# APPIUM Details

Different settings for writing tests on both the android and ios platforms are in capabilities. Below are the differences in the settings.

## iOS - Grid Node Configuration Example json file

|  |
| --- |
| {  "automationName": "XCUITest",  "platformName": "iOS",  "deviceName": "iPhone",  "app": "/Users/mt/RTRK-oblo/appium/OBLO Living.ipa",  "newCommandTimeout": 60,  "udid": "d329bcd8dade1f308f9ddcbc8d1fbe2dcf78e032",  "bundleId": "com.rtrk.obloliving",  "xcodeOrgId": "83FXDN8628",  "xcodeSigningId": "iPhone Developer",  "showXcodeLog": true,  "clearSystemFiles": false,  "platformVersion": "10.3.1"  } |

Figure 1 – Example Capability for iOS

## Android - Grid Node Configuration Example json file

|  |
| --- |
| {  "platformName": "Android",  "deviceName": "LG Nexus 5X",  "app": "/Users/mt/RTRK-oblo/Android/presentation-oblo-release-2.1.1-21112.apk",  "newCommandTimeout": 120,  "noReset": false,  "appPackage": "com.rtrk.oblosmarthome",  "appWaitActivity": "com.rtrk.smarthome.presentation.mvp.login.LoginActivity\_",  "appWaitActivity2": "com.rtrk.smarthome.presentation.mvp.navigation.NavigationActivity\_"  } |

Figure 2 – Example Capability for Android

# APPIUM INSTRUCTIONS

**Step 1: Download Appium from link http://appium.io/**

Step 1a:

> brew install node      # get node.js

> npm install -g appium  # get appium

> npm install wd         # get appium client

> appium &               # start appium

**Step 2 : Setup Android and Java home**

1. Download Android SDK  
   https://developer.android.com/sdk/index.html
2. Extract Android SDK to /usr/local/adt/  
   unzip -qo adt-bundle-\*.zip -d /usr/local/adt
3. Set ANDROID\_HOME and JAVA\_HOME environment variables  
   export JAVA\_HOME=$(/usr/libexec/java\_home)  
   export ANDROID\_HOME=/usr/local/adt/sdk
4. Add ANDROID\_HOME and JAVA\_HOME to sh profile (in order not to set them every time)  
   echo "export JAVA\_HOME=$(/usr/libexec/java\_home)" >> ~/.bash\_profile  
   echo "export ANDROID\_HOME=/usr/local/adt/sdk" >> ~/.bash\_profile
5. Install Android build and platform tools  
   $ANDROID\_HOME/tools/android update sdk --no-ui --obsolete --force
6. Goto some directory and get Appium framework  
   git clone https://github.com/appium/appium.git
7. Configure Appium for Android  
   cd appium  
   ./reset.sh --android --selendroid --verbose
8. Start Appium server with  
   node .

**Step 3: Download and setup WebDriverAgent**

https://github.com/facebook/WebDriverAgent

\*Open WebDriverAgent.xcodeproj in Xcode. For both the WebDriverAgentLib and WebDriverAgentRunner targets, select "Automatically manage signing" in the "General" tab, and then select your Development Team. This should also auto select Signing Ceritificate.

> ./Scripts/bootstrap.sh

And copy pojectBody into /usr/local/lib/node\_modules/appium/node\_modules/appium-xcuitest-driver/WebDriverAgent and repeat step \*.

\*\*\*\*

cd usr/local/lib/node\_modules/appium/node\_modules/appium-xcuitest-driver/WebDriverAgent

brew install carthage

npm i -g webpack

./Scripts/bootstrap.sh -d

open /usr/local/lib/node\_modules/appium/node\_modules/appium-xcuitest-driver/WebDriverAgent/

run project

Start test

<https://www.mutuallyhuman.com/blog/2017/04/20/webdriveragent-getting-started-with-automated-ios-testing>

Create webDriver

**Step 4 : Finish setup**

Two pieces of software are currently necessary to run iOS tests on real devices:

libimobiledevice - install using brew install libimobiledevice --HEAD

ios-deploy - install using npm install -g ios-deploy

https://github.com/appium/appium-xcuitest-driver#external-dependencies

# TEST IMPLEMENTATION SAMPLES

|  |
| --- |
| */\*\*  \* Configurations for iOS.  \*/* public static final class iOS {   public static final String *OS\_NAME* = "ios";   public static DesiredCapabilities getCapabilities() {  DesiredCapabilities capabilities = new DesiredCapabilities();    capabilities.setCapability("xcodeOrgId", "83FXDN8628");  capabilities.setCapability("app", "/Users/mt/RTRK-oblo/appium/OBLO Living.ipa");  capabilities.setCapability("bundleId", "com.rtrk.obloliving");  capabilities.setCapability("automationName", "XCUITest");  capabilities.setCapability("platformName", "iOS");  capabilities.setCapability("deviceName", "iPhone");  //6s plus  capabilities.setCapability("udid", "d329bcd8dade1f308f9ddcbc8d1fbe2dcf78e032");   capabilities.setCapability("platformVersion", "10.3.1");  capabilities.setCapability("newCommandTimeout", 60);  capabilities.setCapability("xcodeSigningId", "iPhone Developer");  capabilities.setCapability("showXcodeLog", true);  capabilities.setCapability("clearSystemFiles", false);  //capabilities.setCapability("fullReset", false);  return capabilities;  } } |

Figure 3 – Configurations for iOS

|  |
| --- |
| */\*\*  \* Configurations for Android.  \*/* public static final class Android {   public static final String *OS\_NAME* = "android";   public static DesiredCapabilities getCapabilities() {  DesiredCapabilities capabilities = new DesiredCapabilities();   capabilities.setCapability("app", "/Users/mt/RTRK-oblo/Android/presentation-oblo-release-2.1.1-21112.apk");  capabilities.setCapability("deviceName","LG Nexus 5X");  capabilities.setCapability("platformVersion", "7.1.2");  capabilities.setCapability("platformName","Android");  capabilities.setCapability("appPackage", "com.rtrk.oblosmarthome");  capabilities.setCapability("appWaitActivity", "com.rtrk.smarthome.presentation.mvp.login.LoginActivity\_");  capabilities.setCapability("appWaitActivity2", "com.rtrk.smarthome.presentation.mvp.navigation.NavigationActivity\_");  capabilities.setCapability("newCommandTimeout", 10000);  capabilities.setCapability("command-timeout", 600);  capabilities.setCapability("idle-timeout", 800);  capabilities.setCapability("max-duration", 10800);  capabilities.setCapability("automationName", "Appium");  //capabilities.setCapability("noReset", true);  capabilities.setCapability("fullReset", true);   return capabilities;  } |

Figure 4 – Configurations for Android

|  |
| --- |
| public interface ConstantsIOS {  */\*\*  \* LOGIN SOURCE  \*/* //------------------------------------------------------------------------   public static String *LOGIN\_FIELD\_EMAIL* = "//XCUIElementTypeTextField";  public static String *LOGIN\_FIELD\_PASSWORD* = "//XCUIElementTypeSecureTextField";  public static String *LOGIN\_BTN\_NEXT* = "Next";  public static String *LOGIN\_ALERT\_NOTIFICATION\_ALLOW* = "Allow";  public static String *LOGIN\_ALERT\_NOTIFICATION\_DONTALLOW* = "Don’t Allow";  //------------------------------------------------------------------------  */\*\*  \* REGISTRATION SOURCE  \*/* //------------------------------------------------------------------------   final String *REGISTRATION\_BTN\_CREATE\_ACCOUNT* = "Create new account";  final String *REGISTRATION\_FIELD\_USER\_MAIL* = "//XCUIElementTypeTextField";  final String *REGISTRATION\_FIELD\_PASSWORD* = "//XCUIElementTypeSecureTextField[1]";  final String *REGISTRATION\_FIELD\_CONFIRM\_PASSWORD* = "//XCUIElementTypeSecureTextField[2]";  final String *REGISTRATION\_CHECK\_BOX\_AGREE* = "check box not selected";  final String *REGISTRATION\_BTN\_ARROW* = "right arrow";  final String *REGISTRATION\_FIELD\_FIRST\_NAME* = "//XCUIElementTypeTextField[1]";  final String *REGISTRATION\_FIELD\_MIDDLE\_NAME* = "//XCUIElementTypeTextField[2]";  final String *REGISTRATION\_FIELD\_LAST\_NAME* = "//XCUIElementTypeTextField[3]";  final String *REGISTRATION\_FIELD\_DATE* = "//XCUIElementTypeOther[1]";  final String *REGISTRATION\_FIELD\_GENDER* = "//XCUIElementTypeOther[2]";  final String *REGISTRATION\_DIALOG\_GENDER\_MALE* = "Male";  final String *REGISTRATION\_DIALOG\_GENDER\_FEMALE* = "Female";  final String *REGISTRATION\_COUNTRY* = "//XCUIElementTypeTextField[1]";  final String *REGISTRATION\_TOWN* = "//XCUIElementTypeTextField[2]";  final String *REGISTRATION\_ADRESS* = "//XCUIElementTypeTextField[3]";  final String *REGISTRATION\_CODE* = "//XCUIElementTypeTextField[4]";  final String *REGISTRATION\_PRIMARY\_NUMBER* = "//XCUIElementTypeTextField[5]";  final String *REGISTRATION\_SECONDARY\_NUMBER* = "//XCUIElementTypeTextField[6]";  final String *REGISTRATION\_DONE* = "Done"; //------------------------------------------------------------------------   */\*\*  \* MENU  \*/* //------------------------------------------------------------------------   final String *MENU\_HOME\_SETUP\_ICON* = "Home Setup";  final String *MENU\_SCENES\_ICON* = "Scenes";  final String *MENU\_TYPE\_ICON* = "Type";  final String *MENU\_LOCATION\_ICON* = "Location";  final String *MENU\_LEFT\_SETTINGS\_ICON* = "reveal icon";  //--}----------------------------------------------------------------------  } |

Figure 5 – Constants for iOS

|  |
| --- |
| package iOS;  import ...  */\*\*  \* Created by mt on 6/30/17.  \*/* public class Login{   IOSDriver driver;   @Before  public void setup() throws MalformedURLException {  driver = new IOSDriver<>(new URL(Constants.*SERVER\_URL*), Config.iOS.*getCapabilities*()); // Breaks here when i debugged  driver.manage().timeouts().implicitlyWait(60, TimeUnit.*SECONDS*);  }   @After  public void teardown() {  driver.quit();  }   */\*\*  \* Logging with valid data  \* After logging click on the menu as a sign of a successful test  \*/* @Test  public void correctLogin() throws InterruptedException {   System.*out*.println("Start correctLogin " +Config.*getTimeStamp*());  driver.findElement(By.*name*(ConstantsIOS.*LOGIN\_ALERT\_NOTIFICATION\_ALLOW*)).click();   Thread.*sleep*(1000);  driver.findElement(By.*xpath*(ConstantsIOS.*LOGIN\_FIELD\_EMAIL*)).sendKeys(Constants.*LOGIN\_CORRECT\_EMAIL*);  driver.findElement(By.*xpath*(ConstantsIOS.*LOGIN\_FIELD\_PASSWORD*)).sendKeys(Constants.*LOGIN\_CORRECT\_PASSWORD*);  driver.findElement(By.*name*(ConstantsIOS.*LOGIN\_BTN\_NEXT*)).click();  Thread.*sleep*(1000);   driver.findElement(By.*name*(ConstantsIOS.*MENU\_LEFT\_SETTINGS\_ICON*)).click();  Thread.*sleep*(1000);   System.*out*.println("Finish correctLogin " + Config.*getTimeStamp*());  } } |

Figure 6 –Login test for iOS

|  |
| --- |
| package iOS;  import ...  */\*\*  \* Created by mt on 6/30/17.  \*/ /\*\*  \* Created by mt on 7/7/17.  \*/* public class Registration {    IOSDriver driver;   @Before  public void setup() throws MalformedURLException {  driver = new IOSDriver<>(new URL("http://0.0.0.0:4723/wd/hub"), Config.iOS.*getCapabilities*()); // Breaks here when i debugged  driver.manage().timeouts().implicitlyWait(60, TimeUnit.*SECONDS*);  }   @After  public void teardown() {  driver.quit();  }    */\*\*  \* Registration with valid data  \* Checks the dialog message and completes the test  \*/* @Test  public void correctRegistration() throws InterruptedException {   System.*out*.println("Start correctRegistration " + Config.*getTimeStamp*());   driver.findElement(By.*name*(ConstantsIOS.*LOGIN\_ALERT\_NOTIFICATION\_ALLOW*)).click();  Thread.*sleep*(1000);   driver.findElement(By.*name*(ConstantsIOS.*REGISTRATION\_BTN\_CREATE\_ACCOUNT*)).click();   driver.findElement(By.*xpath*(ConstantsIOS.*REGISTRATION\_FIELD\_USER\_MAIL*)).sendKeys(Constants.*TEST\_CORRECT\_MAIL*);  driver.findElement(By.*xpath*(ConstantsIOS.*REGISTRATION\_FIELD\_PASSWORD*)).sendKeys(Constants.*TEST\_CORRECT\_PASSWORD*);  driver.findElement(By.*xpath*(ConstantsIOS.*REGISTRATION\_FIELD\_CONFIRM\_PASSWORD*)).sendKeys(Constants.*TEST\_CORRECT\_PASSWORD*);  driver.findElement(By.*id*(ConstantsIOS.*REGISTRATION\_CHECK\_BOX\_AGREE*)).click();  driver.hideKeyboard();  driver.findElement(By.*id*(ConstantsIOS.*REGISTRATION\_BTN\_ARROW*)).click();  Thread.*sleep*(1000);   driver.findElement(By.*xpath*(ConstantsIOS.*REGISTRATION\_FIELD\_FIRST\_NAME*)).sendKeys(Constants.*TEST\_TEXT*);  driver.findElement(By.*xpath*(ConstantsIOS.*REGISTRATION\_FIELD\_MIDDLE\_NAME*)).sendKeys(Constants.*TEST\_TEXT*);  driver.findElement(By.*xpath*(ConstantsIOS.*REGISTRATION\_FIELD\_LAST\_NAME*)).sendKeys(Constants.*TEST\_TEXT*);  driver.findElement(By.*xpath*(ConstantsIOS.*REGISTRATION\_FIELD\_GENDER*)).click();  driver.findElement(By.*id*(ConstantsIOS.*REGISTRATION\_DIALOG\_GENDER\_MALE*)).click();  driver.findElement(By.*id*(ConstantsIOS.*REGISTRATION\_BTN\_ARROW*)).click();  Thread.*sleep*(1000);   driver.findElement(By.*xpath*(ConstantsIOS.*REGISTRATION\_COUNTRY*)).sendKeys(Constants.*TEST\_TEXT*);  driver.findElement(By.*xpath*(ConstantsIOS.*REGISTRATION\_TOWN*)).sendKeys(Constants.*TEST\_TEXT*);  driver.findElement(By.*xpath*(ConstantsIOS.*REGISTRATION\_ADRESS*)).sendKeys(Constants.*TEST\_TEXT*);  driver.findElement(By.*xpath*(ConstantsIOS.*REGISTRATION\_CODE*)).sendKeys(Constants.*TEST\_NUMBER*);  //driver.findElement(By.xpath(ConstantsIOS.REGISTRATION\_PRIMARY\_NUMBER)).sendKeys(Constants.TEST\_NUMBER);  //driver.findElement(By.xpath(ConstantsIOS.REGISTRATION\_SECONDARY\_NUMBER)).sendKeys(Constants.TEST\_NUMBER);  //driver.hideKeyboard();  driver.findElement(By.*name*(ConstantsIOS.*REGISTRATION\_DONE*)).click();  driver.findElement(By.*id*(ConstantsIOS.*REGISTRATION\_BTN\_ARROW*)).click();  Thread.*sleep*(1000);   Alert alert = driver.switchTo().alert();  *assertEquals*((Constants.*REGISTRATION\_MESSAGE\_7*), driver.findElement(By.*id*(ConstantsIOS.*REGISTER\_DIALOG\_7*)).getText());  alert.accept();   Thread.*sleep*(2000);   System.*out*.println("Finish correctRegistration " + Config.*getTimeStamp*());   }   public IOSDriver getDriver() {  return driver;  } } |

Figure 7 – Register test for iOS

# Software Requirements

Appium is an open source, cross-platform test automation tool for native, hybrid and mobile web apps, tested on simulators (iOS, FirefoxOS), emulators (Android), and real devices (iOS, Android, Windows, FirefoxOS).

## SUPPORTED PLATFORMS

* iOS
* Android
* Windows
* FirefoxOS

Your environment needs to be setup for the particular mobile platforms that you want to run tests on. If you want to run Appium via an npm install, hack with or contribute to Appium, you will need node.js and npm v6 or greater (use n or brew install node to install Node.js. Make sure you have not installed Node or Appium with sudo, otherwise you’ll run into problems). We recommend the latest stable version.

To verify that all of Appium’s dependencies are met you can use appium-doctor. Install it with npm install -g appium-doctor (or run it from source), then run appium-doctor and supply the --ios or --android flags to verify that all of the dependencies are set up correctly.

You also need to download the Appium client for your language so you can write tests. The Appium clients are simple extensions to the WebDriver clients. You can see the list of clients and links to download instructions at the Appium clients list.

### IOS REQUIREMENTS

* Mac OS X 10.12 recommended
* XCode 8 recommended
* Apple Developer Tools (iPhone simulator SDK, command line tools)
* Ensure you read our documentation on setting yourself up for iOS testing!

### ANDROID REQUIREMENTS

* Android SDK API >= 17 (Additional features require 18/19)
* Appium supports Android on OS X, Linux and Windows. Make sure you follow the directions for setting up your environment properly for testing on different OSes:
* linux
* osx
* windows

### WINDOWS REQUIREMENTS

* Windows 10