

**Java Selenium**

**Automation Framework**

For

All Test Automation Projects

Version 0.1

Last Updated **1/3/2018**

**Prepared by Mark Elking**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Version** | **Date** | **Author(s)** | **Description of Change** | **Approved By** |
| 1.0 | 12/29/2016 | Mark Elking | Initial Draft |  |
| 2.0 |  |  |  |  |
| 3.0 |  |  |  |  |

Revision History

Table of Contents

[Introduction 5](#_Toc498441436)

[1.0 Application/Project Setup 5](#_Toc498441437)

[1.1 GitHub 5](#_Toc498441438)

[**1.1.1** **Push Code into GitHub Repository for the First Time** 5](#_Toc498441439)

[**1.1.2** **Configure Eclipse to Clone Repository from Git First Time** 6](#_Toc498441440)

[1.2 Directory/Folder Structure 6](#_Toc498441441)

[1.2.1 logs 7](#_Toc498441442)

[1.2.2 src 7](#_Toc498441443)

[1.2.3 maps 7](#_Toc498441444)

[1.2.4 variables 7](#_Toc498441445)

[1.2.5 webservices 7](#_Toc498441446)

[1.2.6 testdata 7](#_Toc498441447)

[1.2.7 bin 7](#_Toc498441448)

[1.2.8 driver 7](#_Toc498441449)

[1.3 Utility 8](#_Toc498441450)

[1.4 Configuration Map 8](#_Toc498441451)

[1.4.1 **ALM** 8](#_Toc498441452)

[1.4.2 **Environment** 8](#_Toc498441453)

[1.4.3 **Global** 9](#_Toc498441454)

[1.5 Test Utility 9](#_Toc498441455)

[1.5.1 **ExecParms** 9](#_Toc498441456)

[1.5.2 **EventNames** 9](#_Toc498441457)

[1.5.3 **Object** 10](#_Toc498441458)

[1.5.4 **Test Cases** 11](#_Toc498441459)

[1.5.5 **Component** 13](#_Toc498441460)

[1.5.6 **Devices** 13](#_Toc498441461)

[2.0 Event to Function Mapping 17](#_Toc498441462)

[3.0 Java Development Standards 28](#_Toc498441463)

[4.0 Execution 30](#_Toc498441464)

[4.1 Eclipse 30](#_Toc498441465)

[4.2 Jenkins 30](#_Toc498441466)

[5.0 Framework 34](#_Toc498441467)

[5.1 Technical Architecture 34](#_Toc498441468)

[5.2 Code Flow 36](#_Toc498441469)

[5.3 Test Utility 36](#_Toc498441470)

[5.4 Troubleshooting & Reporting – Log Parser 37](#_Toc498441471)

[5.5 Troubleshooting & Reporting – ALM Test Set Metrics 38](#_Toc498441472)

Document Glossary of Terms

|  |  |
| --- | --- |
| **Term** | **Description** |
| Example | Brief description or definition |
|  |  |
|  |  |

# Introduction

**This document outlines the detailed setup for automating all applications using the Java Selenium Automation framework established for the TMO Automation Team at Charter.**



# Application/Project Setup

The application/project setup will include GitHub, directory/folder structure, configuration map spreadsheet and Main function setup.

## GitHub

One spreadsheet needs to be created for managing test case data input and test configuration and object

### **Push Code into GitHub Repository for the First Time**

1. Go into GitHub and create new repository called nextgen (as an example). See Mark to have this repository created.

2. Log into your development box using your userID (as an example)

3. create directory C:\Users\{userID}\git\nextgen (nextgen is an example).

* An example folder structure has been created and stored at X:\Automation\Framework\Initial GIT Folder Structure. This structure can be copied over to your development box for the initial GIT push. Please make sure to update all project specific folders and files to your project name. (Example project used for structure is Amr).

4. Launch DOS Prompt

5. cd to that directory just created in Step #3 and issue following DOS commands from that directory

6. git init

7. git add --all

8. git commit -m "20161107 - Baseline"

9. git remote add origin https://{userID}@git.corp.chartercom.com:8443/scm/taa/nextgen.git

NOTE: IF YOU GET ERROR that origin already exists, then run this command first...

git remote rm origin

10. git push -u origin master

NOTE: if you get a Fatal Authentication because credentials to Git are preset and they need to be edited, then go to Control Panel\All Control Panel Items\Credential Manager and edit the Git entry.

### **Configure Eclipse to Clone Repository from Git First Time**

DOS

1. On 6659, log in using svc\_automation user ID
2. Launch DOS
3. Navigate to C:\Users\svc\_automation\git
4. Run following DOS command…

git clone ssh://git@git.corp.chartercom.com:7999/taa/webservices.git

1. Create a readme.txt file inside of C:\Users\svc\_automation\git\webservices
2. Navigate to DOS directory C:\Users\svc\_automation\git\webservices

Eclipse

1. Launch Eclipse

2. In left hand project explorer, select Import

3. Select Git\Projects from Git

4. Select Next

5. Select CloneURI

6. Fill out information

URI: ssh://melking@git.corp.chartercom.com:7999/taa/nextgen.git

Host git.corp.chartercom.com

Repository Path /taa/nextgen.git

Protocol ssh

Port 7999

User melking

Password ..........

click Next button

7. Hit Next

8. Click Finish

## Directory/Folder Structure

The directory folder structure will need to be created for the application/project. There are required folders.

### logs

The Java Selenium execution output log files will be placed in this directory.

### src

All of the Java code will be located in this directory including .java class files and all functions.

### maps

The Configuration Map file will be located in this directory. There will be one .xlsm configuration map file per project/application.

### variables

Many functions create temporary variable files when called and store those temporary variable files in this directory.

### webservices

We have a video for demo/training on how to use/build webservices in the Java Selenium Framework.

[**https://webconference.twcable.com/p6y9fxo48m6/**](https://webconference.twcable.com/p6y9fxo48m6/?OWASP_CSRFTOKEN=84bfe7a34acaffc9874b16b6b0610360c0893926ef231c7ce6e034272cda145e)

All webservice files reside in this directory. There are 2 mandatory subdirectories, 1) requests and 2) responses. Also, there are 2 manadatory subdirectories under requests, namely, a) templates and b) runtime. The templates directory contains all of the XML requests which are parameterized with variables to be substituted. All variables should be prefixed with “XML\_”. An example is “XML\_PHONE\_NUMBER”. During runtime, the XML runtime request is generated and stored in webservices\requests\runtime directory. Also, the response of the xml request is placed into the webservices\responses directory.

### testdata

Contains all of the

### bin

Contains all of the binary files.

### driver

Contains all of the

## Utility

There is now 1 test case creation/maintenance utility. Below video explains how to use.

[**https://webconference.twcable.com/p8cli3ij22h/**](https://webconference.twcable.com/p8cli3ij22h/?OWASP_CSRFTOKEN=05c2b97a27df865a2f3c8d6fb7d77e31aa7b90577ffb7da1f5d3e9bf8f2b4ea1)

## Configuration Map

One spreadsheet needs to be created for managing test case data input and test configuration and object definition parameters. The file format must be .XLSM and must be located under directory…..{Application source directory}\maps\{Application}ConfigurationMap.xlsm. Below is the list of mandatory worksheets in this spreadsheet.

### **ALM**

Contains all of the ALM configuration parameters for the given application

4 columns (Attribute, Value, JenkinsSmoke, JenkinsRegression) to the ALM worksheet also.  NOTE: these column headers are case-sensitive. This gives you the ability to configure different test sets/folders per type of testing you are doing (unit/dev, Jenkins smoke, Jenkins regression).

### **Environment**

Contains list of all environments available for automation of that application

For webservices, On the Environment worksheet, add 2 new columns for each new webservice. The names of the columns should be {webservicename}ENDPOINT and {webservicename}CREDS or if there are shared endpoints across multiple webservices then you do not need to add additional redundant columns – the same column can be referenced using notation (see XMLExecute section of this document for details).

Here are examples….

addOrderENDPOINT & addOrderCREDS

For the corresponding environment for which you plan to execute this webservice, provide the ENDPOINT and CREDS information.

ENDPOINT example <https://ebs-uat.corp.charterom.com/csg_cter/2.06/OrderDetailService.asmx>

CREDS example chtr\svc\_tst\_automation:H2i1fL9!

If Endpoint is dynamic (as stored in the Environments worksheet) then provide the following format…

ENDPOINT example

([https://prov-qa.corp.chartercom.com/video-provisioning-service/services/rest/video/service/v2/accounts/{accountNumber}/services/{serviceId}/activate-video-requests](https://prov-qa.corp.chartercom.com/video-provisioning-service/services/rest/video/service/v2/accounts/%7baccountNumber%7d/services/%7bserviceId%7d/activate-video-requests)

The values inside the curly braces represent the name of the variable file to reference. If the endpoint contains a curly brace, then the value inside the curly braces will be substituted with the value in the corresponding variable files.

### **Global**

Contains all of the Global configuration parameters for the given application

## Test Utility

One test utility is located in maps directory of Framework GitHub repository for managing several entities.

### **ExecParms**

This worksheet has a button that when clicked will read all of the worksheets in the spreadsheet and update the list of SheetName column values with all worksheets in the spreadsheet. Any comments associated with a given TC will be preserved and re-associated. BrowserType dropdown values pull from BrowserType worksheet. When a BrowserType is selected it will automatically update all TCTestData values for all corresponding LaunchBrowser events across all TC prefixed worksheets. Environment dropdown values pull from Environment worksheet. When an Environment is selected it will automatically update all TCTestData values for all corresponding NavigateToURL events across all TC prefixed worksheets. Each TC sheetname will have a corresponding Value column dropdown value of Y or N. If Y is selected, then that test will be executed when Java main function is executed. If N is selected, that test will not be executed. You can have multiple test cases with the value Y and the Java main function will run those tests in the order listed from top to bottom on the ExecParms worksheet.

5 columns (SheetName, Value, JenkinsSmoke, JenkinsRegression, Comments). NOTE: these column headers are case-sensitive. You can configure execution of Development testing via “Value” column, Jenkins smoke test testing via “JenkinsSmoke” column and Jenkins regression testing via “JenkinsRegression” column.

### **EventNames**

List of TC events available. This is a fixed list and corresponds to a function call in the Framework fw\_event function.

### **Object**

Must have 3 columns named,

* + - 1. **COLUMN ObjectName**

the format of the value should be PAGENAME\_FIELDNAME, example is SEARCH\_Address1 where SEARCH is the page name and Address1 is the field label.

* + - 1. **COLUMN TagName,Attribute,Value**

the value should be tagname + “,” + attribute + “,” + value. Examples include:

Button,NA,Search

NA,id,addressLine1

input,xpath,//\*[@id=”container”]/button

input,xpath,//\*[span=’---FILE\_accountnumber---/../..

The developer can define Object definitions with dynamic variables. Here is an example……on the Object worksheet, you would have the following TagName,Attribute,Value definition

* + input,xpath,//\*[text()=’---FILE\_accountnumber---‘/../..
  + Notice the 3 mandatory dashes to the left and right of the variable file reference.
  + FILE\_ must be caps as is our standard. Notice also that accountnumber does NOT have any underscores in the variable file accountnumber – this is a mandatory requirement as well, no underscores in the variable file name.
  + During execution, the substitution is made and would look like the following…..

input,xpath,//\*[text()=’834538293837483‘/../..

* + - 1. **COLUMN ExtraInfo**

Any other information that you need to have associated per field can be used in the ExtraInfo column (if needed)

### **Test Cases**

There must be N test cases named “TC{ALMTestID}”, this is case-sensitive. If you are automating 20 test cases, then you will need 20 corresponding entries in database. Examples include: “TC27473”, “TC27474” or “TC27474-8623”.

fw\_update\_hpalm\_test\_case\_execution\_status –function modified to handle test case id/configuration id combination if needed.

There are instances were manual test cases are created in ALM with corresponding configuration IDs. To uniquely identify those tests in a test set for execution reporting requires a test id/configuration id combination. Format now for test worksheet is TC{alm test id} or TC{alm test id}-{alm configuration id}. If your tests do not require this then you have nothing to change. Currently, Gateway tests have moved to this standard and require TC{alm test id}-{alm configuration id}, an example is “TC27474-8623” where 27474 is the ALMTestID and 8623 is the ALMConfigurationID. NOTE: there can be multiple configuration IDs per test case ID.

Test Cases will have 5 columns, all are case-sensitive.

* + - 1. **COLUMN TCObjectName**

which is a dropdown data validation list linked to ObjectName named range on Object worksheet.

* + - 1. **COLUMN TCTestData**

free form text field. If EnterDataTextbox event, then TCTestData should have some text value. NOTE: all text is automatically Cleared before entering the data into the text field. If you do NOT want to CLEAR the textbox value before entering a value into the textbox, then you would put in “NOCLEAR,3”.

NOTE: if the value of TCEventName is Component, then TCObjectName can be NA and TCTestData value must be the {PageName}

* + - 1. **COLUMN ObjectToLookForAfterObjectEvent**

if you want to control the rate at which the script is executing by interrogating the page document AFTER you click a button (for example), then you can search for a given text on the page document until it’s found or until the loop counter/time you configure is met. Format of this value is “NA” or the following……..

tagname + “,” + text to search for + “,” + number of loops to keep looking + “,” + milliseconds to wait per loop

Examples include:

h4,Search,15,1000

span,Customize Offers,30,1000

Default value is NA

An example is “label,Order Reasons(s),15,1000”. So if I click a button and this column value is NOT NA and it has value of “label,Order Reasons(s),15,1000” then the code will look for the Order Reasons(s) text in the label tagname on the page for a total of 15 seconds, checking every 1 second 15 times. If the text is found at 3 seconds, then it will not continue to look for the additional 12 seconds. Inside control is passed on to the next event in the test case.

For Webservices, do the following…..

If you want the webservice to keep executing until a certain value in the response is found, then update the ObjectToLookForAfterObjectEvent column with the following value

{text to look for in the XML response}--{number of loops to check}--{milliseconds to wait per loop}

Here is an example….

ReciveFromActivation--3--5000

If you have dynamic data to pass, then do something like the following……

ReciveFromActivation<Account>,FILE\_AccountId,</Account>--3--5000

…..where the text highlighted in Yellow is literal text. NOTE: the commas are NOT literal text to be validated but they separate the literal text from the dynamic text. Dynamic text comes from the variable file referenced. So in this example, if account ID value 12345 was in the file “AccountId”, then the text that will be validated in the XML response file will be the following…..

ReciveFromActivation<Account>12345</Account>

And the check will occur every 5 milliseconds for 3 loops. So total of 15 seconds.

* + - 1. **COLUMN MillisecondsToWaitAfterObjectEvent**

default value is 0. If you want the script to wait for some hard coded time interval after the event. NOTE: regardless of whether or not there is an ObjectToLookForAfterObjectEvent specified, this MillisecondsToWaitAfterObjectEvent will occur. The ObjectToLookForAfterObjectEvent (if not NA) will occur first, then the MillisecondsToWaitAfterObjectEvent will occur next.

* + - 1. **COLUMN TCEventName**

which is a dropdown data validation list linked to EventNames worksheet. See Section 2.0 – Event to Function Mapping for a list of the available Event Names.

### **Component**

There can be many components defined per application. The format of the component is the same as test cases, 5 columns. Component is for commonly used events/code which can be re-used by many test cases.

Usability of Component functionality

* + 1. You can Query all components or 1 individual component
    2. You can Update all components or 1 individual component
    3. You cannot update 3 components in same view unless 3 is the total number of components for that given application

### **Devices**

List of TC events available. This is a fixed list and corresponds to a function call in the Framework fw\_event function.

1. Test Utility.xlsm - created Device worksheet in Test Utility.xlsm for maintaining MTA and Worldbox devices in Devices Oracle table
2. created fw\_testdata\_get\_device function to get next available device for automation tester when creating test cases
3. created GetDevice event in fw\_event – please do a Query on EventNames worksheet to see new GetDevice event
4. There are 8 columns of data to maintain on the Device worksheet.
   * 1. Type – T2, T3, T6 or MTA
     2. RealFake – F or R
     3. CMMAC
     4. MTAMAC
     5. VIDEOSerial
     6. VIDEOESTBMAC
     7. VIDEOESMMAC
     8. PortType – 2Port, 4Port, 8Port
     9. Available – Y or N
     10. Env – TEST, QA1, SIT3, etc

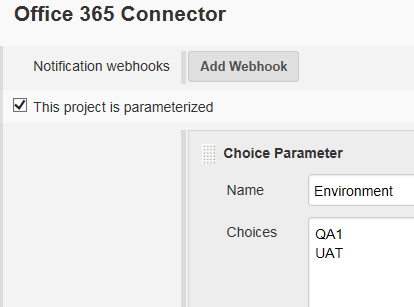
All dropdown values are maintained via DeviceInfo worksheet.

1. There are 2 request types on the Device worksheet.
   * 1. QUERY – will pull back all data in the Devices table.
     2. UPDATE – inserts records into the Oracle table. NOTE: this UPDATE does a backup of all devices data before inserting records into the table. The records that will be inserted into the table will be all of the records in the view on the Devices worksheet before clicking the Submit Button for Request Type of UPDATE.
2. No Devices data can be removed from the Devices table via the Device utility.

Framework release notes 5/10 – Environment to Select for Execution

* The following enhancement allows developers, manual testers to dynamically select which environment they want to execute via runtime instead of configuring the environment to run tests in the Maps spreadsheet.  Here is a link to the recording of how to configure Global and Environment parms and how to configure Jenkins jobs for selecting and Environment via Jenkins……….X:\Automation\Framework\Recordings\Global and Environment Parms Configuration and Execution
* modified fw\_get\_environment\_to\_execute\_tests to pull options from Database for Eclipse developers and let them select via dropdown and for Jenkins pulls from Choice Parameter generated environment selected value
* The environment you select for execution will no longer be configured via the ExecParms worksheet.  Please remove the ExecParms worksheet from the {Application}Maps.xlsm – Make sure to commit your Maps.xlsm change back to GitHub.  NOTE: the only remaining worksheet in that Maps spreadsheet should be ALM.
* If you are running via Eclipse as a developer, you will be prompted with a dropdown box of environments which pulls from the TBL\_ENVIRONMENT table for that application and allows you to select the environment you want to run in.
* Prior to running via Jenkins (Smoke and/or Regression jobs), you will need to make the following Configuration updates one time only.

1. Add a Choice Parameter named “Environment” in the Office 365 Connector section with the Environment values for the respective Application (QA1, UAT, etc).  These values should come from the first column on the Environment worksheet in the Test Utility.xlsm
2. In the Build section you will need to add the command echo %Environment% > c:\Temp\JENKINSENVIRONMENTREGRESSION above the JAR file execution statement (as shown below).  NOTE: the filename should be JENKINSENVIRONMENTSMOKE for Jenkins Smoke jobs.
3. When you execute the Jenkins, you will select “Build with Parameters” option.  Previously, without the Choice Parameter, it was “Build Now”.  Once you select “Build with Parameters”, a dialog box will appear allowing the user to select the environment value, select the environment and click OK and the Jenkins job executes normally.



These changes have been pushed to Framework GitHub.  You will need to do a Fresh pull of Framework code to see the changes.

\*\*\*\*Also, please remove the ExecParms worksheet from the Maps spreadsheet for your respective application and push that application Map change back to the .xlsm into GitHub. – Please have this completed by EOD on 5/10\*\*\*\*\*\*

Also, remember, before running any Jenkins Smoke or Jenkins Regression jobs, you will need to navigate to all Jenkins Slave machines and delete the Framework directory on those Jenkins slaves in the following location: C:\Users\svc\_automation\.m2\repository\com\chtr\tmoauto or C:\Windows\SysWOW64\config\systemprofile\.m2\repository\com\chtr\tmoauto

# Event to Function Mapping

The Java Functions that support this automation solution include the following.

| **ID** | **Event Name (or Calling Function)** | **TCTestData** | **Function/Description** |
| --- | --- | --- | --- |
| 1 | Manual execution via Eclipse or via Jenkins | NA | Main function inside of {Application}Tests.java. See main function example in GatewayTests.java |
| 2 | Main function | NA | fw\_get\_list\_of\_test\_cases\_to\_execute  Called from the Main function. Pulls all test IDs from the ExecParms worksheet which have a corresponding value of “Y”. |
| 3 | Main function | NA | fw\_create\_output\_log\_file – called to create text output log file. |
| 4 | Main function | NA | fw\_event – this function has 7 arguments   1. configuration\_map\_fullpath – full path of input file. 2. tab\_name – worksheet that is to be used i.e. TC24734 or Component 3. tc\_event\_name – TCEventName value (ClickButton, EnterDataTextbox, SelectListValue, SelectCheckbox,GetText,StopExecution,NA). 4. tc\_object\_name – TCObjectName value, example is “OFFER\_Next”. However, if you want to execute all of the objects on a given page, then just indicate the value “OFFER” and it will execute all of the objects on that OFFER page. 5. tc\_test\_data – TCTestData value. Freeform value to enter data into text box or select from listbox or key prefixed value for Component worksheet. 6. object\_to\_look\_for\_after\_object\_event – value from the TC prefixed worksheet ObjectToLookForAfterObjectEvent 7. milliseconds\_to\_wait\_after\_object\_event – value from the TC prefixed worksheet MillisecondsToWaitAfterObjectEvent |
| 5 | LaunchBrowser | IE or CHROME | fw\_launch\_browser  This will open the browser and sets current driver to IE or CHROME. |
| 6 | NavigateToURL | TCTestData value should only be the column name corresponding to the URL as defined by the column name in the Environment worksheet.  Input comes from TCTestData column, TCObjectData name no longer required. The environment value (i.e. QA, UAT, etc) in the TCTestData previously does not need to be provided any longer in TCTestData column. | fw\_navigate\_to\_url  Navigates to url and maximizes window. |
| 7 | fw\_event | Value not needed here | fw\_check\_for\_loading\_page |
| 8 | TerminateWindowProcesses | List of window processes to terminate comma delimited  For example  “excel.exe,chrome.exe” | fw\_terminate\_window\_processes |
| 9 | EnterDataTextbox | Value to enter into textbox  For example “1234”.  If you want to pull value from variable file, then indicate “FILE\_{name of variable file}” to pull from.  If you want to send TAB, then put “KEYTAB,9”…..9 represents the number of times to tab.  If you want to send ENTER, then put “KEYENTER”. | fw\_enter\_data\_into\_text\_field |
| 10 | SelectCheckbox | Value not needed here | fw\_select\_checkbox |
| 11 | ClickButton | Value not needed here | fw\_click\_button |
| 12 | SelectListValueByValue | Value to be selected from the list. | fw\_select\_from\_a\_list\_by\_value |
| 13 | SelectListValueByVisibleText | Value to be selected from the list. | fw\_select\_from\_a\_list\_by\_visible\_text |
| 14 | GetText | Value not needed here | fw\_get\_text |
| 15 | Currently called from fw\_closedown\_test (which is in Logging.java) | NA | fw\_quit\_driver  Quits the active driver. |
| 16 | SwitchToDriver | IE or CHROME | fw\_switch\_to\_driver  This function switches current driver to IE or CHROME. |
| 17 | No Event | Using locator and locatorvalue, the webelement is determined and then……  *driver*.switchTo().frame(webelement); | fw\_switch\_frame |
| 18 |  | Gets current handle, switches to new window, closes new window, then returns back to original window. | fw\_close\_window |
| 19 | GetWindowHandle | Gets current window handle. | fw\_get\_window\_handle |
| 20 | SwitchToNewWindow | Switches to new window | fw\_switch\_to\_new\_window |
| 21 | SwitchToWindow | window\_handle\_value  which gets passed into the following method  *driver*.switchTo().window(window\_handle\_value); | fw\_switch\_to\_window |
| 22 | AcceptAlert | Value not needed here | fw\_accept\_alert |
| 23 | ClickJAVASCRIPT | Value not needed here | fw\_click\_element\_using\_javascript |
| 24 |  |  | fw\_get\_webelements\_object  Returns a list of webelementss |
| 25 |  |  | fw\_get\_element\_object  Returns by value for selecting object. |
| 26 | ValidateText | {Expected Value},{Actual Value}  Example is “Jingle,JingleBells”. | fw\_validate\_text  The validate text function checks to see if the expected value is CONTAINED inside the actual value. It’s not an EQUALS comparison. It’s a contains comparison. |
| 27 | XMLExecute | Below is a video for demo/training on usage of webservices  [**https://webconference.twcable.com/p6y9fxo48m6/**](https://webconference.twcable.com/p6y9fxo48m6/?OWASP_CSRFTOKEN=84bfe7a34acaffc9874b16b6b0610360c0893926ef231c7ce6e034272cda145e)  Values can be passed into an XML template in 2 ways, 1-hard coded in spreadsheet OR 2-by referencing a file which has a value in there for substitution.  Enter values to substitute in TCTestData column in following format 1) XML\_LOCATION\_ID,12345 (hard coded example), or 2) XML\_LOCATION\_ID,FILE\_LocationID (dynamic example where LocationID is the name of the file in the variables directory which contains value of LocationID. To reference the file, make sure the word file is capitalized and has an underscore. | fw\_execute\_xml  Create an Object on the Object worksheet entitled “{webservicetype}\_{webservicenametemplatefilename}” or “{webservicetype}\_{webservicenametemplatefilename}\_{columnnameEnvironmentsworksheet}”.  Webservicetype can be WEBSERVICE or JSON.  Example is “JSON\_addHouse” or is “WEBSERVICE\_addHouse\_CSG”. Make sure TCObject definition is “NA,NA,NA”  Create a template xml file located under webservices\templates directory. Substitute any hard coded values in that template request xml file with variables prefixed with XML. For example, “XML\_ACCOUNT\_NUMBER”. Should be all caps.  XMLExecute section in fw\_event function – code modified to pull input from webservice URL and Credential variable files as defined by environment selected at beginning of test case. |
| 28 | XMLGetValueByTagName | This event gets a value for a specified tagname from the webservice response xml file. After the XMLExecute event is called, a webservice response file is created. This XMLGetValueByTagName event will retrieve the corresponding value out of the response file for the tagname that you specify in TCTestData. NOTE: do NOT include the “<” and the “>” values in the TCTestData. If a tagname is <address> in the response xml file. Then in TCTestData, you would indicate “address” (no < or > needs to be specified). | fw\_get\_value\_from\_xml\_based\_on\_tagname  Below is a video for demo/training on usage of webservices  [**https://webconference.twcable.com/p6y9fxo48m6/**](https://webconference.twcable.com/p6y9fxo48m6/?OWASP_CSRFTOKEN=84bfe7a34acaffc9874b16b6b0610360c0893926ef231c7ce6e034272cda145e) |
| 29 | SetVariable | Put name of the variable (case-sensitive) followed by a comma follow by value you want for this variable. Below example:  Sysprin,82451200 NOTE: A file named “Sysprin” will be created in your “variables” folder with the variable value “82451200”. You must have a variables folder defined in your workspace. NOTE: the TCObjectName should be “NA”. | fw\_set\_variable |
| 30 | CheckForElementExistence | Value something like (10,1000) where 10 is number of loops to check for element and 1000 is the number of milliseconds to wait per loop. | fw\_check\_element\_existence |
| 31 | GetAttribute | Value is attribute name you want to get value of like (img or data-reactid, etc). | fw\_get\_attribute\_value |
| 32 | GetCurrentDate | Value is “{variable name},{format of date}”. Here is an example: “CurrentDate,yyyy-MM-dd”  NOTE: TCObjectName value is NA. | fw\_generate\_datetime\_current |
| 33 | GetFutureDate | Value is “{variable name},{format of date},{number of days in future}”. Here is an example: “FutureDate,yyyy-MM-dd,10”.  NOTE: TCObjectName value is NA. | fw\_generate\_datetime\_future |
| 34 | WriteLogHeader | the header block name that you want to appear in your output log file. | fw\_writeLogEntry |
| 35 | XMLValidateTextinXMLResponse | This event will validate text in an XML Response. The text to validate should be put into TCTestData column.  If you have dynamic data to pass, then do something like the following……  ReciveFromActivation<Account>,FILE\_AccountId,</Account>…..where the text highlighted in Yellow is literal text. NOTE: the commas are NOT literal text to be validated but they separate the literal text from the dynamic text. Dynamic text comes from the variable file referenced. So in this example, if account ID value 12345 was in the file “AccountId”, then the text that will be validated in the XML response file will be the following…..ReciveFromActivation<Account>12345</Account>. NOTE: this validation check will occur only 1 time. | fw\_validate\_text\_in\_xml\_response  Below is a video for demo/training on usage of webservices  [**https://webconference.twcable.com/p6y9fxo48m6/**](https://webconference.twcable.com/p6y9fxo48m6/?OWASP_CSRFTOKEN=84bfe7a34acaffc9874b16b6b0610360c0893926ef231c7ce6e034272cda145e) |
| 36 | XMLGetValueByMultipleTagnames | First delimiter is &&, value to right is the output file that the value will be stored in.  All values to the left are the search criteria delimited by “--“.  First search string is always SEARCHFORWARD.  2nd search string can be forward or backward and you must indicate how many tags to traverse looking for whatever string value you want to look for.  3rd search string in below example searches forward and then once all search is done, then the value to the right between > and < will be placed into output file.  Example is  SEARCHFORWARD--E911 INSERT</osmc:ApsIntent>, SEARCHBACKWARD(5)--FILE\_Number##</osmc:TelNumber>, SEARCHFORWARD(3)--<osmc:ClientOrderNumber&&ClientOrderNumber1 | fw\_get\_value\_from\_xml\_based\_on\_multiple\_tagnames  Below is a video for demo/training on usage of webservices  [**https://webconference.twcable.com/p6y9fxo48m6/**](https://webconference.twcable.com/p6y9fxo48m6/?OWASP_CSRFTOKEN=84bfe7a34acaffc9874b16b6b0610360c0893926ef231c7ce6e034272cda145e) |
| 37 | IncrementValueByOne | {Variable File Name}  Example: AptNumber | fw\_increment\_value\_by\_one  This event will open the file, pull the integer value out of the file and increment it by one and store that incremented value back into the same file. |
| 38 | NA | NA | fw\_get\_value\_from\_file  The file name full path is passed into this function. |
| 39 | NA | NA | fw\_get\_workspace  This function gets the active workspace. |
| 40 | LoginToSSO | The user ID and Pass ID can be passed into the “TCTestData” column value as a comma delimited value “svc\_automation,Aut0m@tion” (is an example) | fw\_login\_to\_SSO – new function in GUI.java with 2 arguments, namely, user ID and pass ID  NOTE: if you call this function and SSO login is needed, then it will handle it. If you call this event and SSO login is needed it will bypass it and continue in the script. So this handles either condition. As you may experience with SSO, some machines require SSO login, some don’t. This event will handle either condition |
| 41 | Component  Logic to execute Component events in Main.java | The value should be the prefix of all of the ObjectNames. So for example, if you have 3 objects named….  SEARCH\_Address1  SEARCH\_ZipCode  SEARCH\_Submit  Then you can have a Component event with a TCTestData value of SEARCH and it will navigate to the Component worksheet and execute all test case events for that given test case which have a prefix of SEARCH. | Component – there is only 1 Component worksheet, and it’s case-sensitive. The format of this worksheet is identical to the TC prefixed worksheet. The intent of this worksheet is to house all of the common code used across many test cases to avoid redundancy in the TC worksheets. For example if you have a set of 10 events on the Customer page which have absolutely identical values for all 5 columns across all test cases, then you can put all of those 10 events into the Component worksheet and then inside each of the TC prefixed remove those 10 rows and replace them with 1 row referencing the Component. |
| 42 | ClickAndEnterData | Value to enter into textbox  For example “1234”.  If you want to pull value from variable file, then indicate “FILE\_{name of variable file}” to pull from. | fw\_click\_and\_enter\_data  This function was created to satisfy a need in T3 but can be used by any application. The problem with T3 fields was that you cannot enter data into a field until you click on it and the object property is different before you click on the field versus after you click on the field. Hence, current solution is to use 2 events, 1) click and then 2) enter data. Now the solution is to use only 1 event, namely, “ClickAndEnterData” which clicks on the field and enters the data into that field using this 1 event. |
| 43 | NA | NA | fw\_create\_environment\_variables– new function created - when execution first starts per test case, there is now a function call to this new function – which will create variable files per Environment column in the Environment worksheet in the Maps spreadsheet. NOTE: each variable created for Environments will be prefixed with “ENV”. This function is called from fw\_create\_output\_log\_file (which is one function call per test case executed). |
| 44 | SQLExecute | {columns to get comma delimited}--{sql query}  Example:  cntval1,cntval2--select count(\*) cntval1, count(\*) cntval2 from solo.tbl\_sodi\_log where account\_number=’<Account>’  NOTE: list of columns comma delimited to get back – each one is stored in a variable file. So in this example 2 variables files created cntval1 and cntval2.  NOTE: any value between “<” and “>” is a variable file referenced. So whatever value is in that file will be populated in sql query.  NOTE: sql queries are restricted to a return of 1 row of data but will populate as many variable files as columns of data you want returned. | fw\_execute\_sql (3 arguments)   * data\_source\_name * sql\_query * values\_to\_get   data\_source\_name is the column header prefix in the Environments worksheet. The Environments worksheet must have 3 columns, namely,  {data\_source\_name}DATASOURCE  {data\_source\_name}USERID  {data\_source\_name}PASSID  Example is:  SOLODATASOURCE  SOLOUSERID  SOLOPASSID  The data\_source\_name value comes from the TCObjectData input column value  DATABASE\_SOLO as highlighted in Yellow. Make sure the first part of the TCObjectData name is DATABASE\_{data\_source\_name}  The SOLODATASOURCE value in the Environments worksheet should look something like the following…..  {hostname}:1521/{dbname}  Example is:  ora-dev01.corp.chartercom.com:1521/SOLO02D.CORP.CHARTERCOM.COM  If you want to continuously execute a SQL statement until a certain condition is met, then do the following….  On the Object to Look for column….below is an example. The FILE\_SOLOSYNCPRIVACY is expected variable value. The SOLICITATION variable is the column/actual value. 5 is the number of loops and 5000 is the number of milliseconds to wait per loop  FILE\_SOLOSYNCPRIVACY--SOLICITATION--5—5000 |
| 45 | NavigateBack | NA | This event will navigate back to previous screen by selecting the Back button on the Browser. |
| 46 | GetDevice | Examples are F\_MTA\_2Port\_2 and F\_T6\_NA\_1. The fakerealflag value is either F for Fake or R for Real. The devicetype value is either MTA, T2, T3, T6. The porttype value is either 2Port, 4Port, 8Port, NA. The indexvalue is suffix on variable file | fw\_testdata\_get\_device  This event will get the next available device out of the tbl\_device table based on the criteria provided. The criteria can be specified in the TCTestData column in following format {fakerealflag}\_{devicetype}\_{porttype}\_{indexvalue}. Examples are F\_MTA\_2Port\_2 and F\_T6\_NA\_1. The fakerealflag value is either F for Fake or R for Real. The devicetype value is either MTA, T2, T3, T6. The porttype value is either 2Port, 4Port, 8Port, NA. The indexvalue is suffix on variable file. If MTA, then the following 2 variable output files will be created devicecmmac{index\_value} and devicemtamac{index\_value}. So if indexvalue of 2 is passed, then devicecmmac2 and devicemtamac2 will be created having the corresponding values.  Video files example…..  devicevideoecmmac1  devicevideoestbmac1  devicevideoserialnumber1 |
| 47 | GetTN | Examples are HOSTED\_METASWITCH\_RESI\_1 | fw\_testdata\_get\_phone\_number  This event will get a phone number out of the Phone table. Provide a TCTestData value of {PhoneType}\_{SwitchType}\_{AccountType}\_{index}. An example is HOSTED\_METASWITCH\_RESI\_1. The output file created will be tnvalue1 in this case.  6 columns are required in the Environment worksheet. 6 column names are….  RateCenterCPRESI  RateCenterMETASWITCHRESI  RateCenterNORTELRESI  RateCenterCPCOMM  RateCenterMETASWITCHCOMM  RateCenterNORTELCOMM |
| 48 | GetSequence | Examplese are AccountNumber\_1 | fw\_testdata\_get\_sequence  This event will get a sequence out of the Sequence table. Provide a TCTestData value of {SequenceName}\_{index}. An example is AccountNumber\_1. The output file created will be sequencevalue1 in this case. |
| 49 | GetSPALocation | The TCTestData value should be in the following format {Env}--{SwitchType}--{RateCenter}--{IndexValue}. Example is "QA--METASWITCH--FORTWORTH--2" or "FILE\_ENV--FILE\_SwitchType--FILE\_RateCenter--4". | fw\_testdata\_get\_spalocation  This event will get the SPA and Location details based on 3 input criteria namely, Environment, SwitchType and Rate Center. 8 output variable files are generated prefixed with "out\_spalocation...." including suffix of index value specified. |
| 50 | GenerateRandomCharacters | The TCTestData value should be in following format {Candidate of Characters}\_{length of random string you want}. Example is "ABCDEFGHIJKLMN\_17" or "ABCDEF12345\_22". | fw\_generate\_random\_characters  This event will generate random characters based on candidate of characters list you provide and the length of the randomized string you want generated. The output variable file generated is generate\_random\_character. |
| 51 | GetCurrentURL | No TCTestData value is required. The output variable file created is currenturl. | Get the current URL (event name is GetCurrentURL, function name is fw\_get\_current\_url) |
| 52 | ConvertStringCase | TCTestData input required in the following format “FILE\_nameofvariablefile,LOWER” or “FILE\_nameofvariablefile,UPPER”. The output variable file created is “outconvertedstringcase”. | Convert a String to upper or lower case (event name is ConvertStringCase, function name is fw\_convert\_string\_case). |

# Java Development Standards

The Java development standards are as follows.

1. One GitHub project repository will equate to one corresponding Maven project in Eclipse.
2. All source code will live under the structure {Java Project Name}\src\main\java

Example is Gateway\src\main\java

1. The package name will be named as follows: com.chtr.tmoauto.{Java Project Name}
2. All MainEngine classes should be named as follows: {Application}Tests.java. The class name should be {Application}Tests.java. An example is TPSITests.java. NOTE: there will only be one function in this Class, namely, main.
3. If any application requires any custom functions to be built, then those custom functions will live in the following class {Application}Functions.java. An example is TPSIFunctions.java
4. The main function should call the following functions before test case looping…

fw\_get\_user\_name

fw\_get\_test\_execparms

fw\_get\_environment\_to\_execute\_tests

1. The main function will loop through that list of tests for execution.
2. The fw\_create\_output\_log\_file will be called inside this loop. Thus, one log file will be generated per test case.
3. The log files will be located at: {Java Project Name}\logs directory. A log file will also be generated at X:\Test Utility\Log Files\SELENIUM OUTPUT LOG FILE – TESTID497 – 20170823142650.txt
4. Dynamic variables will be placed in the {Java Project Name}\variables directory. Ensure this variables directory is created.
5. Webservices files will be located at:

{Java Project Name}\webservices\templates

{Java Project Name}\webservices\runtime\requests

{Java Project Name}\webservices\runtime\responses

1. The number of test case steps to execute will be determined and a loop will be established inside the test case loop and the fw\_event function will be called/executed per test case step.
2. Last function call for each test case is fw\_closedown\_test.
3. All Framework functions are prefixed with “fw\_”. An example is “fw\_enter\_data\_into\_textbox”. No framework functions will be created in {Application}Functions.java
4. All application related custom functions must be prefixed with application name. Examples are “Gateway\_Login” and “Gateway\_Search”. Make sure you do not have some functions prefixed with Gateway and some functions prefixed with GW. Make consistent across all functions.
5. All functions should have standard comment block headers. Example is…..

/\*\*

\* This function is will log you into the ALM application using URL, userID and passID inputs.

\* @param: URL

\* @param: userID

\* @param: passID

\* @since: 11/16/2016

\* @author: Mark Elking

\*/

public void ALM\_Login (URL, userID, passID)

1. Standard Commenting inside function/code

It is encouraged to comment some of the code in order for others maintaining to have some insight into what the code is doing without having to read all the lines of code in order to interpret what the function code is doing.

Comment blocks start with /\* and end with \*/

Single line of comment start with //

1. All business functions should call the fw\_writeLogEntry (log\_message, return\_code)
2. All code should be properly indented for readability

# Execution

The application/project setup will include GitHub, directory/folder structure, configuration map spreadsheet and Main function setup.

## Eclipse

One spreadsheet needs to be created for managing test case data input and test configuration and object

## Jenkins

In email at location referenced below is a link to a video recording of a meeting held on 3/16/2017 on how to configure Jenkins job for executing our Java Selenium Automation Framework tests…….X:\Automation\Framework\Java Development Standards\How to Setup Jenkins Job Tutorial

If you create a Jenkins job for Regression, it must have the word “Regression” in the Jenkins job name.

If you create a Jenkins job for Smoke, it must have the word “Regression” in the Jenkins job name.

Configure the smoke tests to be executed via Jenkins using the “JenkinsSmoke” column on the ExecParms worksheet in the {Application}ConfigurationMap.xlsm. NOTE: populate Y on any of the TC prefixed sheetname rows you want for execution configuration.

Configure the regression tests to be executed via Jenkins using the “JenkinsRegression” column on the ExecParms worksheet in the {Application}ConfigurationMap.xlsm. NOTE: populate Y on any of the TC prefixed sheetname rows you want for execution configuration.

Set Jenkins job checkbox “Execute concurrent builds if necessary” in order to execute multiple instances/executions of same Jenkins job simultaneously so multiple jobs can run in parallel and not queue

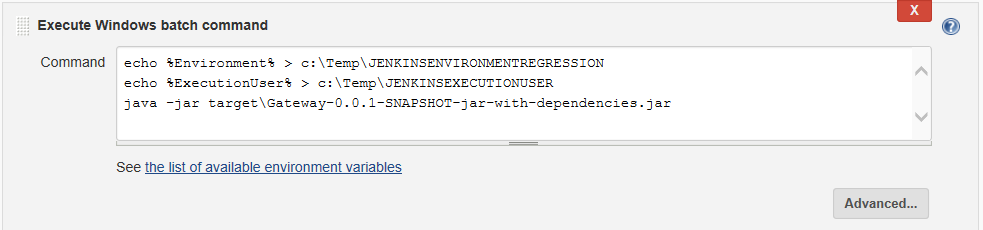
For configuring a specific workstation to be used in a Jenkins job, see the following video for how to set that up.

[**https://webconference.twcable.com/p6nzds27wrh/**](https://webconference.twcable.com/p6nzds27wrh/?OWASP_CSRFTOKEN=05c2b97a27df865a2f3c8d6fb7d77e31aa7b90577ffb7da1f5d3e9bf8f2b4ea1)

**User to Select for Execution via Jenkins**

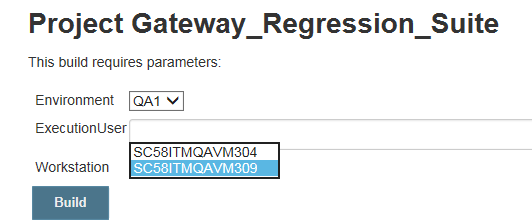
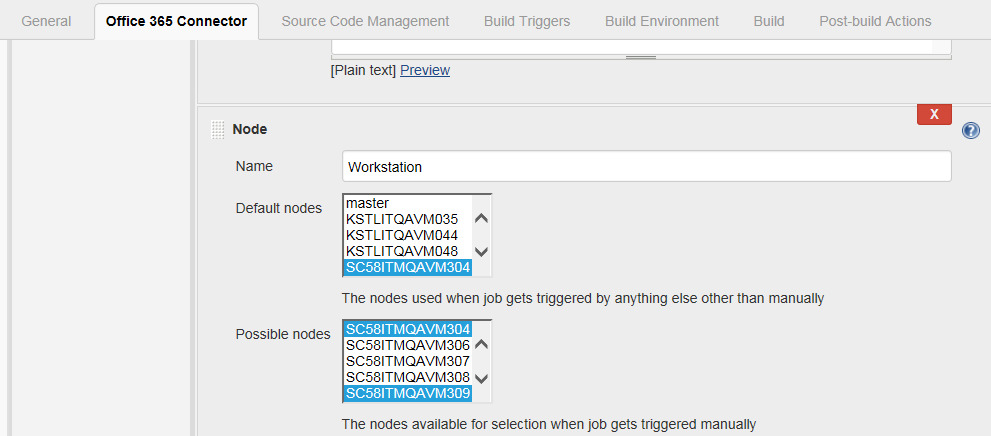
* The following enhancement allows developers, manual testers to dynamically select which user they want to use for execution via runtime in Jenkins.
* modified fw\_get\_user\_name to get Execution User from Jenkins job via a String Parameter and define almcolumnheader value
* modified fw\_get\_alm\_config\_parms to use the almcolumnheader value as defined in fw\_get\_user\_name function
* There are now 10 new users (5 for smoke, 5 for regression).  The execution user names are JS1, JS2, JS3, JS4, JS5, JR1, JR2, JR3, JR4, JR5 where JS = Jenkins Smoke and JR = Jenkins Regression.   NOTE: the JENKINSREGRESSION user and the JENKINSSMOKE user will NO longer be used.
* Going forward, please use Test Utility v7.xlsm.  Discontinue using Test Utility v6.xlsm.
* You will need to update the ALM configuration for each application by adding 10 columns JS1, JS2…..etc and corresponding ALM configuration required.  NOTE: these values are case sensitive and MUST be upper case ONLY.  NOTE: the JenkinsSmoke ALM column and JenkinsRegression ALM configuration columns on the ALM worksheet will need to be removed.  The Value column needs to remain as is.
* If you are running via Jenkins, then you will need to enter your execution user JS1, JS2, or you can use your developer name in Jenkins as well.  NOTE: you can enter this values in lower case if you want (it will make them upper case during execution).
* NOTE: when you are unit testing a test case via Jenkins, make sure to use your user name (first name) and NOT the JSx or JRx users.  The JSx and JRx user configuration should be relatively static/fixed for running automated regression/smoke suites.
* If you are running via Eclipse as a developer, you will continue to be prompted for your username (first name) – nothing has changed here.
* Prior to running via Jenkins (Smoke and/or Regression jobs), you will need to make the following Configuration updates one time only.

1. Add a String Parameter named “ExecutionUser” in the Office 365 Connector section.  Again, the value you will enter into this Textbox field during runtime will be JS1, JS2….JR1, JR2…..or your firstname if you are running a Jenkins unit test.
2. In the Build section you will need to add the command echo %ExecutionUser% > c:\Temp\JENKINSEXECUTIONUSER above the JAR file execution statement (as shown below).
3. When you execute the Jenkins job, you will select “Build with Parameters” option, select the environment value AND now enter the execution user, click OK and the Jenkins job executes normally.



Jenkins Slave Selection During Runtime

* The following enhancement now allows persons executing Jenkins jobs to dynamically select during runtime the Jenkins slave they want the job to run on.
* Add a Node parameter in your respective Jenkins job as shown below (screenshot on left). Give the Name value “Workstation”.
* In the Possible nodes section, only select the workstations which you want to appear during runtime for your Jenkins job. So if you only want workstations “SC58ITMQAVM304” and “SC58ITMQAVM309” to appear during runtime then only select those 2 in the possible nodes section.
* During runtime, you will see the following as an example shown below (screenshot on right).
* X:\Automation\Framework\Java Development Standards\Java Selenium Automation Framework ver2.doc has been updated with this information.



# Framework

The Java Selenium automation framework consists of several parts.

## Technical Architecture

Below is a diagram of the technical architecture for the Java Selenium solution.



1. The Automation Developer on the Developer workstations uses the Test Utility to create test cases, components, objects which are stored in the Test Database
2. The Automation Developer uses Eclipse to build any custom Application Java code if needed and to unit test and debug and to checkin and checkout code to GitHub Apps repositories.
3. The Architect uses Eclipse to create and maintain Framework Java code and to checkin and checkout code to GitHub Framework repository.
4. The Architect users Maven to build the Framework JAR file and deploy it to the Artifactory Server.
5. The Executioner creates, configures, maintains and kicks off Jenkins jobs on the Jenkins Server to execute the test suites.
6. The Jenkins job instructs the Jenkins slave to pull down the Framework JAR file from the Artifactory Server using Maven.
7. The Jenkins job instructs the Jenkins slave to pull down and build the Application JAR file from the GitHub Apps repository for the application configured.
8. The Jenkins job then executes that Application JAR file for the configured Application using the JRE on the Jenkins slave machine for the test cases configured in the Execution Parms per application as defined in the Test Database.
9. During execution, the HP ALM database is updated with the test case status.
10. Test case reporting is performed using HP ALM reporting or the Test Utility reports.

## Code Flow

The chronological flow of the framework Java code is as follows.

1. fw\_get\_user\_name (gets appname, fname, testtype, almuseridlogin variable files)
2. fw\_get\_test\_execparms (gets test cases to execute list)
3. fw\_get\_environment\_to\_execute\_tests (get environment to run tests and put into ENVSELECTED variable file)
4. for x = 1 to number of test cases
   1. fw\_create\_output\_log\_file (generates output log file per test case)
      1. fw\_set\_global\_parms (sets global parm variable files)
      2. fw\_get\_alm\_config\_parms (sets alm parm variable files)
      3. fw\_create\_environment\_variables (sets environment parm variable files based on application and environment selected)
      4. fw\_update\_hpalm\_test\_case\_execution\_status (updates alm with a Run instance with status marked Not Completed)
         1. fw\_write\_logging\_datebase
   2. fw\_get\_test\_case (get test case steps and output to 5 variable files)
   3. fw\_get\_test\_component (get test components and output to 5 variable files)
   4. fw\_get\_test\_object (get test objects and output to 3 variable files)
   5. for y = 1 to number of test steps
      1. fw\_event (executes all of the events in the test case)
   6. next y
   7. fw\_closedown\_test
      1. fw\_update\_hpalm\_test\_case\_execution\_status (updates alm with a Run instance with status marked Passed or Failed)
         1. fw\_write\_logging\_datebase

next x

## Test Utility

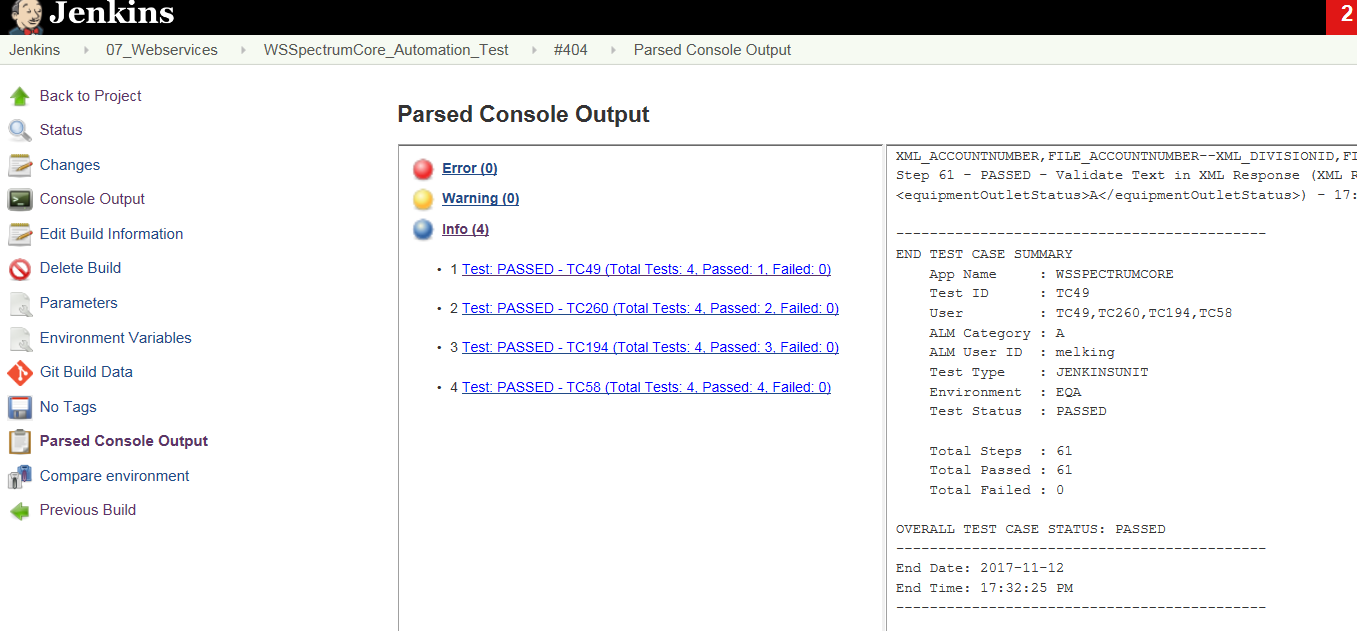
The Test Utility is the tool used to build test objects, test scripts and test components.

## Troubleshooting & Reporting – Log Parser

* modified fw\_closedown\_test with additional logging statement to be use by Jenkins log parser

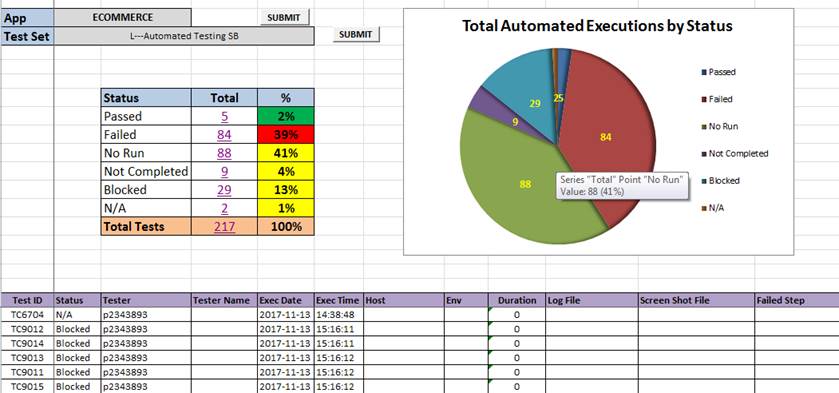
Example output: Test: PASSED - TC49 (Total Tests: 4, Passed: 1, Failed: 0)

* Console output (build log) parsing post build action has been added to all of the Jenkins jobs (for T3, Gateway, Ecommerce, NextGen, Webservices, BPS).
* After each Jenkins job, the following parsing rules will be applied to the console output log file.
  + error / - FAILED - /
  + warn /Warning/
  + warn /Error/
  + info /Test: /
* In Jenkins after job execution, if you click on the Job Build Number and then click on the Parsed Console Output hyperlink, you will get the following view.
* The Info link will display the status for all of the tests executed in the job and give totals for each of the tests executed thus far.  Additionally, if a test case step has failed, clicking on the Error hyperlink will take you to the spot in the log file where this specific failure occurred.



## Troubleshooting & Reporting – ALM Test Set Metrics

* A new reporting view is available to help troubleshoot/analyze failures and view test case execution summary metrics per Test Set
* On the ALMTestSetMetrics worksheet you will see the following view



* Select dropdown value for App and click the Submit button adjacent to the App dropdown.  This will retrieve all of the Test Sets configured for this Application you have selected (which is on HPALM worksheet).
* Select dropdown value for Test Set which you want to pull metrics for and click the Submit button adjacent to the Test Set dropdown.  This will retrieve all of the ALM test set metrics for that selected Test Set.  NOTE: the graph will be updated based on the tabular test summary numbers to its left including Status & Total.  The lower section will list out the Total Tests within that Test Set.  In this example there are 217 tests in Test Set “L—Automated Testing SB”.  If you click on the Total hyperlink per status (i.e. click the Passed total hyperlink, which is a 5, then down below it will list only the 5 tests which have Passed.  Similarly, if you only want to see the Failed tests, then click on the Failed total hyperlink in this case, it’s 84, then down below it will only list the 84 tests have Failed.  NOTE: the last 3 columns can be used to help identify specifically what has failed.
  + Log File – this is a hyperlink if you click it will open the log file for the test case.  There is one log file per test case.
  + Screen Shot File – this is a hyperlink if you click it will open the corresponding screen shot taken (if failure generated via GUI step).
  + Failed Step – this is a failed step description of what step the test case failed on and any specific test data information that might be helpful in understanding what happened and why this step failed.