1. Purpose and Intended Usage

PAF is an in-house automation framework built on using the Selenium API. Selenium is an open-source automation suite that supports web application automation. PAF would utilize the Selenium Locators, Commands and Features to automate web-based as well as windows-based and mobile application scenarios along with databases, APIs, Excel, CSV, PDF automation. PAF would be implemented to support test scenario automation through XML and XPATH, error logging, reporting of test steps results by integrating with HP ALM, SynapseRT, TestRail.

Directly using Selenium Automation suites need technical skills like expertise on JAVA programming. PAF eliminates the need of knowing any software language like JAVA to automate the web application scenarios. Through PAF, with knowledge on XML and XPATH, functional manual tester would be able to automate the scenarios.

The objectives of PAF is as below:

1. To automate web applications, windows-based applications, mobile applications, databases, APIs, Excel, CSV, PDF etc. by utilizing the functional manual tester.
2. To automate RPA processes.
3. To perform UI Validations.
4. Eliminates the need of knowing programming language.
5. To provide user friendly framework to automate scenarios.

2. Pre-requisites

Install JDK 1.8 or higher version if not already installed on your local machine to run automated scenario through PAF.

Through CMD (command prompt), type “java –version” to know the JDK version installed on your machine.

Good to have software - **Notepad++**. Please install from below shared location:

\\inblr-vsfs01\QEDTC\PAF_Tool\Prerequisites_Softwares

3. XML & XPATH Training

To start automation using PAF, users should be trained on XML & XPATH or at least should have basic understanding on XML & XPATH.

Reference for XPATH:

Shared Location - **\\inblr-vsfs01\QEDTC\PAF_Tool\Training Material**
https://msdn.microsoft.com/enus/library/ms256471(v=vs.110).aspx
https://www.w3schools.com/xml/xpath_intro.asp

http://www.protechskills.com/testing/automation-testing/selenium/usage-contains-starts-functions-xpath
http://automationtricks.blogspot.in/2010/09/how-to-use-functions-in-xpath-in.html

How XPATH works:

- XPath uniquely identifies an element on a web page.
- XPath can be used to navigate through elements and attributes in an XML document.
- XPath uses "path like" syntax to identify and navigate nodes in an XML document.
- XPath contains over 200 built-in functions.

Reference for XML:

https://www.w3schools.com/xml/default.asp

https://www.ibm.com/developerworks/library/x-newxml/

Lynda

4. Installation of PAF build

Installation process is updated. Please click here for the steps to install PAF using PAF Installer.

Copy the latest Pace Automation Framework (PAF) build from the following location \\inblr-vsfs01\QEDTC\PAF_Tool to your local machine and Unzip/Extract the folder.

The build folder contains the jar files, library files & drivers. It has **report** folder which will capture the screenshots of execution along with **raw_report** text file which will

capture the logs of script execution. It also has **Project_Test Data_Excel Files** folder which contains excel sheet with test data required for data driven execution.

5. Flow of Execution:

Start.bat → init.properties → flow.xml → activity xml.

The **Project_Automation_Test Library** folder will have the test scripts with flow of execution defined in the flow xml.

6. Init.properties file –

Start.url should be URL of the application under test.

flow.xml.path should be the relative path of the flow.xml (test suite). Do not give absolute path (C:/PAF/build/sample_xml/flow.xml) instead use relative path (./sample_xml/flow.xml).

browser = Chrome, the web application under test will be opened in Chrome browser. (Browser can be **InternetExplorer: FireFox: Chrome: Edge**)

wait.element=true, it will wait until the web element in web page is loaded.

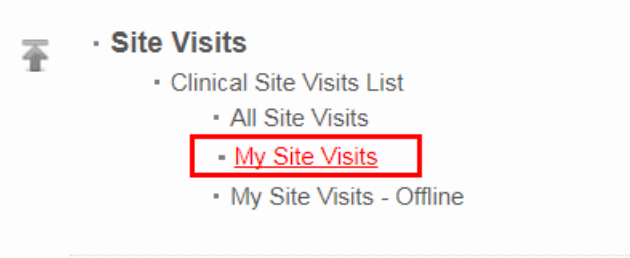
browser.driver.path = ../PAF/drivers/chromedriver.exe (Relative Path). Do not give absolute path (C:\Users\q763607\Desktop\Pace Automation Framework\build\drivers\cromedriver_win32\cromedriver.exe).
Chrome Driver path = ../PAF/drivers/chromedriver.exe
Internet Explorer Driver path = ../PAF/drivers/IEDriverServer.exe
Firefox Driver path = ../PAF/drivers/geckodriver.exe
Edge Driver path = ../PAF/drivers/MicrosoftWebDriver.exe

NOTE: The path should be relevant to browser selected for execution.
loop.flow is the number of times you want the flow to execute.

flow.ids = <if multiple flow IDs then should be separated by commas (,)>. Each flow will be executed once if loop.flow = 1. flow ids will be mentioned in the INIT file once user has completed/defined all the flows in flow xml.

wait.time.out= wait time in millisecond until the web element is loaded or next page is loaded. If the web element is not loaded within the wait time, then page error or timeout may occur.

highlight.color=red. Highlight in RED the web elements on which actions are performed on the web page. See example screenshot below:



raw.report.path= relative path where logs will be captured. There is a **report** folder where logs will be captured in **raw_report** text file.
snapshot.dir= relative path where screenshots of execution will be captured. There is a **report** folder where screenshots will be captured.

bot.only= bot.only property can be set to true in case you don't want to launch the browser, For Example, the database related test cases do not require browser launch.

alm.integration= alm.integration can be set to true in case you want to execute the test script in ALM, If set to false will execute the test case in local system.
Click on the batch file (Start.bat) to start the test script execution. It is preferred to use command prompt to start test execution instead of directly clicking on the batch file.

How to execute from command prompt:

1. Open command prompt. (Click "Start" → Type "cmd" → press "Enter")
2. Give path of the folder where start.bat (batch file) is located.
   Example: cd C:/pace automation framework/build/start.bat. Press Enter

7. Flow xml –

- Flow is defined as the set of activity that completes one test objective.
- Each Flow will have its unique id. This would help when flow execution sequence would be defined in INIT file.
- To design the Flow, CALL command would be used.
- Calling flow within flow file is used for creating test suites.
- Only one flow file is instructed in INIT file, this means execution would be restricted to flow ids of that referred file.

```
<flows>
   <flow id="test flow" name="Test" desc="Test flow" propertyFile=""
excelPath="./data/test1.xlsx" sheetName="Sheet2">
         <call activity="login" xml="./sample_xml/activity.xml"></call>
   </flow>
</flows>
```

flow id = <it should be same as mentioned in the init.properties>

name = any meaningful test case name.

desc = description

excelPath= relative path of the excel sheet from where the data will be picked for this flow.

sheetName= In the excel sheet→worksheet (**example – Sheet1 or Sheet2**) from where the data will be picked for this flow.

<call activity="login" xml="./sample_xml/activity.xml"/>

call is a PAF command which is calling login & in xml = we are giving the relative path of the activity xml where actual login activity is defined.

8. Activity xml –

- Activity is defined as the set of action that achieves specific task.
- Each activity will have its unique id. This is needed when activity sequence is arranged in FLOW file.
- To design the activity, XML, PAF Tags and XPATH knowledge is required.
- While defining the activity, you can use validate tag to ensure required checks.

  Example: After login, you want to validate Home Page is displayed.
- It is advised to keep setup of activities define in one xml file per page.

```
<activities>
	<activity id="login">
		<input name="Username" value="$excel{Username}"></input>
		<input name="Password" value="$excel{pwd}" />
		<click xpath="//a[$excel{variable2}(@onclick, 'Login')]"></click>
		<wait time="30000"></wait>
	</activity>
</activities>
```

Above is the activity xml where <activities> is the root element. activity id = “login”, it should be the same name which is called in the flow xml.

Elements in web page can be identified using ID, name or XPATH. ID will be used only if it is static. If the ID is changing every time the page is loaded/refreshed, then that is dynamic ID & should not be used. **Open the application in chrome, do function (Fn) + F12 - select an element in a web page to inspect it. By this user can identify the web element by ID, name or XPATH.**

input is a selenium function, identifying the username text box (web element) by name (here name = “SWEUserName”). Value can be passed directly like value= “TCTMSUser121” or through excel sheet like value = “$excel{userName}”. userName is the column name in excel sheet (test1.xls).

Identifying the password entry web element by name (here name = “SWEPassword”). Value can be passed directly like value= “Qu1nt1le5” or through excel sheet like value = “$excel{pwd}”. pwd is the column name in excel sheet (test1.xls).

Now, **click** on the login web element using an XPATH (xpath="//a[$excel{variable2}(@onclick, 'Login')]”). click is a selenium function.

Wait is a selenium function where we are defining the wait time in milliseconds (here wait time= “30000”) until the next page or web element is loaded.

9. Description of PAF Tags/Controls

Input:

Syntax - `<input id="" name="" css="" xpath="" snapshotBefore="true/false" snapshotAfter="true/false" value=""></input>`

This control is used to enter or input the text like input username or password on the login screen. The text box (web element) on a web page can be identified using an unique id, name, css or xpath. snapshotBefore & snapshotAfter – whether you want to take screenshot before or after performing an action on web element (true will take screenshot, false will not). Value = “”, this value can be passed directly or through excel sheet.

NOTE: Only one identifier is required for web element - id or name or css or xpath, others can be removed.

Click:

Syntax - `<click id="" name="" css="" xpath="" snapshotBefore="true/false" snapshotAfter="true/false"></click>`

This control is used to click on web element. The web element on a web page can be identified using an unique id, name, css or xpath.

RightClick:

Syntax - `<rightClick id="" name="" className="" css="" xpath="" snapshotBefore="true/false" snapshotAfter="true/false"></rightClick>`

This control is used to right click on web element. The web element on a web page can be identified using a unique id, name, class name, css or xpath.

DoubleClick:

Syntax - `<dblClick id="" name="" className="" css="" xpath="" snapshotBefore="true/false" snapshotAfter="true/false"></dblClick>`

This control is used to double click on web element. The web element on a web page can be identified using a unique id, name, class name, css or xpath.

Wait:

Syntax - `<wait id="" name="" css="" xpath="" time="in milliseconds"></wait>`

This control will make the script wait for certain time (as defined by the user - you can set the wait time in milliseconds). Wait is required when navigating from one web page to another or if there is a delay in loading of certain elements on the web page.

Highlight:

Syntax - `<highlight id="" name="" css="" xpath="" snapshotBefore="true/false" snapshotAfter="true/false"></highlight>`

This control will highlight the web element for the user. The highlight on a web page can be done by using an unique id, name, css or xpath of an element.

Hover:

Syntax - `<hover id="" name="" css="" xpath="" snapshotBefore="true/false" snapshotAfter="true/false"></hover>`

You can hover on an element using hover control. The hover on a web page can be done by using an unique id, name, css or xpath of an element.

Script:

Syntax -

```
<script id="" name="" css="" xpath="" snapshotBefore="true/false" snapshotAfter="true/false" clickElement="true" src="JavaScript file path">
JavaScript
</script>
```

script is an in-house function which will be used for complex scenarios (like identifying dynamic element) in order to accomplish some task that could not be done by tags. The web element can be identified using an unique id, name, css or xpath. If we are clicking on the web element to take an action then **clickElement="true"**. We can also write our own java script under script tag. **src** is the file path if we want to execute the script from certain path.

Redirect:

Syntax - `<redirect url=""></redirect>`

This control will redirect to the web page as mentioned in the url. **Example**: <redirect url= "google.com"> - using this you can redirect to google.com during script execution.

Scroll:

Syntax - `<scroll id="" name="" css="" xpath="" snapshotBefore="true/false" snapshotAfter="true/false"></scroll>`
This control will scroll the web page horizontally or vertically to identify the element which is not visible on the web page. scroll can be done through unique id, name, css or xpath.

Dropdown:

Syntax - `<dropdown id="" name="" css="" xpath="" snapshotBefore="true/false" snapshotAfter="true/false" selected=""></dropdown>`
This control will identify the dropdown list web element. A dropdown can be identified using an unique id, name, css or xpath. Selected = will be the option which should be selected from the dropdown list.

Variable:

Syntax - `<variable id="" name="" css="" xpath="" keyName="" value="" decrypt="" type="text" webElementAttr="" trim="value_to_be_trimmed" expression="" remove="varName" rowNum=""></variable>`

- This control will pick the value of a variable from web page. A variable can be identified using a unique id, name, css or xpath.
- value = will store the value of the variable.
- type="text" will capture text value of a web element

- webElementAttr="attribute" will capture attribute value of a web element
- trim=”value to be trimmed” will trim the variable
- expression=”” lets the script execute any JavaScript expression and update value for variable.
- remove attribute in variable tag: This attribute can be used to remove the attribute value from variables. This should be used specifically to remove resultsetVar variables to free up memory.
- decrypt="encrypted_text" would be used to store encrypted text into a variable
- rowNum gives the number of web elements selected for the given rowXpath (rowNum should be used with rowXpath and not xpath)

valGroup:

- valGroup tag id used to define the validation conditions in the activity.
- valGroup tag would be defined outside the activity
- valGroup tag has a child tag “validate” to define the validation details.

Syntax -

```
<valGroup groupId="Validation1">
      <validate ------------------></validate>
</valGroup>
```

validation:

- valGroup tag would be called inside the activities using validation tag.

Syntax -

```
<validation valGroupIds="Validation1"></validation>
```

validate:

Syntax -

```
<valGroup groupId="">
   <validate id="" css="" xpath="" name="" exists="true/false"
   condition="contains/starts_with/ends_with/equals/not_equals" value=""
   passMsg="" failMsg="" custom="true/false" variable=""
   snapshot="true(default)/false" stopOnFailure="" resultsetVar1=""
   resultsetVar2="" evidence="file Path" file="path"</validate>
</valGroup>
```

Attributes -

- validate is an in-house function, it will check the web element exists on the web page or not. We can also mention different conditions for web element like **"web element contains/starts_with/ends_with/equals"** with value="<pass the value>".
- validate is a child tag for valGroup tag
- For validate tag, you need to give the following combinations at minimum
- id/css/xpath/name, exists
- id/css/xpath/name, condition, value, type=text (for checking labels)
- variable, condition, value
- resultsetVar1, resultsetVar2 (value should also contain a resultSetVar name. Condition is always equals as this would compare all the data in one table with another.)
- evidence can be used with any combination. However, when this attribute is used, screenshots are not uploaded.
- file, exists (to check whether a file exists on local drive)
- highlight="false" attribute added to validation tag to make highlighting optional.
- With validate tag snapshot is by default captured; Screenshot capture can be disabled by using snapshot="false" (*Note: useful for db validation*)

Note –

For validate tag, you need to give the following combinations at minimum

- id/css/xpath/name, exists

- id/css/xpath/name, condition, value, type=text (for checking labels)
- variable, condition, value
- resultsetVar1, resultsetVar2 (value should also contain a resultSetVar name. Condition is always equals as this would compare all the data in one table with another.)
- evidence can be used with any combination. However, when this attribute is used, screenshots are not uploaded.
- file, exists (to check whether a file exists on local drive)

Working on Multiple Frames –

If you want to work on multiple frames in the application, then please use the below code.

Syntax:

```
<frame id/name/className/css/xpath=""/> ‘mention the frame details to switch and
Write the XML tags for the control on frame
<frame id="parent"/> ‘use this to return to the main page/frame
```

Example:

```
<frame id="frame1"/>
<input xpath="//*[@id='PageTitle']" value ="testing" />
<click xpath="//input[@class = 'save']" />
<wait time="2000"/>
<click xpath="//a[@id = 'clickExit']"/>
<frame id="parent"/>
```

Working on Multiple Tabs/ Browser popup –

To work on multiple tabs in the application, then please use the below code.

Syntax:

```
<tab tabNo="tab" closeCurrentTab="true"></tab>
```

- “tab” is the tab number to switch
- Write the XML tags following the tab switch
- closeCurrentTab="true”: use this to close the current tab

Example:

```
<tab tabNo="1"/>
<input xpath="//*[@id='PageTitle']" value ="testing" />
<click xpath="//input[@class = 'save']" />
<wait time="2000"/>
<tab tabNo="1" closeCurrentTab="true"/>
```

Working with Multiple Flows:

Syntax - `<call flow="" excelPath="" sheetName=""></call>`

Multiple flows – calling another flow inside a flow.

Example:

```
<flows>

    <flow id="test1" name="Test1" excelPath="./data/test1.xlsx" sheetName="Sheet1">
            <call activity="activity1" xml="./sample_xml/activity_Multiflow.xml" />
            <call flow="test2" excelPath="./data/test2.xlsx" sheetName="MF" />
    </flow>

    <flow id="test2" name="Test2" excelPath="./data/test2.xlsx" sheetName="Sheet1">
            <call activity="activity2" xml="./sample_xml/activity_Multiflow.xml" />
    </flow>

</flows>
```

Robot:

- "robot" tag be used for keyboard and input operations on file upload/save window.
- During file upload or save there is a pop-up window for selecting the location to save or upload, xpath of such web elements cannot be found hence robot function is used.

Syntax –

```
<robot typeString="" command="" moveMouseToX="" moveMouseToY="" leftClick="" rightClick="" centreClick="" scroll=""></robot>
```

Attributes –

- typeString would be used to send the keyboard input or text inputs, keyboard inputs are usually written with curly brace, i.e.{enter}, {tab} etc.
- Escape character "\" needs to be used if "{" or "\" character needs to be written in typeString attribute of robot command.

  For Example: typeString="abc\{kshshg}\\"
- command would be used to send multiple keyboard inputs.
- moveMouseToX would be used to pass the x co-ordinate in the window
- moveMouseToY would be used to pass the y co-ordinate in the window
- leftClick is for left mouse click and should be used with x and y coordinates
- rightClick is for right mouse click and should be used with x and y coordinates
- centerClick is for center mouse click and should be used with x and y coordinates
- scroll is for mouse scroll and should be used with x and y coordinates

Example –

```
<robot typeString="C:\Rapid\build\data\fileupload.txt"/>
<robot typeString="{tab}"/>
<robot typeString="{tab}"/>
<wait time="2000" />
<robot command="{press}:{shift},{type}:{tab},{release}:{shift}"/>
<robot typeString="{enter}"/>
```

Alert:

This tag should be used in case of any pop-up on the web application which requires action by the user like accept/cancel or input from user to proceed, authenticate using username/password to proceed etc.

Syntax –

```
<alert type="accept/cancel/getText/input/authenticate" user="" pwd="" keyName="" value="" robotScreenshot="true" snapshotBefore="true/false" snapshotAfter="true/false"></alert>
```

Attributes –

- type would be used to instruct the tool how you want to handle the alert.
    - type="accept" if you want to click OK or Yes etc on the alert
    - type="cancel" if you want to click Cancel or No etc on the alert
    - type="getText" if you want to capture text value from the alert
    - type="input" if you want to enter input to the alert
    - type="authenticate" if you want to pass username or password for the alert
- user would be used to pass the username along with type="authenticate"
- pwd would be used to pass the password along with type="authenticate"
- keyName would be used to declare variable name along with type="getText"
- value would be used to enter any input along with type="input"
- robotScreenshot="true" can be used along with either of snapshotBefore or snapshotAfter attributes as applicable to capture screenshot of the alert.

Note-

Please make sure not to use snapshotBefore or snapshotAfter attributes with any tag when the alert is open, any tag preceding the alert tag.