Mind-Alliance Test Automation Framework

Version	Date	Contributor	Comments
1.0	04/11/2011	Quamar Mehmood	Initial Draft.

Table of contents

1. Document Intent

The purpose of this document is to describe test case automation implementation using Selenium for Channels products.

The purpose of this document is to describe the automation framework that will be used for test case automation of Channels products using Selenium. It contains the information about:

- High level framework
- Interaction of components within the framework

2. System Overview

The automation framework is based on data-independent design keeping the data separate from the automation script. This data can be configuration data required by the automation scripts as well as the test data. Automation framework allows selective execution of automated test scripts.

Test cases selected for automation would be written in such a way that they are independent of each other. Similarly, the test scripts developed for the test cases are also proposed to be independent of each other so that any script could be selected for execution and the dependency on the other scripts is nullified.

Figure 1 shows the schematic view of the proposed test automation framework

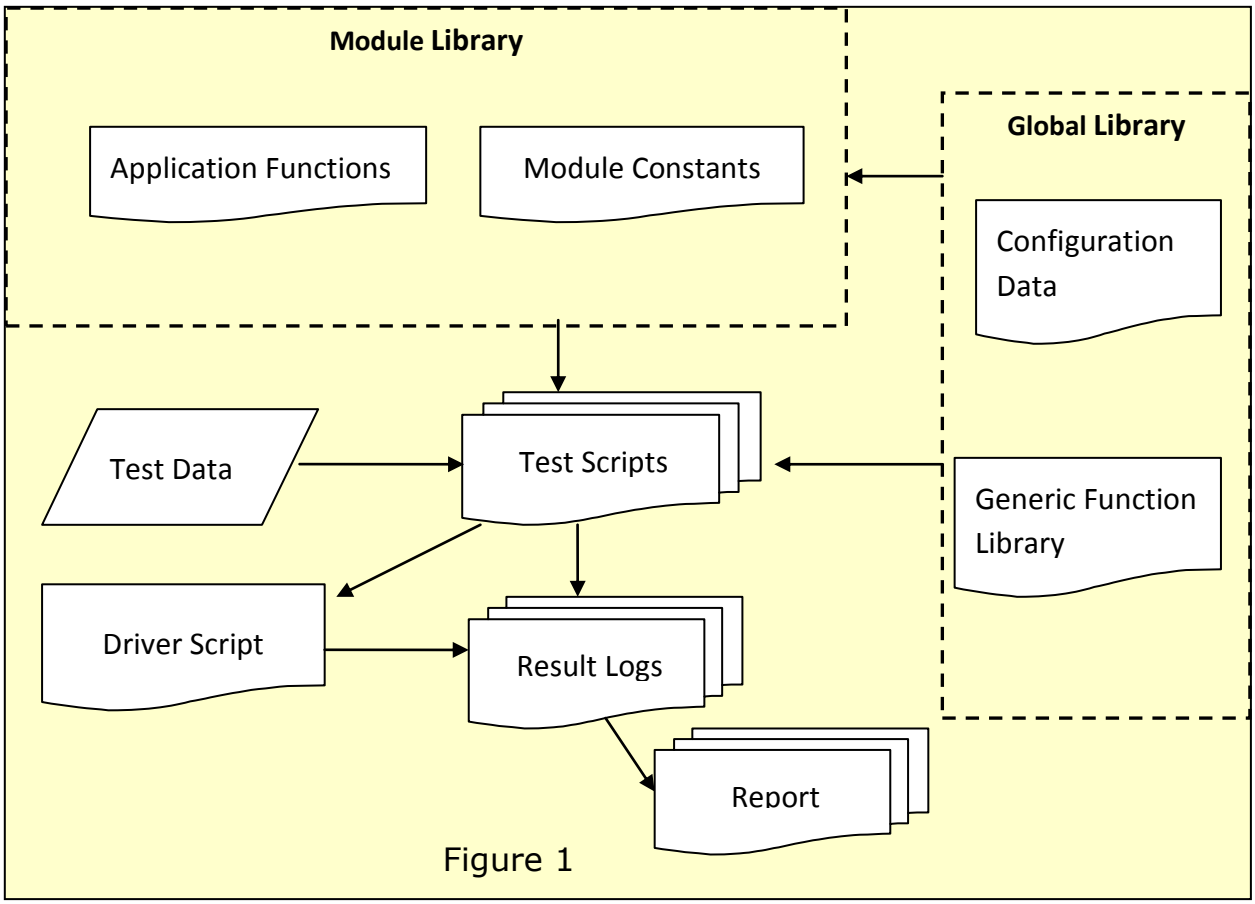


Figure 1

2.1 Driver Script

It facilitates selection of the required set of test scripts for test automation and drives the automation according to the execution sequence provided. The Driver Scripts behaves as Test Plan. This test plan will be formulated by the Form and complimented with a Test case sheet which will decide which test cases will participate in the test execution.

2.2 Configurable data

It is stored in the Configuration txt file (in our case Environment.txt) and contains information like Application URL, Result Log Path etc. Constants used across the test scripts are also stored here. User can change these settings to suit his requirement.

2.3 Test Data

It is kept in the 'Test DataPool'. Each test case owns a Separate/Common DataPool. The DataPool is actually the CSV/txt file which will have the test data required for execution of test case.

a. There will be one CSV/txt file per test case or common across all test cases.
b. The test cases will be independent of each other.
c. The framework will manage 'Test Result' logging.
d. Every test case execution will begin by setting the prerequisites and end with a cleanup task to revert back to the default application state.

2.4 Automation scripts

- There will be one test script file per test case.
- The test scripts will be independent of each other.
- The script file will have error/exception handling sections.
- The script file will manage 'Test Result' logging.
- Every test script will begin by setting the prerequisites and end with a cleanup task to revert back to the default application state.

2.5 Global Library

It consists of two Helper Super Classes.

- **GenericFunctionsLibrary** contains general purpose methods like retrieving data from the test DataPool, result logging and report generation etc.
- **LogFunctions** to log the result of script execution
- **ReportFunctions** to generate the report in html form
- **ApplicationFunctionLibrary** contains methods which are specific to the application. These are the functions to perform common tasks in the application like – invoking application, performing common functionality which is used by most of the test scripts and clean-up etc.

  Application related methods are called from automated test cases inside the test script to perform the basic tasks.
- **GlobalVariables** are the module level application constants.

2.6 Result Log

It contains script execution logs which can be used to keep track of how the test automation execution behaves. Result of test case will consist of the test case name, pass/ fail status, details of pass/fail and error/exception details.

Two types of result logs will be generated:

- Customized CSV result log: The test log in .csv files will contain entries in following sequence – Test Cases ID, Step No., Test Description, Test Result (Pass/Fail) and Comments (If any failure).
- Test Logs: It contains date time event of a particular steps.
- Error Logs: It contains information about any failure or exception in log file naming with TestCaseId.

3. Directory Structure

The recommended Test Project structure is designed to provide reuse of code and independence of action. Figure 3.1 shows the project structure.

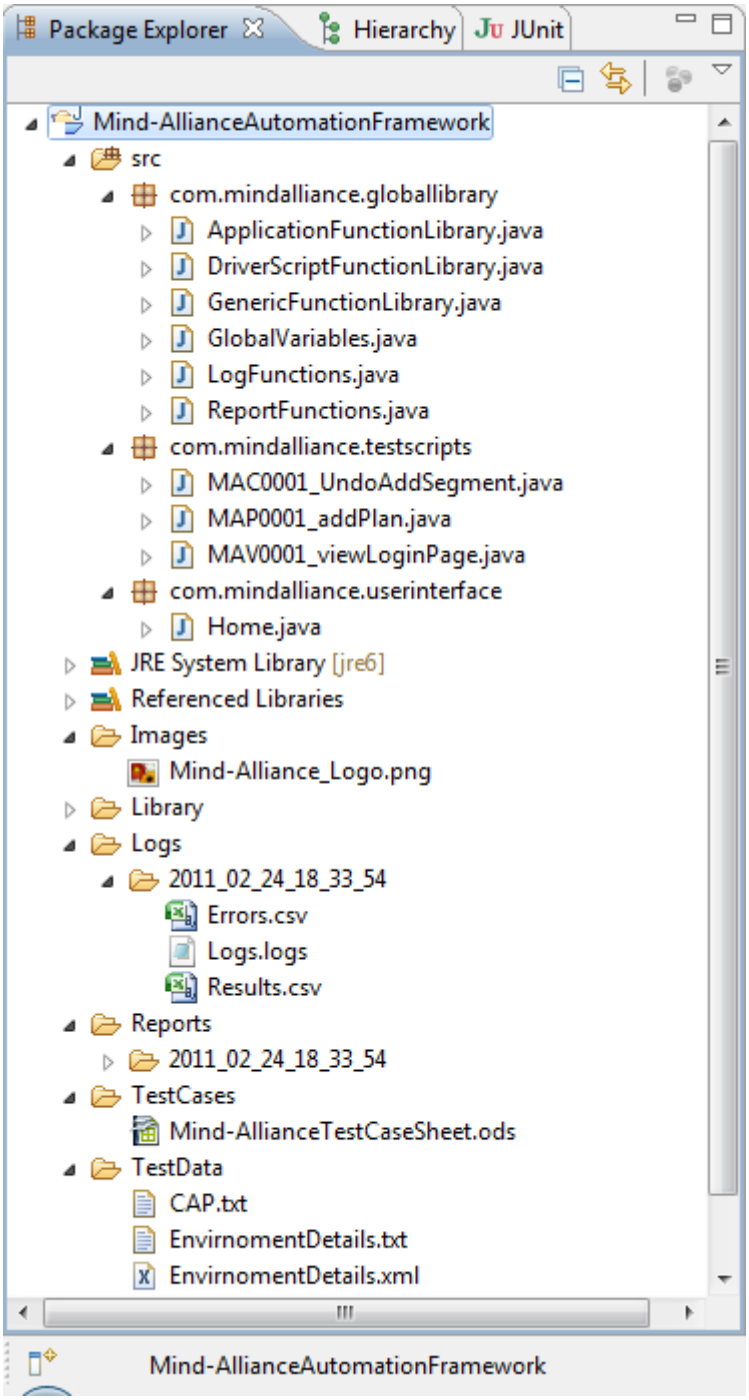


Figure 3.1: Project structure

All the automation (including the Test Script and Test Data) related to a particular test area is kept under one project.

Consider the above displayed Project Structure.
"Mind-AllianceAutomationFramework" is the automation project name. Each test module is represented as folders (e.g. **src, TestData)**. Test script

package inside "**src**" folder contains test scripts. Each script will have a common or separate test data associated with it. For example, in above diagram '**MAV0001_viewLoginPage**' is the test script and a test data with the same name in the form of txt/csv file should be kept inside test data folder.
The '**GlobalLibrary**' package inside "**src**" folder contains the generic functions and application functions in to separate files of type 'Helper SuperClass' (**GenericFunctions** and **ApplicationFunctions**). This library also contains classes GllobalVariables for result logging and reporting.

The Driver scripts are kept in the **src** folder. The driver scripts are responsible for test case execution.

The results and logs will be stored in .csv and .logs files respectively under a folder created with date time stamp "**2011_02_17_10_15_55**" under "**Logs**" folder. The Error logs files will be stored under Errors sub-folder

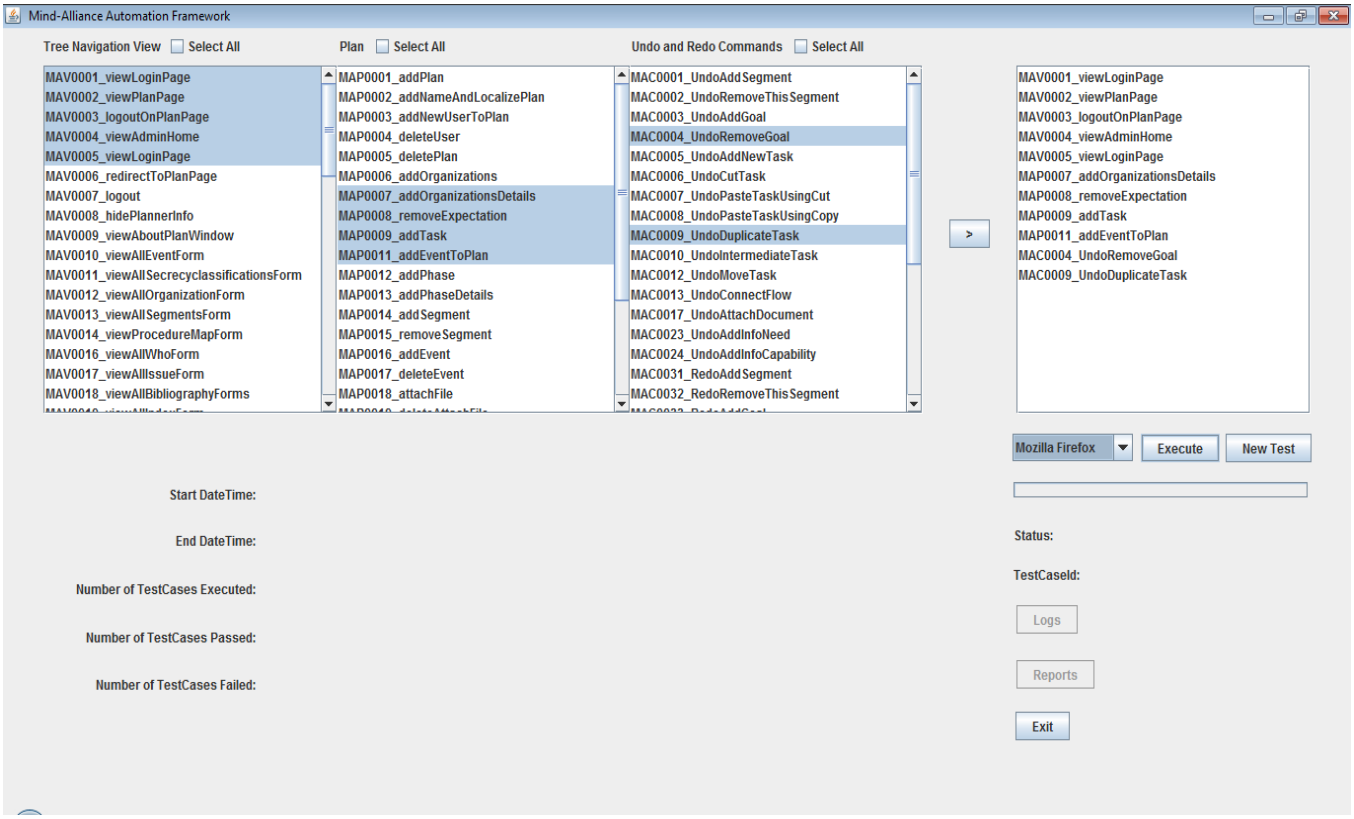
The report will be generated in html and ods files after completing the script execution. The files will be stored under a folder created with date time stamp "**2011_02_17_10_15_55**" under "**Reports**" folder. The TestCaseSheet will be updated in ods file according to the result of script.

TestCases folder contains the excel/ods files consisting of Test cases. The test execution results will be logged against the respective test cases.

4. System Description

4.1 Driver Script

The basic purpose of driver script is to have a common controller through which individual, selective and all scripts can be called for execution.
A driver script read the test case details from test case sheet which is to be executed and display it on UI or run it on console.



4.2 Configuration Data

All the configuration settings, like Result log Path, Time out duration etc. are kept in the Global Library package. User can configure these data to suit his requirements

4.3 Test Data

Individual test scripts will have their own test data or common which will contain all the test data that needs to be used by the script at run time.

4.4 Function Repository

The function repository comprises of Generic Function Library, Log Function Library and Report Function Library.

The generic function library has a set of general-purpose methods.

The application functions library has a set of commonly used methods which are specific to the application. These are the functions to perform common tasks in the application.

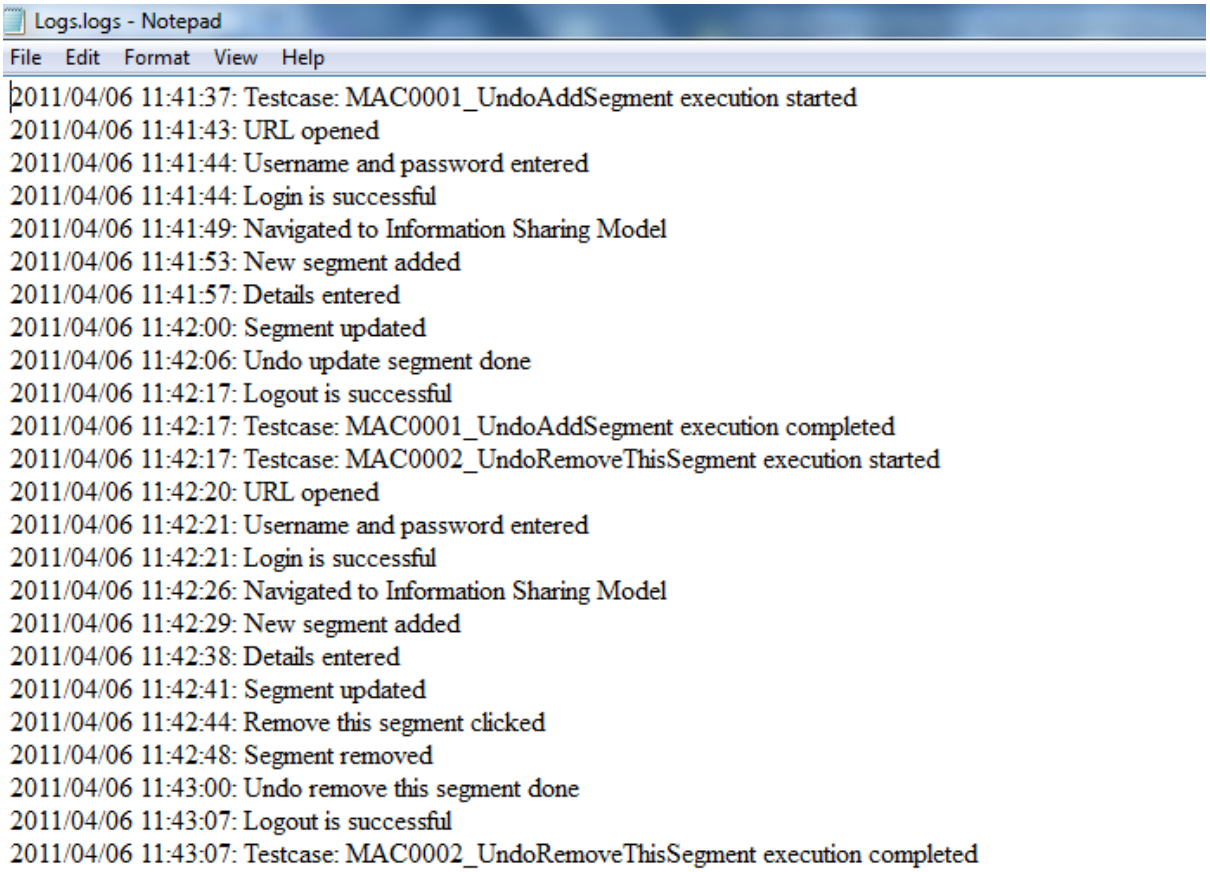
4.5 Result Log

Result.csv

It contains status of each test case steps.

	A	B	C	D	E	F	G
1	TestCaseId	VerificationStepNo	Description	Result	ScriptException	ErrorReport	
2	MAC0001_UndoAddSegment	1	URL opened	PASS			
3	MAC0001_UndoAddSegment	2	Username and password entered	PASS			
4	MAC0001_UndoAddSegment	3	Login is successful	PASS			
5	MAC0001_UndoAddSegment	4	Navigated to Information Sharing Model	PASS			
6	MAC0001_UndoAddSegment	5	New segment added	PASS			
7	MAC0001_UndoAddSegment	6	Details entered	PASS			
8	MAC0001_UndoAddSegment	7	Segment updated	PASS			
9	MAC0001_UndoAddSegment	8	Undo update segment done	PASS			
10	MAC0001_UndoAddSegment	9	Logout is successful	PASS			
11	MAC0002_UndoRemoveThisSegment	1	URL opened	PASS			
12	MAC0002_UndoRemoveThisSegment	2	Username and password entered	PASS			
13	MAC0002_UndoRemoveThisSegment	3	Login is successful	PASS			
14	MAC0002_UndoRemoveThisSegment	4	Navigated to Information Sharing Model	PASS			
15	MAC0002_UndoRemoveThisSegment	5	New segment added	PASS			
16	MAC0002_UndoRemoveThisSegment	6	Details entered	PASS			
17	MAC0002_UndoRemoveThisSegment	7	Segment updated	PASS			
18	MAC0002_UndoRemoveThisSegment	8	Remove this segment clicked	PASS			
19	MAC0002_UndoRemoveThisSegment	9	Segment removed	PASS			
20	MAC0002_UndoRemoveThisSegment	10	Undo remove this segment done	PASS			
21	MAC0002_UndoRemoveThisSegment	11	Logout is successful	PASS			

Logs.logs:

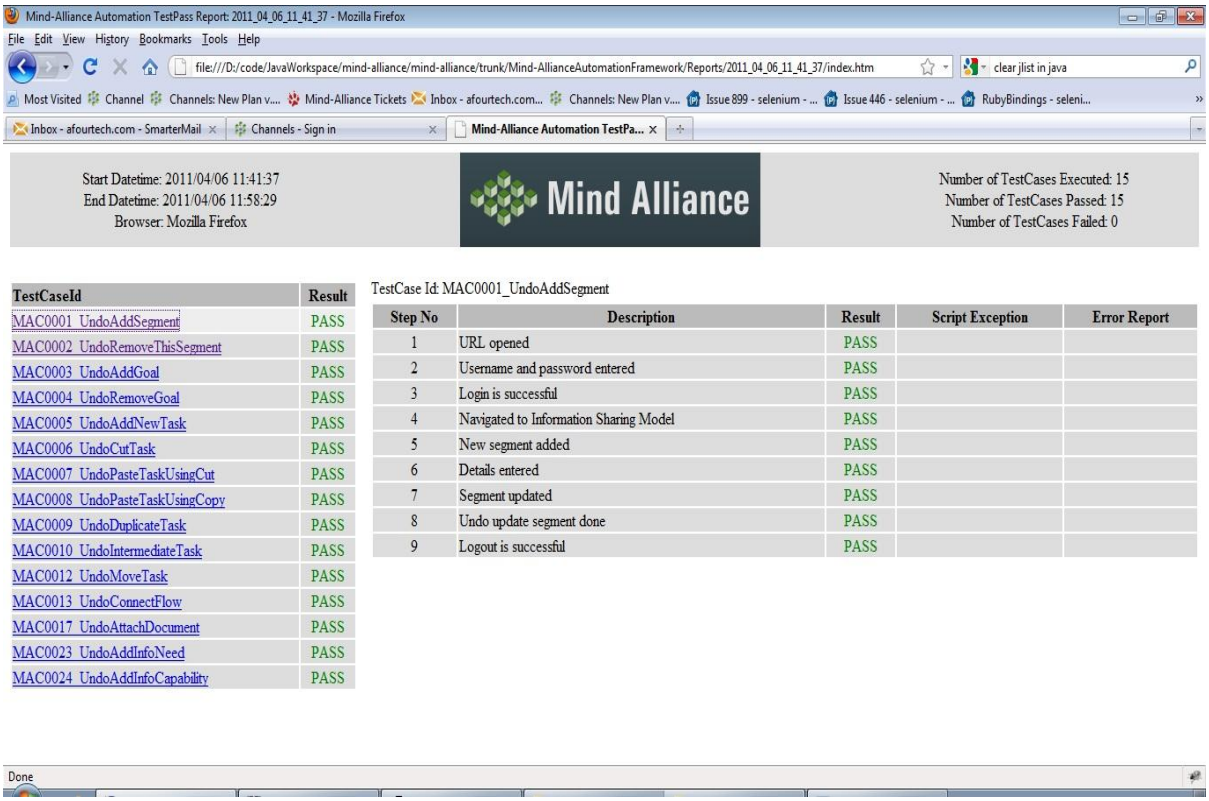


Error.logs:
It stores the stack trace information of the failed test case, so that the developer can easily identify the cause of failure.

4.6 Reports

It shows result of test pass with following details:

a. Tests pass time.
b. Browser details.
c. No. Of test cases executed, passed and failed.
d. Step by step status of each test case.



5. Test Scripts Breakdown Structure

Test cases will be taken from the Test Case sheet. for test automation, these test cases will have to be decomposed / updated to have the following information and then each test case will have one corresponding automated test script

- Each of the test step or the expected result for a condition will be as per the current implementations of the Mind-Alliance products
- Every test case will have a unique test case ID
- For every test step in the test case, there has to be an expected result and a unique step ID. This will help in troubleshooting and maintenance of the scripts.

6. Delivery Mechanism

The entire automation project will be delivered as the automation work. The user will need to import that project in the tool and run the scripts (after setting up the automation environment).

7. Constraints

- The pre-requisites for execution of automation stated in this document must be followed before executing any automated script.
- There can be UI changes over builds. There is a possibility that the automation will fail if these changes are not incorporated in the script. These failures are due to the possible changes in description of window or its object.
- Any changes in functionality or the UI of the application can cause failures in the automated test scripts.