|  |  |  |
| --- | --- | --- |
| |  |  | | --- | --- | | Capgemini_logo_hr_TIF |  | |

Nordea –Rest API Automation Testing

Design Document

Table of Contents

[1. Introduction 4](#_Toc461787158)

[2. Pre-requisites 5](#_Toc461787159)

[2.1 Network drive mapping 5](#_Toc461787160)

[2.2 System Environment Variables 5](#_Toc461787161)

[3. Framework Design 7](#_Toc461787162)

[3.1 Folder Structure 7](#_Toc461787163)

[3.2 Initialization 8](#_Toc461787164)

[3.3 Configuration Parameters 8](#_Toc461787165)

[3.4 Associating Function Libs to Test 9](#_Toc461787166)

[3.4.1 Generic Function Libs 9](#_Toc461787167)

[3.4.2 Module specific Function Libs 9](#_Toc461787168)

[3.5 Associating Object Repository file to Test 11](#_Toc461787169)

[3.6 Batch Execution 11](#_Toc461787170)

[3.7 Test Execution Report 12](#_Toc461787171)

[3.7.1 QTP / UFT execution log 12](#_Toc461787172)

[3.7.2 HTML execution log 13](#_Toc461787173)

[3.7.3 Console Log 13](#_Toc461787174)

[3.7.4 Debug Log 13](#_Toc461787175)

[4. Working with Solution Manager + UFT 13](#_Toc461787176)

[4.1 SolMan settings in UFT 13](#_Toc461787177)

[5. UFT Settings 15](#_Toc461787178)

[5.1 Test Run Settings 15](#_Toc461787179)

[5.2 Test Iteration Settings 16](#_Toc461787180)

[5.3 UFT Add-ins 17](#_Toc461787181)

[6. Appendix 18](#_Toc461787182)

[6.1 UFT+SolMan integration 18](#_Toc461787183)

[6.2 Sample HTML Execution Report 18](#_Toc461787184)

[6.3 Configuring new VM 18](#_Toc461787185)

[7. Glossary 18](#_Toc461787186)

**Revision History**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** | **Reviewer** |
| 12/01/2017 | 0.1 | Initial Draft | Visweswara Rao Kolluri |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

# Introduction

This document briefs about the API automation testing approach with Robot Framework using RIDE tool. This framework uses Gradle as build testing tool and going to use Jenkins as a continuos integration tool. As per initial scope, we thought of using postman and newman tools for automation. However, database connections can not be made using these tools so decided to use robot framework. Also for fetching data from legacy database we use OSB(Oracle Service Bus) as an interface through which the AISP gateway communicates to database.

# Pre-requisites

## Network drive mapping

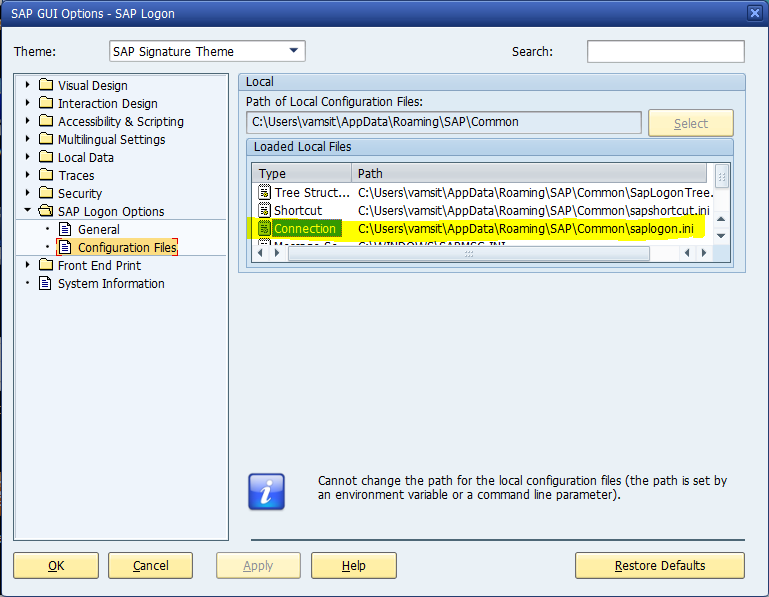
Map Network Drive Y: To [\\na-in3pj-v001\ECH\_Interface\_HPR\TAO](file:///\\na-in3pj-v001\ECH_Interface_HPR\TAO)

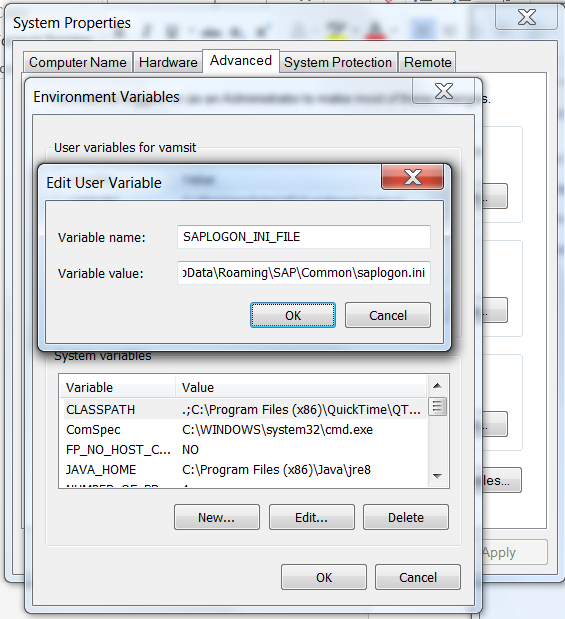
Also, refer section “[Configuring new VM](#_Configuring_new_VM)” in Appendix for list of pre-requisite settings.

## System Environment Variables

As pre-requisite, it’s required to create below system environment variables

* **AUTOMATION\_ROOT\_PATH** – Set the value as “Y:\SolMan\_QTP\”. It’s mandatory to set this variable, as Initialization() functions reads this value at run-time and builds the folder path for all automation artifacts (TestData, Libs, Object Repo etc…)
* **SAPLOGON\_INI\_FILE** – This variable is used by UFT to get the SAP Logon details. If this variable is not set, then SAP logon connection details are not displayed when trying to connect to SolMan from UFT. Set the value of this variable to “<<FolderLocation>>\saplogon.ini”. You can get the folder location of saplogon.ini file SAP GUI options screen as shown below:



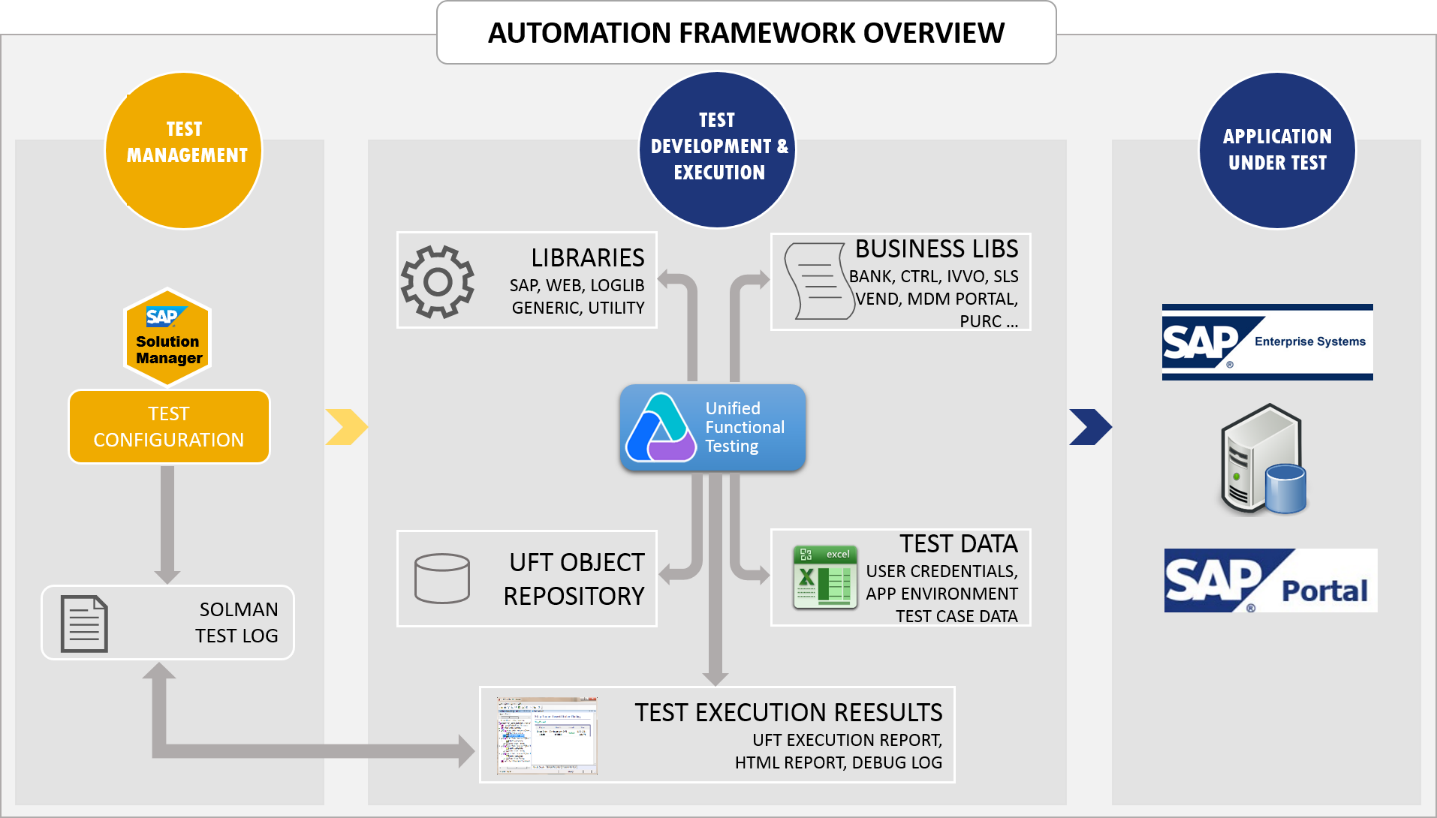


# Framework Design

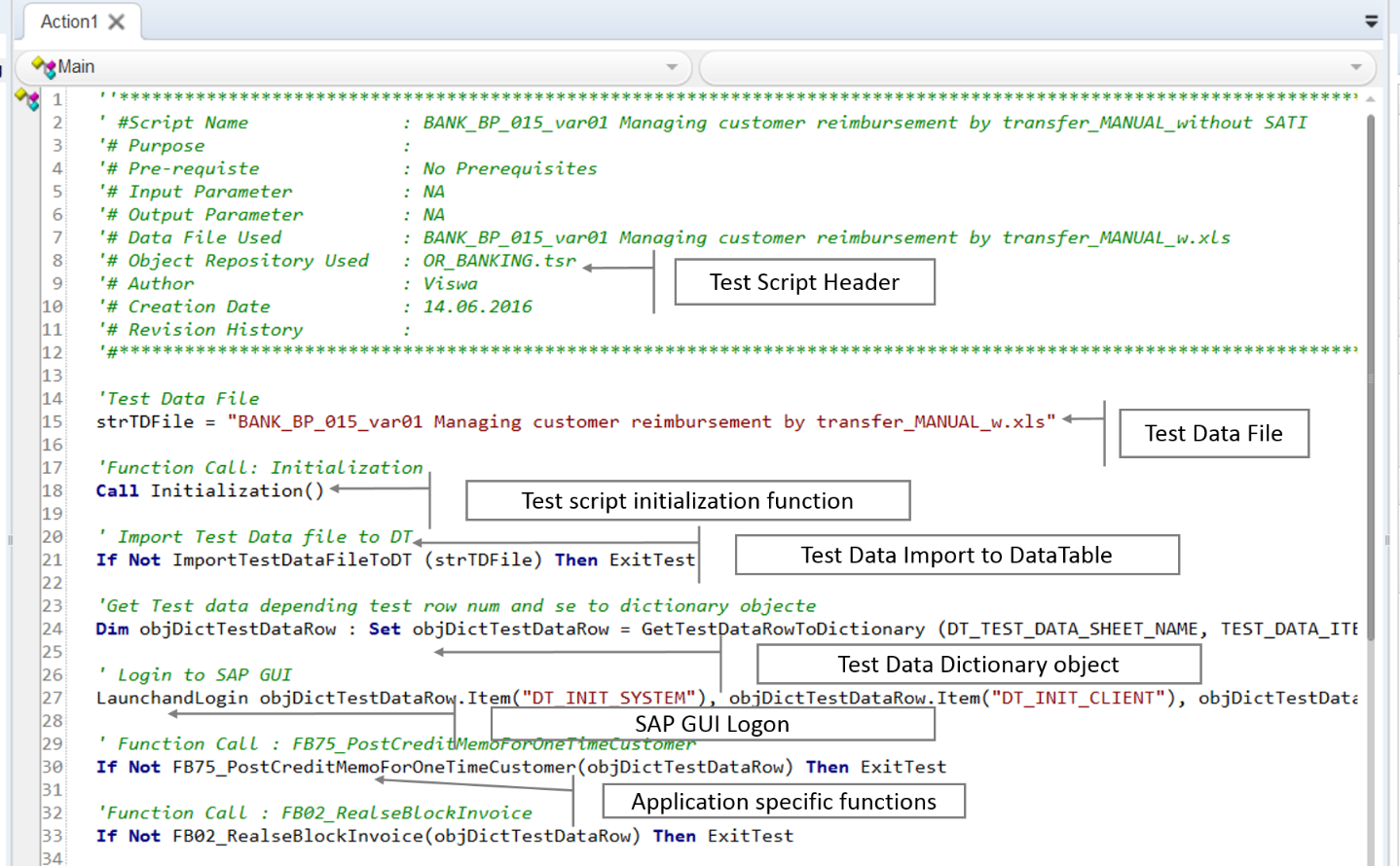
This automation framework is conceptualized based on Business Process Test (BPT) methodology. Each QTP/UFT test script consists of function calls which are called sequentially as per the End-to-End (E2E) flow of test scenario.

Bellow picture depicts the automatoin framework overview. Currently “Test Management’ section depicting the SAP SolMan is not explained as the SolMan environment is not available currently. More information will be added about SolMan implementaiton when the test scripts are migrated to SolMan in the near future.

Currently, we use QTP/UFT for test scripts development and batch execution and the rest of the following sections in this document, explains about the various automation artifacts and the implementation.



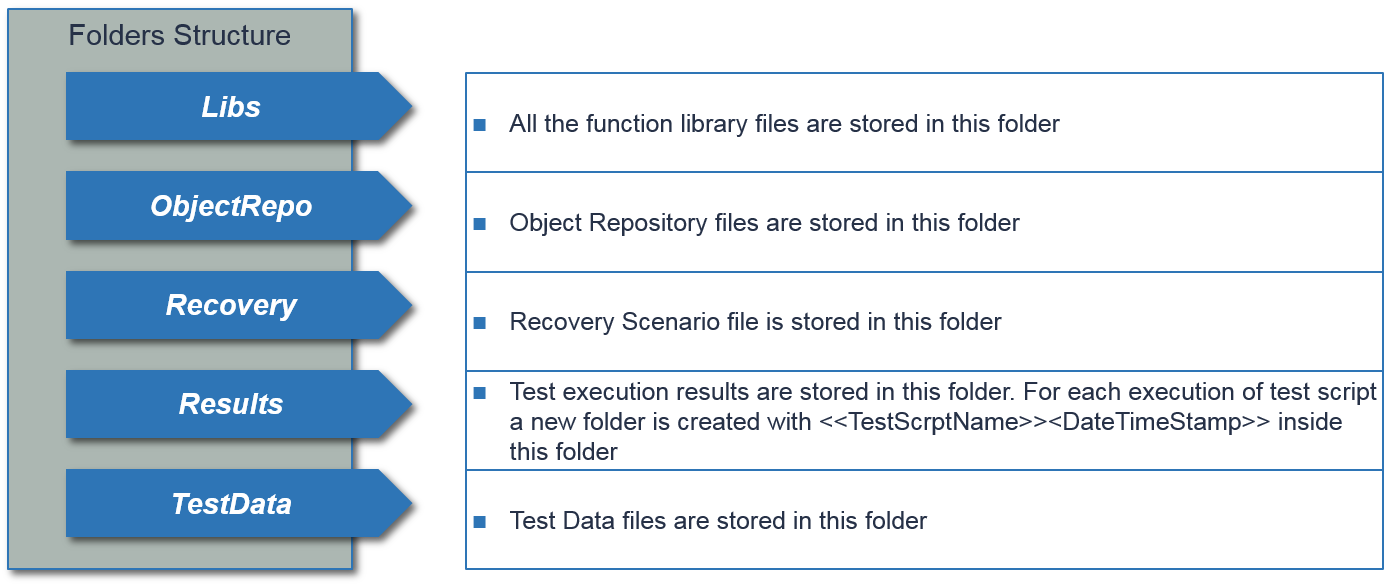
Find below the sample test script template:



At the start of every test script, there are few and variable declarations and common functions calls, which are mandatory to have and in the same sequence as shown in the above script template.

All the UFT test scripts are to be stored in SAP Solution Manager 🡪 Project Configuration (Blueprint)

## Folder Structure

All the automation test artifacts except for UFT test scripts are stored in “Y:\SolMan\_QTP” folder in network drive. UFT test scripts are stored in SAP Solution Manager. Below are the details of folder structure:

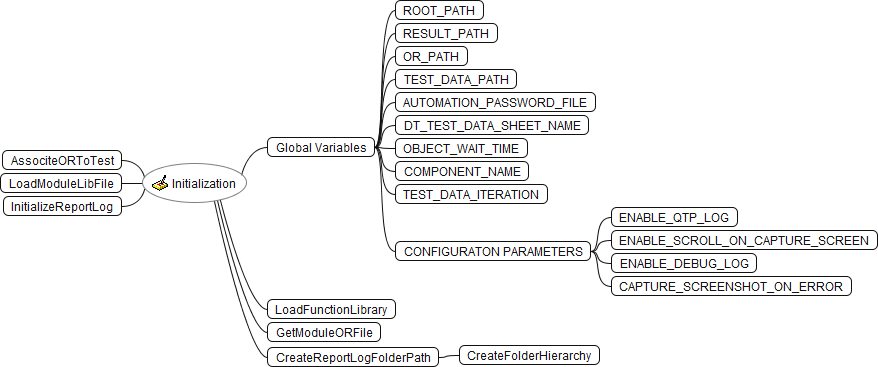
Note: As currently SAP Solution Manager is not ready, so UFT Test scripts are also stored in “Y:\SolMan\_QTP” folder, as an alternative option to execute the Test scenarios.

## Initialization

Initialization function is called as the first statement in every test script. This function is available in “Y:\SolMan\_QTP\Libs\Initialization.vbs” file and this file had to be associated to every test script manually. To associate function library to test script, from UFT

* Select menu File 🡪 Settings.
* Select “Resources” and add “Initialization.vbs” file. This file is available in folder “Y:\SolMan\_QTP\ Libs\”

Below is the class diagram representation of Initialization.vbs



Initialization function performs below set of actions:

* Reads and assign configuration parameters value (config.xml)
* Load Generic and Module specific Object repository files
* Loads Generic and Module specific function libraries
* Initialize test execution report file
* Stores all Global variables

## Configuration Parameters

Configuration parameters are stored in config.xml file available in “Y:\SolMan\_QTP” folder.

|  |  |
| --- | --- |
| CONFIG PARAMETER NAME | DESCRIPTION |
| ENABLE\_DEBUG\_LOG | If value set to “TRUE”, then detailed step information is written to DEBUG log |
| CAPTURE\_SCREENSHOT\_ON\_ERROR | If values set to “TRUE” then, screenshots are captured and are shown as thumbnail in HTML execution report |
| ENABLE\_SCROLL\_ON\_CAPTURE\_SCREEN | If value set to “TRUE”, then screen will capture while scrolling |
| ENABLE\_QTP\_LOG | If value set to “TRUE”, then detailed step information is written to UFT log |

## Associating Function Libs to Test

Functions are re-usable codebase, which can be called from UFT test scripts and can also be called from within from another functions. Initialization() will perform the action of associating library files to UFT test script at run-time.

### Generic Function Libs

Generic function libs are the placeholder for all the generic functions which are application independent. Below are the list of generic function library files, which are placed in “Y:\SolMan\_QTP\Libs\” folder

* GenericLib.vbs
* Common.vbs
* LogLib.vbs
* Sap.vbs
* UtilityLib.vbs
* Web.vbs
* CommonBusinessFunctions.vbs

All the above library files are loaded at runtime by Initialization() function

### Module specific Function Libs

Module specific function library file consists of functions which are used to the functionality of a specific module (e.g. Bank, Controlling, Sales etc…) are placed in “Y:\SolMan\_QTP\Libs\”

Below is the list of module specific function library files

* Bank\_BusinessFunctions.vbs
* Controlling\_BusinessFunctions.vbs
* Invoice\_BusinessFunctions.vbs
* Procurement\_BusinessFunctions.vbs
* Project\_BusinessFunctions.vbs
* Sales\_BusinessFunctions.vbs
* MDMPortal.vbs
* Portal.vbs
* Vendor\_BusinessFunctions.vbs

The above listed module specific functions library files are loaded at runtime during Initialization. Based on the UFT test script naming convention, the corresponding module library is loaded.

For example:

Bank module UFT test script “BANK\_BP\_004\_var02 Managing supplier payments by manual cheques” loads the corresponding module function library file “Bank\_BusinessFunctions.vbs” at runtime.

Initialization() function, gets the test script full name and checks for the corresponding module name by first four characters in the test script name. Based on the module name value, it loads the corresponding module library file.

Below are the list of modules and its corresponding library files:

|  |  |
| --- | --- |
| **Module Name** | **Module specific function library file** |
| BANK | Bank\_BusinessFunctions.vbs |
| CTRL | Controlling\_BusinessFunctions.vbs |
| INVO | Invoice\_BusinessFunctions.vbs |
| PROJ | Project\_BusinessFunctions.vbs |
| PURC | Procurement\_BusinessFunctions.vbs |
| SLS | Sales\_BusinessFunctions.vbs |
| VEND | Portal.vbs  MDMPortal.vbs  Vendor\_BusinessFunctions.vbs |

## Associating Object Repository file to Test

All the application User Interface objects information is captured in shared object repository file and they are placed in “Y:\SolMan\_QTP\ ObjectRepo\”

Shared object repository files are created based on the module wise (e.g. “OR\_BANKING.tsr” will have the entire UserInterface elements information specific to Bank module).

Having the entire User Interface elements in one single shared OR file, will increase the size of OR file and it will increase the memory utilization and slows down the performance during test script execution. To avoid this, we’ve split the OR files based on the modules and stored them in separate OR files. However, some of the common UI elements for all the modules are captured in a separate OR file “OR\_COMMON.tsr”.

Below are the list of modules and its specific shared object repository files:

|  |  |
| --- | --- |
| **Module Name** | **Shared Object Repository file** |
| BANK | OR\_BANKING.tsr |
| CTRL | OR\_CONTROLING.tsr |
| INVO | OR\_INVOICE.tsr |
| PROJ | OR\_PROJ.tsr |
| PURC | OR\_PURC.tsr |
| SLS | OR\_SLS.tsr |
| VEND | OR\_VENDOR.tsr |
|  | OR\_COMMON.tsr – All the common UI elements are stored in this OR file |

The common shared Object Repository “OR\_COMMON.tsr” is loaded by default in Initialization(), other module specific files are loaded at runtime based on the naming convention of UFT test script. For e.g., when Bank module test script “BANK\_BP\_004\_var01 Managing supplier payments by manual cheques” is being execution, Initialization(), will fetch the name of the module from the test script name and loads the corresponding Object Repository file.

## Batch Execution

In Batch execution, the execution of a series of test scenarios on a single batch. Batch are created as per the Modules and executed. Batches are executed in “Y:\SolMan\_QTP\Batch Run” folder.

Below are the modules and its corresponding executed batch:

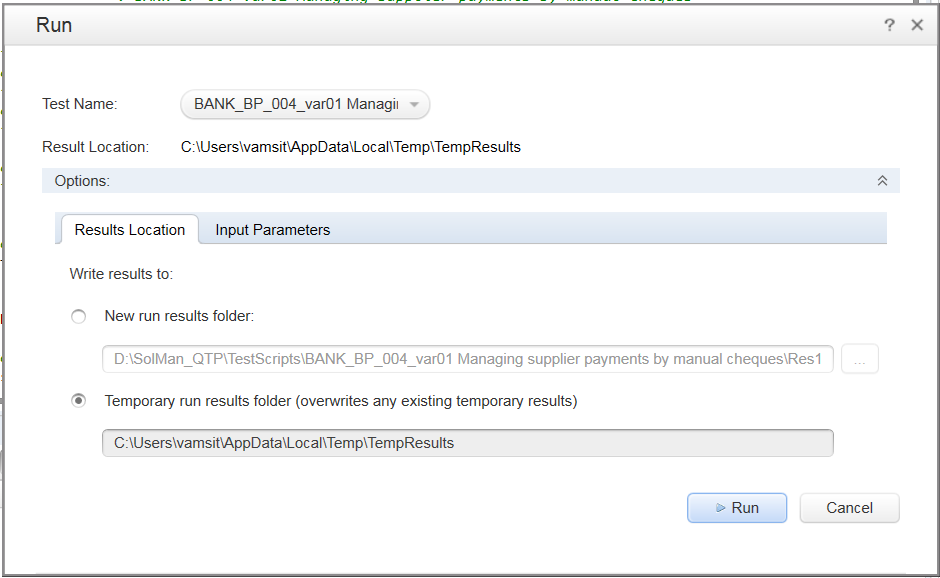
Batch is created as per modules wise, For e.g In BANK module “Bankbatch.mtb” batch is created to pulled all the bank related UFT Test scripts and placed sequentially as per business flow and executed. Other batches are also created in same pattern.

## Test Execution Report

Test execution results are stored in network folder “Y:\SolMan\_QTP\Results\”. There are three different types of reports are generated as mentioned in below sections.

### QTP / UFT execution log

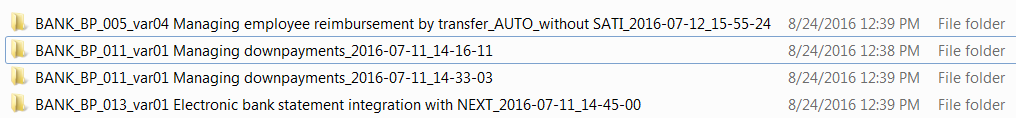
This is the default UFT tool report and is stored based on the UFT Run options as shown below:

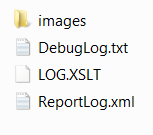
* New run results folder: UFT results are created and stored in specific location as per mentioned path in new results folder.
* Temporary run results folder: UFT results are overwrites each and every time with existing temporary results.

### HTML execution log

HTMLreport is stored in network folder “Y:\SolMan\_QTP\Results\” and it’s more user friendly report to Users as its having much details on each and every verification maintained at each level.

When UFT test script is executed, Initialization() createa a new folder under “Y:\SolMan\_QTP\Results\”, with the name of the folder same as test script name along with date & time stamp as shown below.

Once this folder is created, Initialization(), will create below list of files & folder:

Execution report details are stored in XML file “ReportLog.xml” and screenshots captured during test script executions are stored in folder “images”. To view HTML execution report, open “ReportLog.xml” file in IE browser. ReportLog.xml file is rendered as HTML using the XML stylesheet file “LOG.XSLT”.

Refer to section “[Sample HTML Execution Report](#_Sample_HTML_Execution)” in Appendix to view sample HTML execution report

### Console Log

Console Log is the UFT print view screen. During the UFT test run, all the info regarding current execution UFT statements are shown in the console log.

### Debug Log

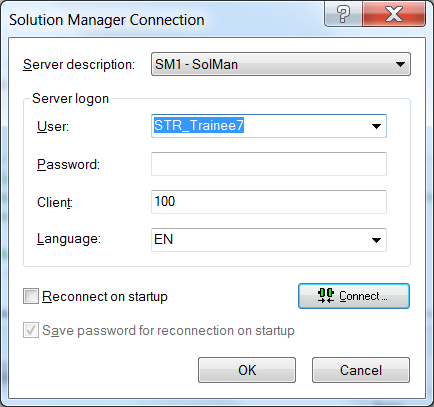
Debug Log is part of HTML report, its gives more details about the logs created in each and every action in test scripts. It stored detailed information about any failure and missing objects.

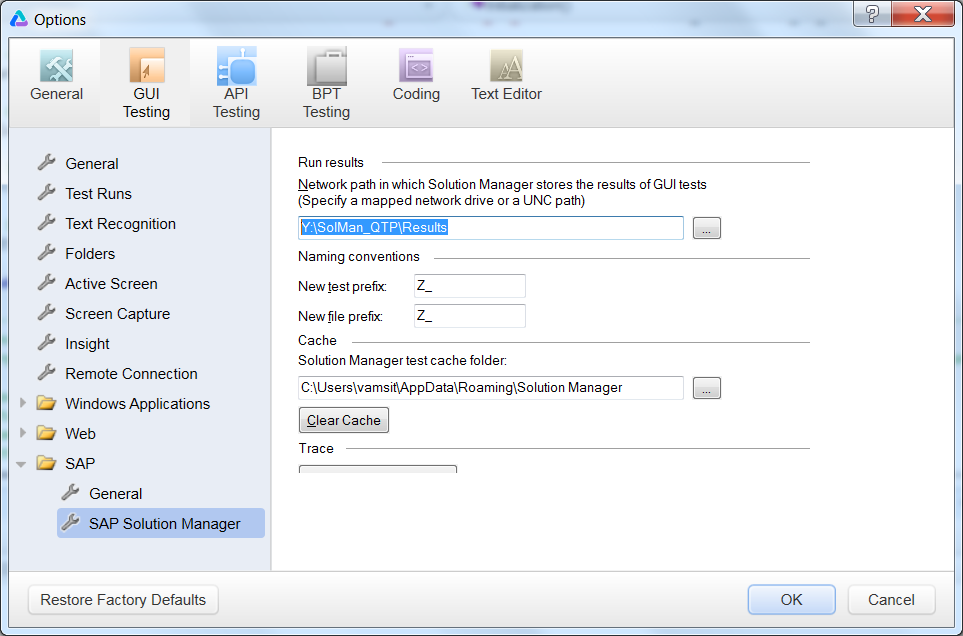
# Working with Solution Manager + UFT

To work with SOLMAN & UFT, it is required perform required configurations / settings as described in section [UFT + SolMan configurations](#_UFT+SolMan_integration) session in Appendix.

## SolMan settings in UFT

There are few basic settings to be done in UFT to work with SolMan.

To connect to SolMan from UFT in standalone mode, select menu Tools🡪Solution Manager Connection

Configure the UFT results location folder. Go to Tools🡪Options. Options window is displayed. Now select “GUI Testing” icon and navigate to “SAP 🡪 SAP Solution Manager” to view SolMan settings pane as shown below. This pane enables you to configure how UFT behaves when you are connected to SAP Solution Manager.

In the Run results section, set the location value as “Y:\SolMan\_QTP\Results”. This is the location in which UFT run results are stored when the test is run from Solution Manager

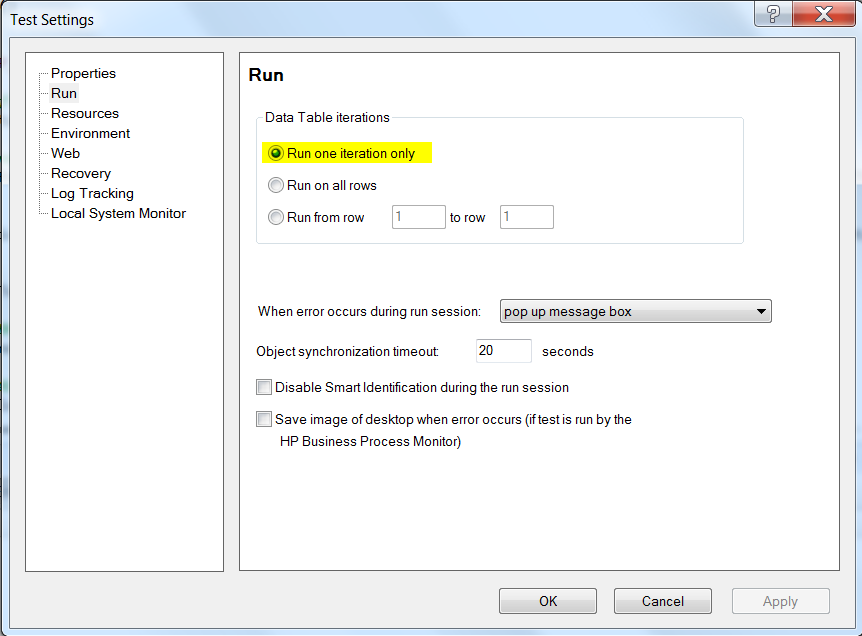
This folder must be a mapped network drive or a path in Universal Naming Convention (UNC) format.

Also, in the Naming conventions section, set the value as “Z\_” in both “New test prefix” and “New file prefix”

# UFT Settings

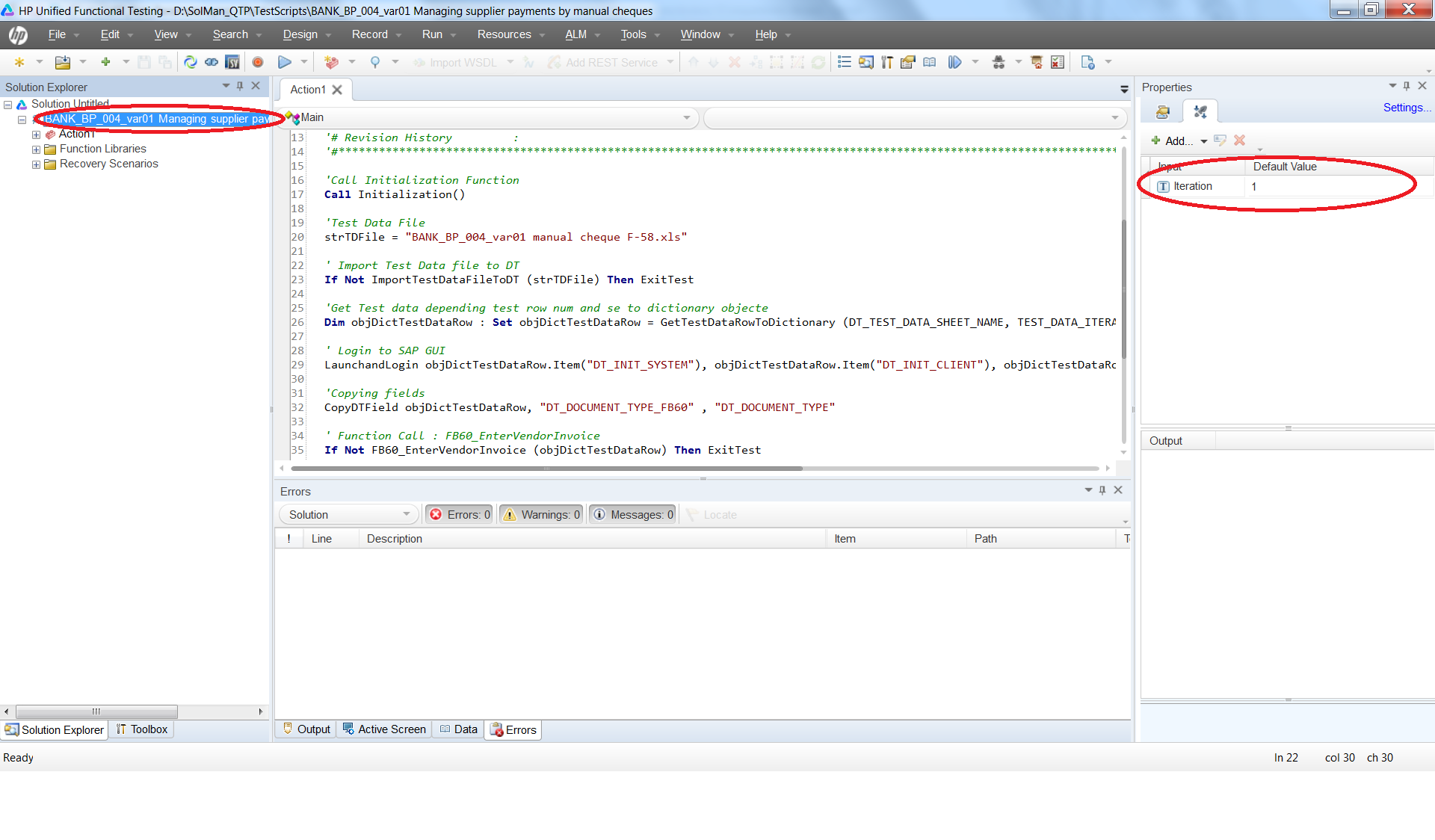
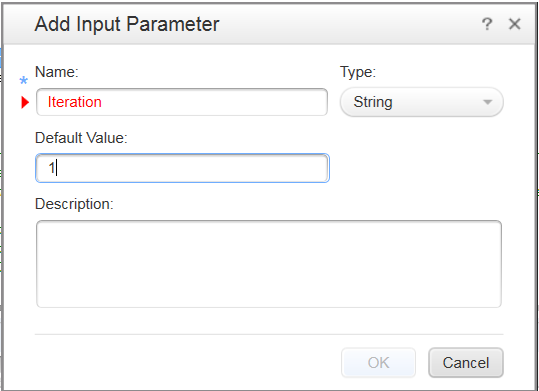
## Test Run Settings

It’s is mandatory to set the UFT test run settings to “Run one iteration only” as shown below. To open test settings, select menu File🡪Settings



## Test Iteration Settings

Test Iterations are managed using UFT test parameters. It is mandatory to create test parameter “Iteration” in every UFT test script. This parameter value is retrieved by Initialization() and stored in global variable “TEST\_DATA\_ITERATION”. This global variable value is used in test script to retrieve the corresponding Datatable row values and used as test data during test script execution. To create Test parameter, once UFT test script opened, select on the name of the test script in Solution explorer pane on the left as shown in the below picture. Test properties are displayed in the “Properties” pane on the right. Click “Add Input parameter” and given name of the parameter as “Iteration” and set default values as “1”



When executing test script from SolMan, the “Iteration” parameter value is managed using “Test Data Containers” (TDC) in SolMan. However, when executing test script outside SolMan, the value of “Iteration” test parameter has to be set from UFT test properties. For e.g., if you want to execute test script with test data in 2nd row of test data file, then set the value “Iteration” parameter value to “2”.

## UFT Add-ins

To improve the UFT tool performance, it is recommended to select only the required Add-ins as per the application under test (AUT) technology. Fine below the UFT Add-ins required in Veolia application technology stack context:

|  |  |
| --- | --- |
| **Application Interface** | **UFT Add-in** |
| SAP GUI | SAP |
| SAP Portal | Web, SAPUI5 |
| SAP GUI + SAP Portal | SAP, Web, SAPUI5 |

# Appendix

## UFT+SolMan integration

Refer the below attached document for detailed steps to integrate UFT + SolMan integration



## Sample HTML Execution Report

Extract the below Zip file to view sample HTML test execution report



## Configuring new VM



# Glossary

Provided relevant terms and abbreviations used

|  |  |
| --- | --- |
| Acronym | Description |
| UFT | HP Unified Functional Testing |
| QTP | HP Quick Test Professional |
| E2E | End-to-End |
| OR | UFT Object Repository file |
| SOLMAN | SAP Solution Manager |