabilities.setCapability(MobileCapabilityType.AUTOMATION_NAME, "XCUITest");
desiredCapabilities.setCapability(MobileCapabilityType.DEVICE_NAME, "iPhone 11");
desiredCapabilities.setCapability(MobileCapabilityType.PLATFORM_NAME, "iOS");
desiredCapabilities.setCapability(MobileCapabilityType.PLATFORM_VERSION, "13.2");
// Install and start an app on Android
desiredCapabilities.setCapability(AndroidMobileCapabilityType.APP_PACKAGE, "com.example.android.apis");
desiredCapabilities.setCapability(AndroidMobileCapabilityType.APP_ACTIVITY, ".ApiDemos");
desiredCapabilities.setCapability(MobileCapabilityType.APP, "path/to/TestApp.apk");
// Install and start an app on iOS
desiredCapabilities.setCapability(MobileCapabilityType.APP, "path/to/TestApp.app.zip");
// Start mobile browser on Android
desiredCapabilities.setCapability(MobileCapabilityType.BROWSER_NAME, "Chrome");
// Start mobile browser on iOS
desiredCapabilities.setCapability(MobileCapabilityType.BROWSER_NAME, "safari");
// Set WebView Context for Hybrid Apps
driver.context(driver.getContextHandles().stream().filter(c -> c.contains("WEBVIEW")).findFirst().orElse(null));
// Use local server instance
driver = new AndroidDriver<AndroidElement>(appiumLocalService, desiredCapabilities); // initialize Android driver on local server instance
driver = new IOSDriver<IOSElement>(appiumLocalService, desiredCapabilities); // initialize iOS driver on local server instance
// Use remote Appium driver
driver = new AndroidDriver<AndroidElement>(new URL("http://127.0.0.1:4723/wd/hub"), desiredCapabilities); // initialize Android remote driver
driver = new IOSDriver<IOSElement>(new URL("http://0.0.0.0:4723/wd/hub"), desiredCapabilities); // initialize iOS remote driver
```

Locators

```
driver.findElementById("android:id/text1");
driver.findElementByClassName("android.widget.CheckBox");
driver.findElementByXPath("//*[@text='Views']");
driver.findElementByAccessibilityId("Views");
driver.findElementByImage(base64EncodedImageFile);
driver.findElementByAndroidUIAutomator("new
UiSelector().text(\"Views\");");
driver.findElementByIosNsPredicate("type == 'XCUIElementBottn' AND
name == 'ComputeSumButton'");
// Find multiple elements
driver.findElementsByClassName("android.widget.CheckBox");
```

Actions

```
// Alert handling
driver.switchTo().alert().accept();
driver.switchTo().alert().dismiss();
// Basic actions;
element.click();
element.sendKeys("textToType");
element.clear();
// Change orientation
driver.rotate(ScreenOrientation.LANDSCAPE); // PORTRAIT
// Clipboard
driver.setClipboardText("1234", "plaintext");
String clipboard = driver.getClipboardText();
// Mobile gestures
TouchAction touchAction = new TouchAction(driver);
touchAction.tap(TapOptions.tapOptions()
.withPosition(PointOption.point(x, y)).withTapsCount(count));
touchAction.press(PointOption.point(x, y));
touchAction.waitAction(WaitOptions
.waitOptions(Duration.ofMillis(200)));
touchAction.moveTo(PointOption.point(x, y));
touchAction.release();
touchAction.longPress(PointOption.point(x, y));
touchAction.perform();
// Simulate phone call (Emulator only)
driver.makeGsmCall("5551237890", GsmCallActions.ACCEPT);
driver.makeGsmCall("5551237890", GsmCallActions.CALL);
driver.makeGsmCall("5551237890", GsmCallActions.CALL);
driver.makeGsmCall("5551237890", GsmCallActions.HOLD);
// Set GSM voice state (Emulator only)
driver.setGsmVoice(GsmVoiceState.DENIED);
driver.setGsmVoice(GsmVoiceState.HOME);
driver.setGsmVoice(GsmVoiceState.OFF);
driver.setGsmVoice(GsmVoiceState.ON);
driver.setGsmVoice(GsmVoiceState.ROAMING);
driver.setGsmVoice(GsmVoiceState.SEARCHING);
driver.setGsmVoice(GsmVoiceState.UNREGISTERED);
// Set GSM signal strength (Emulator only)
driver.setGsmSignalStrength(GsmSignalStrength.GOOD);
driver.setGsmSignalStrength(GsmSignalStrength.GREAT);
driver.setGsmSignalStrength(GsmSignalStrength.MODERATE);
driver.setGsmSignalStrength(GsmSignalStrength.NONE_OR_UNKNOWN);
driver.setGsmSignalStrength(GsmSignalStrength.POOR);
// Set network speed (Emulator only)
driver.setNetworkSpeed(NetworkSpeed.LTE);
// Simulate receiving SMS message (Emulator only)
driver.sendSMS("555-555-5555", "Your code is 123456");
// Toggle services
driver.toggleAirplaneMode();
driver.toggleData();
driver.toggleLocationServices();
driver.toggleWifi();
// Soft keyboard actions
driver.isKeyboardShown(); // returns boolean
driver.hideKeyboard();
// Lock device
driver.isDeviceLocked(); // returns boolean
driver.lockDevice();
driver.unlockDevice();
// Notifications
driver.openNotifications();
// Geolocation actions
driver.location(); // returns Location{Latitude, Longitude,
Altitude}
driver.setLocation(new Location(94.23, 121.21, 11.56));
// Get system time
driver.getDeviceTime(); // returns String
// Get display density
driver.getDisplayDensity(); // returns long
// File actions
driver.pushFile("/data/local/tmp/file", new File("path/to/file"));
driver.pullFile("/path/to/device/file"); // returns byte[]
driver.pullFolder("/path/to/device"); // returns byte[]
```

Application Management - Android

```
// Install app
driver.installApp("path/to/app.apk");
// Remove app
driver.removeApp("com.example.AppName");
// Verify app is installed
driver.isAppInstalled("com.example.android.apis"); // returns bool
// Launch app
driver.launchApp();
// Start activity
driver.startActivity(new Activity("com.example.android.apis",
".ApiDemos"));
// Get current activity
driver.currentActivity(); // returns String
// Get current package
driver.getCurrentPackage(); // returns String
// Reset app
driver.resetApp();
// Close app
driver.closeApp();
// Run current app in background
driver.runAppInBackground(Duration.ofSeconds(10)); // 10 seconds
// Terminate app
driver.terminateApp("com.example.android.apis"); // returns bool
// Check app's current state
driver.queryAppState("com.example.android.apis"); // returns
ApplicationState
// Get app strings
driver.getAppStringMap("en", "path/to/file"); // returns
Map<String, String>
```

Advanced Actions - Common

```
// Multitouch action
TouchAction actionOne = new TouchAction(driver);
actionOne.press(PointOption.point(10, 10));
actionOne.moveTo(PointOption.point(10, 100));
actionOne.release();
TouchAction actionTwo = new TouchAction(driver);
actionTwo.press(PointOption.point(20, 20));
actionTwo.moveTo(PointOption.point(20, 200));
actionTwo.release();
MultiTouchAction action = new MultiTouchAction(driver);
action.add(actionOne);
action.add(actionTwo);
action.perform();
// Swipe using touch action
TouchAction touchAction = new TouchAction(driver);
var element = driver.findElementById("android:id/text1");
Point point = element.getCoordinates().onPage();
Dimension size = element.getSize();
 touchAction
 .press(PointOption.point(point.getX() + 5, point.getY() + 5))
 .waitAction(WaitOptions.waitOptions(Duration.ofMillis(200)))
 .moveTo(PointOption.point(point.getX() + size.getWidth() - 5,
point.getY() + size.getHeight() - 5))
 .release().perform();
// Scroll using JavascriptExecutor
var js = (JavascriptExecutor)driver;
Map<String, String> swipe = new HashMap<>();
swipe.put("direction", "down"); // "up", "right", "left"
swipe.put("element", element.getId());
js.executeScript("mobile:swipe", swipe);
// Scroll element into view by Android UI automator
driver.findElementByAndroidUIAutomator("new UiScrollable(new
UiSelector()).scrollIntoView(new UiSelector().text(\"Views\"));" );
// Update Device Settings
driver.setSetting(Setting.KEYBOARD_AUTOCORRECTION, false);
driver.getSettings(); // returns Map<String, Object>
// Take screenshot
((TakesScreenshot)driver).getScreenshotAs(OutputType.FILE); // BYTES,
BASE64 – returns File, byte[] or String (base64 encoded PNG)
// Screen record
driver.startRecordingScreen();
driver.stopRecordingScreen();
```

Advanced Actions - Android

```
// Add photos
driver.pushFile("/mnt/sdcard/Pictures/img.jpg", new
File("path/to/img.jpg"));
// Simulate hardware key
driver.pressKey(new KeyEvent().withKey(AndroidKey.HOME));
driver.longPressKey(new KeyEvent().withKey(AndroidKey.POWER));
// Set battery percentage
driver.setPowerCapacity(100);
// Set the state of the battery charger to connected or not
driver.setPowerAC(PowerACState.ON);
driver.setPowerAC(PowerACState.OFF);
// Get performance data
driver.getPerformanceData("my.app.package", "cpuinfo", 5); // returns
List<List<Object>>
```

Advanced Actions - iOS

```
// Add photos
driver.pushFile("img.jpg", new File("path/to/img.jpg"));
// Simulate hardware key
Map<String, Object> args = new HashMap<>();
args.put("name", "home"); // volumeup; volumedown
driver.executeScript("mobile: pressButton", args);
// Sending voice commands to Siri
Map<String, Object> args = new HashMap<>();
args.put("text", "Hey Siri, what's happening?");
driver.executeScript("mobile: siriCommand", args);
// Perform Touch ID in iOS Simulator
driver.performTouchID(false); // Simulates a failed touch ID
driver.performTouchID(true); // Simulates a passing touch ID
// Shake device
driver.shake();
```

Application Management - iOS

```
// Install app
Map<String, Object> args = new HashMap<>();
args.put("app", "path/to/app.ipa");
driver.executeScript("mobile: installApp", args);
// Remove app
Map<String, Object> args = new HashMap<>();
args.put("bundleId", "com.myapp");
driver.executeScript("mobile: removeApp", args);
// Verify app is installed
Map<String, Object> args = new HashMap<>();
args.put("bundleId", "com.myapp");
(boolean)driver.executeScript("mobile: isAppInstalled", args);
// Launch app
Map<String, Object> args = new HashMap<>();
args.put("bundleId", "com.apple.calculator");
driver.executeScript("mobile: launchApp", args);
// Switch app to foreground
Map<String, Object> args = new HashMap<>();
args.put("bundleId", "com.myapp");
driver.executeScript("mobile: activateApp", args);
// Terminate app
Map<String, Object> args = new HashMap<>();
args.put("bundleId", "com.myapp");
// will return false if app is not running, otherwise true
(boolean)driver.executeScript("mobile: terminateApp", args);
// Check app's current state
Map<String, Object> args = new HashMap<>();
args.put("bundleId", "com.myapp");
// will return false if app is not running, otherwise true
(ApplicationState)driver.executeScript("mobile: queryAppState", args);
```