



QTP - Quick Guide

Advertisements

⬅ Previous Page

Next Page ➡

QTP - Introduction

QTP stands for **QuickTest Professional**, a product of **Hewlett Packard (HP)**. This tool helps testers to perform an automated functional testing seamlessly without monitoring once script development is complete.

HP QTP uses **Visual Basic Scripting (VBScript)** for automating the applications. The Scripting Engine need not be installed exclusively as it is available part of the Windows OS. The Current version of VBScript is 5.8 which is available as part of Win 7. VBScript is NOT a object oriented language but a object based language.

Testing Tools:

Tools from a software testing context, can be defined as a product that supports one or more test activities right from planning, requirements, creating a build, test execution, defect logging and test analysis.

Classification of Tools

Tools can be classified based on several parameters. It includes,

- The purpose of the tool

- The Activities that are supported within the tool

- The Type/level of testing it supports.

- The Kind of licensing (open source, freeware, commercial)

- The tecnology used

Types of Tools:

S.No#	Tool Type	Used for	Used by
1.	Test Management Tool	Test Managing, scheduling,defect logging, tracking and analysis.	testers
2.	Configuration management tool	For Implementation,execution, tracking changes	All Team members
3.	Static Analysis Tools	Static Testing	Developers
4.	Test data Preperation Tools	Analysis and Design, Test data generation	Testers
5.	Test Execution Tools	Implementation, Execution	Testers
6.	Test Comparators	Comparing expected and actual results	All Team members
7.	Coverage measurement tools	Provides structural coverage	Developers
8.	Performance Testing tools	Monitoring the performance,response time	Testers
9.	Project planning and Tracking Tools	For Planning	Project Managers
10.	Incident Management Tools	For managing the tests	Testers

Where QTP Fits in ?

QTP is a Functional testing tool which is best suited for regression testing of the applications. QTP is a licensed/commercial tool owned by HP which is one of the most popular tools available in the market. It compares the actual and expected result and reports the results in the execution summary

QTP History and Evolution:

HP Quick Test Professional was originally owned by Mercury Interactive and it was acquired by Hp. Its original name was Astra Quick Test and later named as Quick Test Professional but the latest version is known as Unified Functional Tester(UFT).

Version History:

Now let us take a look at the version history of QTP.

Versions	Timelines
Astra Quick Test v1.0 to v5.5 - Mercury Interactive	May 1998 to Aug 2001
QuickTest Professional v6.5 to v9.0 - Mercury Interactive	Sep 2003 to Apr 2006
Hp-QuickTest Professional v9.1 to v11.0 - Acquired and Released by HP	Feb 2007 to Sep 2010
Hp-Unified Functional Testing v11.5 to v11.53	2012 to Nov 2013

Advantages:

Developing automated tests using VBScript doesn't require a highly skilled coder and relatively easy when compared other object oriented programming languages.

Easy to use, ease of navigation, results validation and Report generation.

Readily Integrated with Test Management Tool(Hp-Quality Center) which enables easy scheduling and Monitoring.

Can also be used for Mobile Application Testing.

Since it is a Hp product, the full support is provided by HP and by its forums for addressing technical issues.

Disadvantages:

Unlike Selenium, QTP works in Windows operating system only.

Not all versions of Browsers are supported and the testers need to wait for the patch to be released for each one of the major versions.

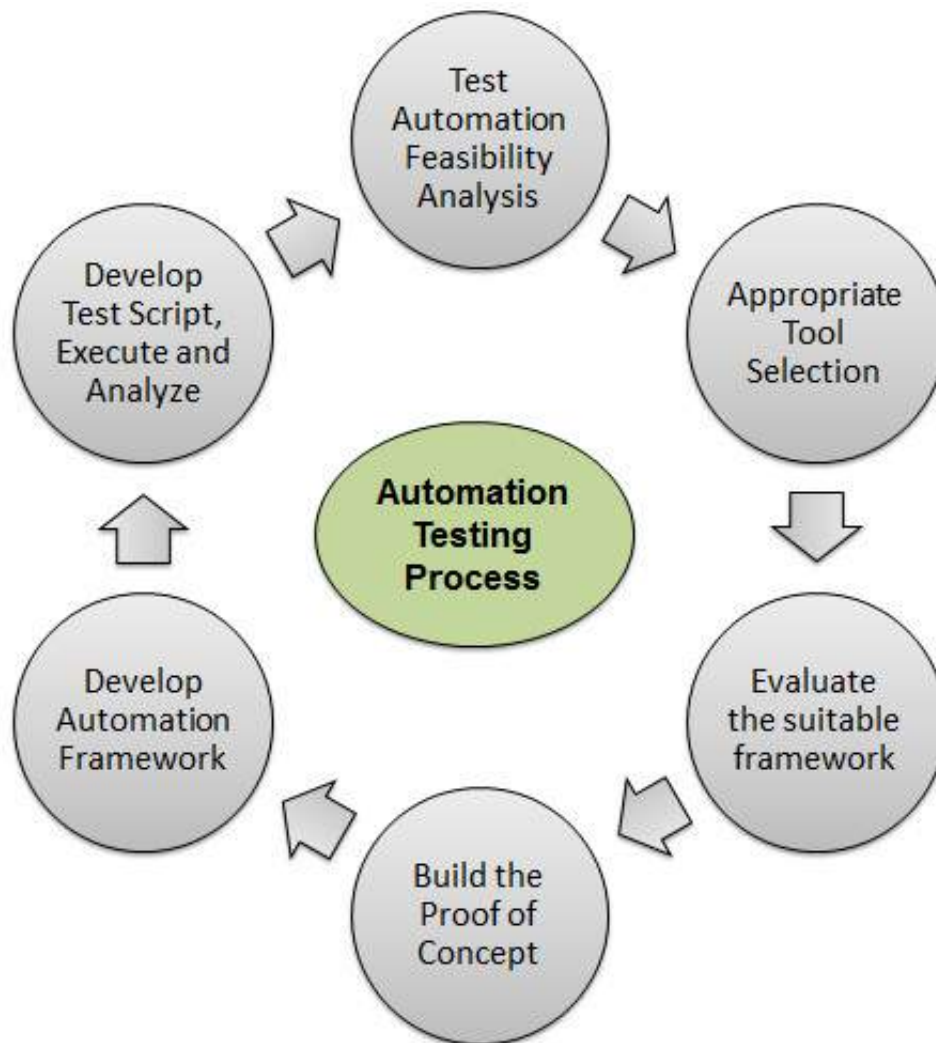
Having said that it is a commercial tool, the licensing cost is very high.

Even though scripting time is less, the execution time is relatively higher as it

puts load on CPU & RAM.

Automated Testing Process:

For any automated tool implementation, the following are the phases/stages of it. Each one of the stages corresponds to a particular activity and each phase has a definite outcome.



Test Automation Feasibility Analysis - First step is to check if the application can be automated or not. Not all applications can be automated due to its limitations.

Appropriate Tool Selection - The Next most important step is the selection of tools. It depends on the technology in which the application is built, its features and usage.

Evaluate the suitable framework - Upon selecting the tool the next activity is to select a suitable framework. There are various kinds of

frameworks and each framework has its own significance. We will deal with frameworks in detail later this chapter.

Build the Proof of Concept - Proof of Concept(POC) is developed with an end to end scenario to evaluate if the tool can support the automation of the application. As it is performed with an end to end scenario which will ensure that the major functionalities can be automated.

Develop Automation Framework - After building the POC, framework development is carried out which is the crucial step for the success of any test automation project. Framework should be build after diligent analysis of the technology used by the application and also its key features.

Develop Test Script, Execute and Analyze - Once Script development is completed, the scripts are executed, results are analyzed and defects are logged, if any. The Test Scripts are usually version controlled.

QTP Installation:


QTP is a commercial tool and trial version can be downloaded from HP site directly. Only the current version which is Unified functional testing(11.5x) is available for download. Below is the URL from where the trial version can be downloaded.

The Download URL : <http://www8.hp.com/us/en/software-solutions/functional-testing.html>

Installation Procedure :

Step 1 - Click "Trials and Demos" link and select "Hp Unified Functional Testing 11.50 CC English SW E-Media Evaluation" as shown below:



www8.hp.com/us/en/software-solutions/functional-testing.html


 For Home For Work Support Search H

Software / Application Lifecycle Management

Functional Testing

Modernize your functional testing practices.

Contact us  Trials and Demos 


Overview Related 

- HP Application Lifecycle Management 11.50 English SW E-Media Evaluation
- HP Quality Center 11.50 SW E-Media Evaluation
- HP Sprinter 11.0 SW E-Media Evaluation
- HP Unified Functional Testing 11.50 CC English SW E-Media Evaluation**

A holistic approach to functional testing

Software Prod

Step 2 - Upon Selecting "Hp Unified Functional Testing 11.50", the download wizard opens. Fill in the Personal details and click next

1 About yourself > 2 Terms of service > 3 Download 


HP Unified Functional Testing 11.50 CC English SW E-Media Evaluation

About yourself

First Name *


Last Name *

Email *

May HP contact you via email? * 

☒ Yes ☐ No

What are your plans for using this software? *

Please select one 

About your company

Company *

Phone *

Country * Area Phone *

Required * [Privacy Statement](#)

Cancel Next

Step 3 - Read the terms of use and click "NEXT".

HP Unified Functional Testing 11.50 CC English SW E-Media Evaluation

Software Download Terms of Use

READ CAREFULLY BEFORE DOWNLOADING THE SOFTWARE.

1. This license agreement (the "Agreement") states the terms between you ("You" or "Your") and Hewlett-Packard Company and its subsidiaries ("HP") for the software that You download from HP's website (the "Software"). **By downloading, copying, or using the Software You agree to this Agreement. If You do not agree to be bound by the terms of this Agreement, do not click on "I Agree" below and do not download, install, copy, or use the Software.**
2. **Terms.** This Agreement includes supporting terms and information referenced by HP, which may be software license information, additional license authorizations, software specifications, published warranties, supplier terms, open source software licenses and similar content ("Supporting Material"). Additional license authorizations are available at: www.hp.com/go/SWLicensing.
3. **Authorization.** If you agree to this Agreement on behalf of another person or entity, you warrant you have authority to do so. This Agreement will be enforceable against You and any entity for which you download, install or use the Product.

BY CLICKING I AGREE AND USING THE SOFTWARE YOU INDICATE YOUR ACCEPTANCE OF THE HP SOFTWARE DOWNLOAD AGREEMENT. IF YOU CLICK I DISAGREE AND DO NOT ACCEPT THE HP SOFTWARE DOWNLOAD AGREEMENT, THEN YOU ARE NOT GRANTED ACCESS TO THE SOFTWARE, YOU ARE NOT AUTHORIZED TO USE THE SOFTWARE, AND YOU MAY NOT DOWNLOAD THE SOFTWARE.

I DISAGREE

I AGREE

Step 4 - Download window opens. Click on "Download" Button.



HP Unified Functional Testing 11.50 CC English SW E-Media Evaluation



Please review this first:

Letter, HP UFT 11.50 English

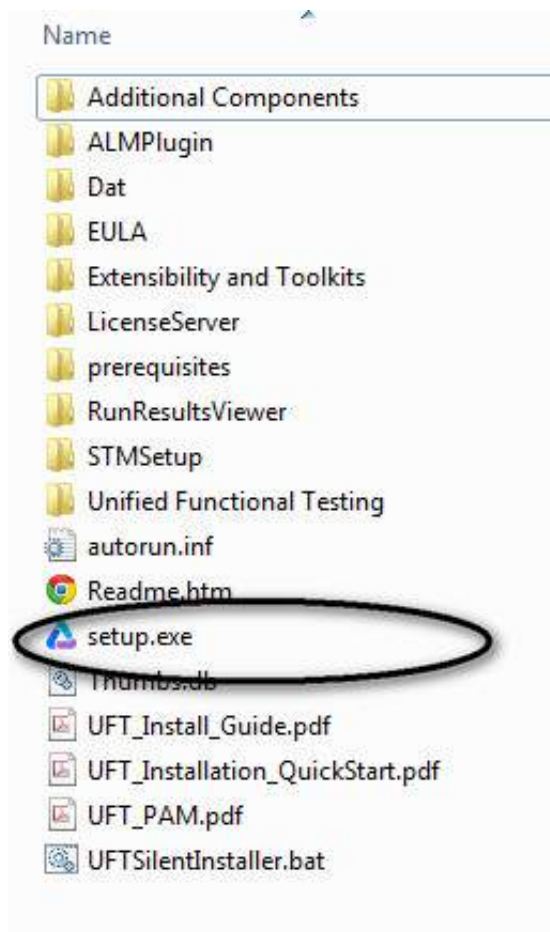
 PDF 0.08 MB

You may now download your software and supporting materials.

For License keys or to download content again, please refer to our download confirmation email sent to tshanmuganathan@gmail.com. Review our [FAQ](#) for more helpful information.

Name	File Size	Download Methods	
		Using HP Download Manager	 
Software, HP UFT 11.50 English (T6510-15 080.zip)	1754 MB	Download	
Letter, HP UFT 11.50 English	0.08 MB	Download	

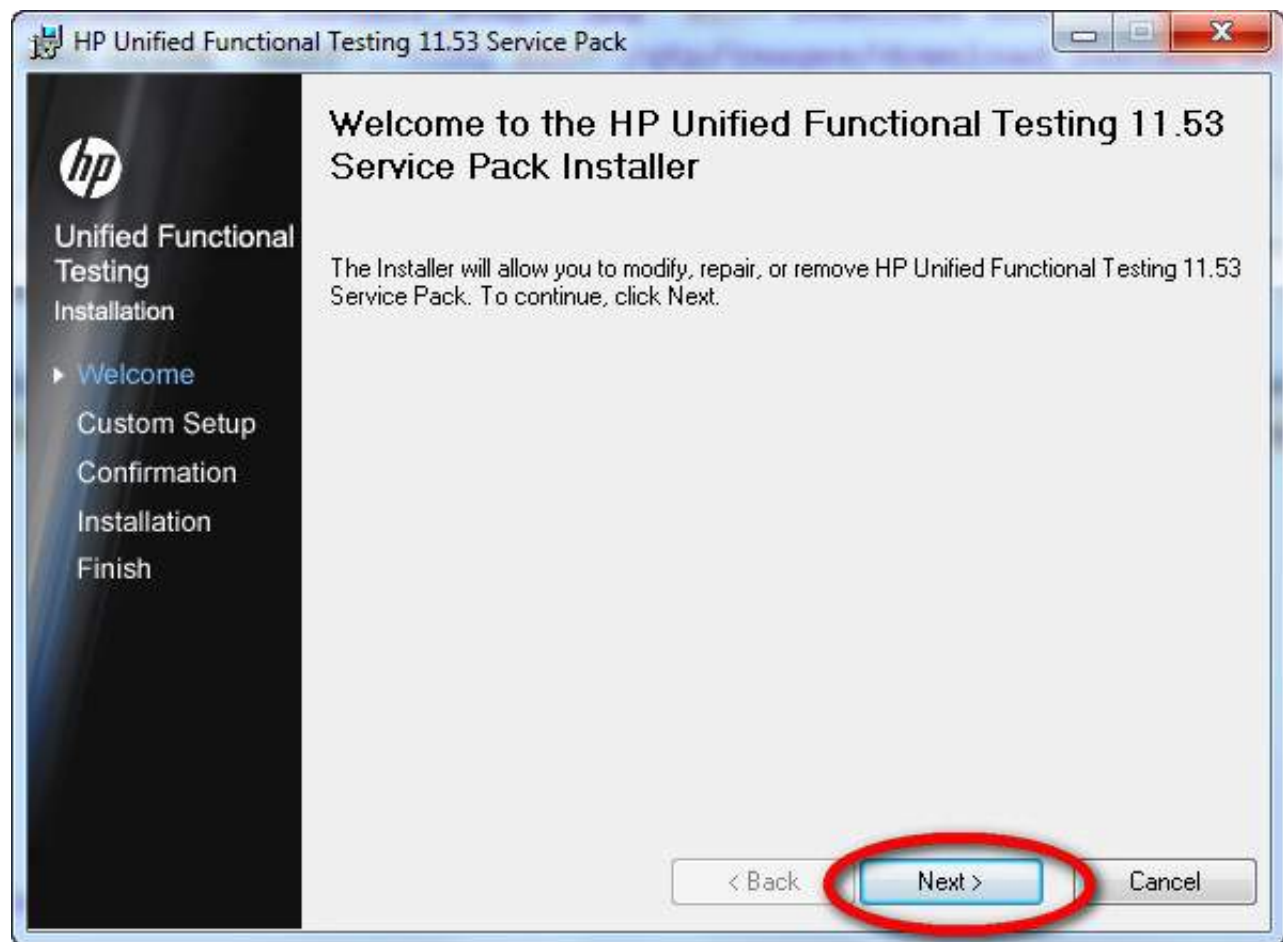
Step 5 - The downloaded file will be of the format .RAR. Now you need to unzip the archive and the folder contents would be as shown below and execute the Setup.exe.



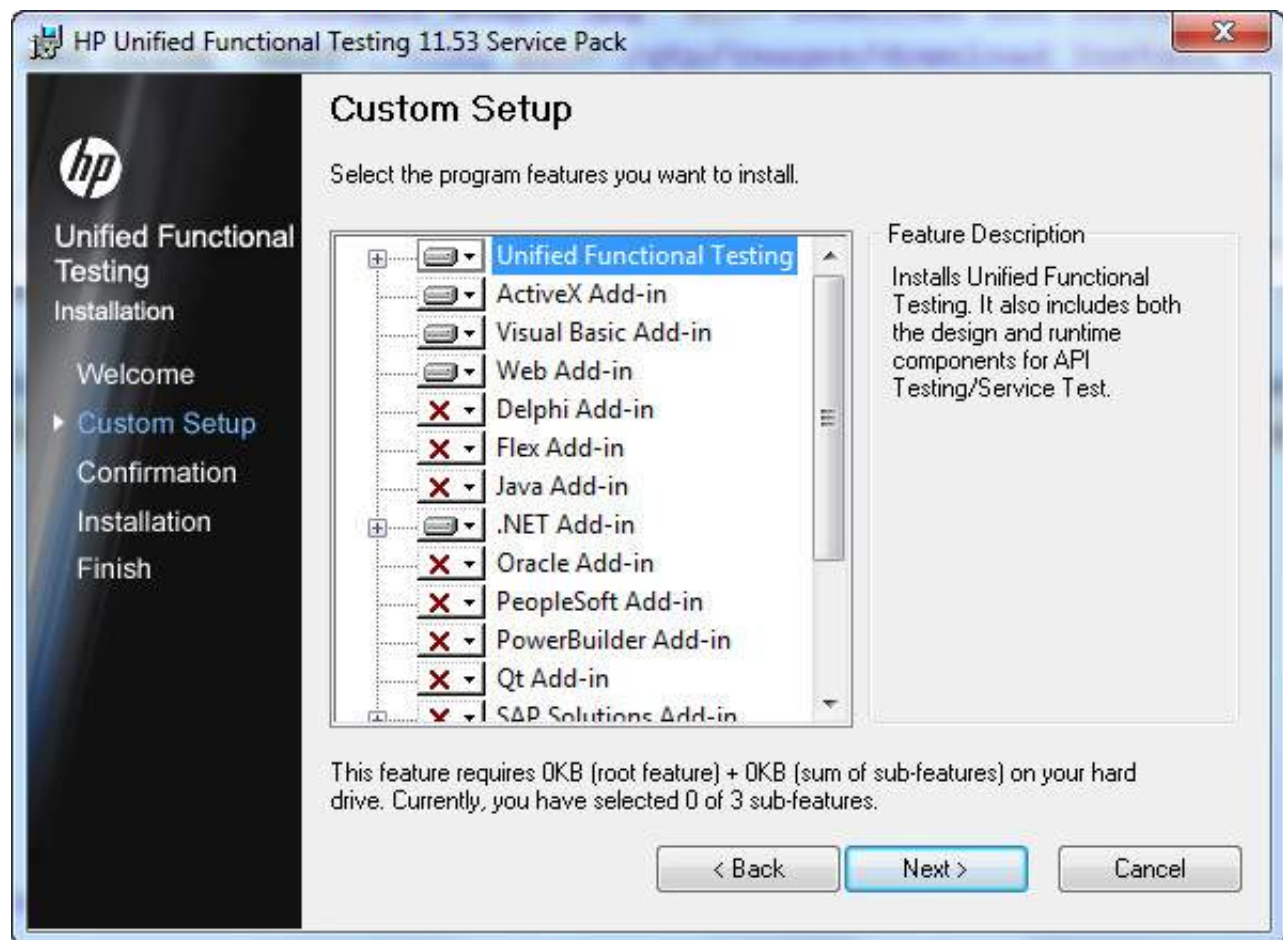
Step 6 - Upon Executing the Setup File, in order to install, select "Unified Functional Testing Set up" from the list as shown below:



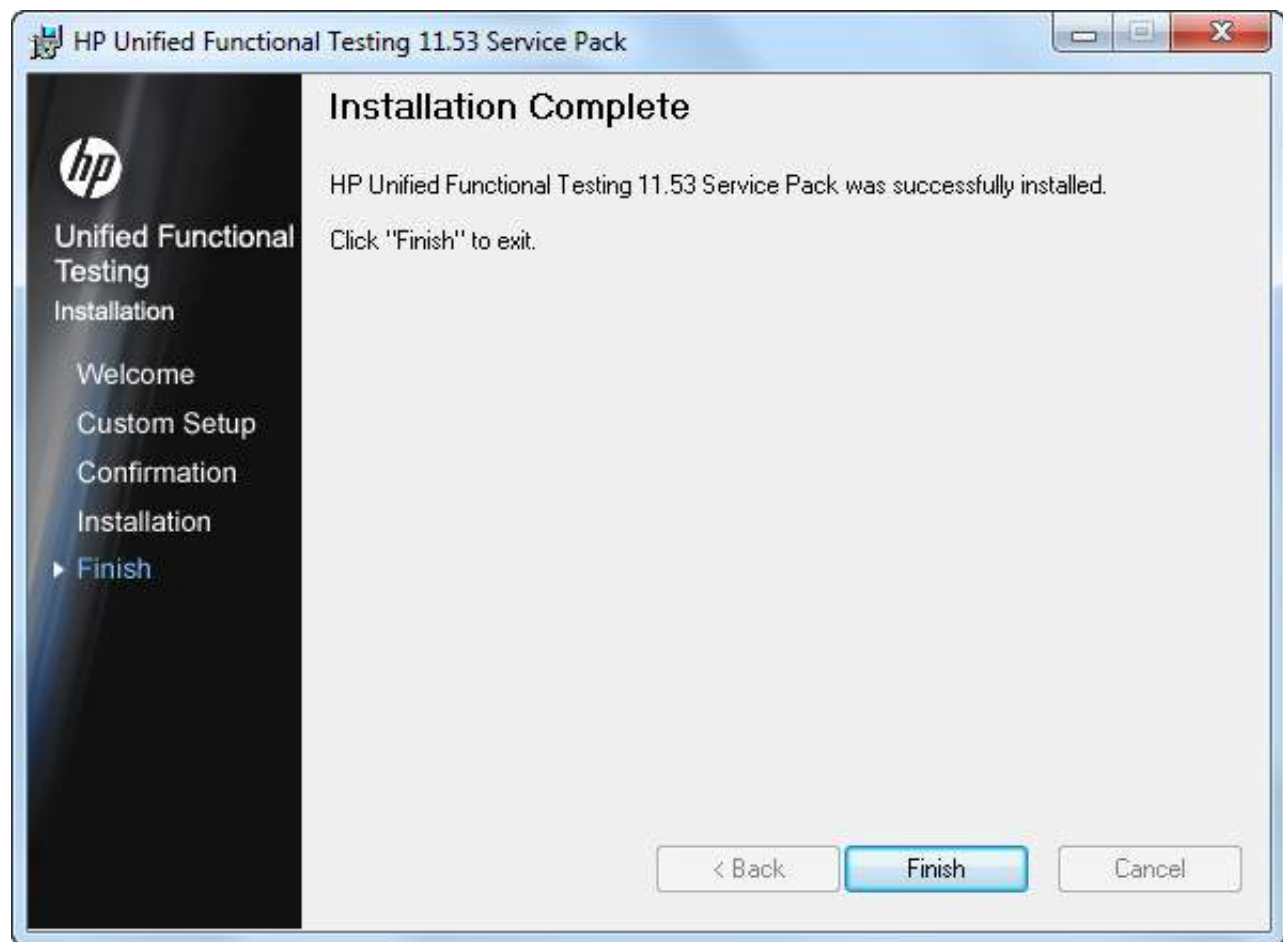
Step 7 - Then Click Next to Continue.



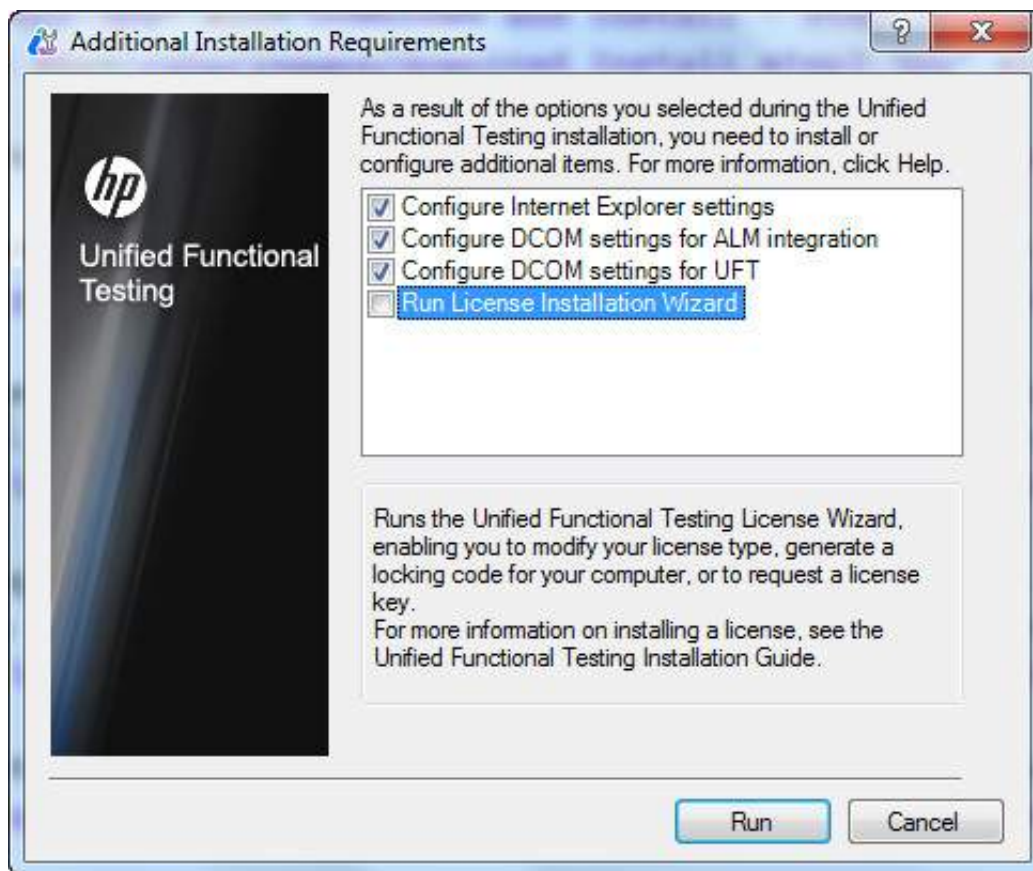
Step 8 - In the Custom Set up Window, select the plugins that are required for your automation. i.e. You Should select the plugins based on the technology of your application under test. For Example, If your application is based on .NET then you should ensure that you select .NET.



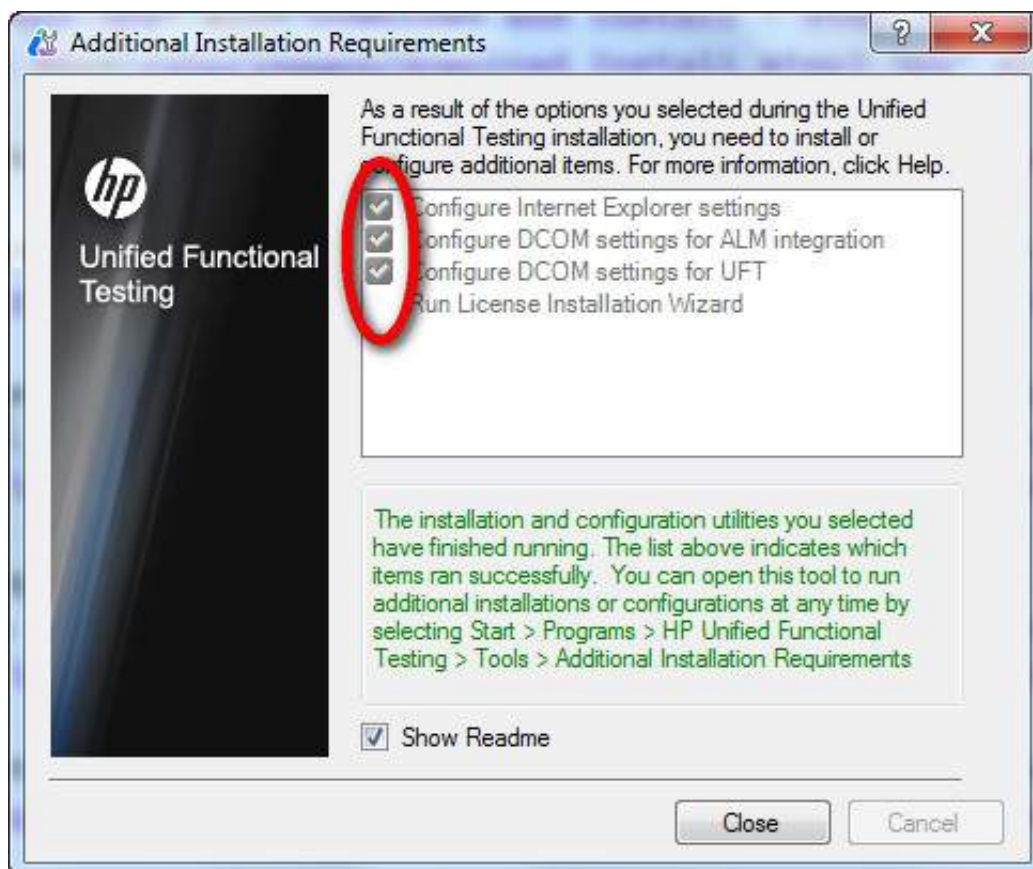
Step 9 - Upon Selecting the required plugins for Installation, Click Next and upon completion of the installation you will end up with a Finish button Window.



Step 10 - Once you complete your installation, the "Additional Installation Requirements" Dialog box opens. Select everything in the list other than "Run License Installation Wizard" and click "RUN". We Need NOT select "Run License Installation Wizard" because we are installing the trial version which by default gives a license for 30 days.

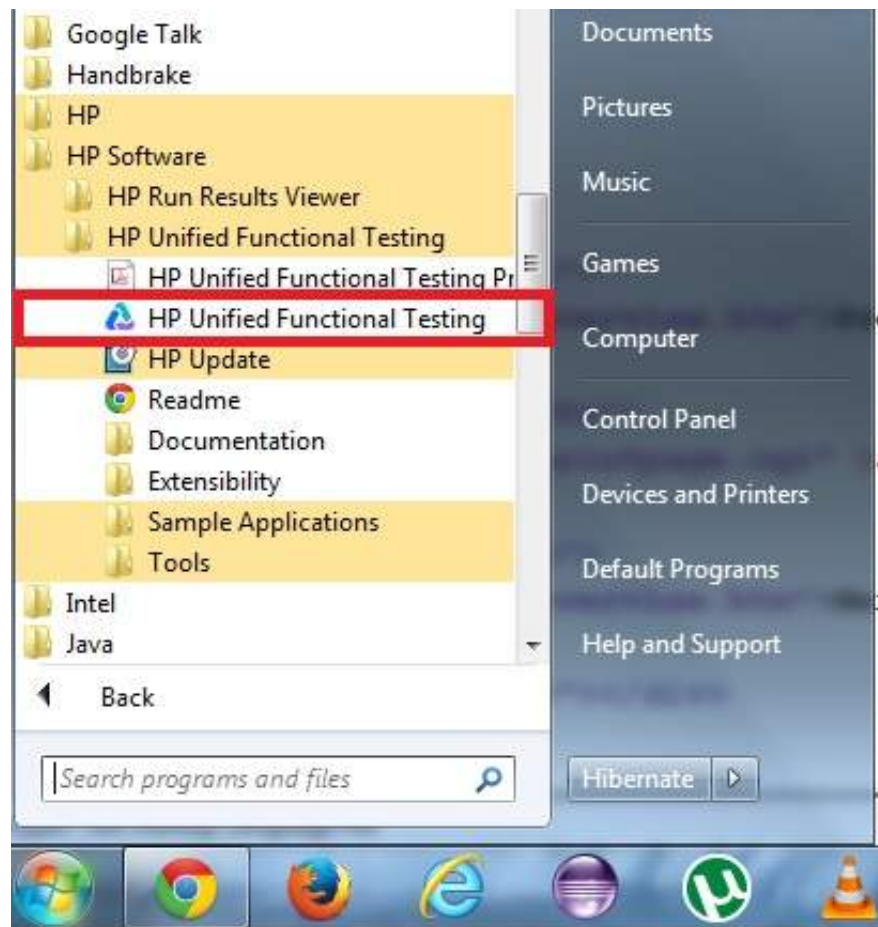


Step 11 - Upon completion of Additional Installation Requirements, a tick mark is shown which inturn states that the components are installed successfully and click close.



Launching UFT and Addins Page:

Step 1 - After Installation, application can be launched from the Start Menu as shown in the figure.

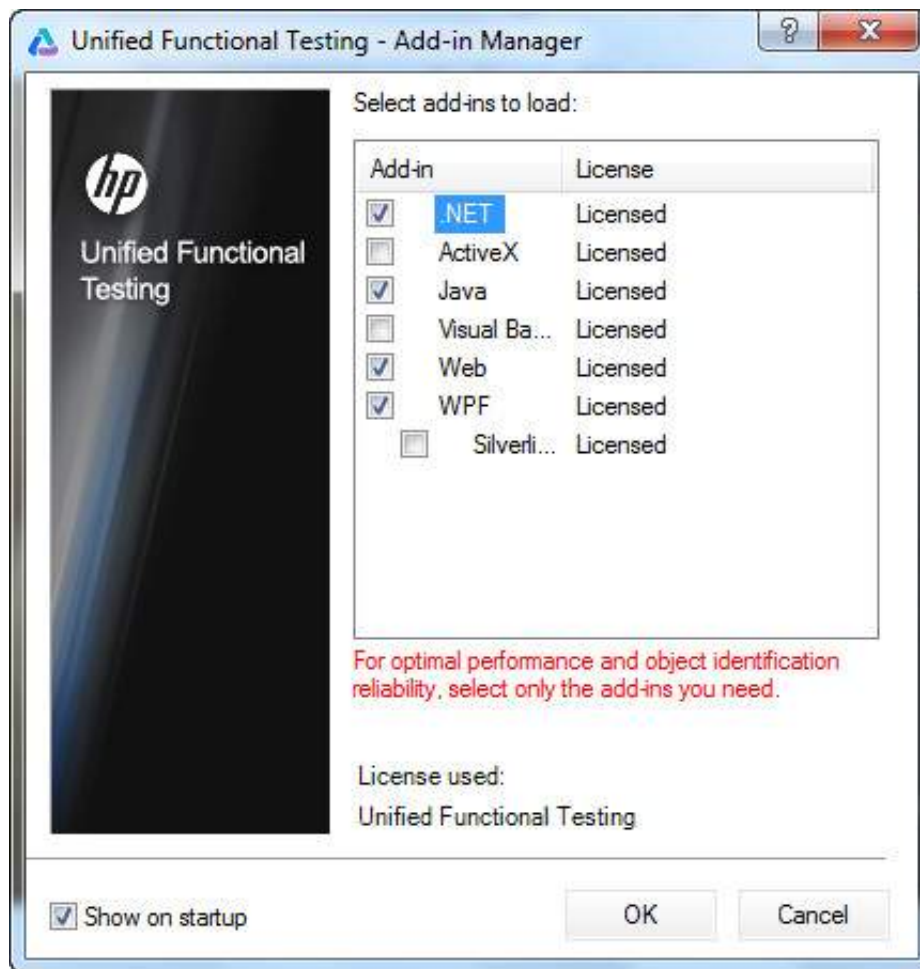


Step 2 - The License page appears. You can click on continue as we have installed the trial license

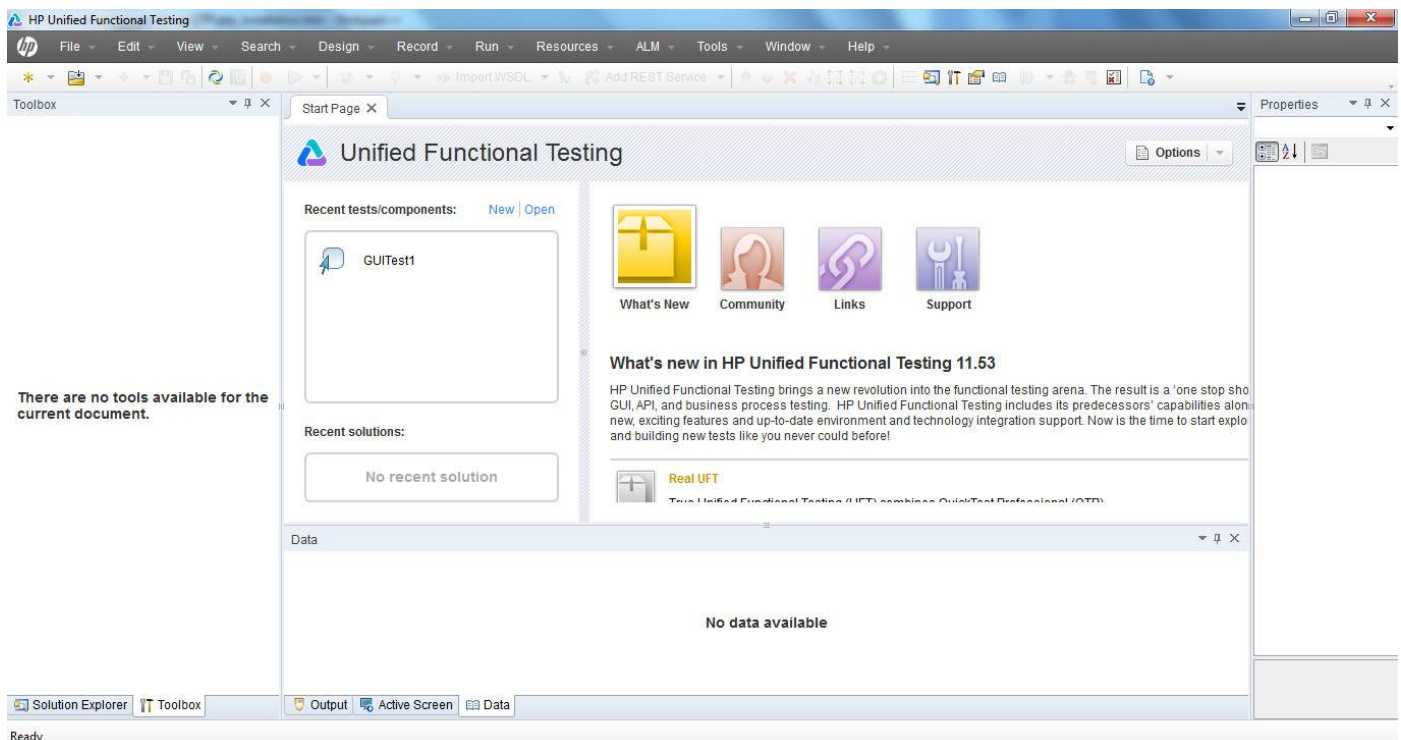


Step 3 - The Addins Dialog box opens for the user to select the required addins'

DONOT load all the addin's but just the required addins and click "Ok" button



Step 4 - Upon loading the required addins the UFT 11.5 tool opens for the user and the first glimpse of the UFT looks as shown below:

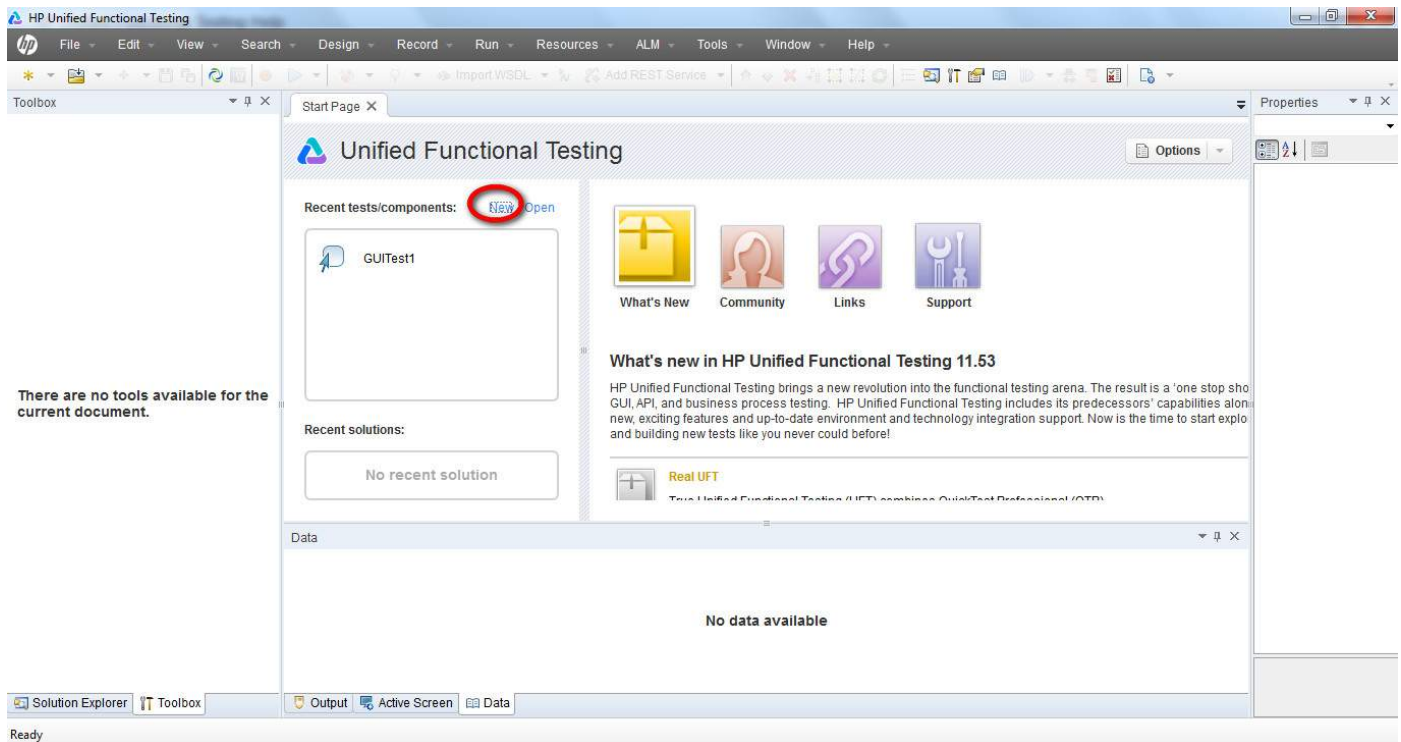


Record and Playback

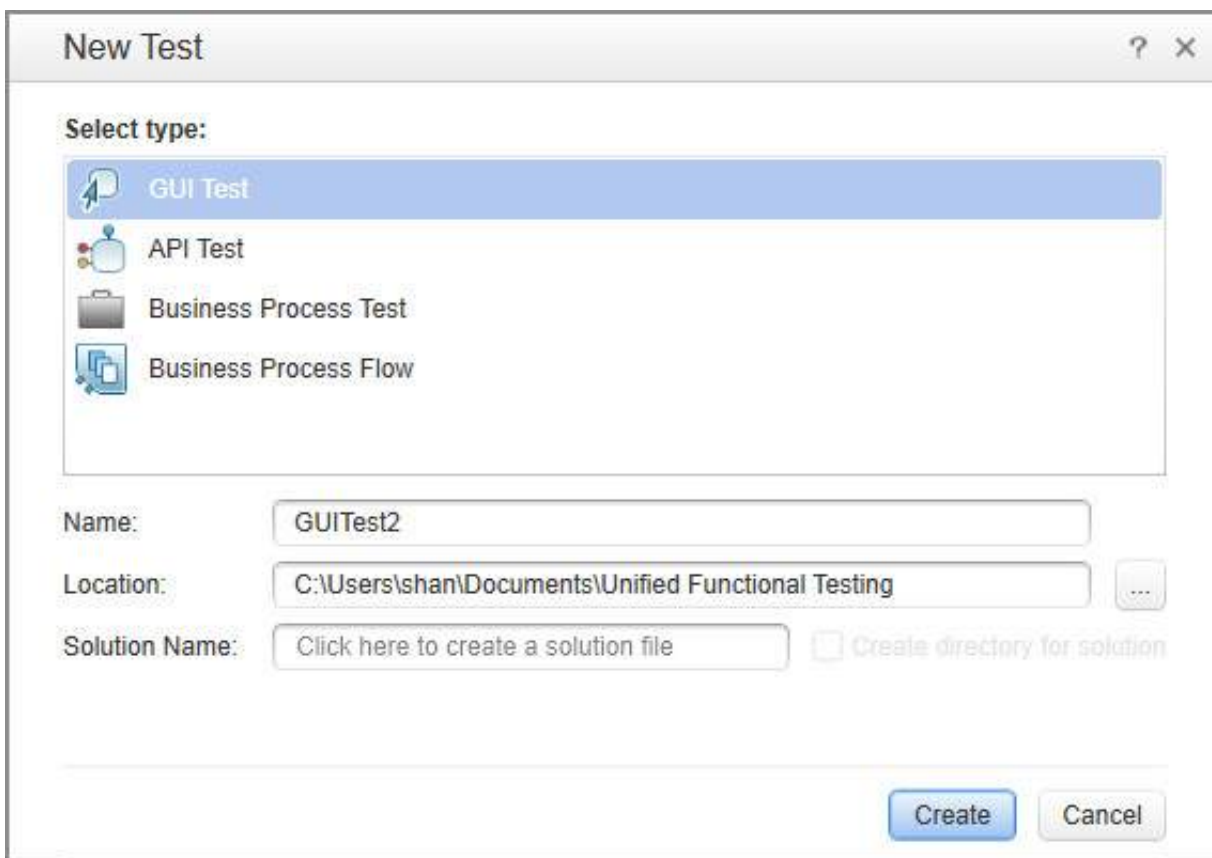
Recording a test corresponds to recording the user actions of the application under test so that UFT automatically generates the scripts that can be played back. Record and Playback can give us the first impression if the tool can support the technology or NOT if the initial settings are done correctly.

Steps for Record and Playback is as follows:

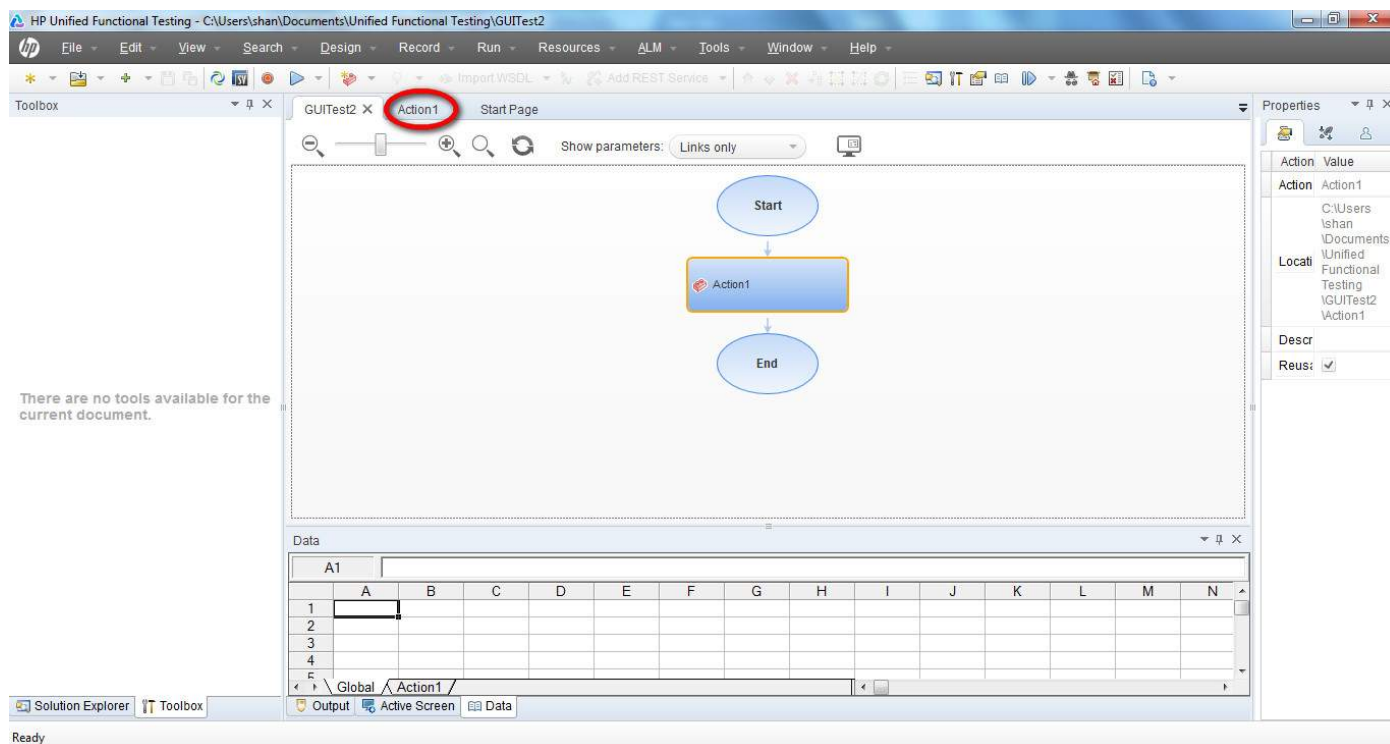
Step 1: Click on "New" test from the Start Page as shown below:



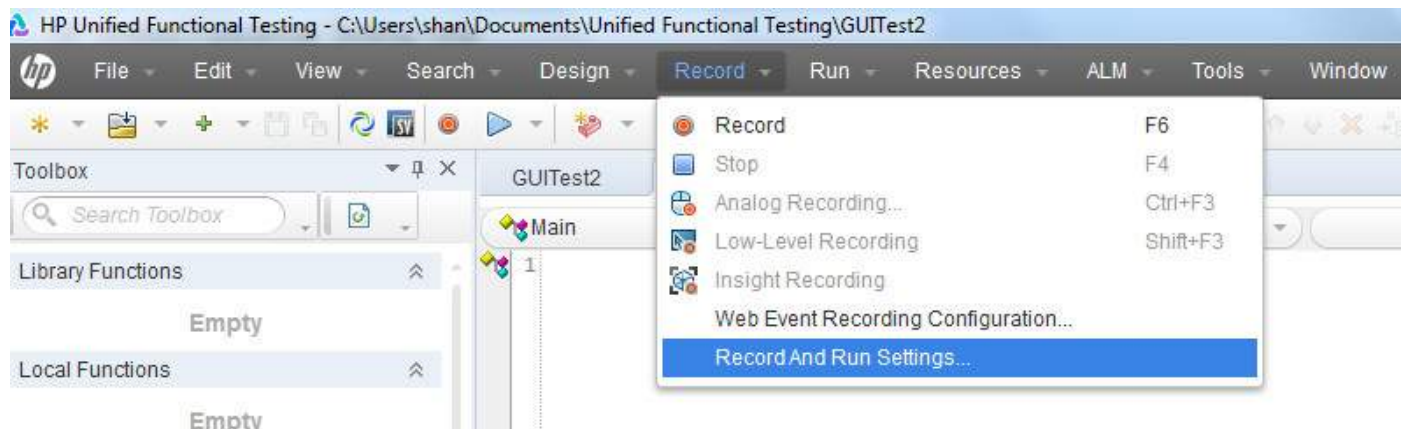
Step 2: Upon Clicking, "New" Link, the new test window opens and the user need to select the test type. Select "GUI Test", give a name for the test and also the location where it needs to be saved.



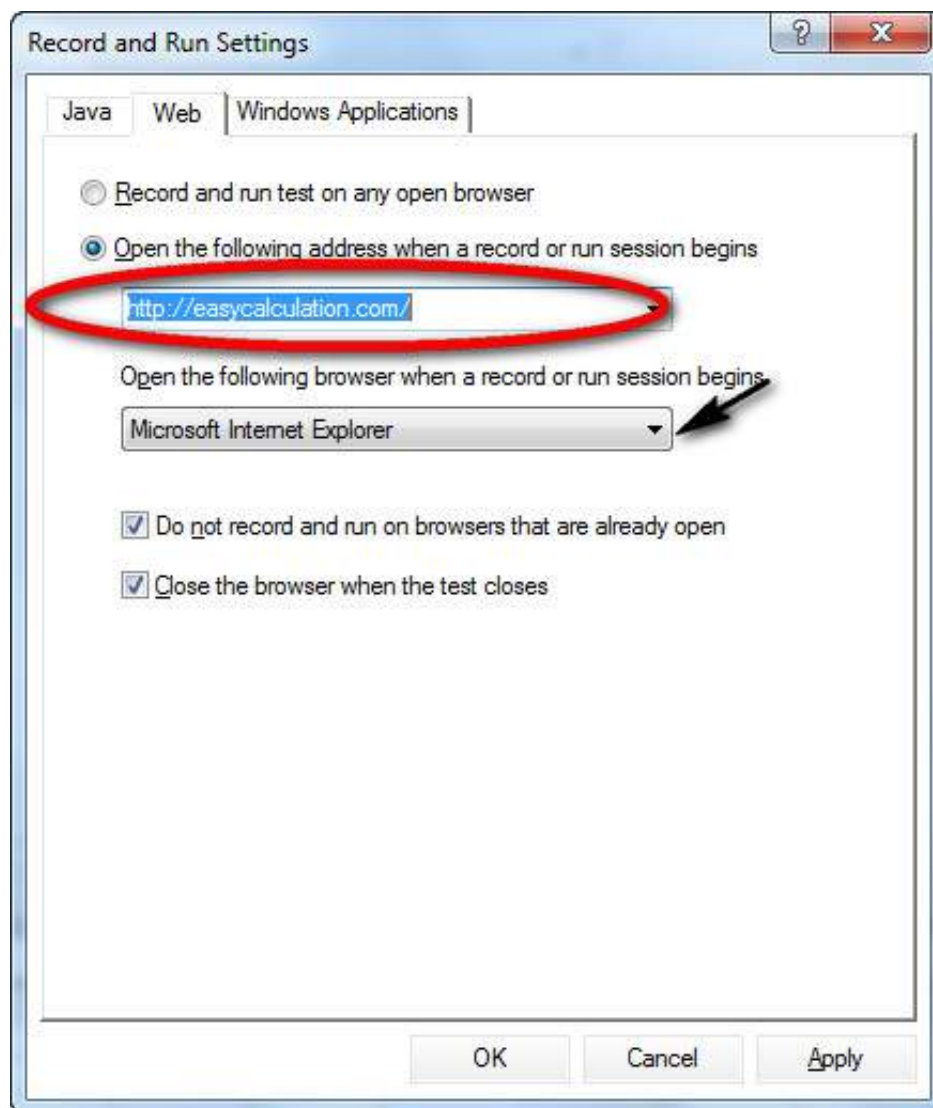
Step 3. Once a New test is created, the new test screen opens as shown below and click on "Action1" Tab which is created with 1 action by default.



Step 4. Click on "Record" Menu and select "Record and Run Settings" as shown below:

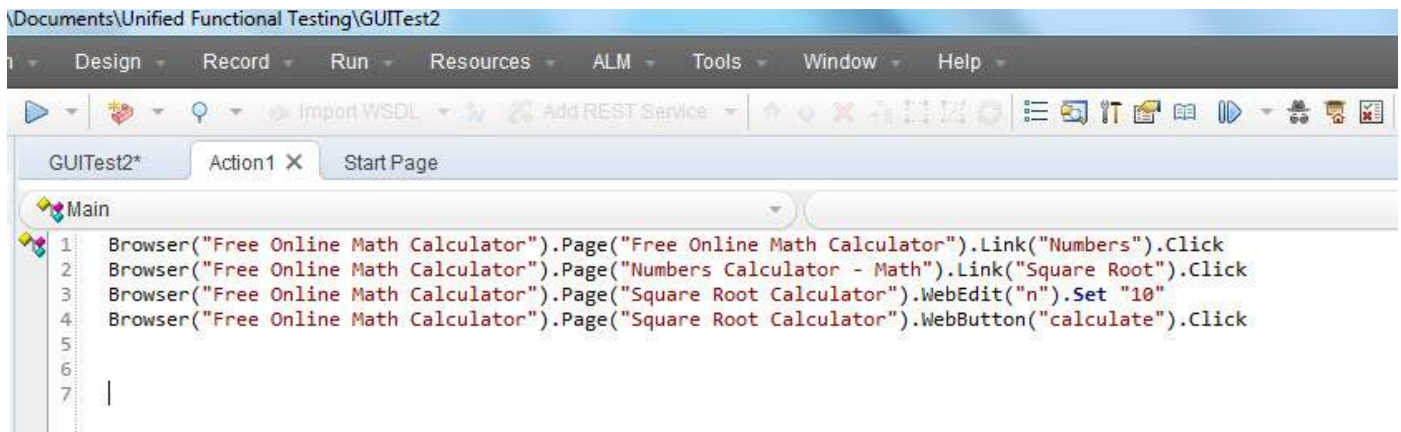


Step 5. The Record and Run Settings Dialog opens and based on the type of application, one can select i.e Web, Java, Windows Applications. For Example, We will record a Web Based Application (<http://easycalculation.com/>)

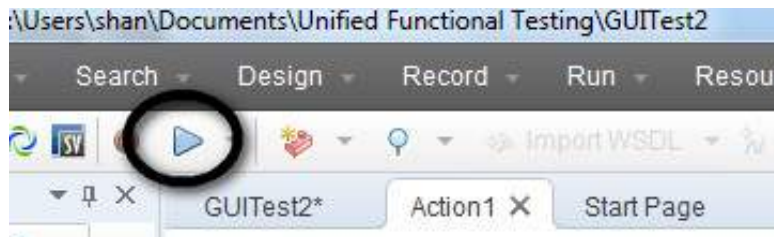


Step 6. Click Record Button, the Internet Explorer opens automatically with the webaddress <http://easycalculation.com/> as per the settings. Click "Numbers" link under "Algebra" and key in a number and hit "calculate". Upon completion of the action click "Stop" button in the record panel. You will notice that the script is

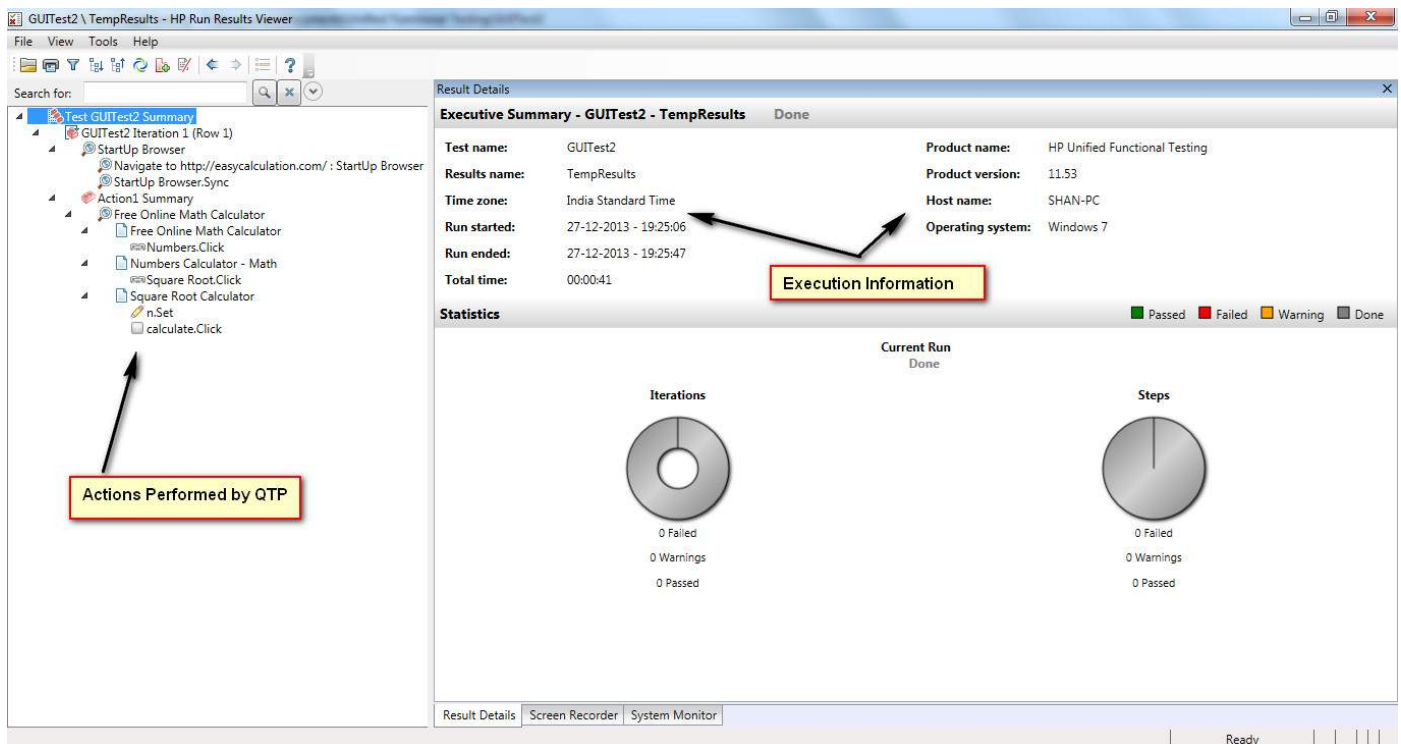
generated as shown below:



Step 7. Now playback the script by clicking on the playback button. The Script replays and result is displayed.



Step 8. The result window is opened by default which exactly shows the timestamp of execution, pass and failed steps.



Significance of Record and Playback:

1. It is used as the preliminary investigation method to verify if UFT can support

the technology/application.

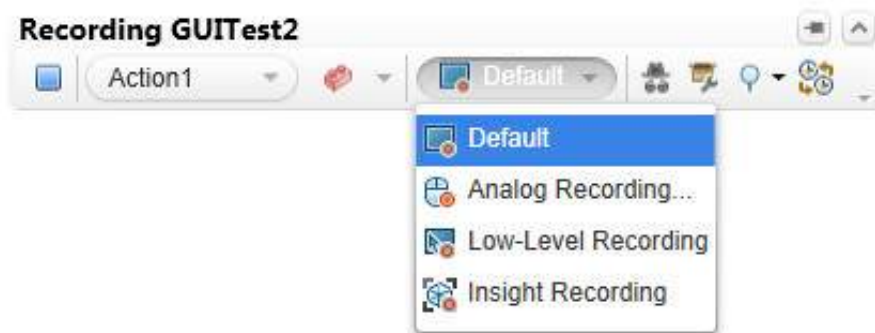
2. Used to create a test a basic functionality of an application or feature that does not require long-term maintenance.
3. It can be used for recording both mouse movements and keyboard inputs.

Modes of Recording:

1. **Normal Recording** : This is the default Recording mode that records the objects and the operations performed on the application under test.
2. **Analog Recording** : This records not only the keyboard actions but also the mouse movements relative to the screen or the application window.
3. **Low-Level Recording** : This records the exact co-ordinates of the objects independent of the fact whether UFT recognizes the object or NOT. It just records the co-ordinates, hence does NOT record mouse movements.
4. **Insight Recording** : UFT records operation based on its appearance and NOT based on its native properties.

How to Choose Recording Modes:

Upon clicking on Recording button, user can choose the recording mode from the recording pane that appears on the screen once recording starts. The selection can be made from any the ones that has been discussed above.



The Below Scenario is recorded in all the modes and see how the same action has been recorded under various circumstances.

1. Launch IE - <http://easycalculation.com/>
2. Click "Numbers" under "Algebra"

3. Click "Square Root" link
4. Enter a value to calculate the square root. Let us say 10
5. Hit Calculate

Script Recorded under Default, Analog and Low Level Recording Mode.

```
' DEFAULT RECORDING MODE
Browser("Free Online Math Calculator").Page("Free Online Math Calculator").Link("Numbers").Click
Browser("Free Online Math Calculator").Page("Numbers Calculator - Math").Link("Square Root").Click
Browser("Free Online Math Calculator").Page("Square Root Calculator").WebEdit("n").Set "10"
Browser("Free Online Math Calculator").Page("Square Root Calculator").WebButton("calculate").Click


' ANALOG RECORDING MODE
Desktop.RunAnalog "Track1"

' LOW LEVEL RECORDING MODE
Window("Windows Internet Explorer").WinObject("Internet Explorer_Server").Click 235,395
Window("Windows Internet Explorer").WinObject("Internet Explorer_Server").Click 509,391
Window("Windows Internet Explorer").WinObject("Internet Explorer_Server").Click 780,631
Window("Windows Internet Explorer").WinObject("Internet Explorer_Server").Type "10"
Window("Windows Internet Explorer").WinObject("Internet Explorer_Server").Click 757,666
```

The Recordings using insight recording mode will be as shown below:

```
' INSIGHT RECORDING MODE

Browser("Free Online Math Calculator").InsightObject( ).Click
Browser("Free Online Math Calculator").InsightObject( ).Click
Browser("Free Online Math Calculator").InsightObject( ).Click
Browser("Free Online Math Calculator").InsightObject( ).Click
Window("Windows Internet Explorer").WinObject("Internet Explorer_Server").Type "10"
Browser("Free Online Math Calculator").InsightObject( ).Click
Browser("Free Online Math Calculator").InsightObject( ).Click
```



Object Repository:

Object Repository is a collection of object and properties with which QTP will be able to recognize the objects and act on it. When a user records a test, the objects and its properties are captured by default. Without understanding objects and its properties, QTP will NOT be able to play back the scripts.

Click on each one of the below topics to know more about Object Repository and its

associated features.

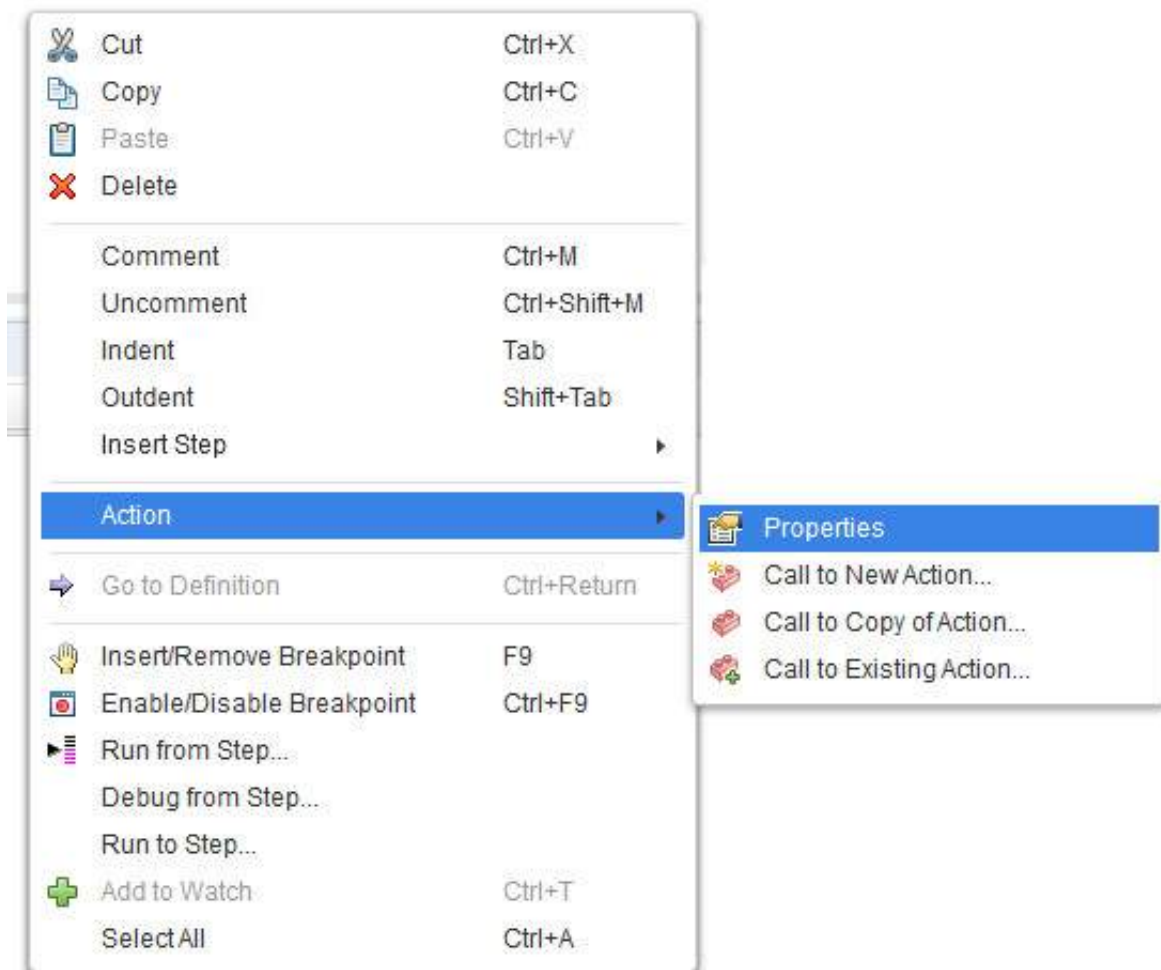
Topic	Description
Object Spy and its Features	To Understand the usage of object Spy and its associated functionalities.
Working with Object Repository	Adding,Editing, Deleting Objects from a Object Repository and its associated functionalities.
Types of Object Repository	Deals with Shared Object and Local Object Repository and their context with respect to scripting.
User-defined Objects	Deals with the circumstances to use the User-Defined Objects.
Object Repository in XML	Deals with coverting OR's to XML and how to use the object Repository as XML.
Comparing and Merging OR	Operations such as Compare OR', Merge OR's to effectively work with Object Repository.
Ordinal Identifiers	Circumstances when the ordinal identifiers are used and their advantages.
Child Objects	Using Child Objects for effective scripting

Actions:

Actions helps testers to divide scripts into groups of QTP statements called actions. Actions are similar to functions in VBScript, however there are few differences. By Default QTP creates a test with 1 action.

Actions	Functions
Actions are inbuild feature of QTP.	VBScript Functions are supported by both VBScript and QTP.
Actions parameters are passed byvalue only.	Function parameters are passed either by byvalue or byref.
Actions have extension .mts	Functions are saved as .vbs or .qfl
Actions may or maynot be reusable.	Functions are always reusable.

The properties of the action can be accessed by Right Clicking on the Script Editor Window and Selecting "Properties".



Action properties contains following information

Action Name

Location

Reusable Flag

Input Parameters

Output Parameters

Types of Actions:

There are three types of actions:

Non-reusable action - An action that can be called only in that specific test in which it has been designed and can be called only once

Reusable action - An action that can be called multiple times any test in which it resides and can also be used by any other tests

External Reusable action - It is a reusable action stored in another test. External actions are read-only in the calling test, but it can be used locally with the editable copy of the Data Table information for the external action

Working with Actions:

There are three options to insert an action. Click on each one of those to know more about the selected type of action.

Action Type	Description
Insert Call to New Action	Inserts a New Action from the existing action
Insert Call to Copy of Action	Inserts a copy of an existing action
Insert Call to Existing Action	Inserts a call to existing re-usable action

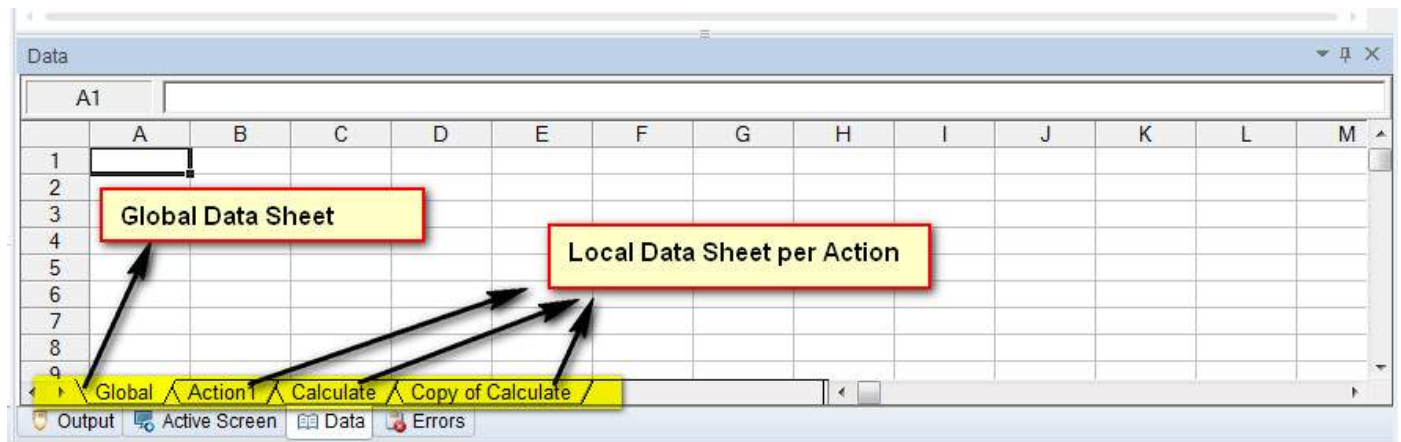
Datatables:

A DataTable, similar to Microsoft Excel helps testers to create data driven test cases that can be used to run an Action multiple times. There are two types of Datatables.

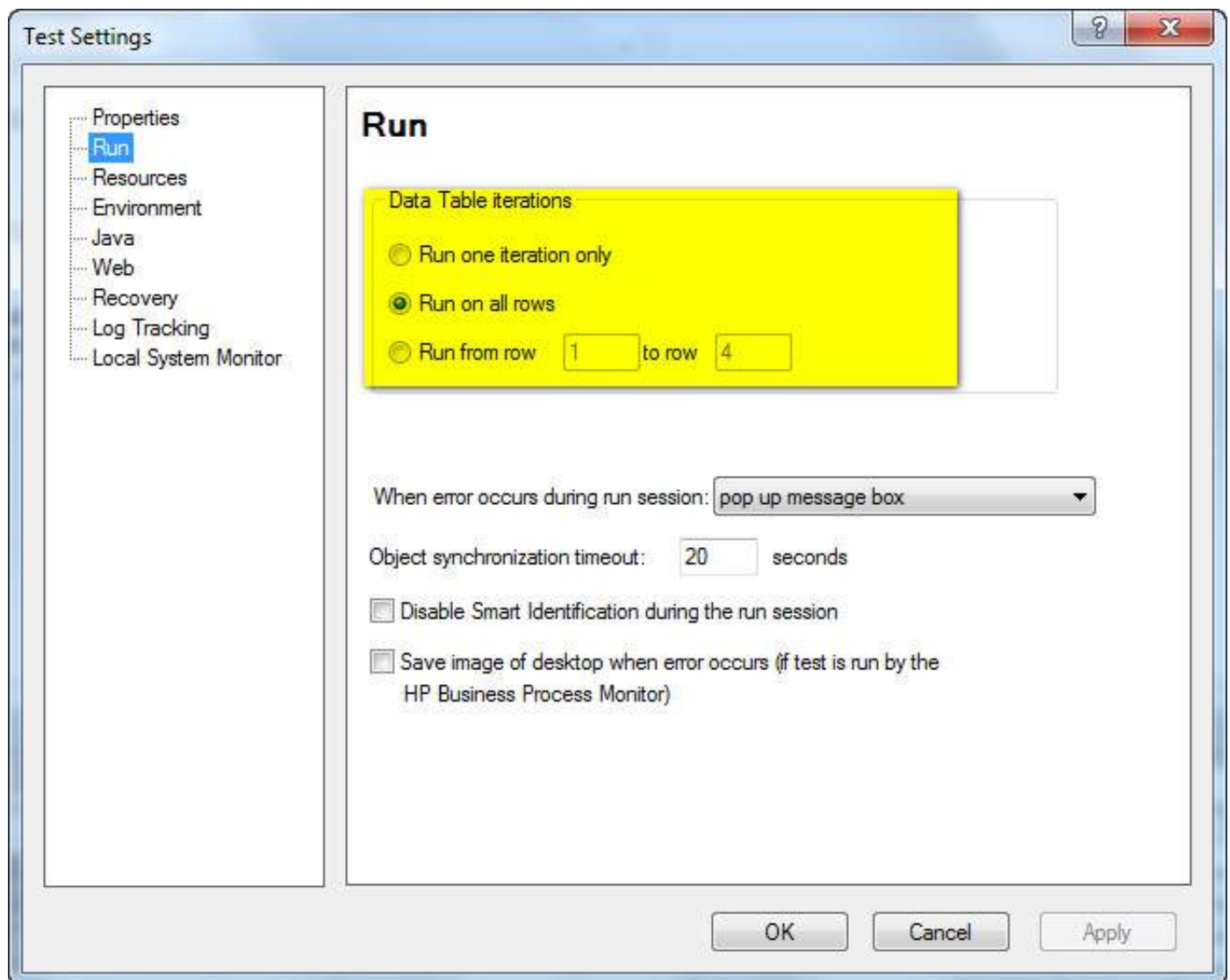
Local Data Table - Each action has its own private data table also known as local data table which is can also be accessed across actions.

Global Data Table - Each test has one global data sheet that is accessible across actions.

The Data sheet can be accessed from the "Data" Tab of QTP as shown below:



To execute a test case for some specified number of iterations, one can set the iterations of global data table in the Test Settings dialog, that can be accessed using File -> Settings -> Run(Tab) as shown below:



Example:

For Instance, if user want to parameterize "compound Interest" of

"http://easycalculation.com/" that can be accessed using "http://easycalculation.com/compound-interest.php". The Parameters can be created as shown below. Most of the functionalities of Excel can be used in Data table as well.

Data			
	A3	1322	
	Principal	Rate	Time
1	232	3.26	4
2	2556	7%	5
3	1322	6.50%	6
4	32.21	3.32%	4.4
5			
Global / Action1 / Calculate			

Data Table Operations:

There are three types of objects to access Data Table. Data Table Operations can be well understood by traversing through the below link:

Object Type	Description
Data Table Methods	Gives Detailed information about the data table methods.
DTPParameter Object Methods	Gives Detailed information about the DTPParameter methods.
DTSheet Object Methods	Gives Detailed information about the DTSheet methods.

What are CheckPoints?

Checkpoints, as the name says it all, it refers to a validation point that compares the current value for specified properties or current state of an object with the expected value which can be inserted at any point of time in the script.

Types:

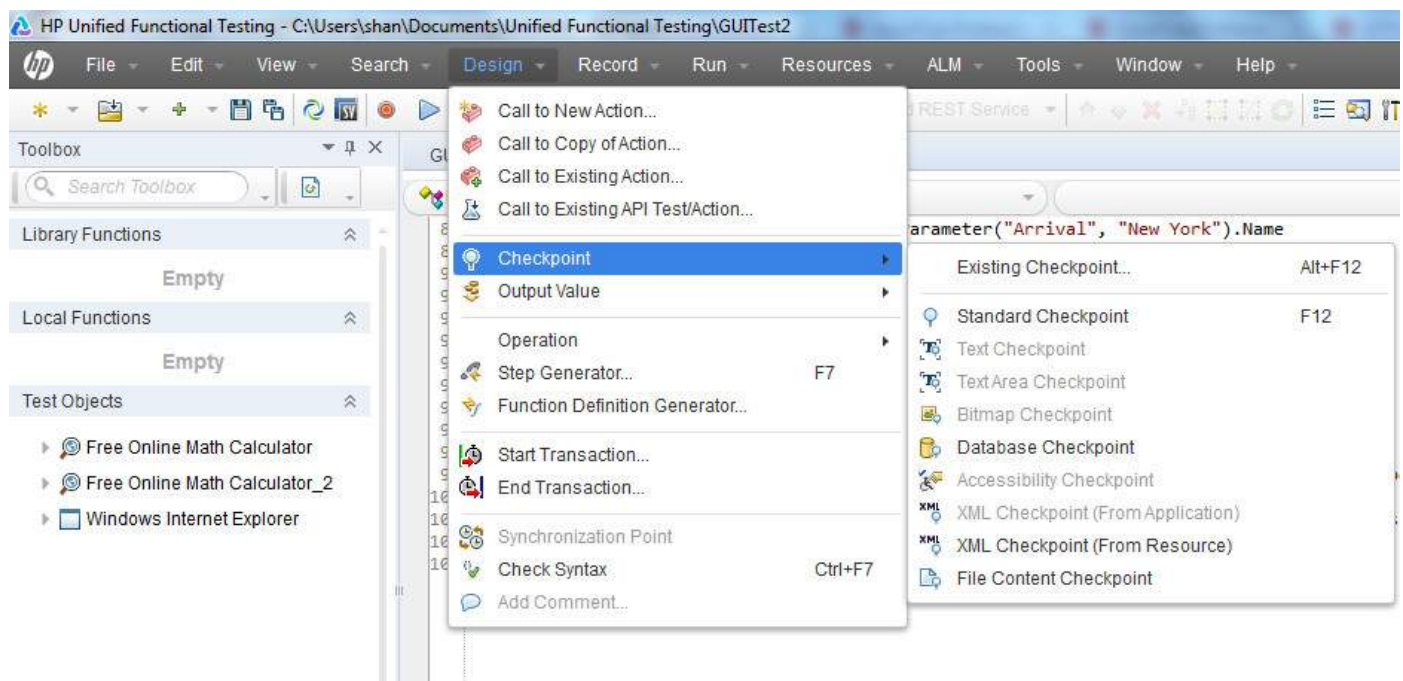
Type	Description
Standard Checkpoint	Verifies the property values of an object in application under test and supported by all add-in environments.
Bitmap Checkpoint	Verifies an area of your application as a bitmap

File Content Checkpoint	Verifies the text in a dynamically generated or accessed file such as .txt,.pdf
Table Checkpoint	Verifies the information within a table. Not all environments are supported.
Text Checkpoint	Verify if the text that is displayed within a defined area in a Windows-based application, according to specified criteria.
Text Area Checkpoint	Verifies if the text string is displayed within a defined area in a Windows-based application, according to specified criteria.
Accessibility Checkpoint	Verifies the page and reports the areas of the Web site that may not conform to the World Wide Web Consortium (W3C) Web Content Accessibility Guidelines
Page Checkpoint	Verifies the characteristics of a Web page. It can also check for broken links.
Database Checkpoint	Verifies the contents of a database accessed by the application under test.
XML Checkpoint	Verifies the content of the .xml documents or .xml documents in Web pages and frames.

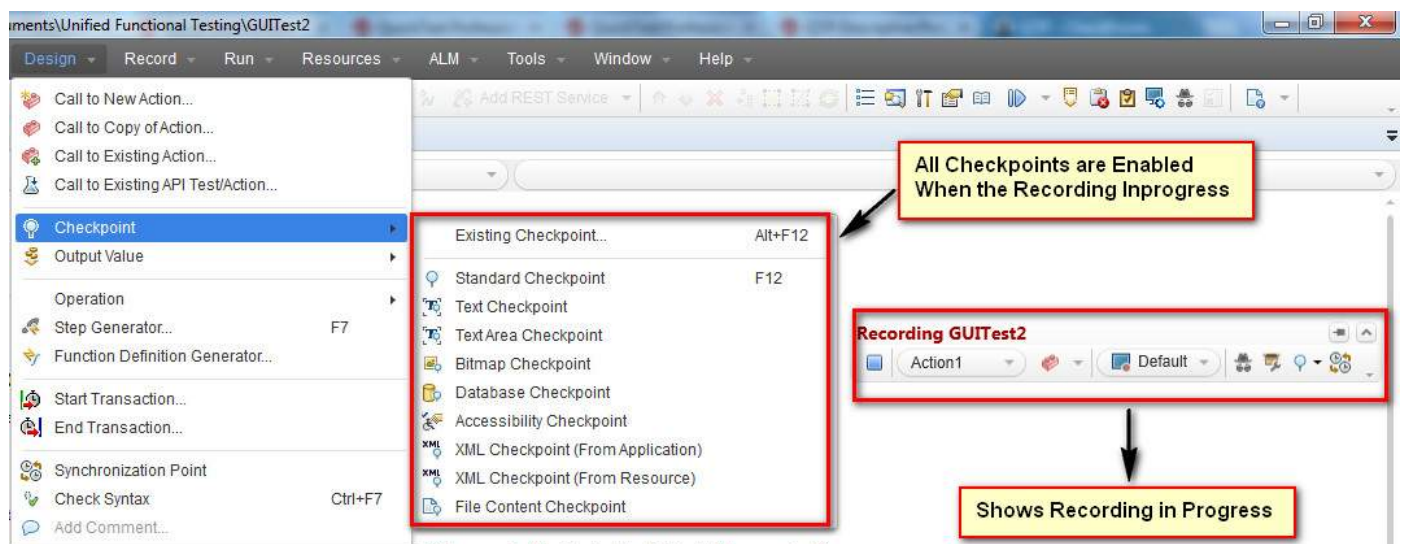
Inserting CheckPoint

When the user wants to insert a checkpoint, one has to ensure that most of the checkpoints are supported during the recording sessions only. Once the user stops recording, checkpoints are NOT enabled.

Below is the checkpoint menu, when the user is NOT in the recording mode.



Below is the checkpoint menu, when the user is in the recording mode.



Example:

The checkpoints are added for the application under test - "http://easycalculation.com/"

```
' 1. Inserted Standard Checkpoint
Status = Browser("Math Calculator").Page("Math Calculator").Link("Numbers").Check CheckPoint("Numb

If Status Then
    print "Checkpoint Passed"
Else
    Print "Checkpoint Failed"
End if

' 2. Inserted BitMap Checkpoint
imgchkpoint = Browser("Math Calculator").Page("Math Calculator").Image("French").Check CheckPoint(
```

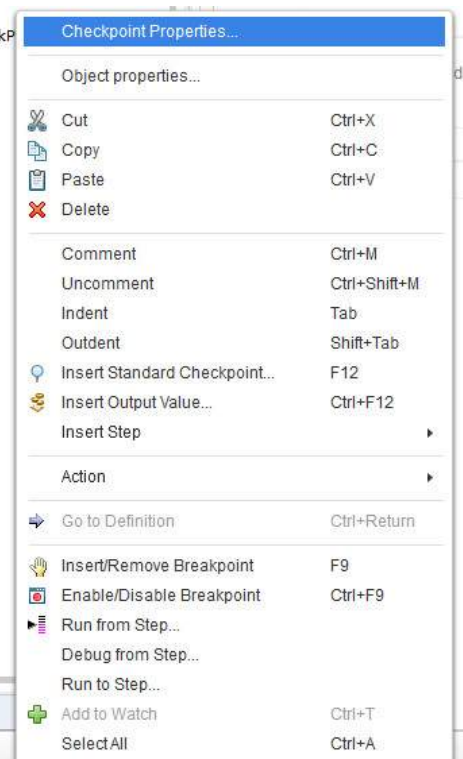


```
If imgchkpoint Then  
    print "Checkpoint Passed"  
Else  
    Print "Checkpoint Failed"  
End if
```

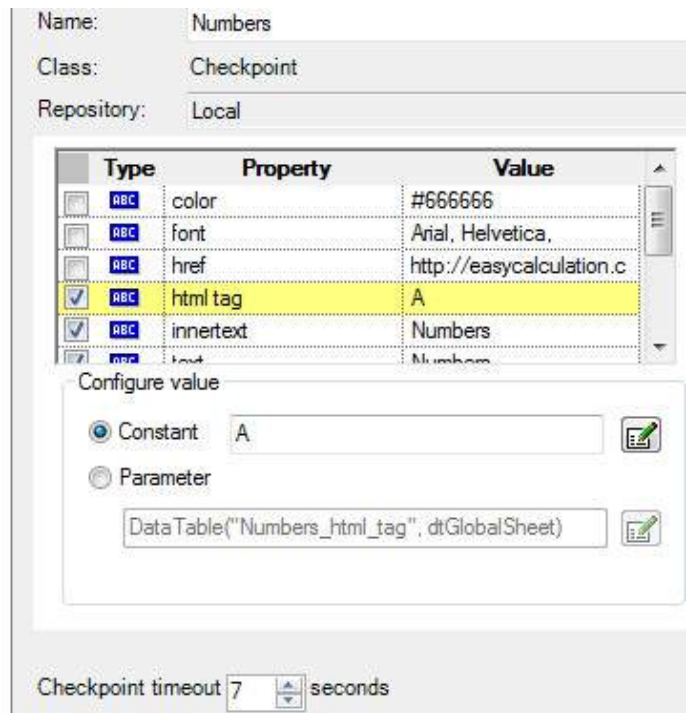
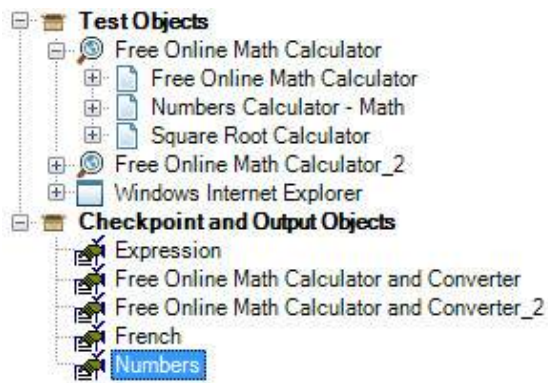
Viewing Checkpoint Properties

After Inserting, incase a tester want to change the values, we can do so by performing right click on the keyword 'checkpoint' of the script and navigating to "Checkpoint Properties" as shown below:

```
~("Free Online Math Calculator_2").Page("Free Online Math Calculator").Link("Numbers").Check CheckP
```



You can locate the same checkpoints in object repository as well as shown below. It exactly shows what type of checkpoint and what are the expected values, time out values.



What is Synchronization?

Synchronization point is the time interface between Tool and Application under test. Synchronization point is a feature to specify delay time between one step and another of the test script.

For Example, clicking on a link may load the page is 1 second, sometimes 5 seconds or even it might take 10 seconds to load it completely. It depends on various factors such as the application server response time, network bandwidth , client system capabilities etc.

If the time is varying then the script will fail unless the tester handles these time differences intelligently.

Ways to Insert Sync Point:

WaitProperty

Exist

Wait

Sync(only for web based apps)

Inserting QTP Inbuilt Synchronization points.

Let us say we need to insert a sync point between clicking on "numbers" link and clicking on "simple Interest" calculator of in "http://easycalculation.com/". We will now take a look at all the 5 ways to insert sync point for the above scenario.

Method 1: WaitProperty

WaitProperty is a method that takes the property name, Value and Timeout value as input to perform the sync. It is a dynamic wait and hence this option is encouraged.

```
' Method 1 - WaitProperty with 25 seconds
Dim obj
Set obj = Browser("Math Calculator").Page("Math Calculator")
obj.Link("Numbers").Click

obj.Link("Simple Interest").WaitProperty "text", "Simple Interest",25000
obj.Link("Simple Interest").Click
```

Method 2: Exist

Exist is a method that takes the Timeout value as input to perform the sync. Again it is a dynamic wait and hence this option is encouraged.

```
' Method 2 : Exist Timeout - 30 Seconds
Dim obj
Set obj = Browser("Math Calculator").Page("Math Calculator")
obj.Link("Numbers").Click

If obj.Link("Simple Interest").Exist(30) Then
    obj.Link("Simple Interest").Click
Else
    Print "Link NOT Available"
End IF
```

Method 3: Wait

Wait is a hardcoded sync point which waits independent of the event happened or NOT. Hence usage of Wait is discouraged and can be used for shorter wait time such as 1 or 2 seconds.

```
' Method 3 : Wait Timeout - 30 Seconds
Dim obj
Set obj = Browser("Math Calculator").Page("Math Calculator")
obj.Link("Numbers").Click
wait(30)
Browser("Math Calculator").Page("Math Calculator").Link("Simple Interest").Click
```

Method 4: Sync Method

Sync Method can be used only for web applications where there is always a lag

between page loads.

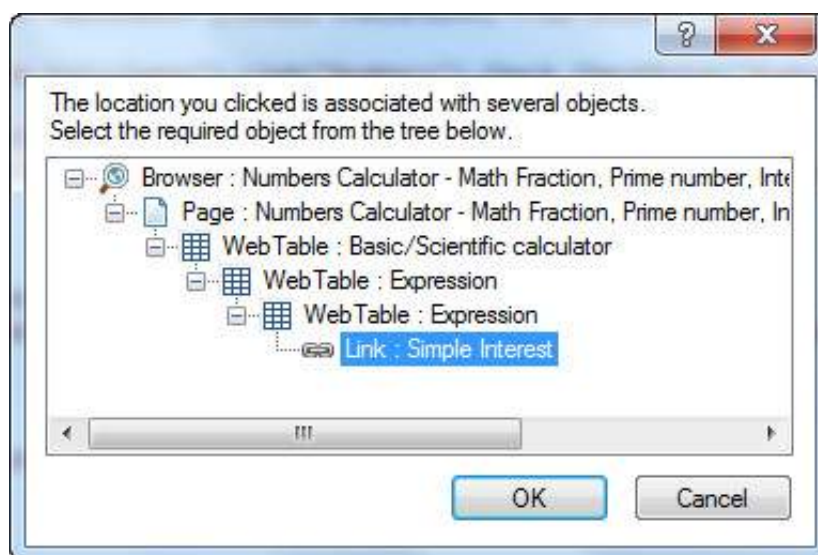
```
' Method 4 :  
Dim obj  
Set obj = Browser("Math Calculator").Page("Math Calculator")  
obj.Link("Numbers").Click  
  
Browser("Math Calculator").Sync  
Browser("Math Calculator").Page("Math Calculator").Link("Simple Interest").Click
```

Method 5 : Inserting QTP Inbuilt Synchronization points:

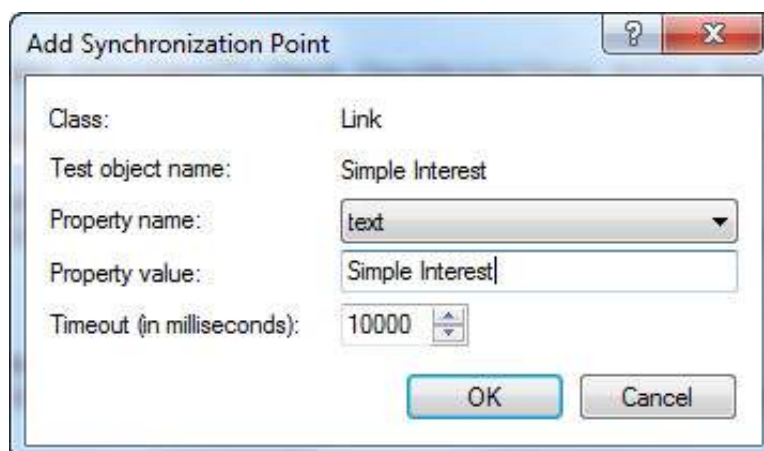
Step 1 : Get into Recording Mode. This Option Would be Disabled if the user is NOT in Recording Mode.

Step 2 : Goto "Design" -> "Synchronization Point" .

Step 3 : We need to Select the object which we want to be the Sync Point. After Selecting the object, object window opens as shown below:



Step 4 : Click Ok, the "Add Synchronization Window" Opens up. Select the Property, Value and Time out value and click ok as shown below:



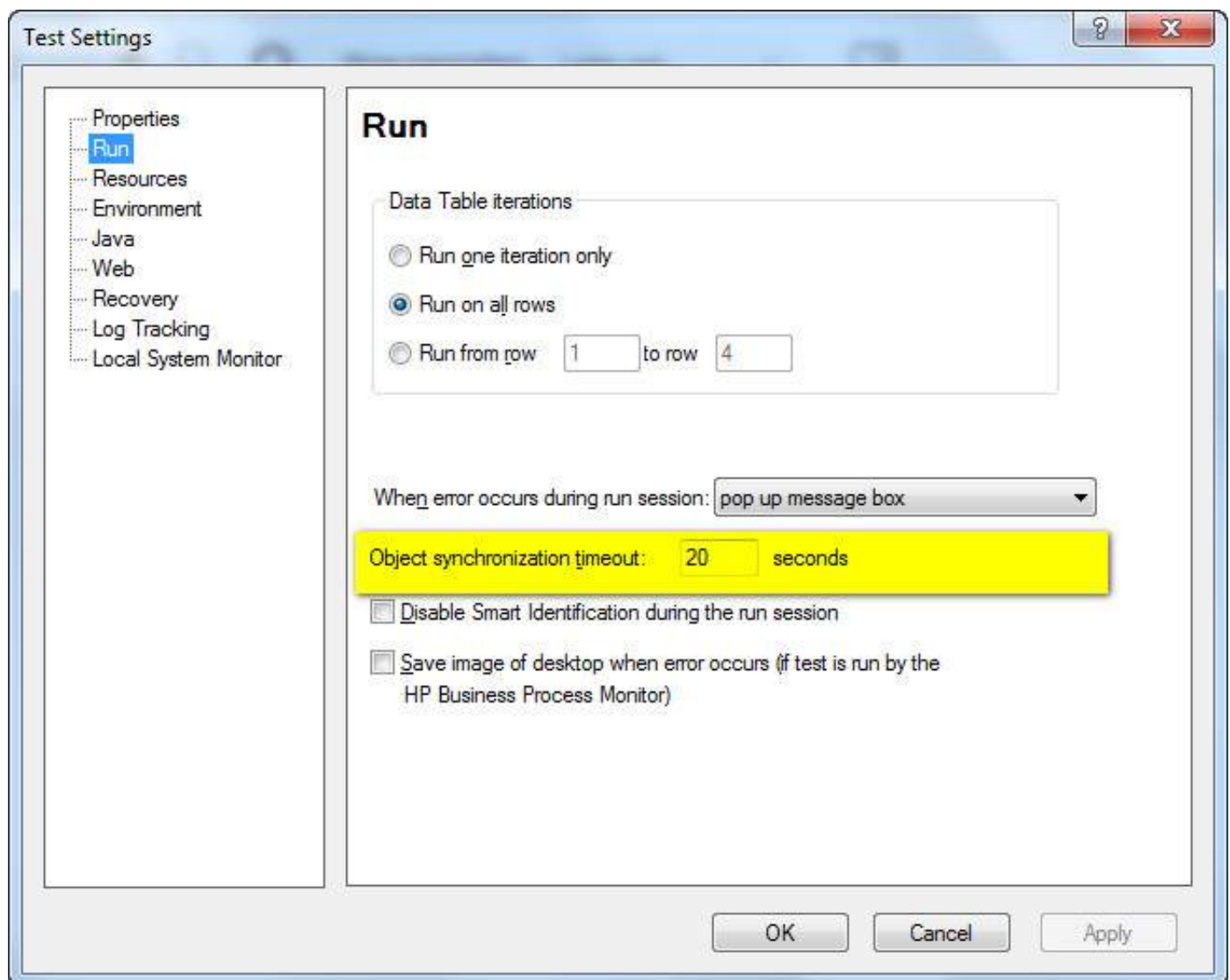
Step 5 : The Script would be generated as shown below which is the same as that of the WaitProperty(Method 1) that we had already discussed:

```
Browser("Math Calculator").Page("Math Calculator").Link("Numbers").Click  
Browser("Math Calculator").Page("Math Calculator").Link("Simple Interest").WaitProperty "text", "S
```

Default Synchronization:

When user hasn't used any of the above sync methods, still QTP has inbuilt Object synchronization timeout which can be adjusted by the user.

Navigate to "File" >> "Settings" >> Run Tab >> Object Synchronization Time out as shown below.



Smart Identification:

Sometimes, QTP is unable to find any object that matches the recognized object

description or it may find more than one object that fits the description, then QTP ignores the recognized description and uses the Smart Identification mechanism to recognize the object.

QTP's Smart Identification uses two types of properties:

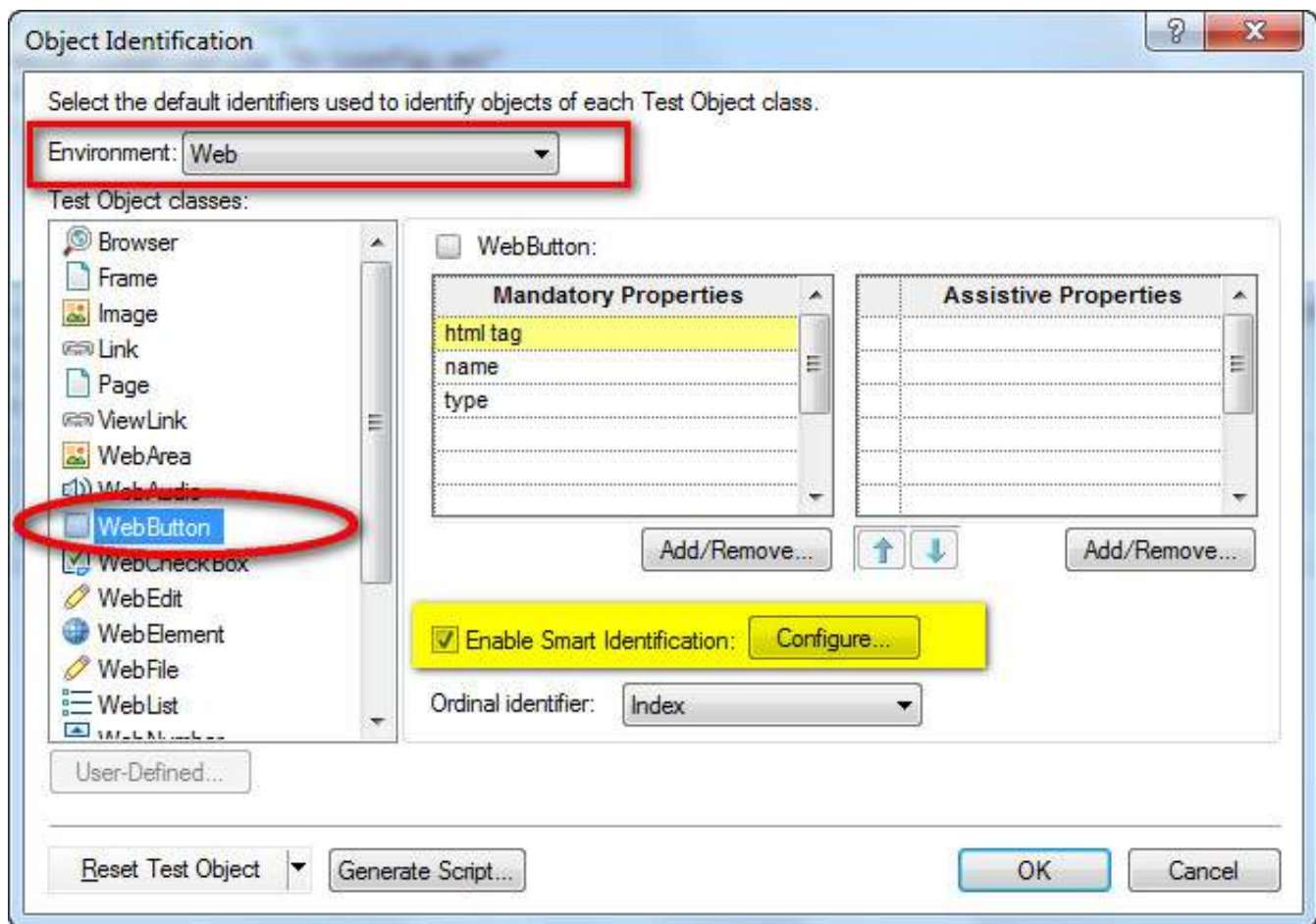
Base Filter Properties - The basic properties of a particular test object class whose values cannot be changed without changing the essence of the original object.

Optional Filter Properties - Other properties also assist in identifying the objects of a particular class whose properties are unlikely to change often but can be ignored if they are no longer applicable.

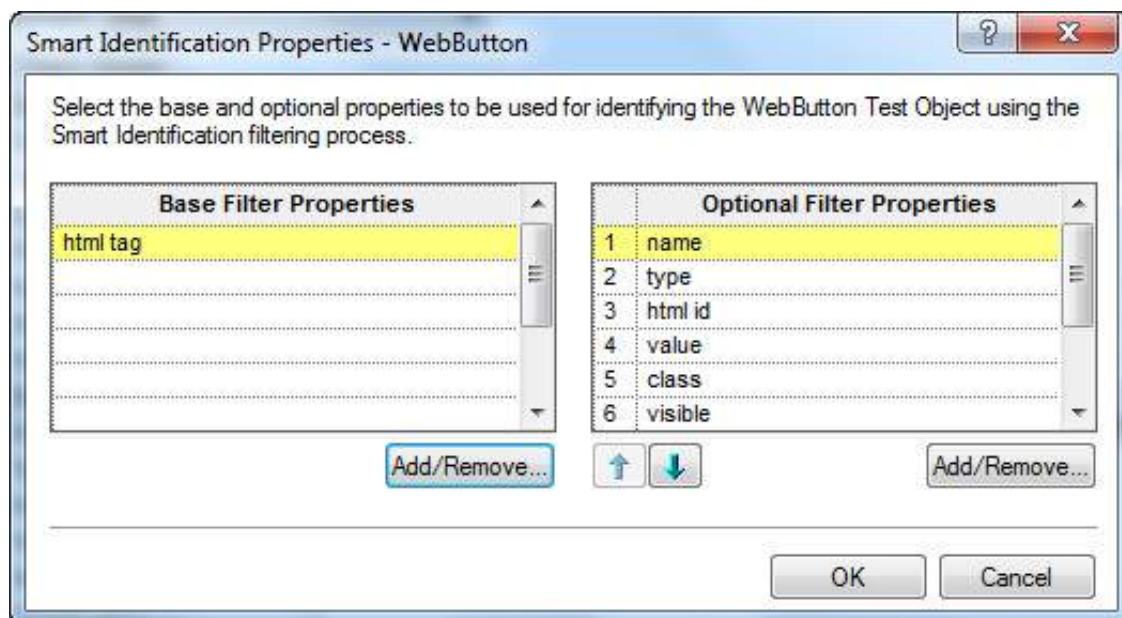
Enabling Smart identification for an object:

Step 1 : Navigate to "Tools" -> "Object Identification". Object Identification Dialog Opens.

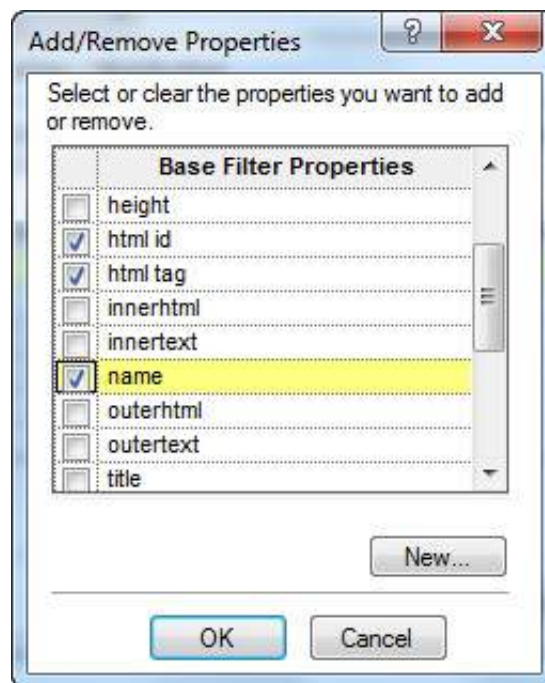
Step 2 : Choose the Environment, Object Class and Turn ON "Enable Smart Identification" as shown below:



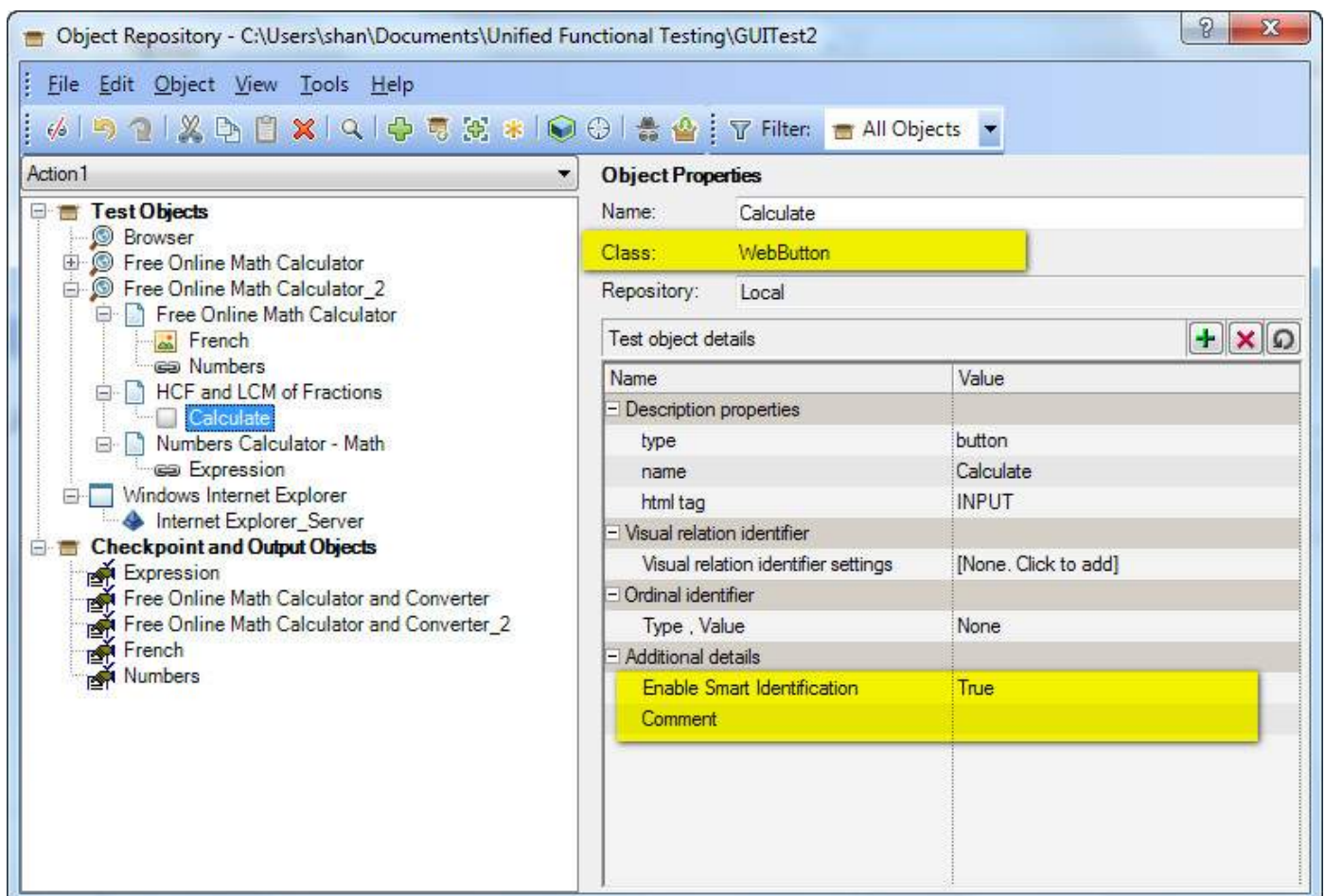
Step 3 : Click Configure and choosing the base and Optional Filter Properties.



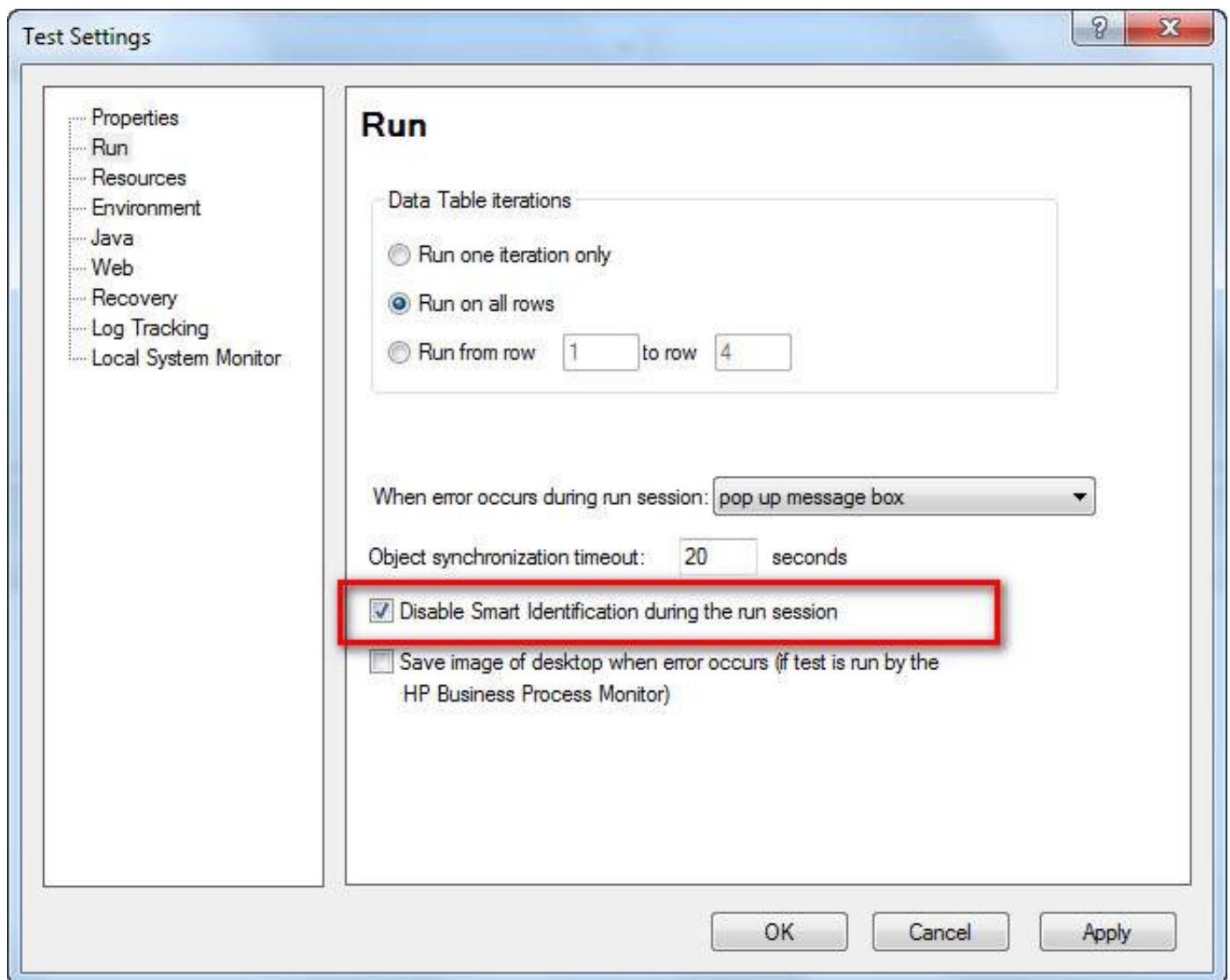
Step 3 : Add Properties in Base Properties apart from the Default one and also add/remove Optional Filter Properties. Please Note that Same properties cannot be part of both Mandatory and Assistive Properties and click "OK".



Step 4 : Verifying if the Smart Identification is Enabled after Adding object of that type in the Object Repository. Smart Identification is Set to TRUE. We can also make it False in case we dont want to enable Smart Identification.



Step 5 : We can even disable a test Level by applying at test script level under "Settings" of "File" Menu as shown below:



Step 6 : If the Smart Identification is disabled as per Step# 6 then it won't apply smart identification for any object during script execution.

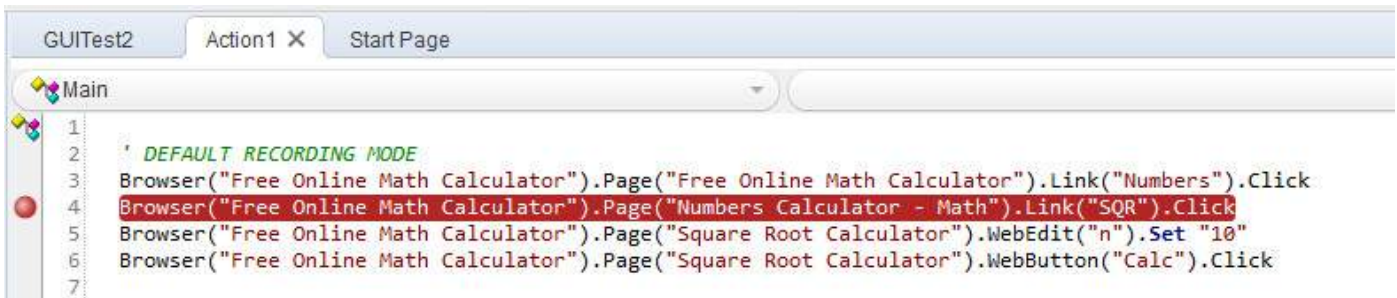
Step 7 : In case objects are added with Smart Identification as Off, QTP won't use Smart Identification for recognizing in future, even though we have enabled it afterwards.

Debugging:

Debugging, in automation testing context, is a systematic process of spotting and fixing the coding issues in the automation scripts so that the script will be more robust and can spot the defects in the application.

There are various ways to perform debugging using break points in QTP. Break Points can be inserted just by pressing "F9" or by using the Menu option "Run" -> "Inserting/Removing Break Point".

After Inserting the Break point the "Red Coloured" Dot and the line will be highlighted in RED as shown below:



Method	ShortCut	Description
Step Into	F11	Used to execute each and every Step. Steps into the Function/Action and executes line by line. It pauses on each line after execution.
Step Over	F10	Used to Step over the Function. Step Over runs only the current step in the active document.
Step Out	Shift+F11	After Step Into the function, you can use the Step Out command. Step Out continues the run to the end of the function and then pauses the run session at the next line.

Options in Break Point:

Various Options in Break Point can be accessed by Navigating 'Run' Menu.

ShortCut	Description
F9	Insert/Remove BreakPoint
Ctrl+F9	Enable/Disable BreakPoint
Ctrl+Shift+F9	Clear All BreakPoint
Use Only Menu	Enable/Disable All BreakPoints

Debugging Pane:

The Following are the panes in the debugging window:



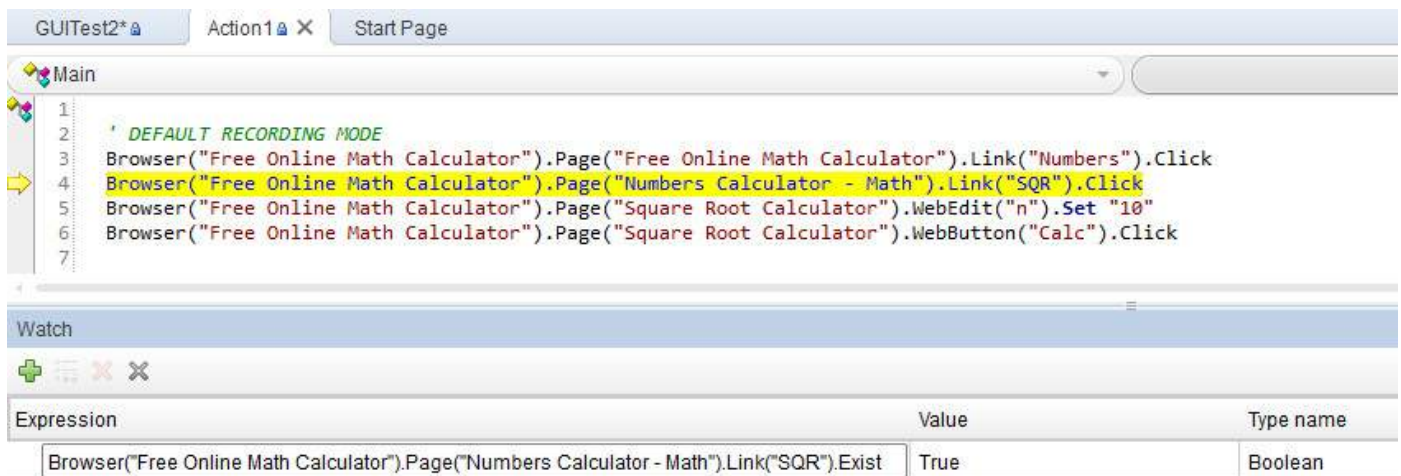
Output - This Tab displays all the Output of the Print Statements.

Watch - This Tab displays the Boolean output of the Given Expression.

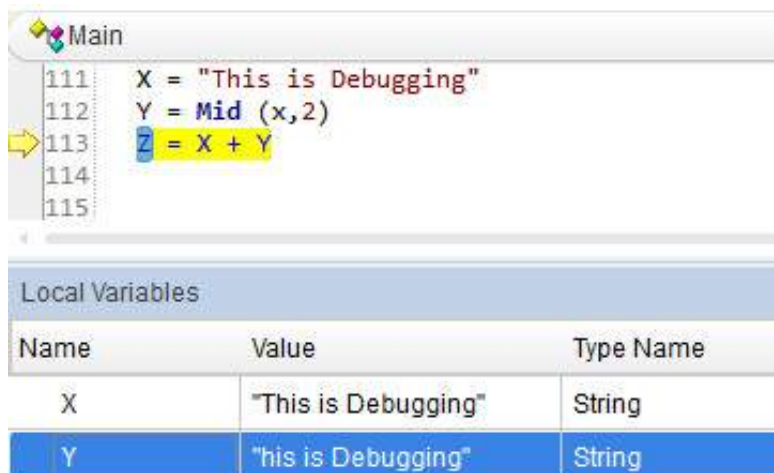
LocalVariables - This Tab displays the Output of the Local Variables.

Example:

The Watch Pane shows the output expression as shown below:



The Local Variables Pane shows the values held by the local variables as shown below:



What is Error Handling?

There are various ways on handling errors in QTP. There are three possible kinds of error type one would encounter while working with QTP.

Syntax Errors

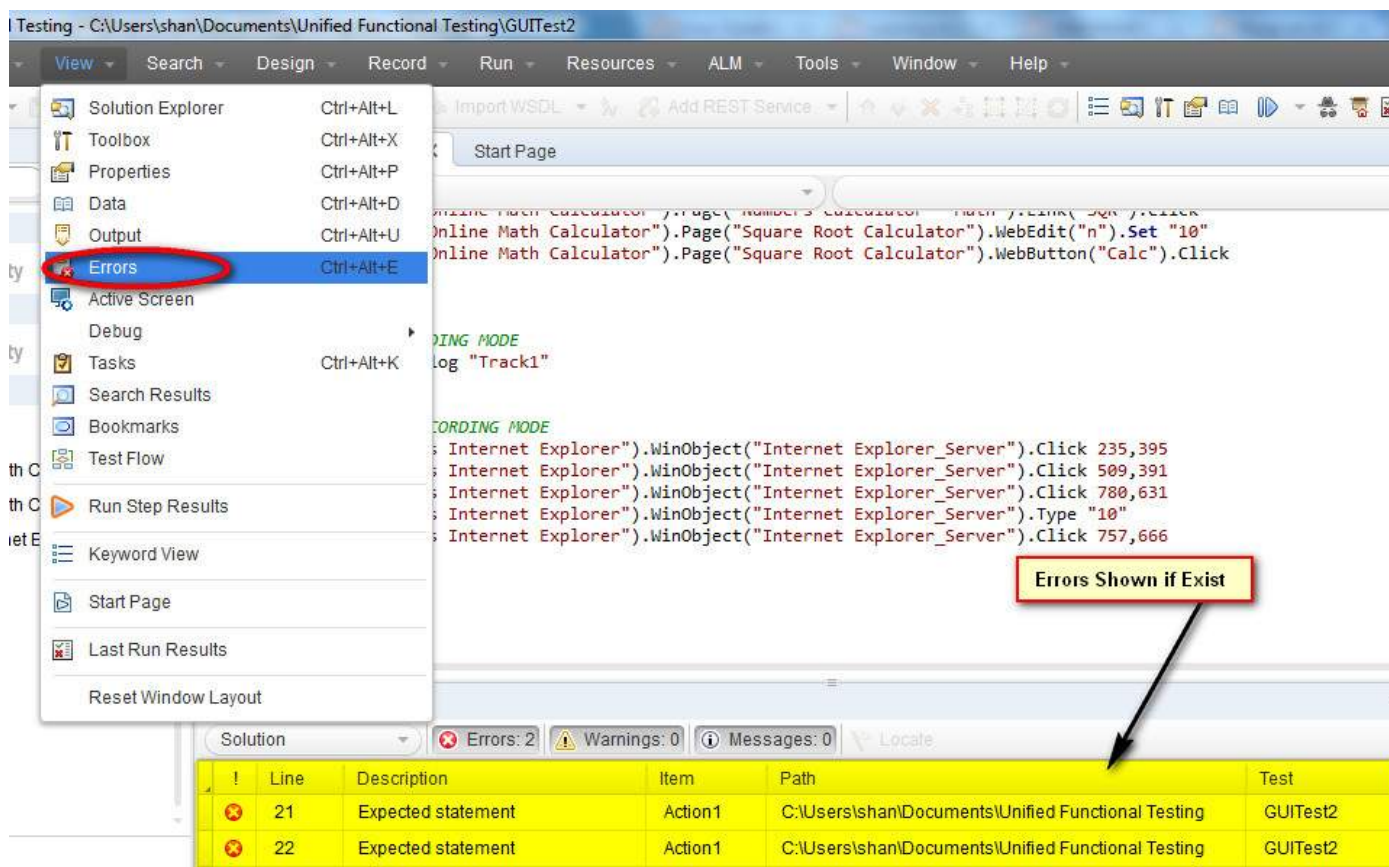
Logical Errors

Run Time Errors

Error Types:

Syntax Errors:

Syntax errors are the typos or a piece of the code that does not confirm with the VBScripting language grammar. Syntax errors occur at the time of compilation of code and cannot be executed until the errors are fixed. To verify the syntax one use the keyboard shortcut as Ctrl+F7 and the result is displayed as shown below. If the window is NOT displayed one can navigate to "View" -> "Errors".



Logical Errors:

If the script is syntactically correct but it produces unexpected results. Logical error usually does not interrupt the execution but produces incorrect results. Logical errors could occur due to variety of reasons, viz- wrong assumptions or misunderstanding of the requirement and sometimes incorrect program logics(using do-while instead of do-Until) or Infinite Loops.

One of the ways to detect a logical error is to perform peer reviews and also verifying the QTP output file/result file to ensure the tool has performed what it has intended to do.

RunTime Errors:

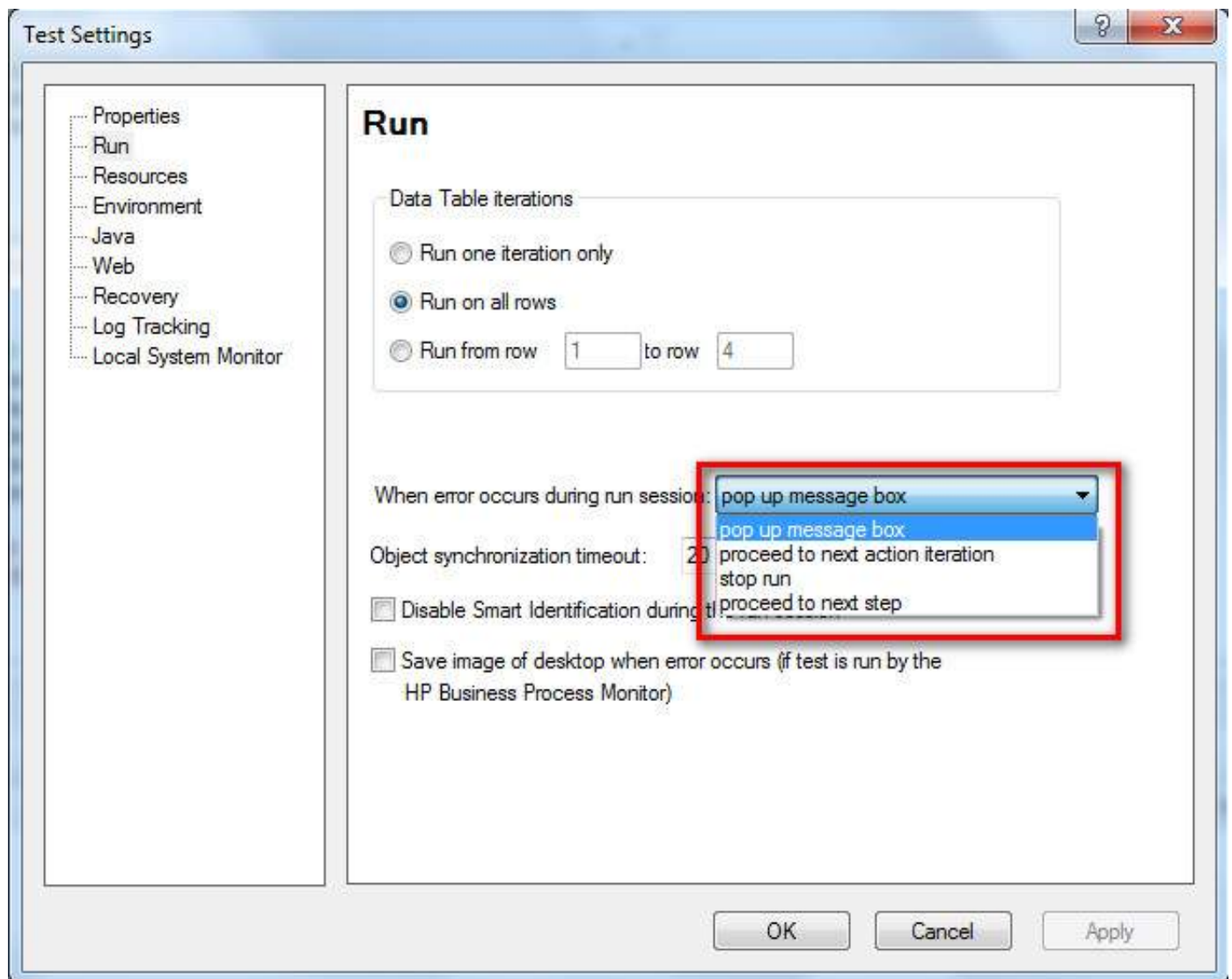
As The name states, this kind of Error happens during Run Time. The reason for such kind of errors is that the script trying to perform something but it is unable to do so and the script usually stops as it is unable to continue with the execution. Classic Examples for Run Time Errors are,

1. File NOT found but the script trying to read the file.
2. Object NOT found but scripts is trying to act on that particular object.
3. Dividing a number by Zero.
4. Array Index out of bounds while accessing array elements.

Handling Run-Time Errors:

There are various ways to handle errors in the code.

1. Using Test Settings - Error handling can be defined the Test Settings by Navigating to "File" >> "Settings" >> "Run" Tab as shown below. We can select any of the specified settings and click "OK".



2. Using On Error Statement - On Error statement is used to notify the VBScript engine of intentions to handle the run-time errors by tester, rather than allowing the VBScript engine to display error messages that are not user friendly.

On Error Resume Next - On Error Resume Next informs the VBScript engine to proceed executing the next line of code when an error is encountered.

On error Goto 0 - This helps the testers to turn off the error handling.

3. Using Err Object - Error object is an inbuilt object within VBScript that captures the run time error number and error description with which we will be able to debug the code easily.

Err.Number - The Number property returns or Sets a numeric value specifying an error. If Err.Number value is 0 then No error had occurred.

Err.Description - The Description property returns or sets a brief description about an error.

Err.Clear - The Clear method resets the Err object and clears all the previous values associated with it.

Example:

```
'Call the function to Add two Numbers
Call Addition(num1,num2)

Function Addition(a,b)

On error resume next
If NOT IsNumeric(a) or IsNumeric(b) Then
    Print "Error number is " & err.number & " and description is : " & err.description
    Err.Clear
    Exit Function
End If

Addition = a+b

'disables error handling
On Error Goto 0

End function
```

Using Exit Statement - Exit Statements can be used along with Err object to exit from a test or action or iteration based on the Err.Number value. Let us see each one of those Exit statements in detail.

ExitTest - Exits from the entire QTP test no matter what the run-time iteration settings are.

ExitAction - Exits the current action.

ExitActionIteration - Exits the current iteration of the action.

ExitTestIteration - Exits the current iteration of the QTP test and proceeds to the next iteration.

5. Recovery Scenarios - Upon encountering an error, recovery scenarios are triggered based on certain conditions and it is dealt in detail in a separate chapter.

6. Reporter Object - Reporter Object helps us to report an event to the run results. It helps us to identify if the concerned action/step is pass/fail.

```
'Syntax: Reporter.ReportEventEventStatus, ReportStepName, Details, [ImageFilePath]
```

' Example

```
Reporter.ReportEvent micFail, "Login", "User is unable to Login."
```

Recovery Scenarios

While executing the QTP scripts, we might get some Unexpected errors. Inorder To recover the test and continue executing the rest of the script from these unexpected errors, Recovery Scenarios are used. The Recovery Scenario Manager can be accessed by Navigating to "Resources" -> Recovery Scenario Manager as shown below:



Steps to create Recovery Scenario:

Step 1 : Upon Clicking "New" Recovery Scenario button, the Recovery Scenario Wizard opens as shown below:



Step 2 : Next Step is to choose the Trigger Event. It corresponds to event which arises It can be any of the below four events.

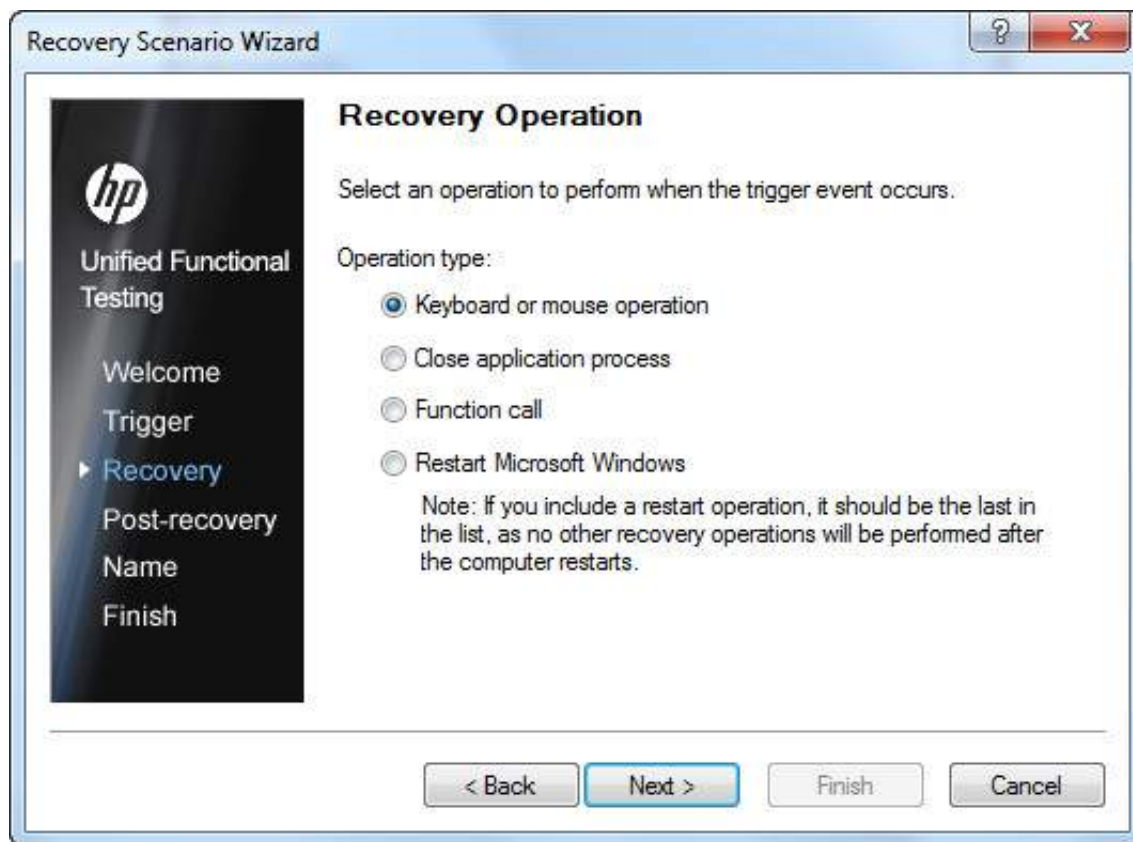
Pop-Up Window

Object State

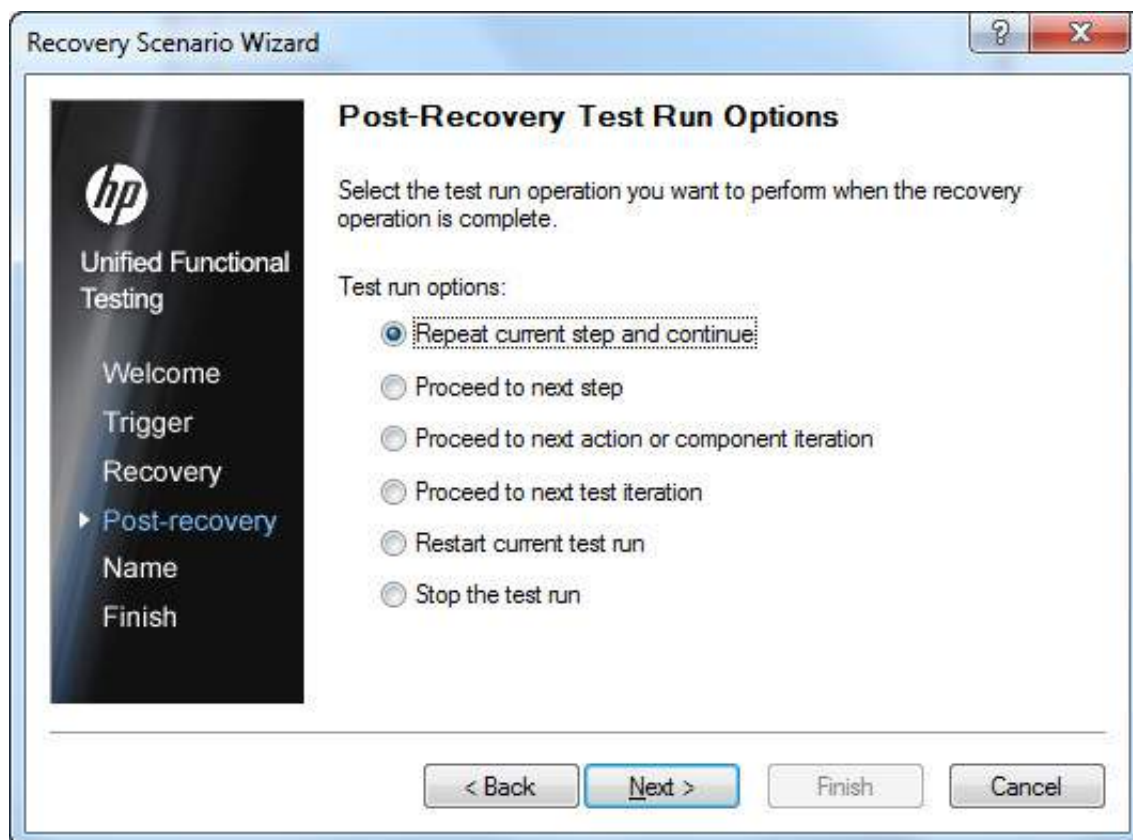
Test Run Error

Application Crash

Step 3 : Recovery Operations Window Opens. Recovery Operation can any of the following Operation as shown below:



Step 4 : Upon Specifying the appropriate Recovery Operation, we need to specify the Post Recovery Operation as well as shown below:

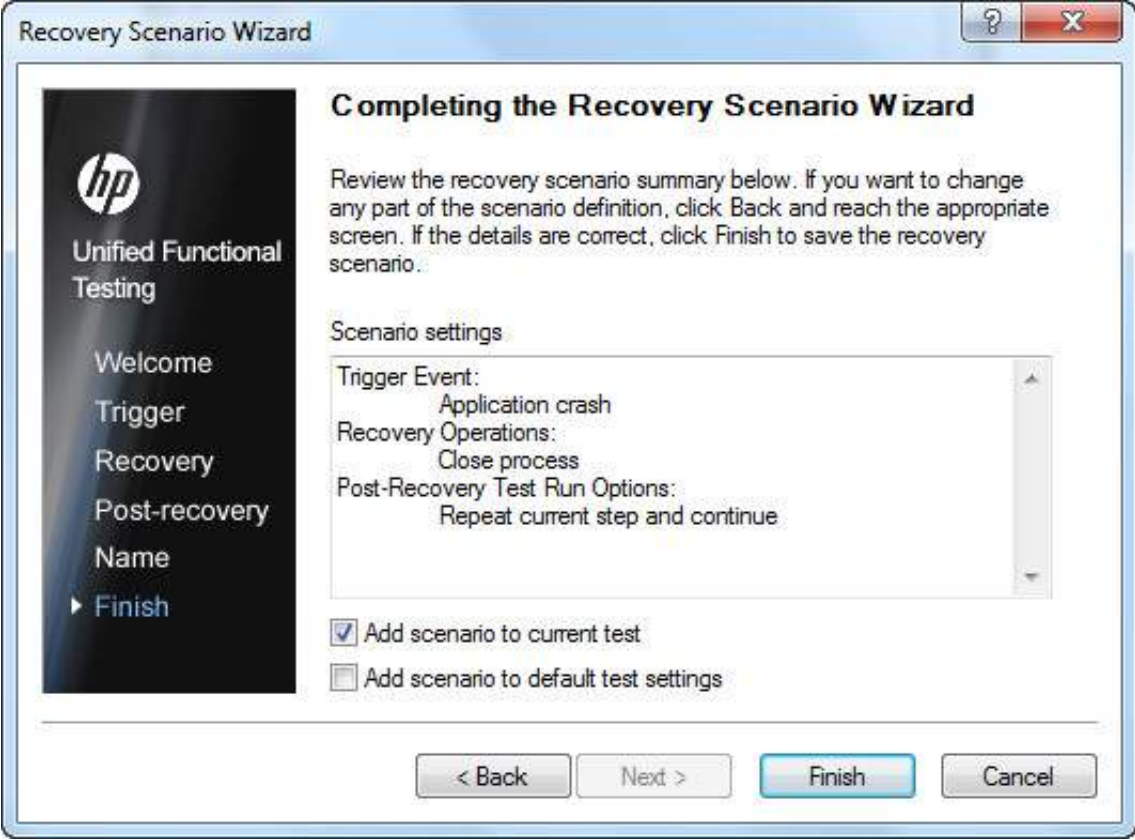


Step 5 : Upon Specifying the Post Recovery Operation, the recovery scenario should be named and added to the Test so that it will be activated.



The screenshot shows the 'Recovery Scenario Wizard' window. On the left is a sidebar with the HP logo and a list of steps: 'Unified Functional Testing', 'Welcome', 'Trigger', 'Recovery', 'Post-recovery', 'Name' (highlighted with a blue arrow), and 'Finish'. The main area is titled 'Name and Description' and contains the instruction: 'Provide a name and description for your recovery scenario.' Below this are three input fields: 'Recovery file:' with the text '<New Recovery Scenario File>', 'Scenario name:' with the text 'Recover_Application_Crash', and a large 'Description:' text area. At the bottom are four buttons: '< Back', 'Next >', 'Finish', and 'Cancel'.

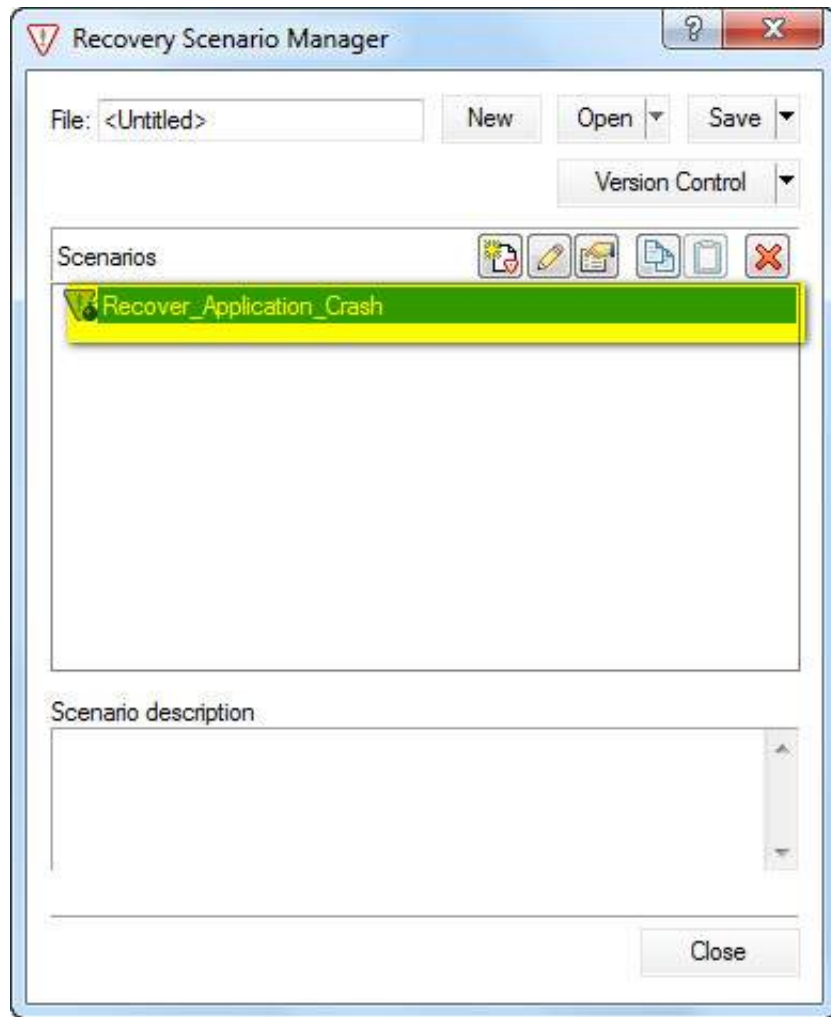
Step 6 : The Recovery Scenario creation is complete and needs to be mapped to the current Test by checking the option "Add Scenario to the current Test" and click "Finish"



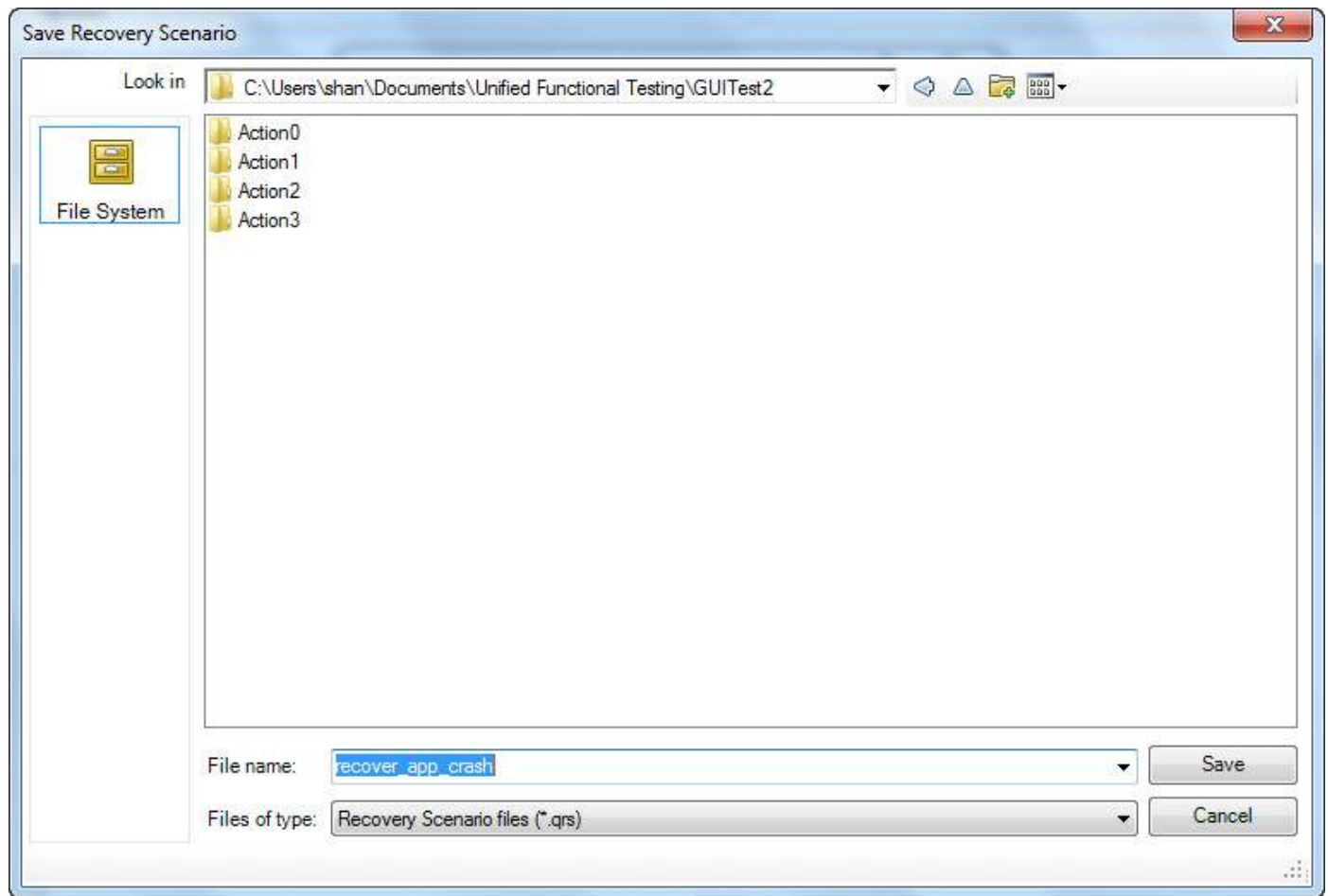
The screenshot shows the 'Recovery Scenario Wizard' window at the 'Completing the Recovery Scenario Wizard' step. The sidebar on the left is the same as in Step 6, but 'Finish' is now highlighted with a blue arrow. The main area contains the instruction: 'Review the recovery scenario summary below. If you want to change any part of the scenario definition, click Back and reach the appropriate screen. If the details are correct, click Finish to save the recovery scenario.' Below this is a 'Scenario settings' section with a list box containing: 'Trigger Event: Application crash', 'Recovery Operations: Close process', and 'Post-Recovery Test Run Options: Repeat current step and continue'. At the bottom are two checkboxes: 'Add scenario to current test' (checked) and 'Add scenario to default test settings' (unchecked). The bottom buttons are '< Back', 'Next >', 'Finish' (highlighted), and 'Cancel'.

Step 7 : The Added Recovery Scenario will be as shown below and click on "Close"

Button to continue.

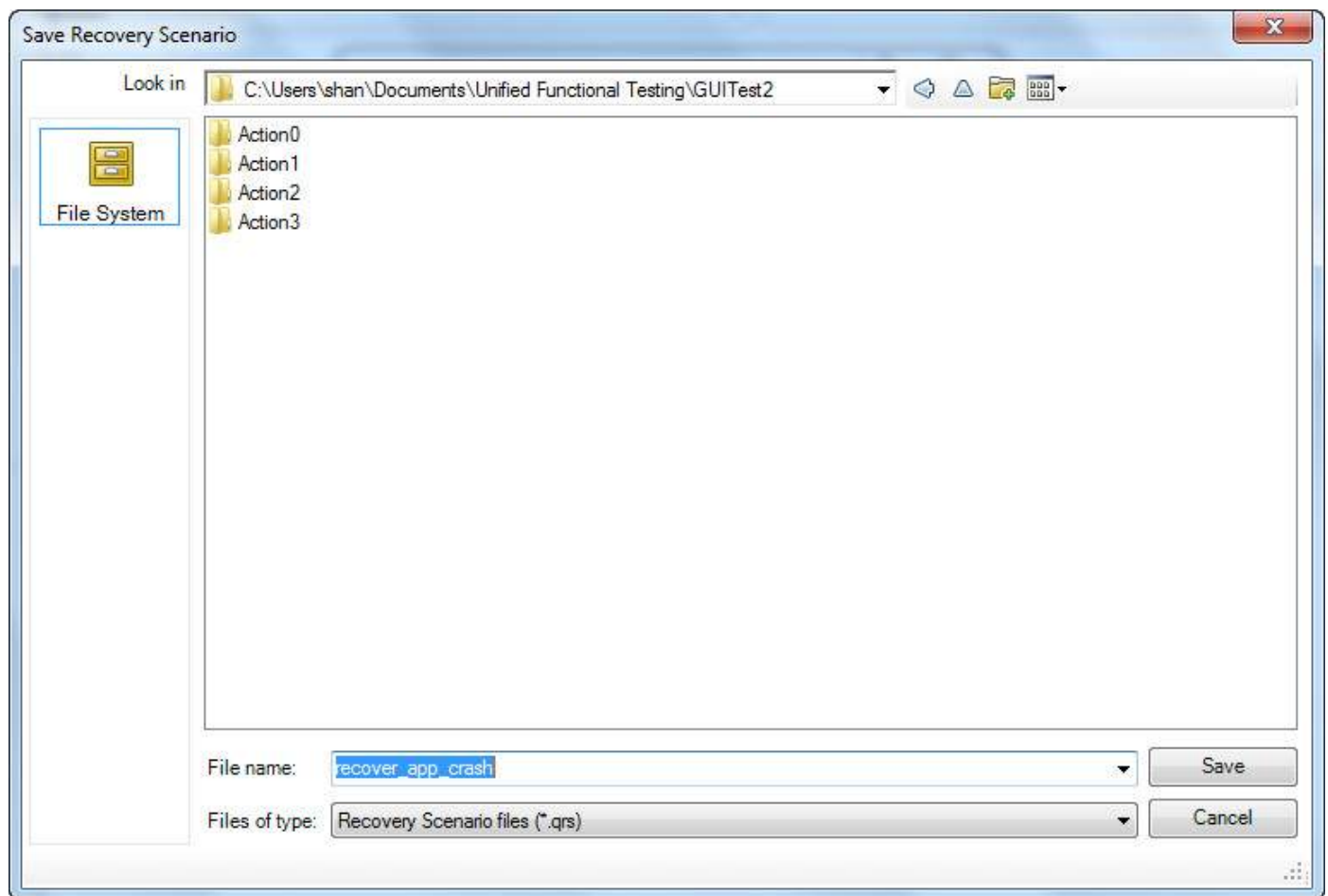


Step 8 : Upon Clicking on Close Button, QTP would Pop up user to Save the created Recovery Scenarion. It will be saved with the extension .qrs and the wizard would be closed.



Verification:

The Created Recovery Scenario should be part of the test now and can be verified by navigating to "File" -> "Settings" -> "Recovery" Tab.

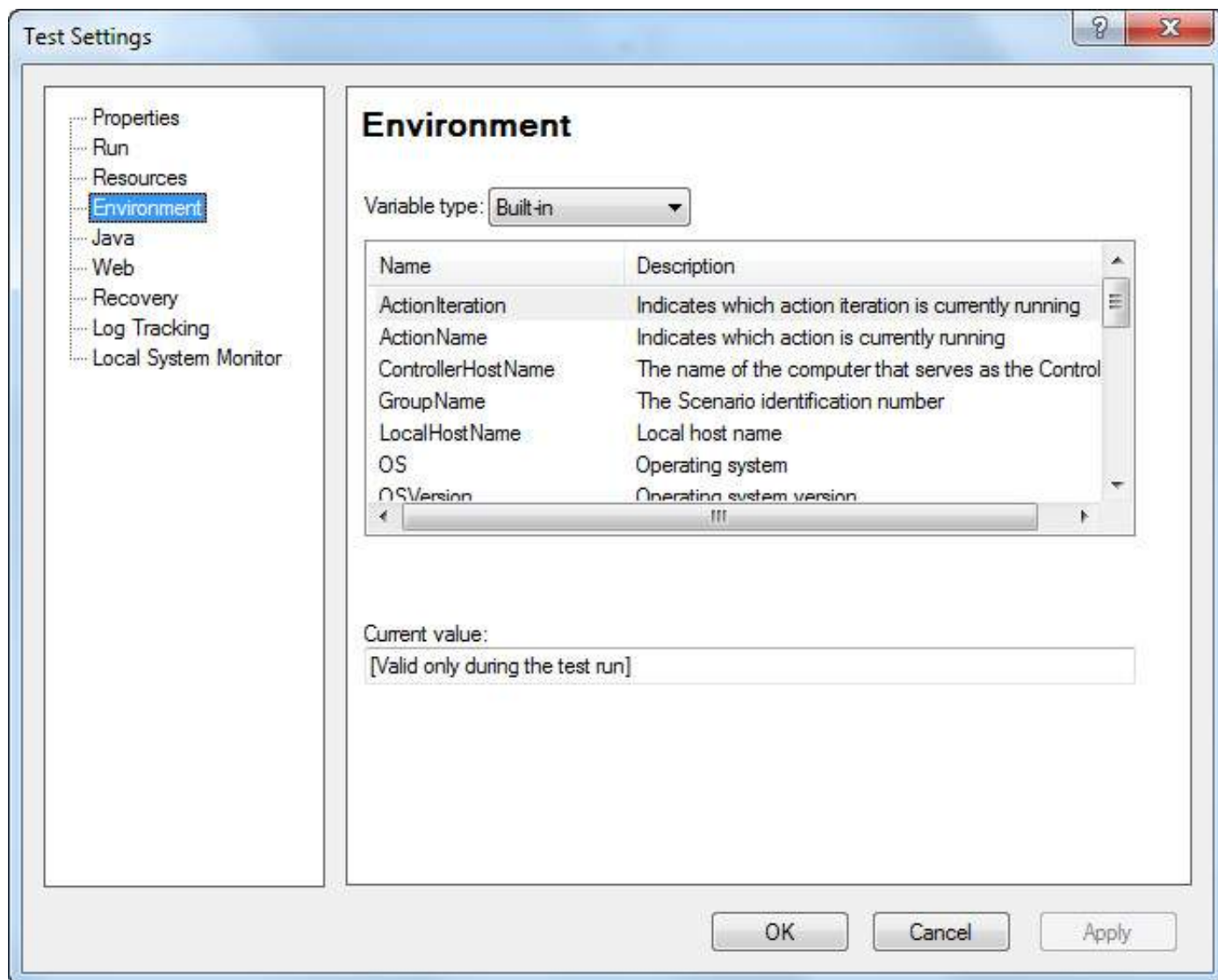


Environment Variables

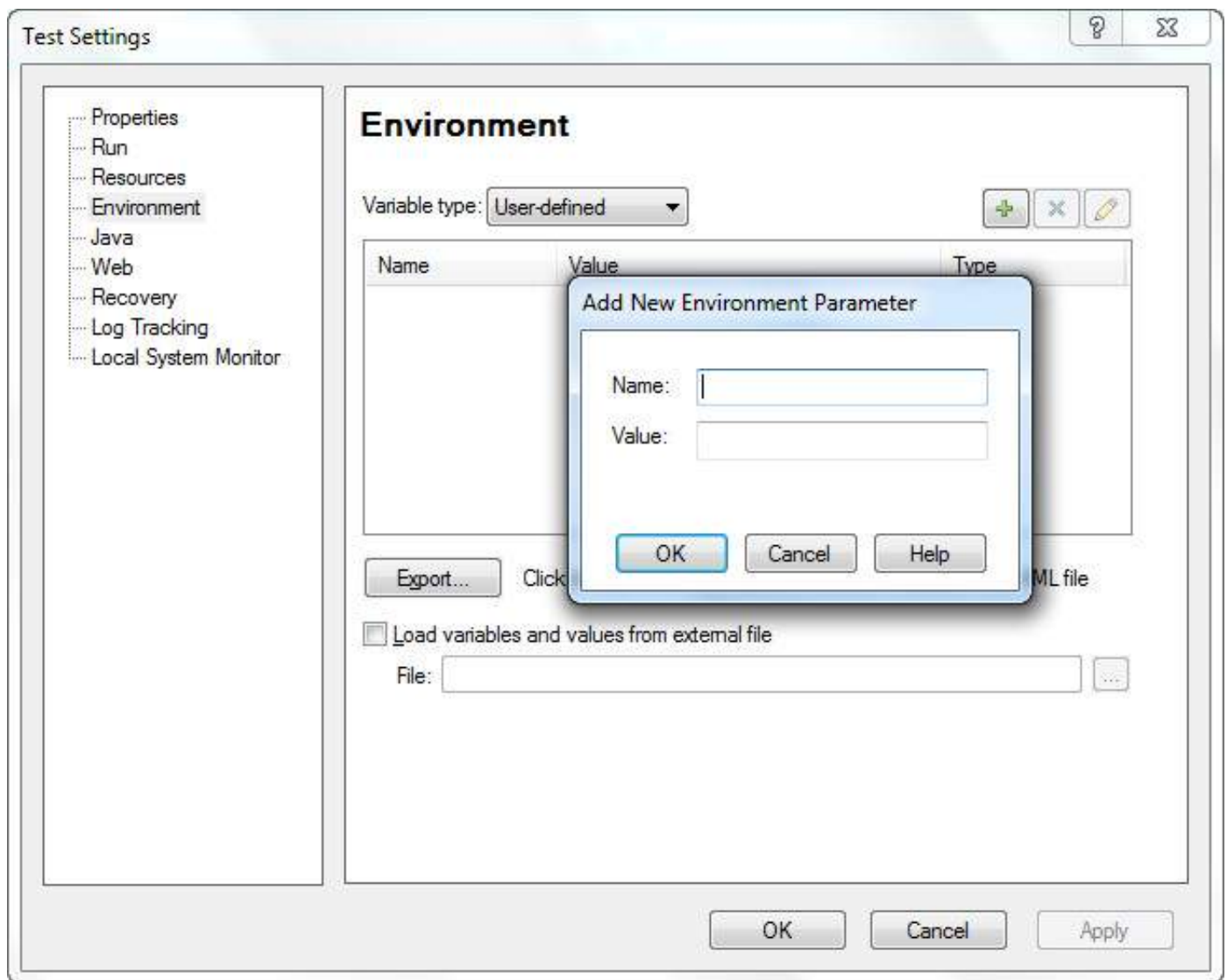
QTP environment variables are special types of variables that can be accessed by all actions, function libraries and recovery Scenarios. There are inbuilt environment variables for Windows that are available to all the applications running on that particular system whereas QTP environment variables are only available to that test script during run-time.

Types of Environment Variables

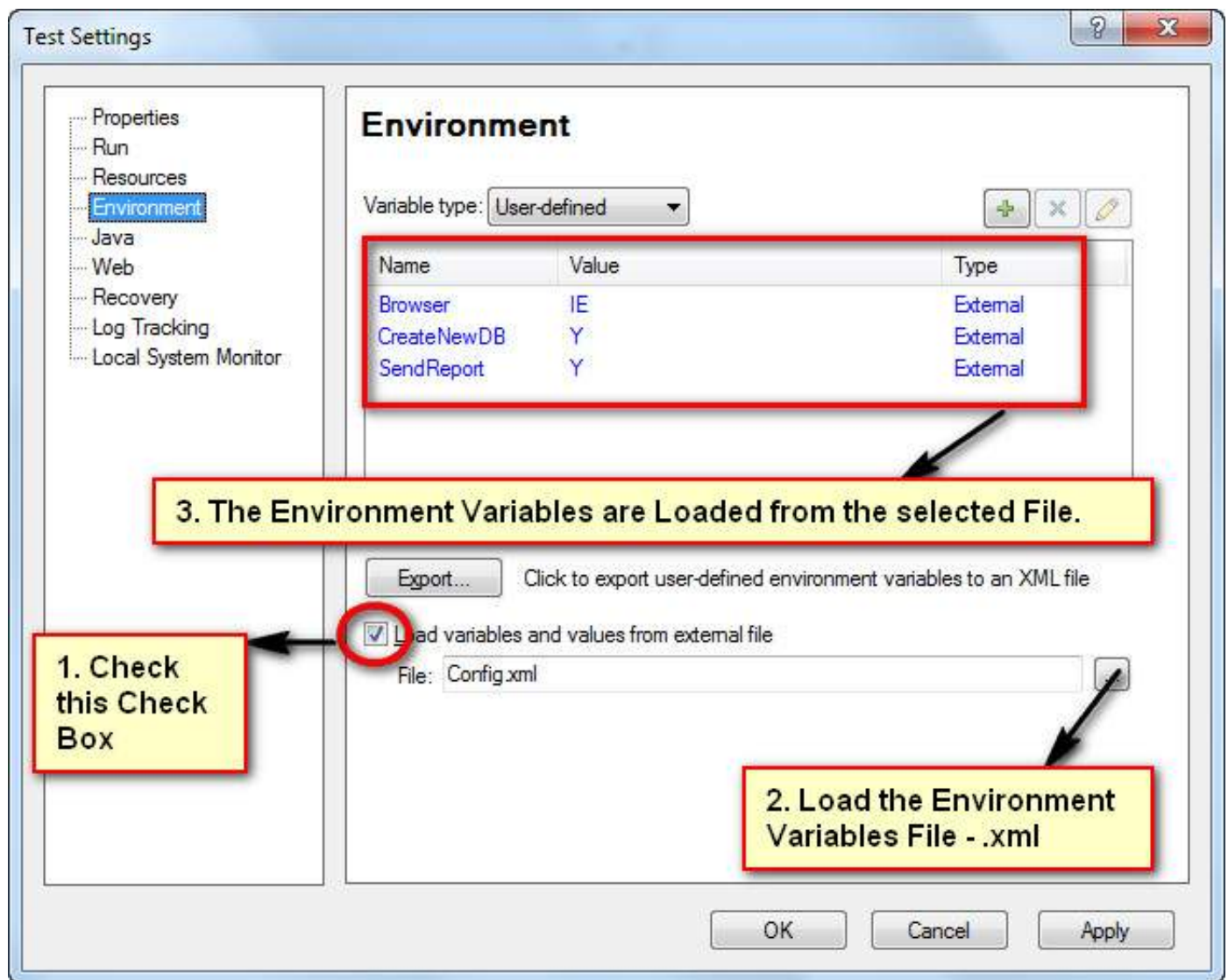
Built-in Environment Variables - provides a range of environment parameters that can provide information such as test name, action name, the test path, local host name, the operating system name, type and its version. The Environment Variable names can be accessed by Navigating to "File" -> "Test Settings" -> "Environment" Tab



User defined Internal - User Defined Variables can be saved by Selecting "User Defined" in the Environment Tab Window. The "+" button is Clicked to enter Parameter Name and Value as shown below:



User Defined External - User Defined Variables can be stored in a external file as an .xml and can be loaded into the test as shown in the below figure or can also be loaded dynamically during run-time as explained below in one of the examples.



Environment Variables - Supported Methods:

1. ExternalFileName Property - Returns the name of the loaded external environment variable file specified in the Environment tab of the Test Settings dialog box. If no external environment variable file is loaded, this property returns an empty string

```
x= Environment.ExternalFileName  
print x
```



2. LoadFromFile Method - Loads the specified environment variable file(.xml) dynamically during run time. When using this method, the environment variables need NOT be added manually into the Environment Tab.

```
Environment.LoadFromFile "D:\config.xml"
b = Environment.Value("Browser")
print b
```

The screenshot displays the QTP interface during a test run. On the left, the 'Main' pane shows the following code:

```
122 Environment.LoadFromFile "D:\config.xml"
123 b = Environment.Value("Browser")
124 print b
```

A red box highlights this code, with an arrow pointing to a yellow callout box that reads: "2. Dynamically Loaded the Env File and accessed the Browser Parameter which has value 'IE'".

Below the code, the 'Output' pane shows the text "IE" printed, with a yellow callout box stating: "3. It has Printed 'IE' as Expected in the Output Window".

On the right, the 'Environment' tab is open, showing an XML structure:

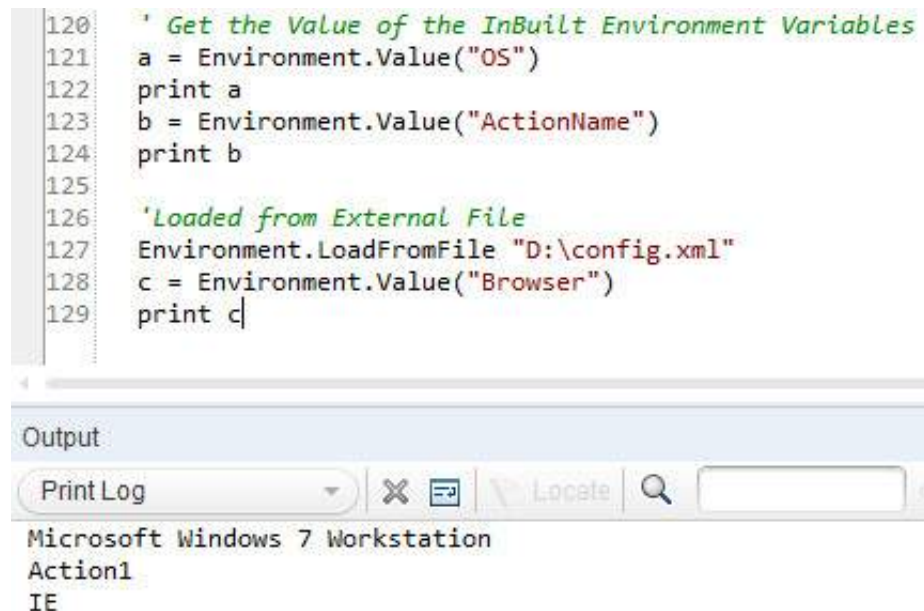
```
1 <Environment>
2   <Variable>
3     <Name>Browser</Name>
4     <Value>IE</Value>
5   </Variable>
6   <Variable>
7     <Name>CreateNewDB</Name>
8     <Value>Y</Value>
9   </Variable>
10  <Variable>
11    <Name>SendReport</Name>
12    <Value>Y</Value>
13  </Variable>
14 </Environment>
```

A red box highlights the first variable, with an arrow pointing to a yellow callout box that reads: "1. Environment Variable XML File".

3. Value Property - Retrieves the value of environment variables. We can also set the value of user-defined internal environment variables using this property.

```
' Get the Value of the InBuilt Environment Variables
a = Environment.Value("OS")
print a
b = Environment.Value("ActionName")
print b

'Loaded from External File
Environment.LoadFromFile "D:\config.xml"
c = Environment.Value("Browser")
print c
```



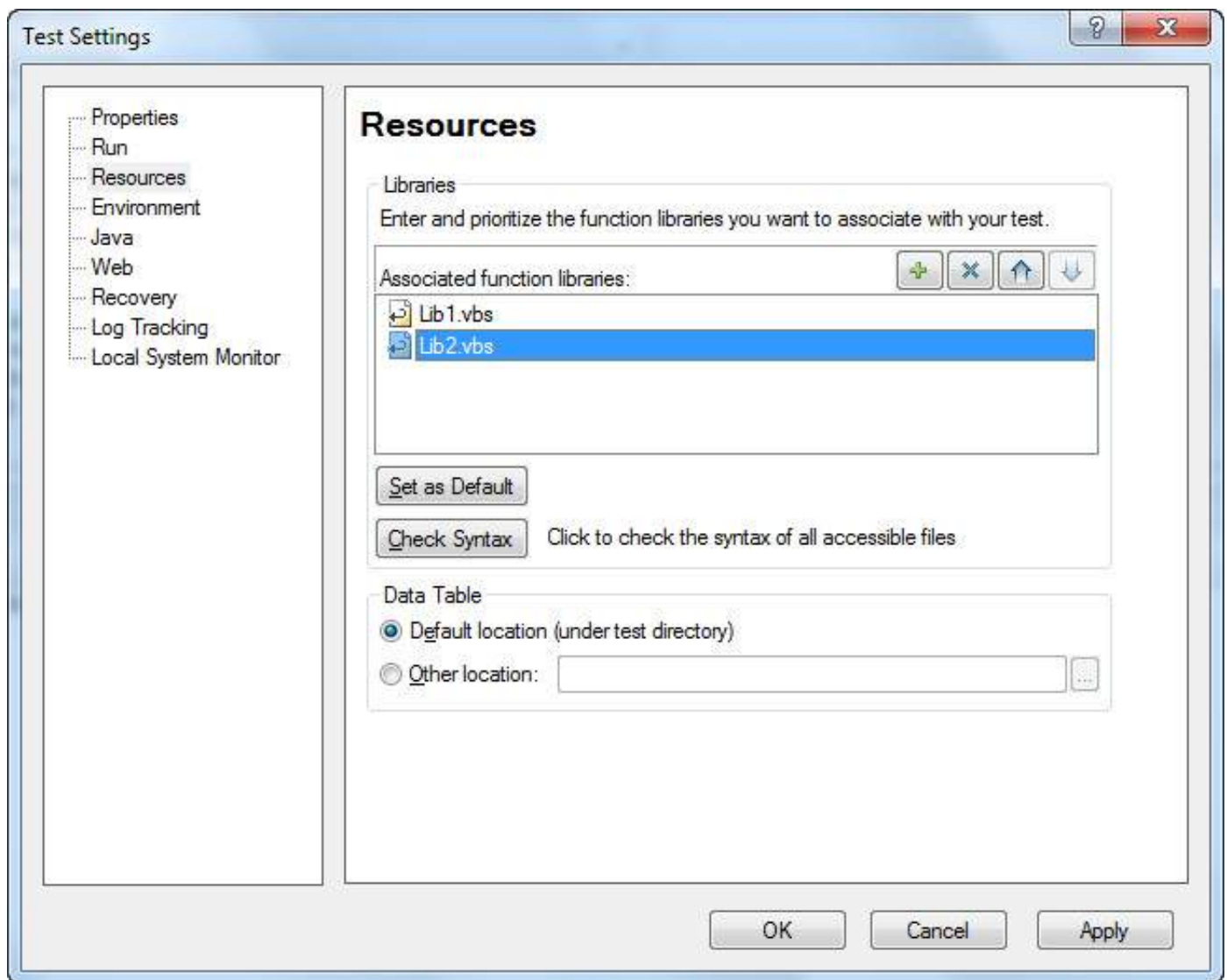
Library Files

Inorder to modularize the script, library Files are added to the QTP Script. It contains variable declaration, Functions, Classes etc. They enable reusability that can be shared across test scripts. They are saved with an extension .vbs or .qfl

A New Library File can be Created by Navigating to "File" >> "Function Library"

Associating Function Libraries

Method#1 : By using "File" > "Settings" > Resources > Associate Function Library option. Click on "+" Button to Add Function Library File and add it using actual path or relative path as shown below:

**Method#2 : Using ExecuteFile method.**

```
'Syntax : ExecuteFile(Filepath)
ExecuteFile "C:\lib1.vbs"
ExecuteFile "C:\lib2.vbs"
```

Method#3 : Using LoadFunctionLibrary Method.

```
'Syntax : LoadFunctionLibrary(Filepath)
LoadFunctionLibrary "C:\lib1.vbs"
LoadFunctionLibrary "C:\lib2.vbs"
```

Method#4 : Automation Object Model(AOM) - It is a mechanism using which we can control various QTP operations outside QTP. Using AOM, we can launch QTP, Open the Test, Associate Function Libraries etc. The Following Vbscript should be saved with Extension .vbs and upon executing the same, QTP will be launched and test would start executing. AOM will be discussed in detail in the later chapters.

```
'Launch QTP
Set objQTP = CreateObject("QuickTest.Application")
```

```

objQTP.Launch
objQTP.Visible = True

'Open the test
objQTP.Open "D:\GUITest2", False, False
Set objLib = objQTP.Test.Settings.Resources.Libraries

'Associate Function Library if NOT associated already.
If objLib.Find("C:\lib1.vbs") = -1 Then
    objLib.Add "C:\lib1.vbs", 1
End

```

Test Results:

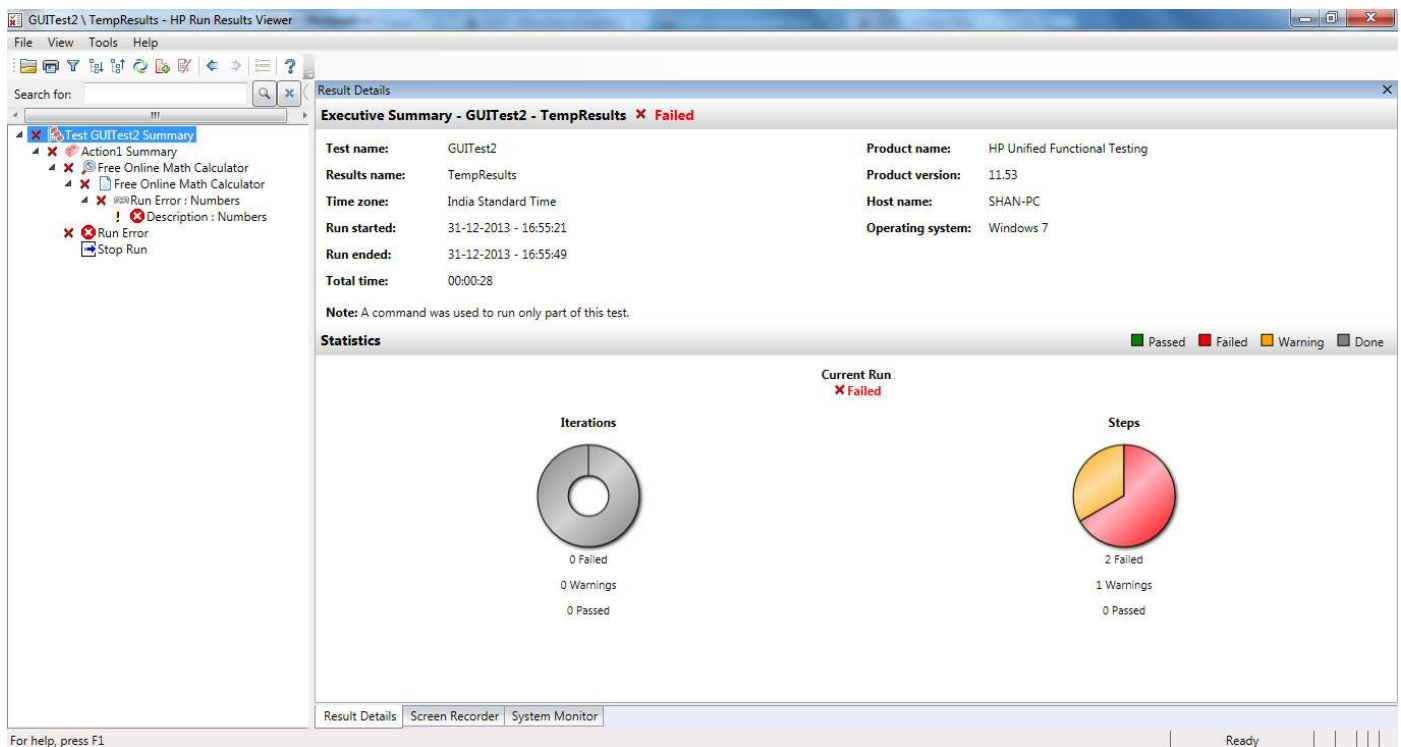
The Test Results Window gives us sufficient information to show the steps passed, failed etc. Results window opens automatically after execution of the test(as per default settings).

Steps Passed

Steps Failed

Environment Parameters

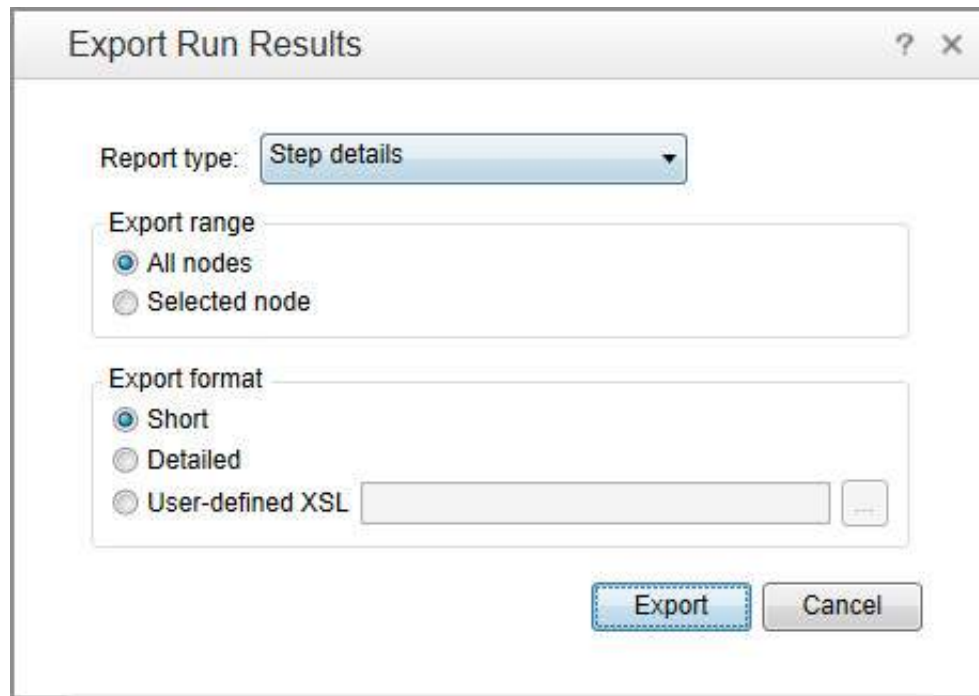
Graphical Statistics



Operations performed in Test Results:

Converting Results to HTML

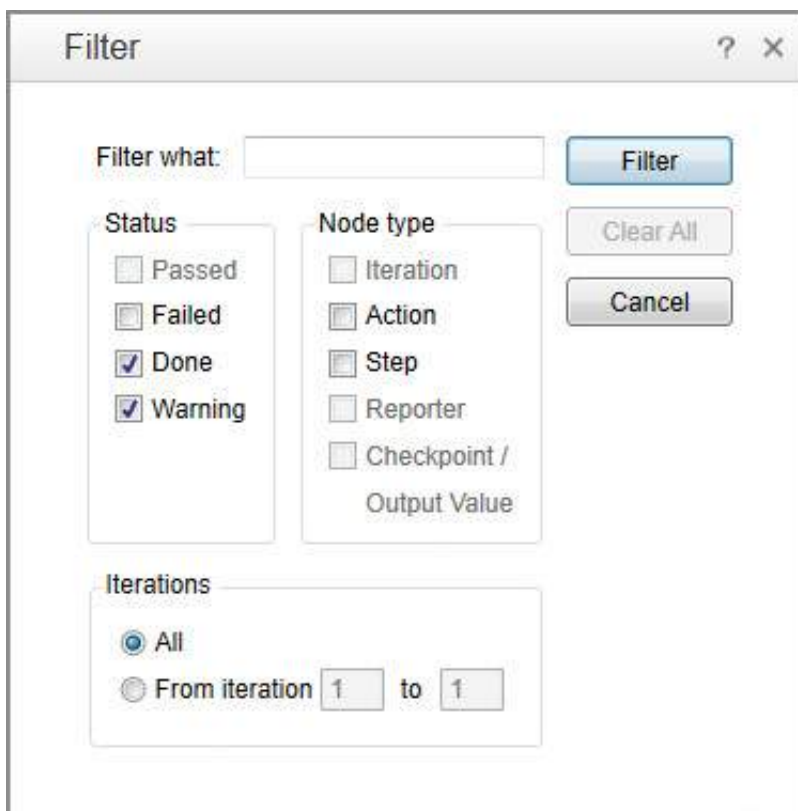
In the Results Viewer window, Navigate to "File" -> "Export to File", Export Run Results dialog box opens as shown below:



We can choose what type of report to be exported. It can be short results, Detailed Results or even we can select nodes. Upon Selecting the File Name and exporting it, the file would be saved as .HTML File

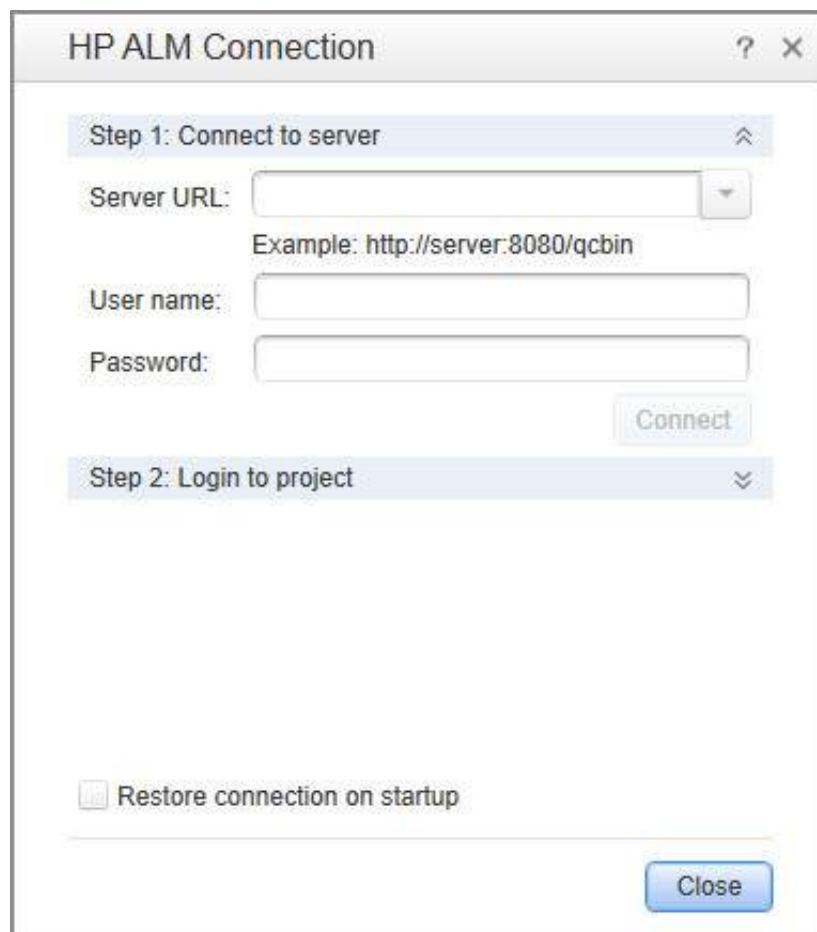
Filtering the Results:

Results can be filtered based Status, Node Type, Iterations. It can be accessed by using Filter Button in the "Test Results Window"



Raising Defects

Defects can be logged into QC directly from the Test Results Window pane by accessing "Tools" -> "Add Defect" which opens connection to ALM as shown below:



The image shows a dialog box titled "HP ALM Connection". It has a title bar with a question mark and a close button. The dialog is divided into two sections. The first section, "Step 1: Connect to server", is expanded and contains a "Server URL:" label with a text box and a dropdown arrow. Below it is an example URL: "Example: http://server:8080/qcbin". There are also "User name:" and "Password:" labels with corresponding text boxes. A "Connect" button is located to the right of the password box. The second section, "Step 2: Login to project", is collapsed. At the bottom of the dialog, there is a checkbox labeled "Restore connection on startup" and a "Close" button.

HP ALM Connection

Step 1: Connect to server

Server URL: ▼

Example: http://server:8080/qcbin

User name:

Password:

Connect

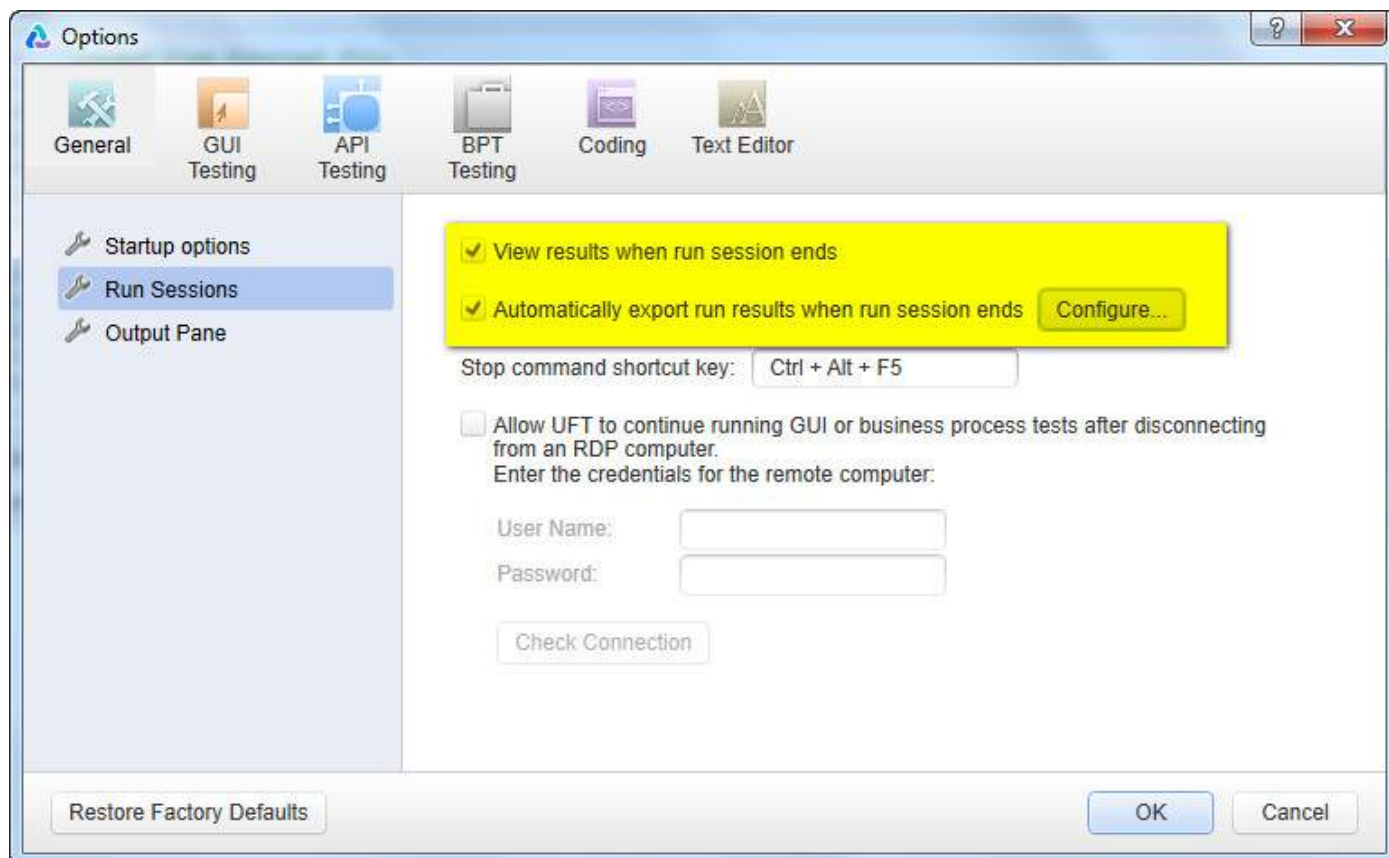
Step 2: Login to project

☐ Restore connection on startup

Close

Test Results:

The Automatic Test Results Window can be configured under "Tools" -> "Options" -> "Run Sessions" Tab. We can turn it off if required and also we can switch ON "Automatically Export Results when session Ends"

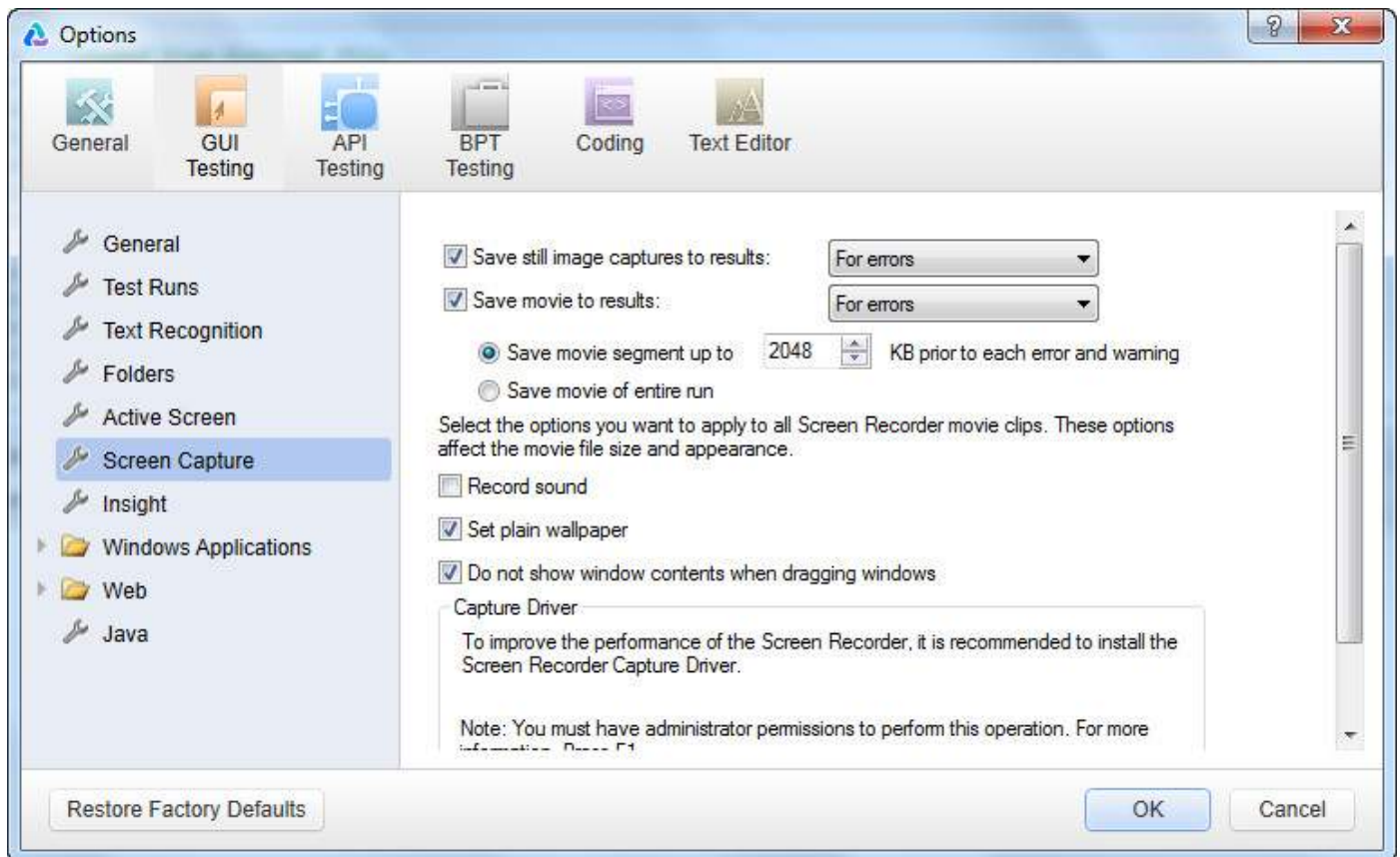


Upon Errors, the screenshot or the movie can be recorded based on the settings. The same can be configured under "Tools" -> "Options" -> "Screen Capture" Tab. We can save the screenshot or movies based on 3 conditions.

For Errors

Always

For Errors and Warnings



Working with GUI Objects:

There are various GUI objects with which QTP interacts during the script execution. Hence it is important to know the basic methods for the key GUI objects using which we will be able to work on it effectively.

Working with Text Box

Below are the methods using which we access text box during Run Time.

Set - Helps the tester to Set Values into the Text Box

Click - Clicks on the Text Box

SetSecure - Used to set the text in the password boxes securely.

WaitProperty - Waits Till the Property value becomes true.

Exist - Checks for the existence of the Text Box

GetROProperty("text") - Gets the Value of the Text Box

GetROProperty("Visible") - Returns a Boolean value if visible.

Example:

```
Browser("Math Calculator").Sync
Set Obj = Browser("Math Calculator").Page("SQR Calc").WebEdit("n")

'Clicks on the Text Box
Obj.Click

'Verify if the Object Exist - Returns Boolean value
a= obj.Exist
print a

'Set the value
obj.Set "10000" : wait(2)

'Get the Runtime Object Property - Value of the Text Box
val = obj.GetROProperty("value")
print val

'Get the Run Time Object Property - Visibility - Returns Boolean Value
x= Obj.GetROProperty("visible")
print x
```

Working with Check Box

Following are some of the key methods with which one can work with Check Box.

Set - Helps the tester to Set the checkbox value "ON" or "OFF"

Click - Clicks on the check Box. Even checks ON or OFF but user won't be sure about the status.

WaitProperty - Waits Till the Property value becomes true.

Exist - Checks for the existence of the Check Box

GetROProperty("name") - Gets the Name of the check Box

GetROProperty("Visible") - Returns a Boolean value if visible

Example:

```
'To Check the Check Box
Set Obj = Browser("Calculator").Page("Gmail").WebCheckBox("PersistentCookie")
Obj.Set "ON"

'To UnCheck the Check Box
Obj.Set "OFF"

'Verifies the Existence of the Check box and returns Boolean Value
val = Obj.Exist
```

```
print val

'Fetches the Name of the CheckBox
a= Obj.GetROProperty("name")
print a

'Verifies the visible property and returns the boolean value.
x = Obj.GetROProperty("visible")
print x
```

Working with Radio Button

Following are some of the key methods with which one can work with Radio Button.

Select(RadioButtonName) - Helps the tester to Set the Radio Box "ON"

Click - Clicks on the Radio Button. Even Radio Button ON or OFF but tester can't get the status.

WaitProperty - Waits Till the Property value becomes true.

Exist - Checks for the existence of the Radio Button

GetROProperty("name") - Gets the Name of the Radio Button

GetROProperty("Visible") - Returns a Boolean value if visible

Example:

```
'Select the Radio Button by name "YES"
Set Obj = Browser("Calculator").Page("Forms").WebRadioGroup("group1")
Obj.Select("Yes")

'Verifies the Existence of the Radio Button and returns Boolean Value
val = Obj.Exist
print val

'Returns the Outerhtml of the Radio Button
txt = Obj.GetROProperty("outerhtml")
print text

'Returns the boolean value if Radio button is Visible.
vis = Obj.GetROProperty("visible")
print vis
```

Working with Combo Box

Following are some of the key methods with which one can work with Combo Box.

Select(Value) - Helps the tester to Select the value from the ComboBox

Click - Clicks on the object.

WaitProperty - Waits Till the Property value becomes true.

Exist - Checks for the existence of the Combo Box

GetROProperty("Text") - Gets the Selected Value of the Combo Box

GetROProperty("all items") - Returns all the items in the combo Box

GetROProperty("items count") - Returns the number of items in the combo Box

Example:

```
'Get the List of all the Items from the ComboBox
Set ObjList = Browser("Math Calculator").Page("Statistics").WebList("class")
x = ObjList.GetROProperty("all items")
print x

'Get the Number of Items from the Combo Box
y = ObjList.GetROProperty("items count")
print y

'Get the text value of the Selected Item
z = ObjList.GetROProperty("text")
print z
```

Working with Buttons

Following are some of the key methods with which one can work with Buttons.

Click - Clicks on the Button.

WaitProperty - Waits Till the Property value becomes true.

Exist - Checks for the existence of the Button

GetROProperty("Name") - Gets the Name of the Button

GetROProperty("Disabled") - Returns a boolean value if enabled/disabled

Example:

```
'To Perform a Click on the Button
Set obj_Button = Browser("Math Calculator").Page("SQR").WebButton("Calc")
obj_Button.Click

'To Perform a Middle Click on the Button
```



```
obj_Button.MiddleClick
```

```
'To check if the button is enabled or disabled.Returns Boolean Value
x = obj_Button.GetROProperty("disabled")
print x
```

```
'To fetch the Name of the Button
y = obj_Button.GetROProperty("name")
print y
```

Working with webTables

In Today's web based application, webtables have become very common and testers need to understand how Web Tables work and how to perform an action on web Tables. This Topic will help you to work with the web Tables Effectively.

Statement	Description
if statement	An if statement consists of a boolean expression followed by one or more statements.
if..else statement	An if else statement consists of a boolean expression followed by one or more statements. If the condition is True, the statements under If statements are executed. If the condition is false, Else part of the script is Executed
if...elseif..else statement	An if statement followed by one or more ElseIf Statements, that consists of boolean expressions and then followed by an optional else statement, which executes when all the condition becomes false.
nested if statements	An if or elseif statement inside another if or elseif statement(s).
switch statement	A switch statement allows a variable to be tested for equality against a list of values.

html id - If the table has an id tag then it is best to make use of this property.

innerText - Heading of the Table.

sourceIndex - Fetches the Source Index of the Table

ChildItemCount - Gets the number of ChildItems present in specified Row

RowCount - Gets the number of Rows in the Table

ColumnCount - Gets the number of Columns in the Table

GetcellData - Gets the Value of the Cell based on the column and Row Index

Example:

```
Browser("Tutorials Point").Sync
' WebTable
Obj = Browser("Tutorials Point").Page("VBScript Decisions").WebTable("Statement")
' Fetch RowCount
x = Obj.RowCount
print x

' Fetch ColumnCount
y = Obj.ColumnCount(1)
print y

' Print the Cell Data of the Table
For i = 1 To x Step 1
    For j = 1 To y Step 1
        z = Obj.GetCellData(i,j)
        print "Row ID : " & i & " Column ID : " & j & " Value : " & z
    Next
Next

'Fetch the Child Item count of Type Link in a particular Cell
z = Obj.ChildItemCount(2,1,"Link")
print z
```

What are Virtual Objects?

Sometimes, application under test may contain standard window object but are NOT recognized by QTP. Under these circumstances objects can be defined as virtual object (VO) of type button, link etc so that user actions can be simulated on the virtual objects during execution.

Example

Let us say we are automating a scenario in Microsoft Word. I activated MS word application and I click on any icon in the ribbon. For example, In the Insert Ribbon, User clicks on "Picture" button. A Button is recognized as WinObject hence importance of virtual objects is pronounced.

```
Window("Microsoft Word").WinObject("Ribbon").Click 145,45
Window("Microsoft Word").WinObject("Ribbon").WinObject("Picture...").Click 170,104
```

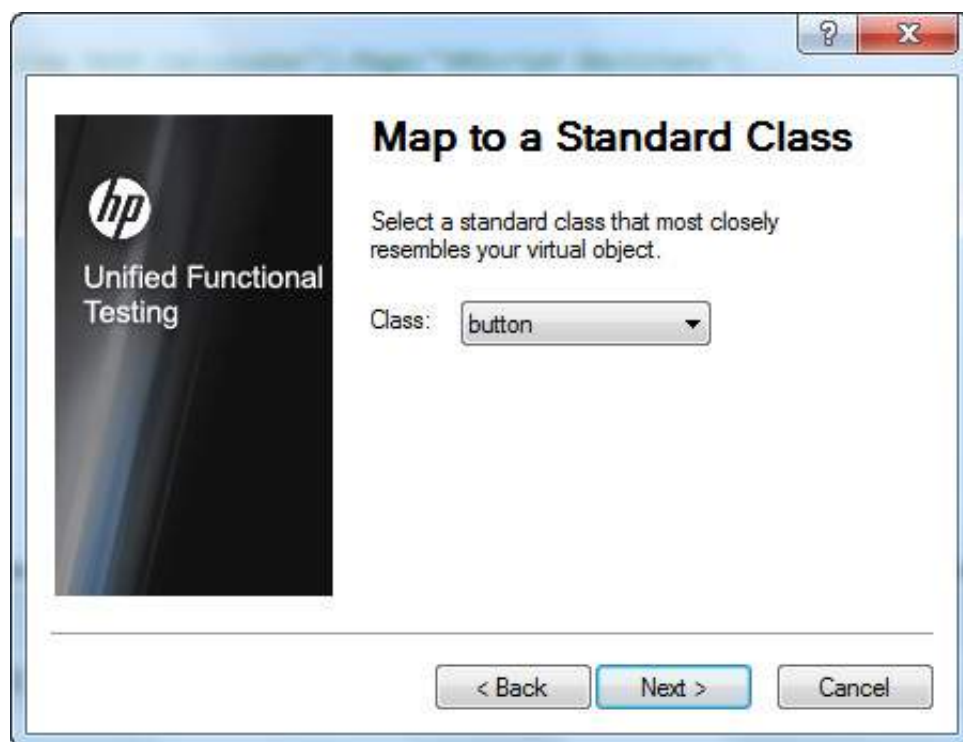
Creating a Virtual Object

Step 1 : In such scenarios, virtual Objects are created using Virtual Object Manager or New Virtual Object from "Tools" >>

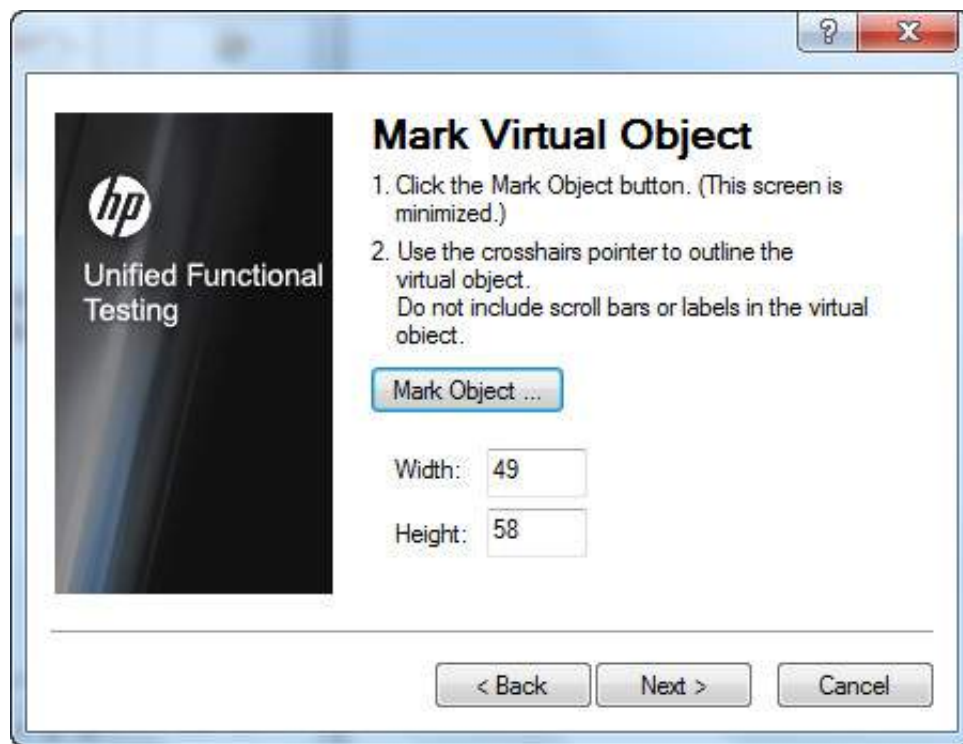
"Virtual Object" >> "New Virtual Object" and click "Next" Button



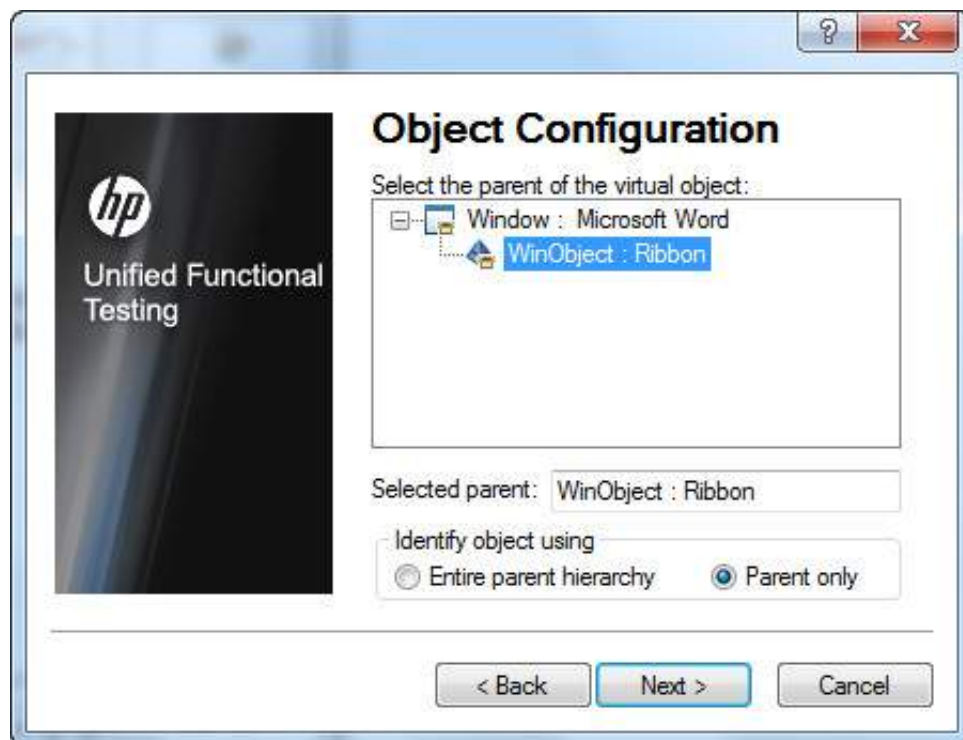
Step 2 : Map the Object against the Class Type and click "Next".



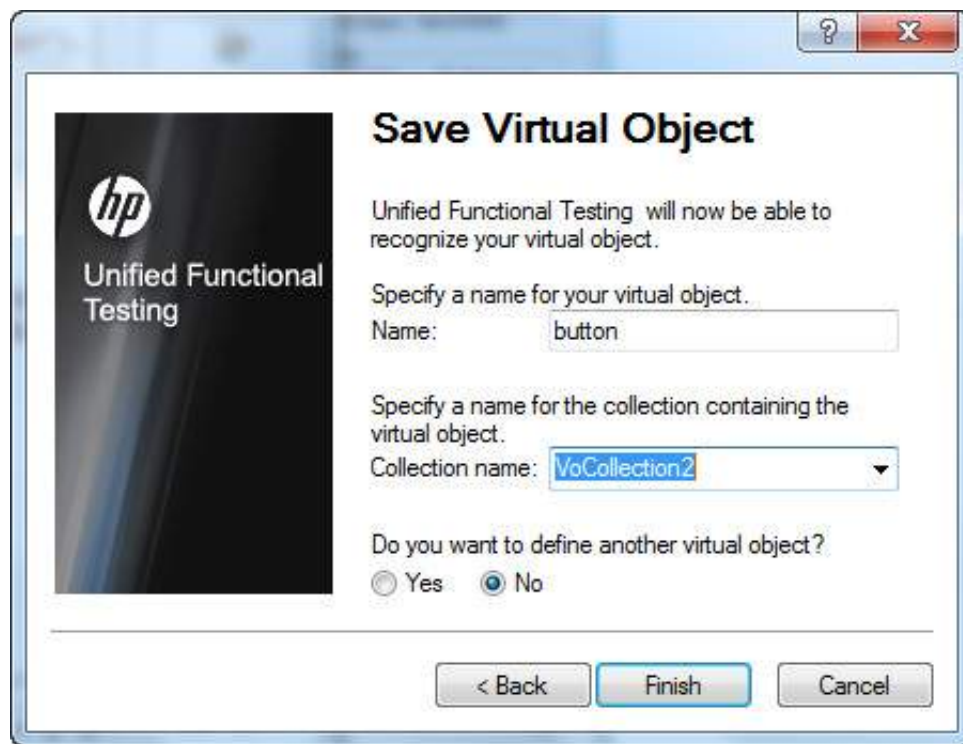
Step 3 : Click "Mark Object" Button, cross hair cursor would appear and mark the object that you would like to map and click "Next".



Step 4 : Selecting the parent of the Virtual object and click "Next".



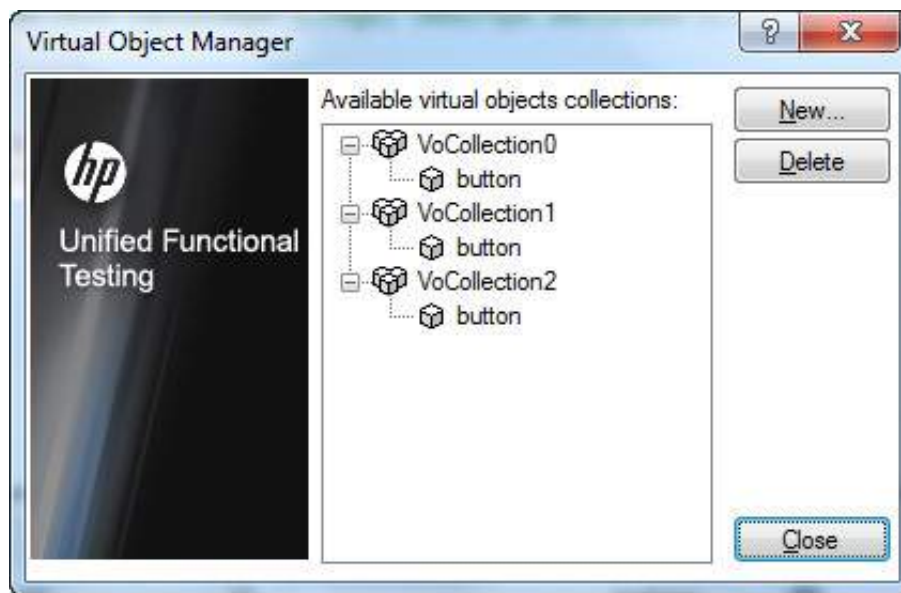
Step 5 : Name the collection in which you would like to store the virtual object and click "Finish".



Virtual object Manager

Virtual object Manager manages the collections of Virtual objects. Testers can add or Delete the Virtual Objects from the Virtual object manager.

Navigation to Virtual object Manager : "Tools" >> "Virtual Object Manager" as shown below:



Using Virtual Objects

After creating the Virtual Objects the created object can be used as shown below:

```
Window("Microsoft Word").WinObject("Ribbon").VirtualButton("button").Click
```

Virtual Object Limitations

QTP doesn't support virtual objects for analog or low-level recording.

Checkpoints cannot be added on Virtual Objects.

Virtual Object are NOT controlled by Object Repository.

Though we map an object to a particular class (button or List), all the methods of the native objects are not supported by Virtual objects.

Object Spy cannot be used on Virtual Object.

The test execution will fail if the screen resolution changes as the co-ordinates change.

Application Window should be of same screen size so that Virtual objects are captured correctly.

Accessing Databases:

As such QTP doesn't provide any built-in support to connect to database, however using VBScript testers will be able to connect and interact with databases using ADODB objects

ADODB has 4 properties or methods with which we will be able to work with the databases.

ADODB.Connection - Used to establish a connection to the Database

ADODB.Command - Used to execute a SQL command(Queries or Stored Procedures)

ADODB.Fields - Used to fetch a particular column from a record set after executing a query/stored proc

ADODB.Recordset - Used to fetch data from a database

How to connect to Database?

Databases can be connected using Connection strings. Each database differ the way

we connect to the them, however the connection strings can be build with the help of <http://www.connectionstrings.com/>

Now Let us see how to connect to the database with the following parameters.

Database Type - MSSQL SERVER

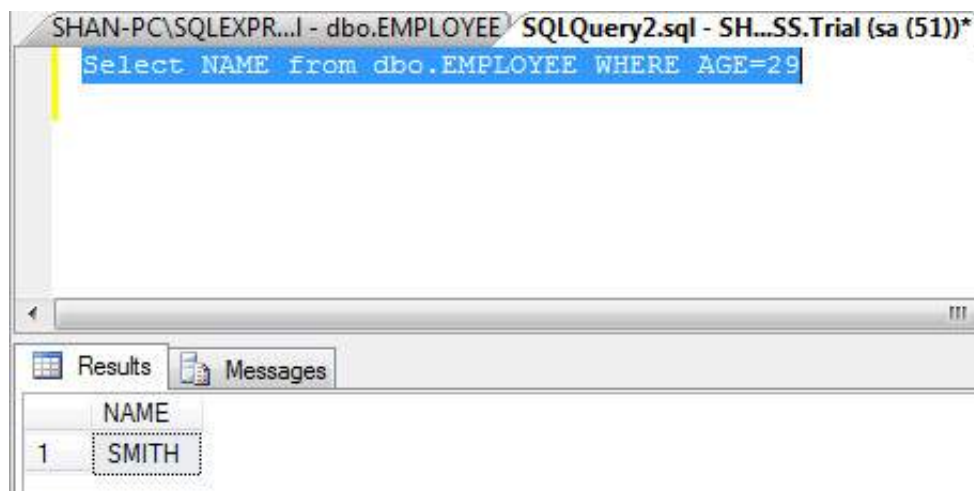
Server Name - SQLEXPRESS

Database Name - Trial

User Id - sa

password - Password123

The Output of the Query is shown in the SQL Server Management Studio as follows:



```
Dim objConnection
'Set Adodb Connection Object
Set objConnection = CreateObject("ADODB.Connection")
Dim objRecordSet

'Create RecordSet Object
Set objRecordSet = CreateObject("ADODB.Recordset")

Dim DBQuery 'Query to be Executed
DBQuery = "Select NAME from dbo.EMPLOYEE where AGE = 29"

'Connecting using SQL OLEDB Driver
objConnection.Open "Provider=sqloledb.1;Server=.\SQLEXPRESS;User Id=sa;Password=Password123;Data Source=SHAN-PC\SQLEXPRESS;Initial Catalog=SH...SS.Trial"

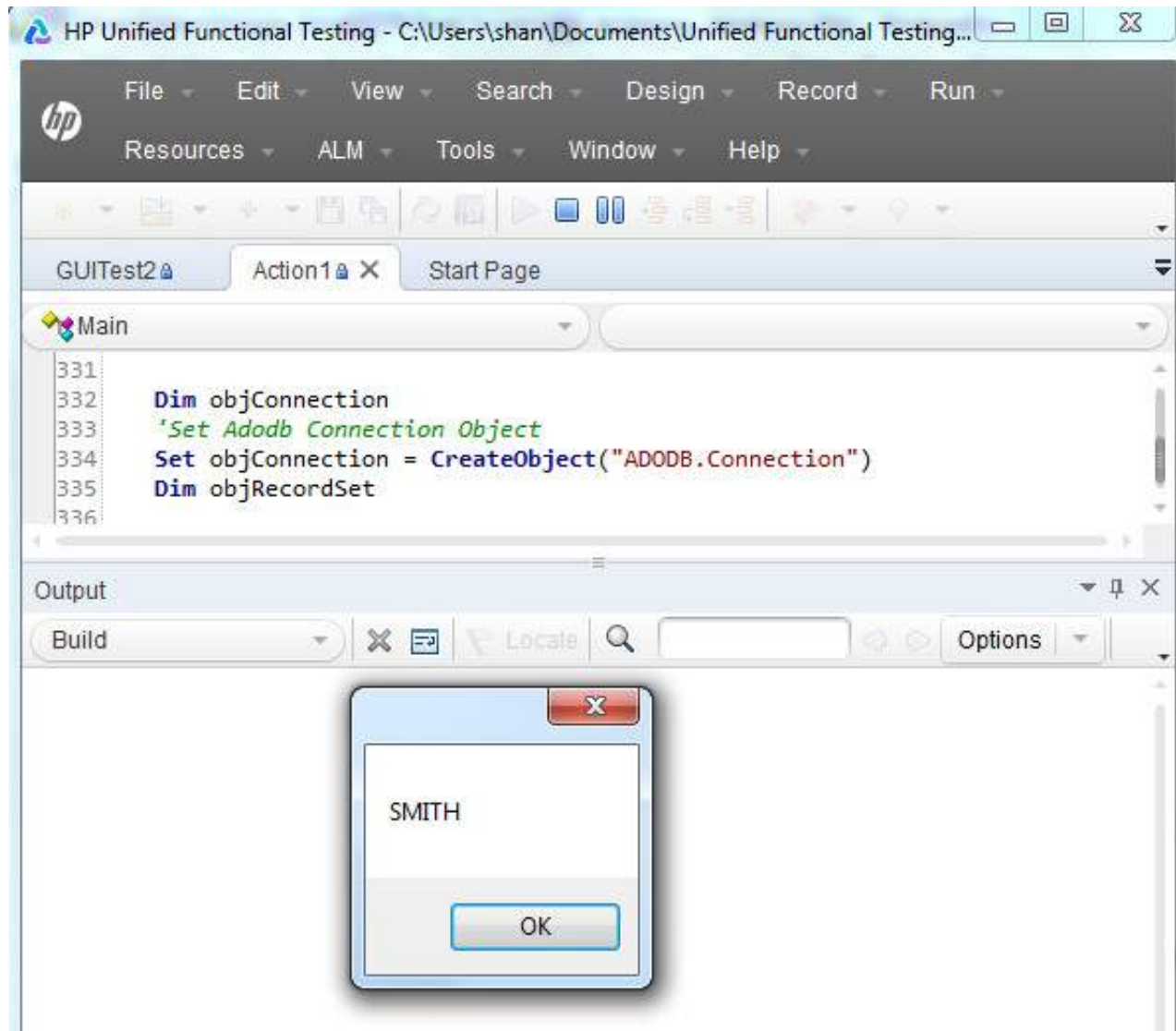
'Execute the Query
objRecordSet.Open DBQuery,objConnection

'Return the Result Set
Value = objRecordSet.fields.item(0)
msgbox Value
```

```
' Release the Resources  
objRecordSet.Close  
objConnection.Close  
  
Set objConnection = Nothing  
Set objRecordSet = Nothing
```

Result

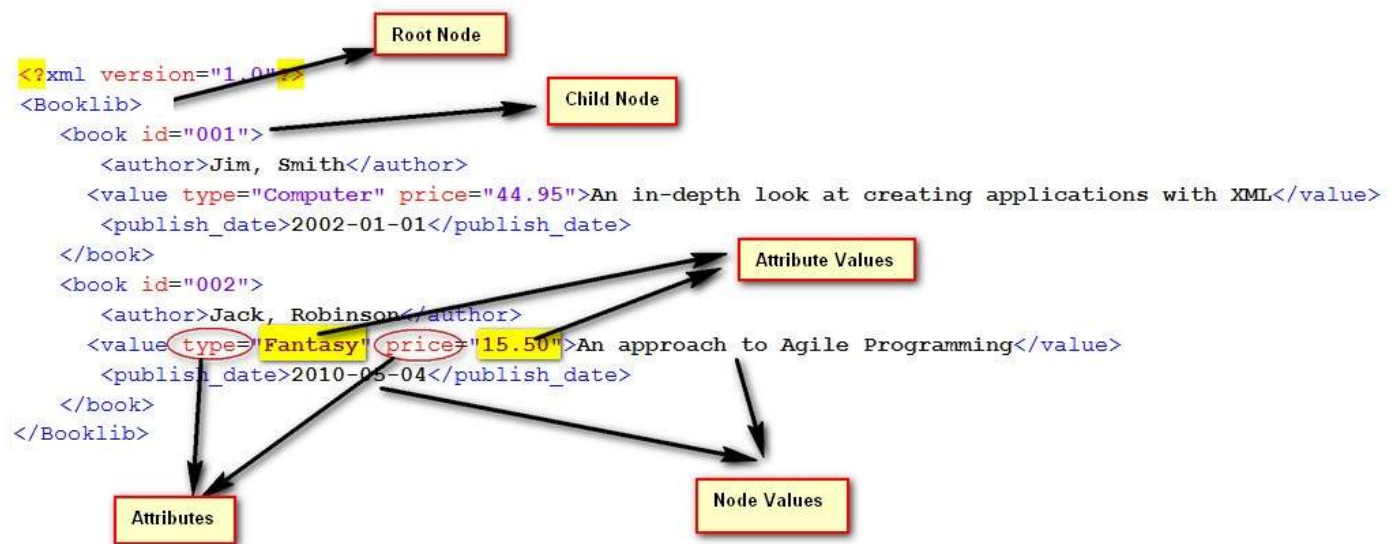
Upon Executing the above script the output is shown in the message box as shown below:



What is an XML?

XML is a markup language designed for how to store data that is in the form that both human and machine readable format. Using XML, data can also be easily exchanged between computer and database systems.

Sample XML and their key elements are represented below:



Accessing XML

```

Const XMLDataFile = "C:\TestData.xml"
Set xmlDoc = CreateObject("Microsoft.XMLDOM")
xmlDoc.Async = False
xmlDoc.Load(XMLDataFile)

' Getting the number of Nodes (books)
Set nodes = xmlDoc.SelectNodes("/bookstore/book")
Print "Total books: " & nodes.Length    ' Displays 2

' get all titles
Set nodes = xmlDoc.SelectNodes("/Booklib/book/value/text()")

' get their values
For i = 0 To (nodes.Length - 1)
    Title = nodes(i).NodeValue
    Print "Title is" & (i + 1) & ": " & Title
Next
  
```

Comparing XML

We can compare Two given xml's.

```

Dim xmlDoc1
Dim xmlDoc2

' Load the XML Files
Set xmlDoc1 = XMLUtil.CreateXMLFromFile ("C:\File1.xml")
Set xmlDoc2 = XMLUtil.CreateXMLFromFile ("C:\File2.xml")

' Use the compare method of the XML to check if they are equivalent
Comp = xmlDoc1.Compare (xmlDoc1, xmlDoc2)
  
```

```
'Returns 1 if the two files are the same
If Comp = 1 Then
    MsgBox "XML Files are the Same"
Else
    MsgBox "XML Files are Different"
End if
```

Descriptive Programming:

QTP scripts can execute only if the objects are present in the Object Repository. If the descriptions of the Objects are created using Descriptive programming when testers want to perform an operation on an object that is not present in the object repository.

When objects in the application are very dynamic in nature.

When the Object Repository grows big which will result in poor Performance as the size of the Object Repository increases.

When the framework is built such that it has been decided not to use Object Repository at all.

When testers want to perform an action on the application at run-time without having the knowledge of object's unique properties.

Syntax

There are two ways to script using Descriptive Programming technique. They are

1. Description Objects
2. Description Strings

Description Objects

Script is developed using description Objects that depends upon the properties used and their corresponding values. Then these descriptions are used to build the script.

```
'Creating a description object
Set btnCalc = Description.Create()

'Add descriptions and properties
btnCalc("type").value = "Button"
btnCalc("name").value = "calculate"
btnCalc("html tag").value = "INPUT"
```

```
' Use the same to script it
Browser("Math Calc").Page("Num Calculator").WebButton(btncalc).Click
```

Description Strings

The description of the objects are developed using the properties and values as strings as shown below.

```
Browser("Math Calc").Page("Num Calculator").WebButton("html tag:=INPUT","type:=Button","name:=calc
```

Child Objects

QTP provides the ChildObjects method which enables us to create a collection of objects. The parent objects preceeds ChildObjects.

```
Dim oDesc
Set oDesc = Description.Create
oDesc("micclass").value = "Link"

'Find all the Links
Set obj = Browser("Math Calc").Page("Math Calc").ChildObjects(oDesc)

Dim i
'obj.Count value has the number of links in the page
For i = 0 to obj.Count - 1
    'get the name of all the links in the page
    x = obj(i).GetROProperty("innerHTML")
    print x
Next
```

Ordinal Identifiers

Descriptive programming is used to script based on ordinal identifiers which will enable QTP to act on those objects when two or more objects have same properties.

```
' Using Location
Dim Obj
Set Obj = Browser("title:=.*google.*").Page("micclass:=Page")
Obj.WebEdit("name:=Test","location:=0").Set "ABC"
Obj.WebEdit("name:=Test","location:=1").Set "123"

' Index
Obj.WebEdit("name:=Test","index:=0").Set "1123"
Obj.WebEdit("name:=Test","index:=1").Set "2222"

' Creation Time
Browser("creationtime:=0").Sync
Browser("creationtime:=1").Sync
```

```
Browser("creationtime:=2").Sync
```

Automation Object Model:

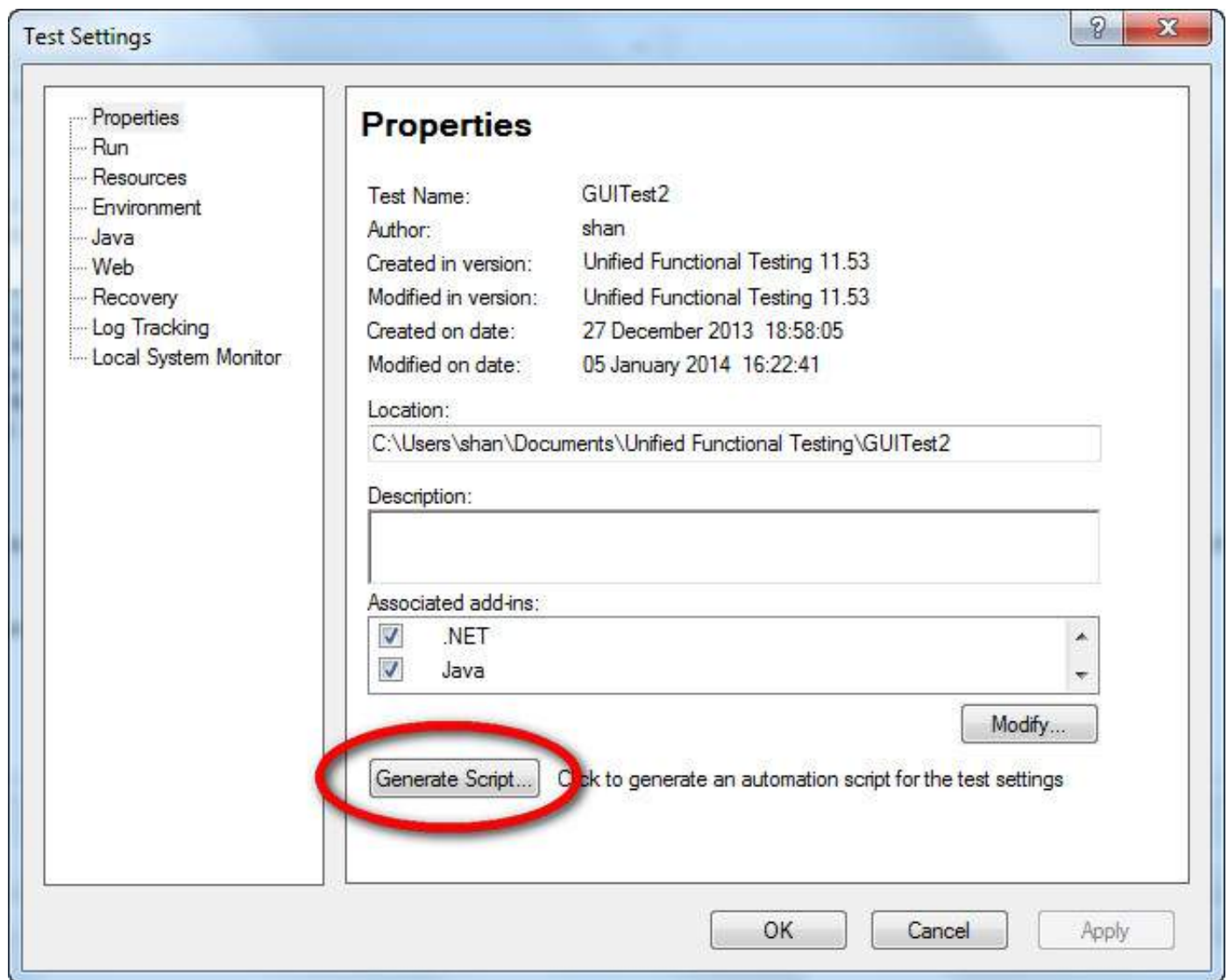
QTP itself can be automated using the COM interface that is provided by Hp-QTP. Automation object model is a set of objects, methods, and properties that helps testers to control the configuration settings and execute the scripts using the QTP interface. The Key Configurations/actions that can be controlled are listed below but not limited to

1. Loads all the required add-ins for a test
2. Makes QTP visible while execution
3. Opens the Test using the specified location
4. Associates Function Libraries
5. Specifies the Common Object Sync Time out
6. Start and End Iteration
7. Enable/Disable Smart Identification
8. On Error Settings
9. Data Table Path
10. Recovery Scenario Settings
11. Log Tracking Settings

QTP 11.5x provides an exclusive documentation on Automation Object model that can be referred by navigating to "Start" >> "All Programs" >> "HP Software" >> "HP Unified Functional Testing" >> "Documentation" >> "Unified Functional Testing Automation Reference"

Generate AOM Script:

Tester Can generate AOM Script from QTP itself using "Generate Script" Option. Navigate to "Run" >> "Settings" >> "Properties" Tab >> "Generate Script" as shown below:



Example

```
' A Sample Script to Demonstrate AOM
Dim App 'As Application
Set App = CreateObject("QuickTest.Application")
App.Launch
App.Visible = True

App.Test.Settings.Launchers("Web").Active = False
App.Test.Settings.Launchers("Web").Browser = "IE"
App.Test.Settings.Launchers("Web").Address = "http://easycalculation.com/"
App.Test.Settings.Launchers("Web").CloseOnExit = True

App.Test.Settings.Launchers("Windows Applications").Active = False
App.Test.Settings.Launchers("Windows Applications").Applications.RemoveAll
App.Test.Settings.Launchers("Windows Applications").RecordOnQTDendants = True
App.Test.Settings.Launchers("Windows Applications").RecordOnExplorerDendants = False
App.Test.Settings.Launchers("Windows Applications").RecordOnSpecifiedApplications = True

App.Test.Settings.Run.IterationMode = "rngAll"
App.Test.Settings.Run.StartIteration = 1
App.Test.Settings.Run.EndIteration = 1
```

```
App.Test.Settings.Run.ObjectSyncTimeOut = 20000
App.Test.Settings.Run.DisableSmartIdentification = False
App.Test.Settings.Run.OnError = "Dialog"

App.Test.Settings.Resources.DataTablePath = ""
App.Test.Settings.Resources.Libraries.RemoveAll

App.Test.Settings.Web.BrowserNavigationTimeout = 60000
App.Test.Settings.Web.ActiveScreenAccess.UserName = ""
App.Test.Settings.Web.ActiveScreenAccess.Password = ""

App.Test.Settings.Recovery.Enabled = True
App.Test.Settings.Recovery.SetActivationMode "OnError"
App.Test.Settings.Recovery.Add "D:\GUITest2\recover_app_crash.qrs", "Recover_Application_Crash", 1
App.Test.Settings.Recovery.Item(1).Enabled = True

' .....
' System Local Monitoring settings
' .....
App.Test.Settings.LocalSystemMonitor.Enable = false
' .....
' Log Tracking settings
' .....

With App.Test.Settings.LogTracking
    .IncludeInResults = False
    .Port = 18081
    .IP = "127.0.0.1"
    .MinTriggerLevel = "ERROR"
    .EnableAutoConfig = False
    .RecoverConfigAfterRun = False
    .ConfigFile = ""
    .MinConfigLevel = "WARN"
End With
```

What is a Software Framework?

A Framework defines a set of guidelines/best practices that enforces a set of standards which makes it easy to use for the end users to work with. There are different types of automation frameworks and the most common ones are listed below:

Keyword-Driven Framework

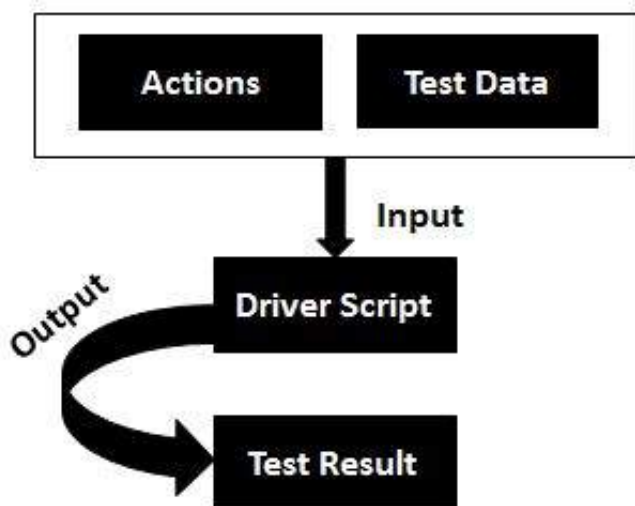
Data-Driven Framework

Hybrid Framework

Keyword-Driven Framework

Keyword driven testing is a type of functional automation testing framework which also known as table-driven testing or action word based testing.

In Keyword-driven testing we use a table format, usually a spreadsheet, to define keywords or action words for each function that we would like to execute.



Advantages:

- It is best suited for novice or a non technical tester.

- Enables writing tests in a more abstract manner using this approach.

- Keyword driven testing allows automation to be started earlier in the SDLC even before a stable build is delivered for testing.

- There is a high degree of reusability.

Disadvantages:

- Initial investment in developing the keywords and its related functionalities might take longer.

- It might act as a restriction to the technically abled testers.

Data Driven Framework

Data-driven testing is creation of test scripts where test data and/or output values are read from data files instead of using the same hard-coded values each time the test

runs. This way testers can test how the application handles various inputs effectively. It can be any of the below data files.

datapools

Excel files

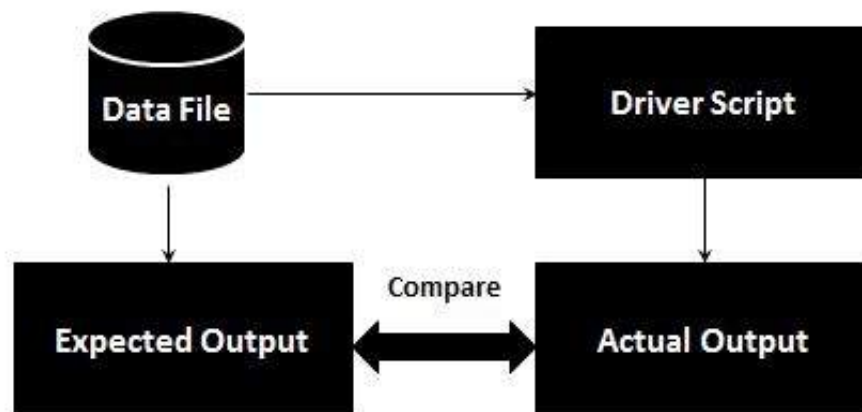
ADO objects

CSV files

ODBC sources

Flow Diagram:

Data Driven Testing can be best understood by the following diagram:



Advantages:

Data driven framework results in less amount of code.

Offers greater flexibility for maintaining and fixing the scripting issues.

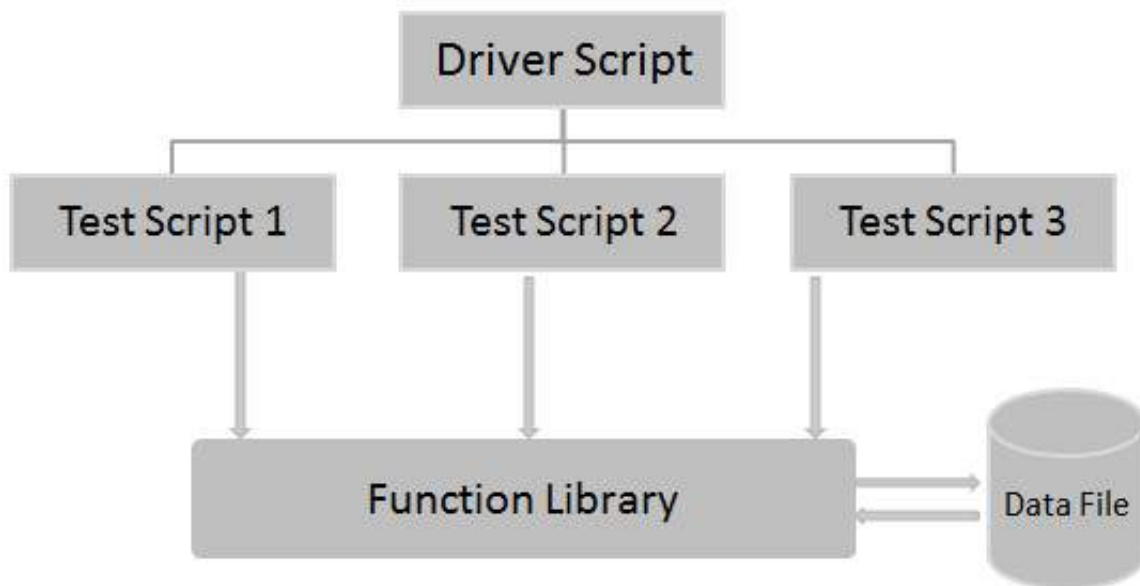
Test Data can be developed

Disadvantages:

Each script needs to be different to understand different sets of data.

Hybrid Framework

Hybrid Framework is a combination of Keyword driven and data Driven framework that can be best described using the following flow diagram.



Affecting Factors

Following are the parameters one should take into account while developing the framework. The affects factors are listed below

Framework Files Should Support Versioning Controlling Software such as SVN, CVS, MS Source Control

Framework should support executing the scripts in different environments viz- QA, SAT, DEV

Upon Object Changes, Scripts should execute with minimal changes.

Framework should configure itself and take care of prerequisite such as creating folders/databases.

Framework Should have Robust Reporting Structure so that issues in the script/application can be easily spotted

Framework Should have greater flexibility so that it should be easy to use

Framework Should follow coding standards so that files, functions and history of changes are maintained correctly.

Designing a Framework

Let us design a simple framework by taking a sample application. We will automate few scenarios of the application under test and write reusable functions.

Please click the below link to learn more about designing frameworks. - **QTP - Designing Framework**

[⬅ Previous Page](#)[Next Page ➡](#)

Advertisements



[Write for us](#) [FAQ's](#) [Helping](#) [Contact](#)

© Copyright 2016. All Rights Reserved.