

**eApp Automation POC**



BRIGHTHOUSE

Intelligent e-App

Version 1.0

# Revision History

| Version | Date | Revision Description | Revised by |
| --- | --- | --- | --- |
| 1.0 | 08/20/2019 | Initial Draft | Soumendra Prasad Sahoo |
|  |  |  |  |
|  |  |  |  |

# Table of Contents

[1. Objective 4](#_Toc17225642)

[2. Tools/Software Used 4](#_Toc17225643)

[3. Framework Selection 5](#_Toc17225644)

[4. POC Result/Effort saving 6](#_Toc17225645)

[5. Areas that cannot be Automated 6](#_Toc17225646)

[6. Future Plan 7](#_Toc17225647)

# Objective

The objective of this program is to create a POC (Proof of Concept) for Automating the Smoke Testing of BHF eApp Application. The POC will include automating the FRA Product for smoke testing. As part of automation, a new FRA policy will be created and submitted to nbA. During the data entry process, in each page, some client and server-side error message validation will be performed.

# Tools/Software Used

Selenium with Java will be used to create the POC.

Other Software used:

* Apache POI – For reading/writing data from/to excel.
* Extent Report API – For test report generation.

Platform/OS – Windows 10.

**Advantages of Selenium:**

* **Selenium is an Open Source Software.**

All Selenium Projects (Selenium IDE, Selenium RC, Selenium WebDriver and Selenium Grid) released under the Apache license are free to use.

* **Selenium supports various programming languages to write programs (Test scripts).**

Java, C#, Perl, Python, Ruby and PHP. We will be using Java for eApp Automation.

* **Selenium supports various Browsers (Mozilla Firefox, Google Chrome, IE, Opera, Safari etc.)**

Once we create Test cases then execute the Test Cases using all Popular Browsers without any changes in Test Cases. Browser driver only varies from one Browser to another Browser, but Test cases are same.

Note: Selenium IDE (one of the tools in Selenium’s Tool suite) supports Mozilla Firefox Browser only. Since, eApp is certified in Chrome and IE only, we can take help of IDE to create the Automation Suite.

**Disadvantages of Selenium:**

* **Technical Support is not robust**

Since it’s Open Source software, no dedicated support is available. Only user and professional community support is available.

* **Complex in comparison to record and run tools like UFT**

Selenium WebDriver (Powerful tool in Selenium’s Tool suite) has Programming interface only, no IDE, so we must write steps for each task. They take more time and require technical knowledge.

* **Limited support for Image Testing.**
* **New features cannot be injected right away**

Since it is open source software nobody is responsible for new features usage, they may or may not work properly. Proper verification or compatibility checking is required before using the new feature.

* **No Built-in Reporting facility.**

Selenium WebDriver doesn’t have built in Result Report facility, we need to take help from either JUnit or TestNG Testing Framework to generate Test Reports or we can create our own test report using open source APIs like Extent Reports.

# Framework Selection

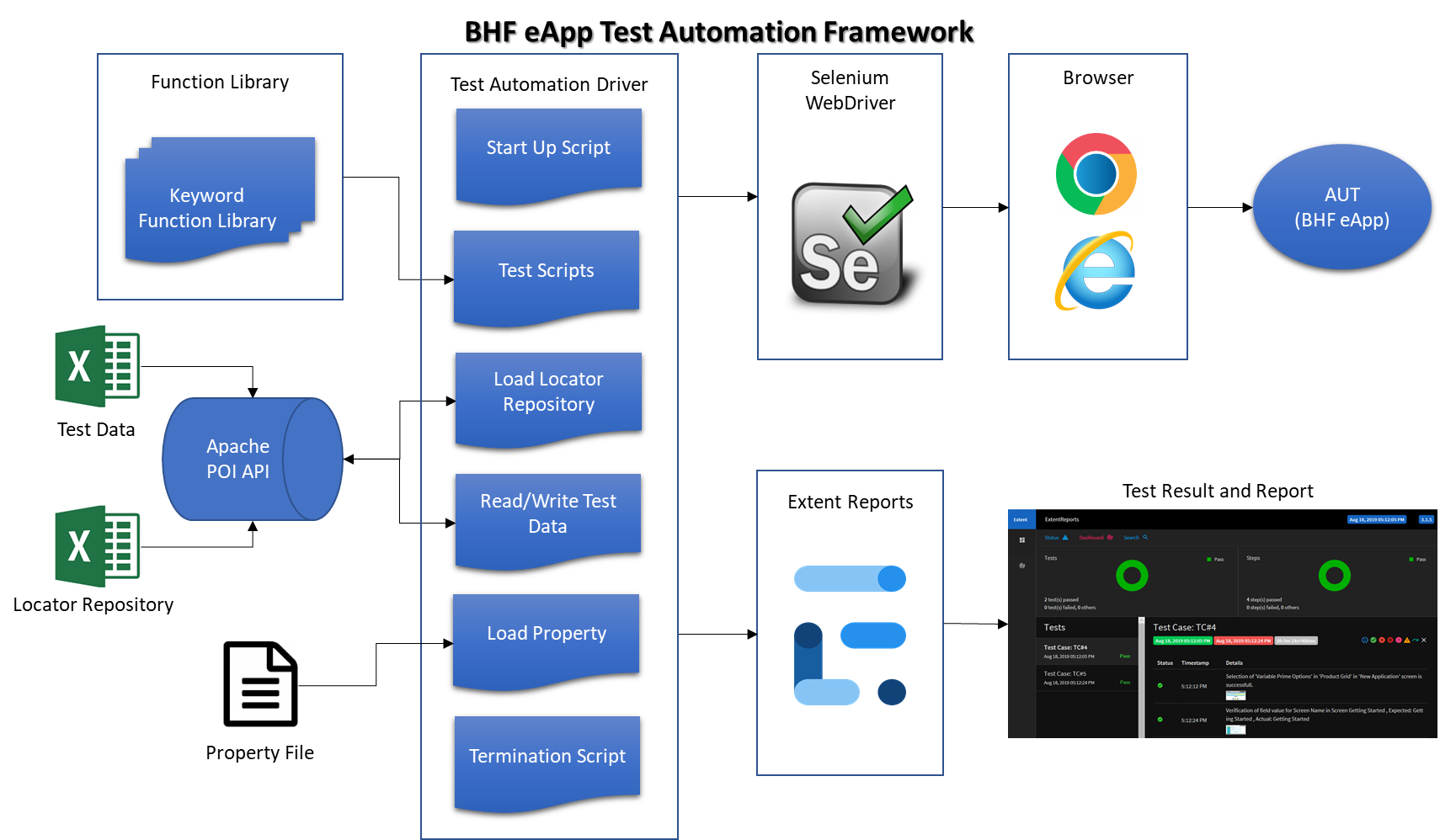
Hybrid Framework will be used to create the Test Automation for eApp. It will be a combination of Keyword, Page Object and Data Driven Framework. The keywords will be used to drive the test steps in each test case. All page element locators will be stored in external library and can be modified/updated at any time without modifying the code. Test data will also be stored in external excel file and can be updated/added/removed without any code changes.

Eg. of Sample Keyword:

SetValue – To put a value in a web field.

VerifyClientMessage – To verify client-side error message for a field.

VerifyPolicyStatus – Verify the policy status in In-Progress or Submitted Index.



**Advantages of this framework:**

* Easy to Maintain – We can easily modify the script or add a new keyword function without affecting rest of the functionality.
* Reusable – Same function and keyword can be used across test steps and cases.
* Complex technical knowledge is not required in creating a new case.
* New page elements can be added without/with limited technical knowledge.
* Can be used to test across Chrome and IE browser without any code modification.

(Note – Some settings are required for testing in IE browser. Since these settings are disabled by administrator, hence it’s not tested on IE. This POC is done in Chrome browser only.)

**Disadvantages of this framework:**

* Parallel Testing cannot be achieved with this framework. Test cases will run sequentially one after other.
* Technical knowledge (Java + Selenium) is required to maintain the framework.
* Since its built on open source tools, through investigation is needed in injecting new JARs to the framework.

# POC Result/Effort saving

A simple FRA Product has been automated for smoke testing.

Total time taken for running the case through automation: 3-4 Mins

Time taken for running the same case manually: 20-25 Mins

Effort Saving Chart:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No. of Products | 1 | 3 | 5 | 7 |
| Manual Effort for Policy Submission (20-25 Mins/Product) (a) | 25 | 75 | 125 | 175 |
| Automation Effort (3-4 Mins/Product) (b) | 4 | 12 | 20 | 28 |
| Time taken for case Acknowledgement (c) | 15 | 15 | 15 | 15 |
| Total Manual Execution Time (a+c) | 40 | 90 | 140 | 190 |
| Total Automation Execution Time (b+c) | 19 | 27 | 35 | 43 |
| **% of Effort saving** | **53%** | **70%** | **75%** | **77%** |

Note – All efforts above are in minutes.

Avg No of Builds Per Month = 5

Total Smoke Testing Effort Per Month (Considering 7 plans) = 190 Mins \* 5, Approx 16 Hrs i.e 2 PD

Smoke Testing through Automation = 43 Mins \* 5, Approx 3.5 Hrs i.e 0.4 PD

1.6 PD/Month effort can be saved, if all 7 Products will be automated for BHF eApp.

Time to Create the Automation Test Cases for Other 6 Plans (8 Hrs/Plan) = 48 Hrs = 5.3 PD

With an effort saving of 1.6 PD/Month, ROI can be achieved in 5.3/1.6 = 3.3 Months

# Areas that cannot be Automated

* GUI look and feel verification.
* Accord xml verification.
* PDF Generation and Data Population verification