Blue text on a black background

Description automatically generated 

# Maturity Claims - Tax Integration Test Automation POC

# TABLE OF CONTENTS

Table of Contents

[Maturity Claims - Tax Integration Test Automation POC 1](#_Toc152949351)

[TABLE OF CONTENTS 2](#_Toc152949352)

[1.0 – Purpose of the Document 3](#_Toc152949353)

[1.1 Scope 3](#_Toc152949354)

[1.2 Overview 3](#_Toc152949355)

[2.0 – Automation Tool 3](#_Toc152949356)

[2.1 UFT 3](#_Toc152949357)

[3.0 – UFT Specific Guidelines 4](#_Toc152949358)

[3.1 Object Repository (OR) 4](#_Toc152949359)

[3.2 UFT Add-ins & Add-in Manager 5](#_Toc152949360)

[4. 0 – Highlights of Maturity Claims Test Automation Framework 5](#_Toc152949361)

[5. 0 – Pre-Requisites 6](#_Toc152949362)

[6. 0 – MATURITY CLAIMS Test Automation Framework Architecture 9](#_Toc152949363)

[6.1 Framework Architecture: 9](#_Toc152949364)

[7. 0 – Framework Components 10](#_Toc152949365)

[8.0 – Framework Libraries 11](#_Toc152949366)

[8.1 Generic Function Library 11](#_Toc152949367)

[8.2 – Business Functions 11](#_Toc152949368)

[8.3 – Framework Environment Variables 12](#_Toc152949369)

[9.0 – Recommended Scripting Standards / Guidelines 12](#_Toc152949370)

[9.1 Comment Standards 12](#_Toc152949371)

[9.2 Scripting Guidelines 12](#_Toc152949372)

[10.0 – Framework Usage Guidelines 13](#_Toc152949373)

[10.1 Driver Script 13](#_Toc152949374)

[10.2 Test Data 13](#_Toc152949375)

[10.3 Function Libraries 14](#_Toc152949376)

[10.4 Object Repository 14](#_Toc152949377)

[10.5 Results 14](#_Toc152949378)

[10.6 Autogenerated Email 15](#_Toc152949379)

## 1.0 – Purpose of the Document

The purpose of this document is to describe how to get started with the framework, to get knowledge about the automation tools used for the POC of maturity claim Automation and process to be followed when using the existing transaction business specific functions. This document will help ensure consistency across the team, resulting in increased usability and maintainability of the developed code.

### 1.1 Scope

The scope of this document is to understand the framework, steps for designing and developing Test Automation scripts for various regression scenarios.

### 1.2 Overview

This document provides guidelines for:

* Automation Tools used for maturity claim Automation
* Highlights of UFT Test Automation Framework
* UFT Test Automation Framework Architecture
* Framework Components
* Recommended Scripting Standards / Guidelines
* Comment standards
* General guidelines
* Business Functions and procedures

## 2.0 – Automation Tool

### 2.1 UFT

**UFT** stands for **U**nified **F**unctional **T**esting. It was earlier known as **QTP (Q**uick Test **P**rofessional). QTP/UFT was Launched in 2002 (Nov) by Mercury Interactive. Later taken over by HP, in 2007.

* It is an icon-based tool that automates the regression and [Functional](https://www.guru99.com/functional-testing.html) [Testing](https://www.guru99.com/functional-testing.html) of an application
* Both technical, as well as a non-technical tester, can use UFT
* It provides both features- Record as well as Playback
* We can test Desktop as well as the Web-based applications
* Micro Focus’s UFT uses VBScript to automate applications
* It supports the largest pool of software development environments like SAP, Oracle, JAVA, .Net, Power Builder, UI, Terminal Emulator etc.
* UFT tool helps the testers to perform an automated functional testing uninterrupted.

## 3.0 – UFT Specific Guidelines

UFT provides the ability to write functional tests that simulate the behaviour of an end user. We write these tests in scripts, which are designed to exercise feature functionality as it is described in the functional requirements. In UFT, test scripts are composed of statements coded in Microsoft’s VBScript programming language.

### 3.1 Object Repository (OR)

Object Repository is a collection of Test Objects and information that is recognized by UFT for working on it. When a user records a test, the objects and their properties are captured by default. There are 2 types of object repositories in UFT.

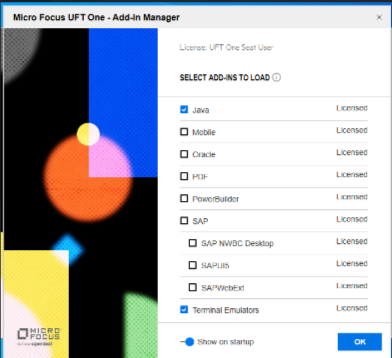
* Local Object Repository
* Shared Object Repository

As the name suggests, the Local Object Repository is applicable only for that action. As we know, that UFT creates a New Test with 1 action by default. Local Object Repository can be opened by traversing to Resources → Object Repository. This is the default OR in UFT.

The Object Repository is shared across actions/modules, which would be mapped for two or more actions. Local objects can be exported to be saved into Shared Object Repository by using the option "Export Local Objects" option. Shared Object Repository can be opened by traversing to Resource → Object Repository Manager

### 3.2 UFT Add-ins & Add-in Manager

After launching UFT, you will get an UFT Add-in Manager screen as below.



Add-ins in “UFT One” are a way to ensure UFT One identifies objects in a corresponding environment.

**For example:** If you are testing Java based UI controls, you would need Java add-in. For Siebel app, you need Siebel add-in.

**Note**: You can select the required add-in according to your application built in technology. For Maturity Claims we will be selecting the **Java** add-in along with **Terminal Emulator**

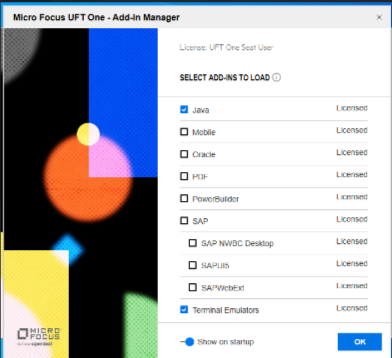
## 4. 0 – Highlights of Maturity Claims Test Automation Framework

* This framework is designed keeping in mind the concepts of Hybrid Driven Framework.
* The execution summary results are displayed in HTML pages in a well-organized fashion.
* Along with execution status the screenshots are captured for both passed and failed validation steps
* Automatic Email will be triggered for sending the test results through an email.

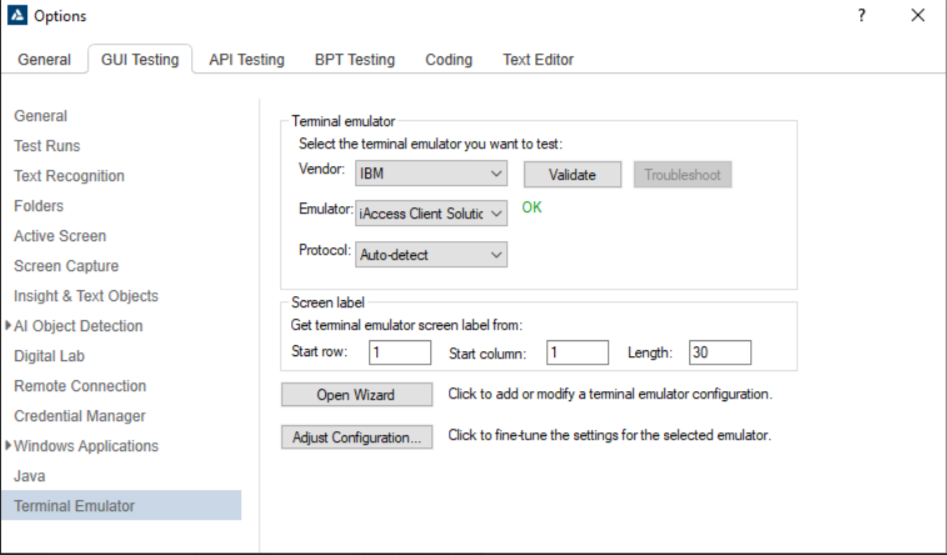
## 5. 0 – Pre-Requisites

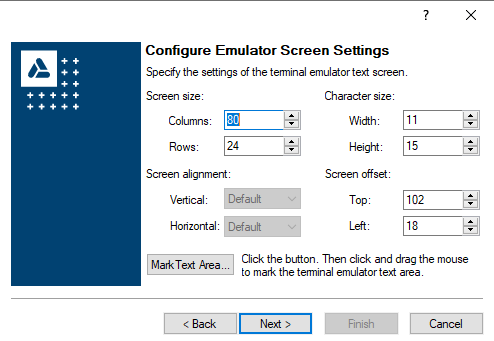
For automation framework setup in a new machine, we need to follow below mentioned prerequisite.

* Java and Terminal Emulator add-in should be enabled in UFT



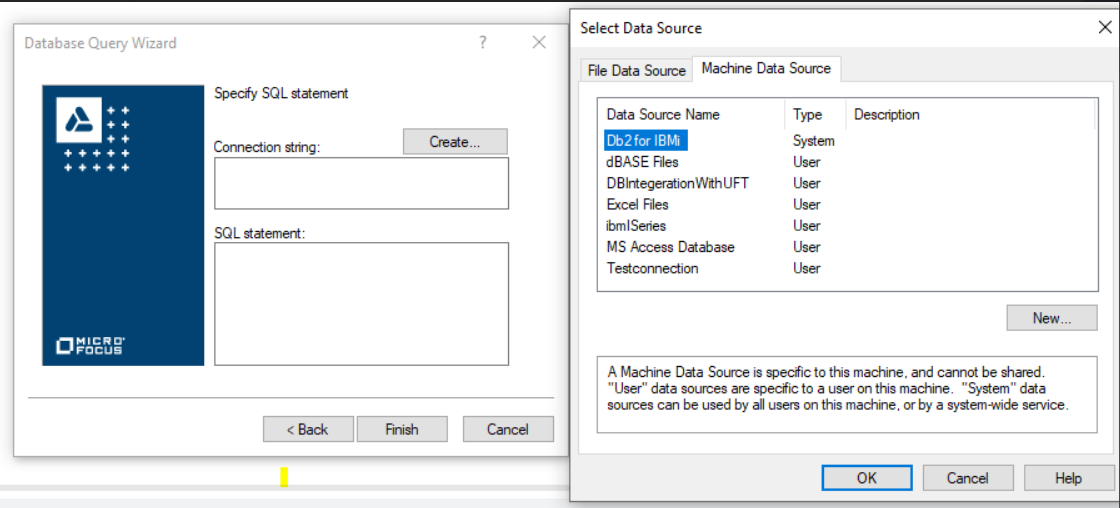
* Make sure to add above **pcshll32.dll** file in the “C:\Program Files (x86) \IBM\EHLLAPI” path on your machine if it is not existed.
* Terminal Emulator for IAccess Client vendor with proper row, column, screen offset, character length and width should be configured and activate the emulator.





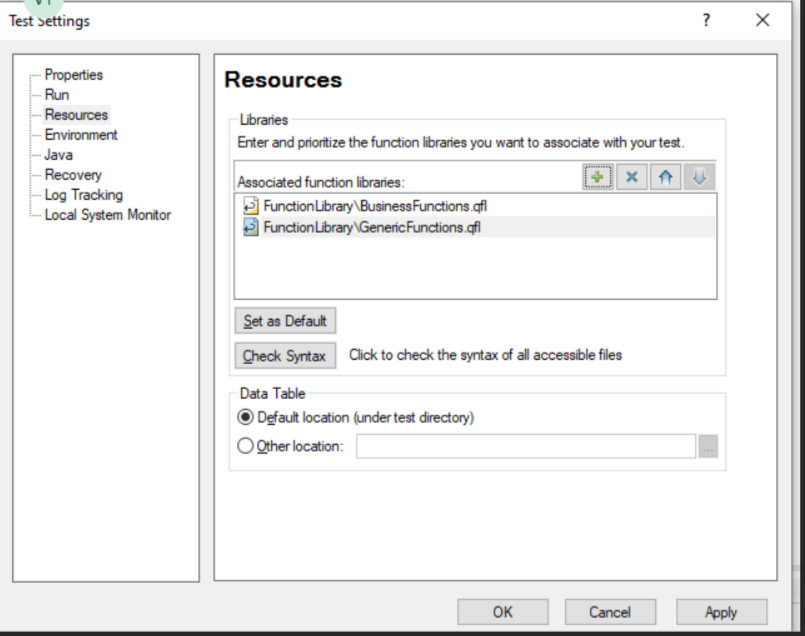
* Emulator screen settings should be configured in the Terminal Emulator configuration as mentioned in above screenshot, if it is changed check with the developer and update the dimensions. Open Wizard🡪Next🡪Next🡪Configure Emulator Screen Settings🡪Next🡪check the Save Terminal emulator settings to file checkbox 🡪 Finish
* Valid test inputs should be provided in the Test data sheet for the scenarios automated
* Connection String should be configured for Database integration with the UFT

Steps to follow Design->Checkpoint->Database Checkpoint->Specific SQL statement->Select the Db2 for IBM in the Machine Data Source.

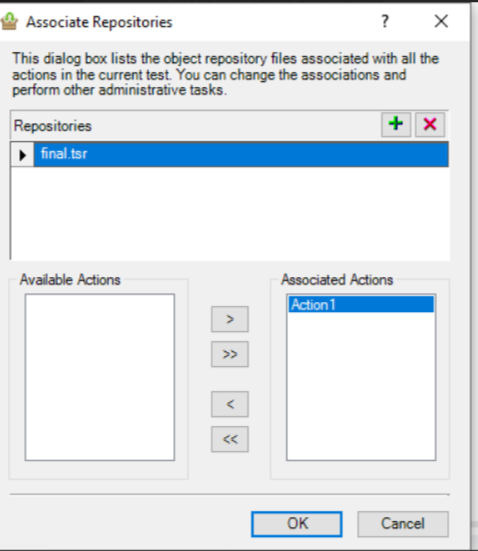


* System should be idle during the automation executions as we have used keyboard actions for handling keystrokes in the terminal emulator screens
* Function libraries and Shared Object Repositories should be associated with the driver script.

1)Steps to follow to associate the Function Library->File ->Setting->Resources->Associated function Libraries



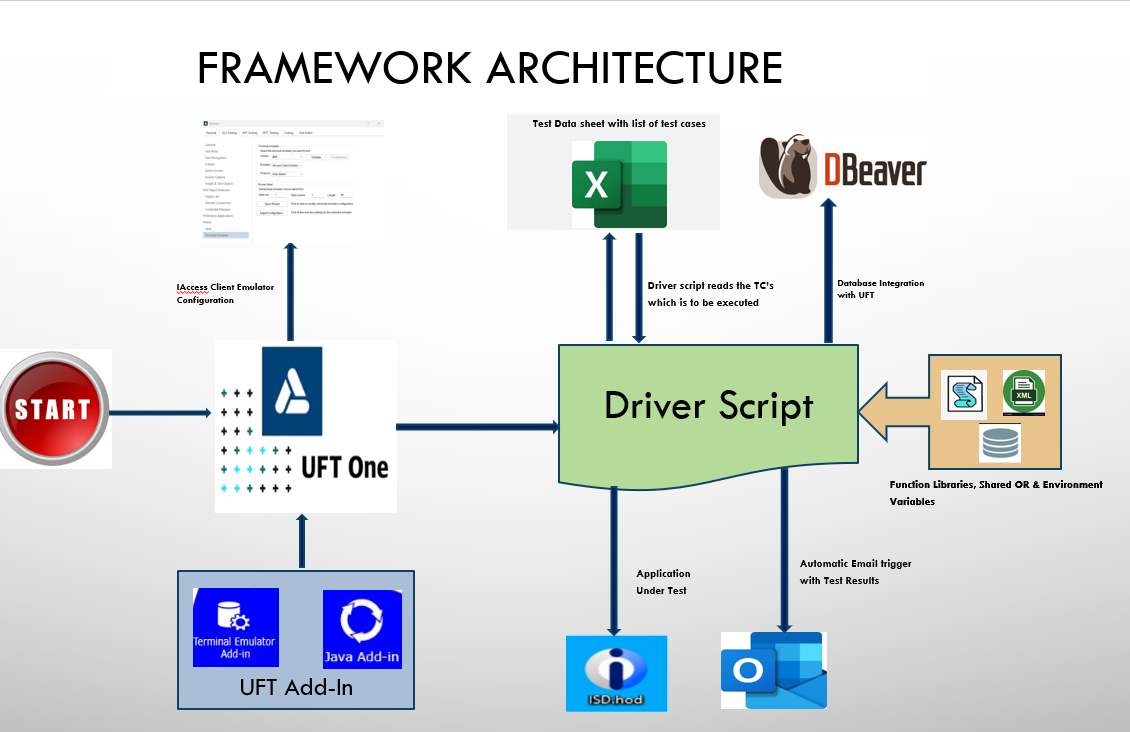
2)Steps to follow to associate the Object repository Resources->Associate Repositories



# 6. 0 – MATURITY CLAIMS Test Automation Framework Architecture

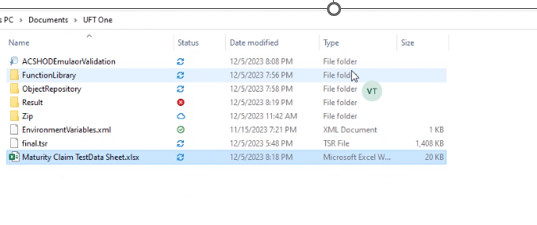
We have designed this a hybrid driven framework considering mainframes applications. Below are the UFT automation architecture/ Key components of Framework that can used for any kind of application.

### 6.1 Framework Architecture:



# 7. 0 – Framework Components

Following is the folder structure for storing automation scripts and library files across the teams



**● <Automation>**

o **< ACSHODEmulatorValidation >** This is the Driver Script/Action Script which drives entire automation framework.

* **<Function Library>** Includes files like BusinessFunctions.qfl, GenericFunctions.qfl,

o <**Maturity Claim Test Data Sheet>** -This is the test datasheet used for poc scenarios.

o **<Object Repository>** - This flat file consists of all the object properties of the application identified so far. Framework has used a specific format to define the list of properties for the object in application.

* **<Results>** - This component consists of all the execution results generated. For each regression suite execution separate results directory is created with summary index html file. There will be screen snapshot taken on which is also given as a hyperlink in the html report.

**Note**: Detailed explanation of each framework component is provided in the same document of Framework Usage Guidelines Section.

## 8.0 – Framework Libraries

To maintain the Framework effectively and reuse the appropriate libraries it is recommended to have separate (.vbs/.qfl) files to differentiate the categories of the functions. As of now we have the following library files. As mentioned above there will 2 categories of Function Library in the framework.

* Generic Function Library
* Business Function Library

### 8.1 Generic Function Library

Generic functions are the common functions/methods that can be used irrespective of any Web, Desktop application or Mainframes application.

**For Example**, if we create generic functions like Clicking on object, Dynamic wait, Validating the Alpha numeric value etc, the same generic functions can be used for any other applications in automation.

Below mentioned Generic function is Dynamic wait function it will wait until the object exist. This can be used across the applications in Automation.



### 8.2 – Business Functions

Business functions are the functions/methods that are created specific to the application/functionality. It cannot be reused for any other business application.

For Example: Below highlighted functions are created specific to Maturity claim (i.e., Our project) application transaction.

**fn\_ValidationOfMaturityClaim**, **fn\_incorrectContractNumberValidation, *fn\_CorrectContractNumberValidation*, etc.**

It can be found under this library. These functions can be created by all automation script developers but must be approved by QA group/Core Automation Team before making it to the library.

### 8.3 – Framework Environment Variables

Contains all the global variables, environment related data and arrays that are used with in libraries and scripts.

## 9.0 – Recommended Scripting Standards / Guidelines

### 9.1 Comment Standards

Significant lines in the code should be provided with inline comments to better explain the line of code's purpose and make it easier for subsequent developers to understand the code faster and more thoroughly.

Example:

'If the Object is not specified - exit the function

If obj is nothing then Exit Function

End If

### 9.2 Scripting Guidelines

Every test script in the test suite has to meet generic coding standards and a modular approach to design. Every script/library functions should follow standards below:

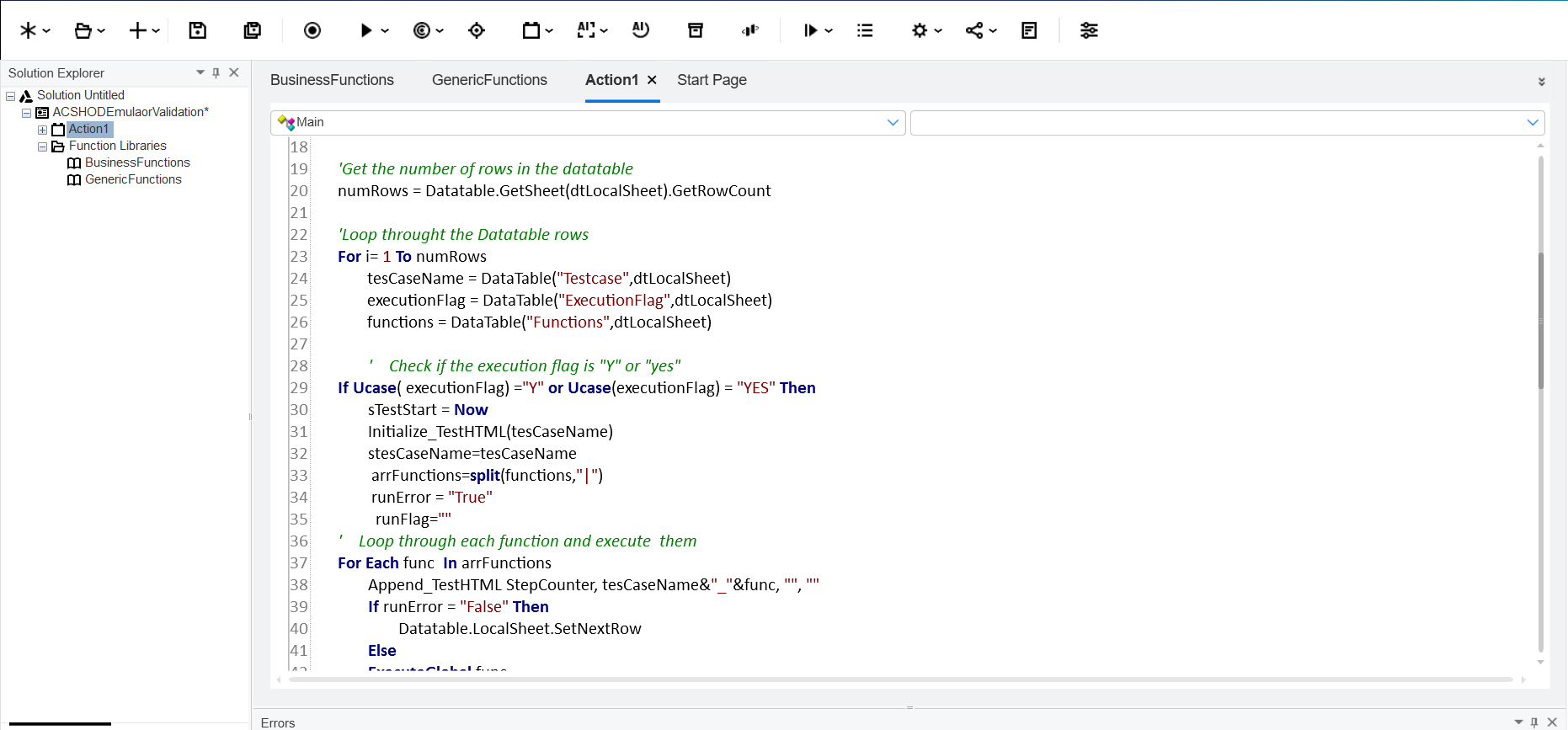
* Every script/functions should contain a header that includes (in following order): ' *'\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* HEADER \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**  
  *'Function Name:*   
  *'Description    :*   
  *'Created By :*   
  *'Created Date :*   
  *'\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*HEADER\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**
* Declare all variables used. Each variable must be declared before being used.
* Every test suite and every test script should check if test data required for successful script execution is available in the application.
* If a Set of script statements are re-usable across the business functions, it should be converted to a common function and make it available as Generic Function

## 10.0 – Framework Usage Guidelines

### 10.1 Driver Script

A driver script triggers and monitors other scripts to test an application. A driver script typically performs no testing of its own, although it may use conditional and other control logic to manage optional execution of other tests within the driver.

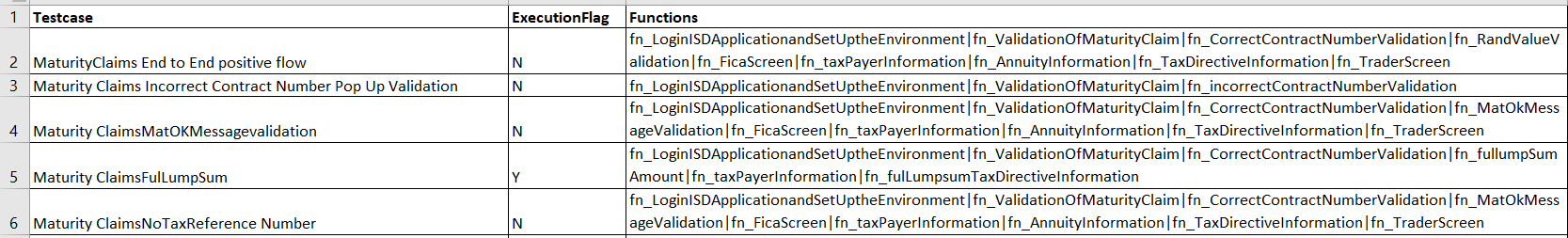
The Driver script will check the testcasename and execution flag in datasheet. For the testcasename if execution flag is “Y” or “Yes” execution will star.



### 10.2 Test Data

Implemented a Hybrid driven framework test data sheet is the combination of data and keywords.

The framework will support file-based configuration for the test data. Each automated testcase must have the respective functions. For example, if user has testcase as “Maturity Claims End to End positive flow” directory then, respective functions are mentioned under “**Functions**” column as shown below and all the data parameters need to updated accordingly.



### 10.3 Function Libraries

To maintain the Framework effectively and reuse the appropriate libraries it is recommended to have separate (.vbs/.qfl) files to differentiate the categories of the functions. As of now we have the following library files. As mentioned above there will 2 categories of Function Library in the framework.

* Generic Function Library
* Business Function Library

The Function Library should be associated with Driver Script as mentioned above

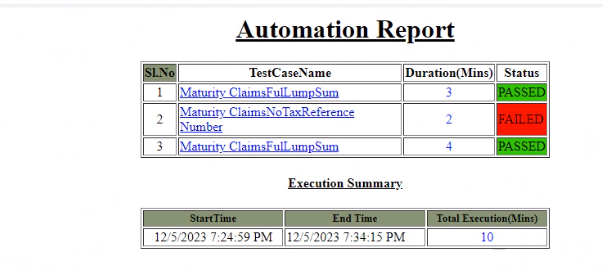
### 10.4 Object Repository

We are using the Shared Object Repository in this framework. The object repository must be associated with the Driver Script.

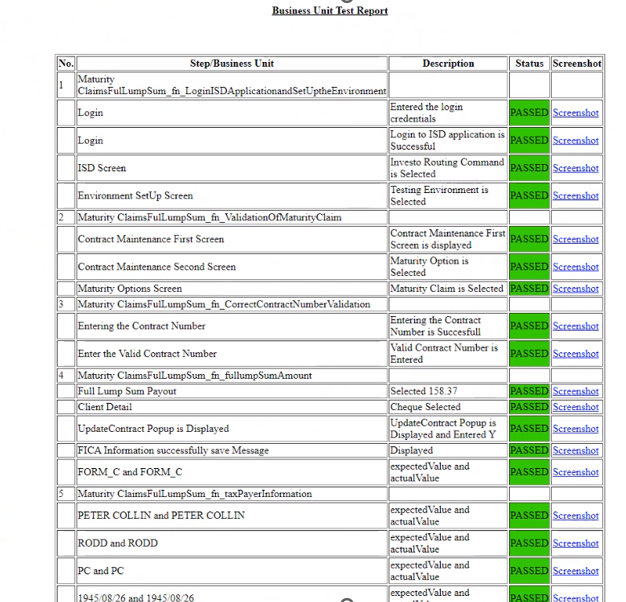
### 10.5 Results

Every execution generates an html index report with summary of all the tests of the automation suite has passed and failed and stored under “Results” directory. Below is the screenshot of the index file with only 1 script in automation suite. Contains the HTML Results generated post Every Execution.

Note: In this framework we can capture the screenshots for both Passed and Failed steps.



Each Script name (ex: Maturity ClaimsFulLumpSum) in the index html file is a hyperlink which will display the detailed business key navigation and validation status as shown in below screenshot.



### 10.6 Autogenerated Email

After execution an autogenerated email will be triggered to respective recipients. In the email we can share the complete batch execution results along with screenshots in a zip file or if needed we can share the index.html report which consist of testcases and execution status and time taken for execution.

