**How-To Use the SLX Mobile Client, Selenium WebDriver Test Automation Framework**

The purpose of this document is help guide you in setting up and using the SLX Mobile Client test automation framework for Selenium WebDriver.

**Preparation:**  
It is assumed that you have the following setup and ready:

* Test VM with Windows 7 (or greater) with at least 2GB of RAM and 30GB of hard-disk space.
* Java JDK:  
  <http://www.oracle.com/technetwork/java/javase/downloads/index.html>
* Selenium WebDriver:  
  <http://www.seleniumhq.org/download>
* Eclipse IDE:  
  <http://www.eclipse.org/downloads>
* TestNG Framework:  
  <http://testng.org/doc/download.html>
* Git and GitHub account:

<https://help.github.com/articles/set-up-git>

* Access to the <NAME OF GITHUB REPO TBD>:

<https://github.com/TBD>

**Step:** Download or pull-down the Framework Test Scripts

Option 1 (recommended): Use Git commands to pull the framework from the GitHub repo.  
 <https://githubrepopath-TBD>

Option 2: Copy the test framework folder from a network shared folder. Folder path:  
 [\\tlautomation\selenium\selenium\_projects](file:///\\tlautomation\selenium\selenium_projects)

Option 3: Download a .ZIP of the test framework from the GitHub repo.  
 (click the **Download ZIP** button from the right-rail of the repo page)

**Step:** Import the SLX8Mobile3x project into your Eclipse IDE workspace

1. Launch the Eclipse IDE.
2. Import the **SLX8Mobile3x** framework project into the Package Explorer.
3. Select the “Existing Projects into Workspace” option under the General group. Then use the framework file path determined from Step 1 for the “Select root directory” field. Be sure to also check the “Copy projects into workspace” check box.  
   (reference:<http://help.eclipse.org/juno/index.jsp?topic=%2Forg.eclipse.platform.doc.user%2Ftasks%2Ftasks-importproject.htm>)
4. You should see the SLX8Mobile3x project listed in the Package Explorer window.
5. You may right-click on the **SLX8Mobile3x** project folder in Package Explorer and select the **Properties** menu item to see the local path folder of the project.

**Step:** Setup the *app.properties* test configuration file

1. Expand the **SLX8Mobile3x** project to see all sub-files and folders.
2. Click to open and edit the ***app.properties*** file under the SLX8Mobile3x project. The ***app.properties*** file defines the test configuration and other important variables.
3. Below is the list of the key variables to modify to suite your SLX 8.x test environment:  
   - base\_url: IP/server name portion of the SLX 8.x site URL where the Mobile Client can be   
    accessed from. (e.g. “http://107.21.243.158/”)  
   - mobile\_url: folder portion of the SLX 8.x site URL (e.g. “Slxmobile”).  
   - user\_name: username of the SLX test user (e.g. “Lee”)  
   - user\_pwd: password of the SLX test user; leave un-set for blank password  
   - version\_lbl: full version string from the Mobile Client Log On page; this should be modified  
    only if there is a version label change (e.g. “Mobile v3.0.0 / Saleslogix v8.0.0”)
4. Save and close the *app.properties* file after s are completed.

The *app.properties* file can be modified if there is any changes to the test environment.

**Step:** Test Classes and test methods

1. All executable test classes under the **src/test/java/argos.saleslogix.selenium** package have a “…**Test**” within the .java filename (e.g. AccountEntityViews**Test**.java). All other .java files are non-executable.
2. Test methods within a .java class all start with a “test…” in the method name. (e.g. “**test**01\_SeTestTCAccountListView()”).
3. The number immediately after the “test” and before the first underscore “\_” denotes the order of test execution (i.e. “test01\_...” is executed before “test02\_...”).
4. Individual test methods can be enabled or disabled via the “**@Test(enabled = [true|false])**” annotation. To enable a test, set the “enabled=true”. Conversely, set “enabled=false”.

**Step:** Test Class Execution under TestNG

1. All test methods within a test class can be executed from the right-context menu .
2. Click to expand the Test .java class file node.
3. Right-click the test class node (denoted by a “C” within a green circle), then select “**Run As -> TestNG Test**”.
4. All enabled test methods within the test class will be executed in their order.

**Step:** Test Suite execution using a *testng.xml* file

1. A test suite can be setup by defining what test classes to execute within a ***testng.xml*** file.
2. A ***testing.xml*** file has the following format:  
     
   <?xml version="1.0" encoding="UTF-8"?>  
   <!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd">  
   <suite name="Mobile3x Full-Test Suite" verbose="3" parallel="classes">  
     
    <test name="Test Class A - FF">  
    <parameter name="browser" value="FF"></parameter>  
    <classes>  
    <class name="argos.saleslogix.selenium.test.FirstTest"/>  
    </classes>  
    </test> <!-- Default test -->

<test name="Test Class B - Chr">  
 <parameter name="browser" value="Chrome"></parameter>  
 <classes>  
 <class name="argos.saleslogix.selenium.test.SecondTest"/>  
 </classes>  
 </test> <!-- Default test -->  
  
</suite>

1. The sections in blue color text are the header and footer of the .xml file.
2. The test suite name is defined by the “suite name” property and the corresponding value.
3. Individual test classes (section in green text) for execution are specified between the header and footer.
4. The test parameters can be specified within the individual test sections.
5. The tests will be executed in the list order within the ***testng.xml*** file.
6. The ***testng.xml*** can be executed as a whole by right-clicking the file-name and selecting the “**Run As -> TestNG Test**” context menu.

**Step:** The BrowserSetup.java class

* The BrowserSetup class defines the main test parameters and utility methods.
* Test class execution handling is controlled by the annotations defined in this class.
* Each Test class is required to extend the BrowserSetup class.  
    
  public class AccountEntityViewsTest **extends BrowserSetup** {  
  }
* There is no need to instantiate this class within test methods. All test classes have access to the members and methods in this class.

**Step:** The CommonNavigation.java class

* The CommonNavigation class defines WebElements of all/most of the common GUI controls and pages (e.g. the Global Navigation Menus).
* This class also defines a collection of commonly-used test methods and utility routines.
* It is highly recommended to include an instance of the CommonNavigation class within each test method.  
    
  CommonNavigation commNav = PageFactory.initElements(driver, CommonNavigation.class);

**Step:** Relation between Test .java classes and corresponding ViewsElements.java classes

* Some Test .java classes define their required WebElements and utility methods within an accompanying “…ViewsElements.java” class (e.g. AccountViewsElements.java and AccountEntityViewsTest.java).
* The purpose of the separation between the Test .java class and ViewsElements.java class is to make it easier to manage, organize and maintain the tests. In short, tests can be defined in one .java file while the dependencies can be defined in another.

**Step:** Console log and test reports

* Raw test results – including check point passes/fails, errors, exceptions and call stack traces – are listed within the console log. The console log is very useful for tracing through and debugging any script errors.
* The TestNG test reports provide a more user-friendly results reporting mechanism than the console log. There is a **\test-output** folder under the test project. This folder contains an **emailable-report.html** that contains a test summary with links to the individual test method results. The emailable-report.html and linked resources are overwritten on each completed test run. The **\test-output** folder can be copied to under another folder outside the project in order to preserve the test results.
* For any test error encountered, a screen shot of the browser application is captured and stored as .PNG files under the **\test-output** folder. It is recommended to clean out any old screen shot .PNG files that are no longer needed.