About the Tutorial

HP QuickTest Professional (QTP), an automated functional testing tool that helps testers to perform automated regression testing in order to identify any gaps, errors/defects in contrary to the actual/desired results of the application under test.

This tutorial will give you an in-depth understanding on HP QuickTest Professional, its way of usage, record and play back of tests, object repository, actions, checkpoints, sync points, debugging, test results etc. and other related terminologies.

Audience

This tutorial is designed for Software Testing Professionals with a need to understand the QTP in detail along with its simple overview, and practical examples. This tutorial will give you enough ingredients to start with QTP from where you can take yourself at higher level of expertise.

Prerequisites

Before proceeding with this tutorial, you should have a basic understanding of software development life cycle (SDLC). A basic understanding of VBScript is also required. You can also go through the basics of VBScript here.

Copyright & Disclaimer

© Copyright 2016 by Tutorials Point (I) Pvt. Ltd.

All the content and graphics published in this e-book are the property of Tutorials Point (I) Pvt. Ltd. The user of this e-book is prohibited to reuse, retain, copy, distribute or republish any contents or a part of contents of this e-book in any manner without written consent of the publisher.

We strive to update the contents of our website and tutorials as timely and as precisely as possible, however, the contents may contain inaccuracies or errors. Tutorials Point (I) Pvt. Ltd. provides no guarantee regarding the accuracy, timeliness or completeness of our website or its contents including this tutorial. If you discover any errors on our website or in this tutorial, please notify us at contact@tutorialspoint.com
Table of Contents

About the Tutorial .............................................................................................................. i
Audience ............................................................................................................................. i
Prerequisites ....................................................................................................................... i
Copyright & Disclaimer ................................................................................................. i
Table of Contents ............................................................................................................. ii

1. QTP – Introduction ........................................................................................................ 1
   Testing Tools ................................................................................................................... 1
   QTP – History and Evolution ......................................................................................... 2
   Advantages ..................................................................................................................... 3
   Disadvantages ............................................................................................................... 3

2. QTP – Automated Testing Process .............................................................................. 4

3. QTP – Environment Setup ........................................................................................... 6
   Installation Procedure ................................................................................................... 6
   Launching UFT and Add ins Page ................................................................................ 13

4. QTP – Record and Playback ....................................................................................... 15
   Significance of Record and Playback: .......................................................................... 19
   Modes of Recording ..................................................................................................... 19
   How to Choose Recording Modes .............................................................................. 19

5. QTP – Object Repository ............................................................................................ 21
   QTP – Object Spy ......................................................................................................... 21
   QTP – Working with Object Repository ...................................................................... 24
   QTP – Object Repository Types .................................................................................. 27
   QTP – User Defined Objects ....................................................................................... 29
   QTP – Object Repository as XML ............................................................................. 30
   QTP – Comparing and Merging ORs ......................................................................... 31
   Merging Object Repositories ..................................................................................... 32
   QTP – Ordinal Identifiers ............................................................................................ 34
   QTP – Child Objects .................................................................................................... 36

6. QTP – Actions ............................................................................................................... 38
   Working with Actions .................................................................................................... 39
   QTP – Call to New Action ............................................................................................ 39
   QTP – Call to Copy of Action ...................................................................................... 41
   QTP – Call to Existing Action ..................................................................................... 44

7. QTP – Datatables ......................................................................................................... 47
   DataTable Operations .................................................................................................. 48
   QTP – DataTable Object Methods ............................................................................ 48
   QTP – DTParameter Object Properties ..................................................................... 50
   QTP – DTSheet Methods ............................................................................................. 51

8. QTP – CheckPoints ..................................................................................................... 53
   Types of Checkpoints .................................................................................................... 53
   Inserting CheckPoint .................................................................................................... 53
   Viewing Checkpoint Properties .................................................................................. 55
<table>
<thead>
<tr>
<th>Chapter</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>9.</td>
<td>QTP – Synchronization</td>
<td>57</td>
</tr>
<tr>
<td></td>
<td>Ways to Insert Sync Point</td>
<td>57</td>
</tr>
<tr>
<td></td>
<td>Default Synchronization</td>
<td>60</td>
</tr>
<tr>
<td>10.</td>
<td>QTP – Smart Identification</td>
<td>61</td>
</tr>
<tr>
<td></td>
<td>Enabling Smart Identification for an Object</td>
<td>61</td>
</tr>
<tr>
<td>11.</td>
<td>QTP – Debugging</td>
<td>65</td>
</tr>
<tr>
<td></td>
<td>Options in Break Point</td>
<td>65</td>
</tr>
<tr>
<td>12.</td>
<td>QTP – Error Handling</td>
<td>67</td>
</tr>
<tr>
<td></td>
<td>Error Types</td>
<td>67</td>
</tr>
<tr>
<td></td>
<td>Handling Run-Time Errors</td>
<td>69</td>
</tr>
<tr>
<td>13.</td>
<td>QTP – Recovery Scenarios</td>
<td>71</td>
</tr>
<tr>
<td>14.</td>
<td>QTP – Environment Variables</td>
<td>77</td>
</tr>
<tr>
<td></td>
<td>Types of Environment Variables</td>
<td>77</td>
</tr>
<tr>
<td></td>
<td>Environment Variables – Supported Methods</td>
<td>79</td>
</tr>
<tr>
<td>15.</td>
<td>QTP – Library Files</td>
<td>82</td>
</tr>
<tr>
<td></td>
<td>Associating Function Libraries</td>
<td>82</td>
</tr>
<tr>
<td>16.</td>
<td>QTP – Automated Testing Process</td>
<td>84</td>
</tr>
<tr>
<td></td>
<td>Operations performed in Test Results</td>
<td>85</td>
</tr>
<tr>
<td></td>
<td>Raising Defects</td>
<td>86</td>
</tr>
<tr>
<td></td>
<td>Test Results</td>
<td>86</td>
</tr>
<tr>
<td>17.</td>
<td>QTP – Working with GUI Objects</td>
<td>88</td>
</tr>
<tr>
<td></td>
<td>Working with Text Box</td>
<td>88</td>
</tr>
<tr>
<td></td>
<td>Working with Check Box</td>
<td>89</td>
</tr>
<tr>
<td></td>
<td>Working with Radio Button</td>
<td>90</td>
</tr>
<tr>
<td></td>
<td>Working with Combo Box</td>
<td>90</td>
</tr>
<tr>
<td></td>
<td>Working with Buttons</td>
<td>91</td>
</tr>
<tr>
<td></td>
<td>Working with webTables</td>
<td>92</td>
</tr>
<tr>
<td>18.</td>
<td>QTP – Virtual Objects</td>
<td>94</td>
</tr>
<tr>
<td></td>
<td>Creating a Virtual Object</td>
<td>94</td>
</tr>
<tr>
<td></td>
<td>Virtual Object Manager</td>
<td>97</td>
</tr>
<tr>
<td></td>
<td>Virtual Object Limitations</td>
<td>97</td>
</tr>
<tr>
<td>19.</td>
<td>QTP – Accessing Databases</td>
<td>98</td>
</tr>
<tr>
<td></td>
<td>How to connect to Database?</td>
<td>98</td>
</tr>
<tr>
<td>20.</td>
<td>QTP – Working with XML</td>
<td>101</td>
</tr>
<tr>
<td></td>
<td>Accessing XML</td>
<td>101</td>
</tr>
<tr>
<td></td>
<td>Comparing XML</td>
<td>102</td>
</tr>
<tr>
<td>21.</td>
<td>QTP – Descriptive Programming</td>
<td>103</td>
</tr>
<tr>
<td></td>
<td>Syntax</td>
<td>103</td>
</tr>
<tr>
<td></td>
<td>Child Objects</td>
<td>104</td>
</tr>
<tr>
<td></td>
<td>Ordinal Identifiers</td>
<td>104</td>
</tr>
</tbody>
</table>
22. QTP – Automation Object Model .......................................................................................... 105
   Generate AOM Script .............................................................................................................. 105

23. QTP – Frameworks .............................................................................................................. 108
   Keyword-Driven Framework .................................................................................................. 108
   Data Driven Framework ....................................................................................................... 109
   Flow Diagram ........................................................................................................................ 109
   Hybrid Framework .............................................................................................................. 110
   Affecting Factors .................................................................................................................. 110

24. QTP – Designing a Framework ............................................................................................ 111
   Master Driver Script .............................................................................................................. 111
   Library Files ........................................................................................................................... 115
   Object Repository .................................................................................................................. 120
   DataTable .............................................................................................................................. 121
   The Execution Log ................................................................................................................ 122
1. 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 as a part of the Windows OS. The Current version of VBScript is 5.8, which is available as a part of Win 7. VBScript is NOT an object-oriented language but an 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 technology used

**Types of Tools**

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

**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 the 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.

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

- Developing automated tests using VBScript does not require a highly skilled coder and is relatively easy when compared to 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 an HP product, 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 the CPU & RAM.
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 in this tutorial.
• **Build 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. It is performed with an end-to-end scenario, which ensures that the major functionalities can be automated.

• **Develop Automation Framework** - After building the POC, framework development is carried out, which is a crucial step for the success of any test automation project. Framework should be built 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 is a commercial tool and the trial version can be downloaded from HP site directly. Only the current version, which is Unified functional testing (11.5x) is available for download. Following is the URL from where the trial version can be downloaded.


### Installation Procedure

**Step 1** – Click the "Trials and Demos" link and select "HP Unified Functional Testing 11.50 CC English SW E-Media Evaluation" as shown below:
**Step 2** - After Selecting "HP Unified Functional Testing 11.50", the download wizard opens. Fill in the Personal details and click Next.

**Step 3** - Read the 'Terms of Use' and click "NEXT".
Step 4 – The Download window opens. Now, click the "Download" button.

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 Setup 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 - After selecting the required plugins for installation, click Next. After the completion of 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** – After the completion of Additional Installation Requirements, a tick mark is shown, which in turn states that the components are installed successfully. Now, click Close.
Launching UFT and Add ins Page

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

![Image of Start Menu](image)

**Step 2** - The license page appears. You can click Continue as we have installed the trial license.

![Image of License Warning](image)
**Step 3** - The Add-ins dialog box opens for the user to select the required add-ins.

**Note**: Do not load all the add-ins but just the required add-ins and click OK.

![Add-ins dialog box](image)

**Step 4** - After loading the required add-ins, the UFT 11.5 tool opens for the user and the first glimpse of the UFT looks, as shown below:

![UFT 11.5 interface](image)
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 of the tool, whether it can support the technology or not, if the initial settings are done correctly.

Steps for Record and Playback are as follows:

**Step 1**: Click "New" test from the Start Page as shown below:
Step 2: Clicking "New" Link, a new test window opens and the user needs to select the test type. Select "GUI Test", give a name for the test and the location where it needs to be saved.

Step 3: Once a New test is created, the new test screen opens as shown below. Now, click the "Action1" tab, which is created with 1 action by default.
**Step 4**: Click the "Record" Menu and select "Record and Run Settings" as shown below:

![Record and Run Settings Dialog Box](image)

**Step 5**: The Record and Run Settings dialog box opens and based on the type of application, one can select Web, Java, or Windows Applications. For Example, here, we will record a Web Based Application (http://easycalculation.com/).

![Record and Run Settings Dialog Box](image)
**Step 6:** Click Record. The Internet Explorer opens automatically with the web address `http://easycalculation.com/` as per the settings. Click the "Numbers" link under "Algebra" and key in a number and hit "Calculate". After completion of the action, click the "Stop" button in the record panel. You will notice that the script is generated as shown below:

```
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").WebButton("calculate").click
```

**Step 7:** Now playback the script by clicking the playback button. The Script replays and the 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:**

- It is used as the preliminary investigation method to verify if UFT can support the technology/application.
- Used to create a test a basic functionality of an application or feature that does not require long-term maintenance.
- It can be used for recording both mouse movements and keyboard inputs.

**Modes of Recording**

- **Normal Recording**: This is the default Recording mode that records the objects and the operations performed on the application under test.
- **Analog Recording**: This records not only the keyboard actions but also the mouse movements relative to the screen or the application window.
- **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.
- **Insight Recording**: UFT records operations, based on its appearance and not based on its native properties.

**How to Choose Recording Modes**

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

You will see that the following scenario is recorded in all the modes and the same action has been recorded under various circumstances.

- Launch IE - http://easycalculation.com/
- Click "Numbers" under "Algebra"
- Click "Square Root" link
Enter a value to calculate the square root. Let us say 10
Hit Calculate

Script Recorded under Default, Analog and Low Level recording mode is given below:

' 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 following topics to know more about Object Repository and its associated features.

<table>
<thead>
<tr>
<th>Topic</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object Spy and its Features</td>
<td>To understand the usage of object spy and its associated functionalities.</td>
</tr>
<tr>
<td>Working with Object Repository</td>
<td>Adding, editing, deleting objects from an Object Repository and its associated functionalities.</td>
</tr>
<tr>
<td>Types of Object Repository</td>
<td>Deals with shared Object and local Object Repository and their context with respect to scripting.</td>
</tr>
<tr>
<td>User-defined Objects</td>
<td>Deals with the circumstances to use the User-Defined Objects.</td>
</tr>
<tr>
<td>Object Repository in XML</td>
<td>Deals with converting OR’s to XML and using the object Repository as XML.</td>
</tr>
<tr>
<td>Comparing and Merging OR</td>
<td>Operations such as Compare OR’, Merge OR's to effectively work with Object Repository.</td>
</tr>
<tr>
<td>Ordinal Identifiers</td>
<td>Circumstances where the ordinal identifiers are used and its advantages.</td>
</tr>
<tr>
<td>Child Objects</td>
<td>Using Child Objects for effective scripting</td>
</tr>
</tbody>
</table>

QTP – Object Spy

Object Spy is a utility/option within QTP to add objects to the Object Repository. Object Spy can be accessed from the tool bar as shown below:
**Step 1**: Clicking the Object Spy icon, the Object Spy Dialog box opens. The Objects can be added to the repository on clicking the pointing hand.
**Step 2:** After Spying the object, the object hierarchy will be shown. Let us say, we are spying the "Numbers" link at "http://easycalculation.com/". The Object properties will be as shown below.

**Step 3:** After Spying an object, click the "Highlight" option to highlight the object in the application.

**Step 4:** For adding the object into the Object Repository, click the "Add objects" button in the Object spy dialog.

**Step 5:** The properties and its values are displayed for the selected object in the dialog box, which should be unique for QTP, to recognize the objects while the script executes.
Step 6: The supported operations on the object can be retrieved by clicking the operation tab. Operations such as "click" for a button, "Set" for a text box are retrieved from the "operations" tab as shown below:

![Object Spy screenshot]

**QTP – Working with Object Repository**

**Adding Objects into OR**

After Spying the object, adding the objects into Repository is the first step. The script can execute successfully if and only if the objects are added into the Object Repository. Upon Clicking "Add Objects to OR", the objects are added into Object repository.

Even when a user does a recording, the objects and its properties are captured automatically. Hence, we are able to replay the script successfully.

**Object Repository – Features**

**Step 1:** After adding objects to the OR, we can verify by navigating to "Resources" -> "Object Repository". The Object Repository Window opens and we can locate the added object in the Repository as shown below:
Step 2: One can add the properties additionally apart from the default ones by clicking the "+" button and remove it using the "x" button. If we want to restore the defaults, we can click on the "circular arrow" button.
**Step 3:** One can change the object's name. Here the name of the object is "Numbers" that can be renamed to "num", which will not have any effect on identifying the object uniquely. If there is a change in logical name, the same name should be used while scripting. Only the Object's Name can be changed and not its properties.

**Note:** Properties of any object must be unique so that QTP will be able to recognize the objects and act on it. If the object properties were same for two or more objects, then during execution an error would be thrown that "More than one object is matched for the specified properties".

**Step 4:** After adding the objects, the same can be used in script by simple drag-drop as shown in the figure. When the object is dragged and dropped, the default operation is set. For example, click for a button, Set for a Text Box etc.
QTP – Object Repository Types

Based on Context, the Object Repository is of two types.

**Local Object Repository** - As the name suggests, the Object Repository is applicable only for that action. As we know, that QTP creates a New Test with 1 action by default. Local Object Repository can be opened by traversing to Resources → Object Repository. This is the default OR in QTP.
Shared Object Repository - The Object Repository is shared across actions/modules, which would be mapped for two or more actions. Local objects can be exported to be saved into Shared Object Repository by using the option "Export Local Objects" option. Shared Object Repository can be opened by traversing to Resource → Object Repository Manager.

Following are the major difference between Local and shared ORs.

<table>
<thead>
<tr>
<th>Local Object Repository (LOR)</th>
<th>Shared Object Repository (SOR)</th>
</tr>
</thead>
<tbody>
<tr>
<td>This Object Repository is available; one for each action.</td>
<td>This type of OR is available for multiple tests and for multiple actions.</td>
</tr>
<tr>
<td>This is the OR that is available for each tests, by default.</td>
<td>This type of OR is usually used in frameworks considering reusability and maintainability.</td>
</tr>
<tr>
<td>Local Object Repository is editable in Object Repository.</td>
<td>Shared Object Repository is read-only by default but can be edited in Object Repository Manager.</td>
</tr>
<tr>
<td>It is NOT a standalone file that can edited.</td>
<td>SOR is a standalone file that can be edited easily.</td>
</tr>
</tbody>
</table>
It is saved with an Extension .bdb

It should be used when not many tests are working on the same screens of the application under test.

SOR is saved with an extension .tsr

SOR Should be used when there are different scripts interacting with the same type of the object.

### QTP – User Defined Objects

Sometimes, not all objects are recognized by QTP, in case the application does not use Standard Windows Classes. QTP uses Class Name to find the type of Object. Sometimes, the object is expected to behave like a button or a Combo Box etc.

When we try to add such kind of button, it might recognize as Winobject.

Hence, we can map that WinObject to behave like "CheckBox" Object type by Navigating to Tools -> Object Identification, and select Environment as "Standard Windows" and click "User Defined" button. Please Note this option will not work in any other environment.
In the following example, an object of a specific class is made to recognize as an object of Type "button". Hence, this object inherits all properties of a button and we can use the objects that are supported by button.

QTP – Object Repository as XML

Object Repository can be saved in XML format so that the size of the Object repository is reduced. The OR can be saved as XML by "Exporting as XML" from Shared Object Repository Window as shown below:

The Same object repository can be imported from XML, can be edited/deleted, and exported back to XML.
QTP – Comparing and Merging ORs

Comparing Object Repositories

Many a times, we might be in a position to compare two object repositories to spot the difference and merge it, if in case, some objects are missing in the main repository.

These tools help us largely to spot the differences in the Object Repository

Steps to Compare OR’s

Step 1 : Navigate to "Resources" >> "Object Repository Manager"

Step 2 : Go to "Tools" >> "Object Repository Merge Tool"

Step 3 : The Object Repository comparison Window opens and the user needs to select the two Object Repository files to be compared.
Step 4: It performs the comparison and displays the differences one by one as shown below:

Step 5: It can be filtered based on three parameters viz - unique objects, identical objects, partial match objects.

Merging Object Repositories

Sometimes, there are circumstances when Object repository needs to be merged, as maintaining two or more object repositories become additional overheads.

Merging Option is a great boon for testers who need to merge two-object repository without losing the object hierarchy.
Steps to Merge OR's

Step 1: Navigate to "Resources" >> "Object Repository Manager"

Step 2: Go to "Tools" >> "Object Repository Comparison Tool"

Step 3: The Object Repository Merge window opens and the user needs to select the two Object Repository files to be merged.
Step 4: After merging, the statistics are shown to the user.

Step 5: It can be filtered based on two parameters viz - show all objects, show only conflicting objects.

QTP – Ordinal Identifiers

Sometimes, there are series of objects with same class name and properties. Let us say, in a window, there are series of checkboxes with the same set of properties. If we want to act on those objects, we need to uniquely identify them so that QTP will be able to act on it.

An Ordinal Identifier assigns a numerical value to the test objects, which indicates its location or order relative to its group. The Ordered value enables QTP to recognize it uniquely when the inbuilt properties are NOT sufficient to do so.

There are three Ordinal Identifiers in QTP that can be used in different context:

- Index
- Location
- Creation Time
Index

An object appearing first in the page/Window will have a smaller Index value when compared to another object that comes later in the same page/Window.

<table>
<thead>
<tr>
<th>Object Name</th>
<th>Index Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>TextBox 1</td>
<td>0</td>
</tr>
<tr>
<td>TextBox 4</td>
<td>1</td>
</tr>
<tr>
<td>TextBox 2</td>
<td>2</td>
</tr>
<tr>
<td>TextBox 5</td>
<td>3</td>
</tr>
<tr>
<td>TextBox 3</td>
<td>4</td>
</tr>
<tr>
<td>TextBox 6</td>
<td>5</td>
</tr>
</tbody>
</table>

Location

The Location property works vertically from top to bottom and from left to right. Hence, for the same case, the value of location for the group of text boxes will be as follows:

<table>
<thead>
<tr>
<th>Object Name</th>
<th>Index Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>TextBox 1</td>
<td>0</td>
</tr>
<tr>
<td>TextBox 2</td>
<td>1</td>
</tr>
<tr>
<td>TextBox 3</td>
<td>2</td>
</tr>
<tr>
<td>TextBox 4</td>
<td>3</td>
</tr>
<tr>
<td>TextBox 5</td>
<td>4</td>
</tr>
<tr>
<td>TextBox 6</td>
<td>5</td>
</tr>
</tbody>
</table>
Creation Time

The Creation Time property holds good only for web based application. When we open two browser sessions of the same website, QTP will not be able to recognize the window, as both the windows will have the same set of properties. Hence, we can use creation time with which QTP will be able to act on the window.

```
'Will have CreationTime value = 0
SystemUtil.Run "iexplore.exe", "http://www.google.com"

'Will have CreationTime value = 1
SystemUtil.Run "iexplore.exe", "http://www.yahoo.com"

'Will have CreationTime value = 2
'Will have CreationTime value = 3
```

Hence, to work on a specific browser, we need to explicitly mention the Creation time in OR or we can use the description of objects, which we will see in detail in descriptive programming section.

```
'Sync's www.google.com
Browser("creationtime:=" ).Sync

'Gets the RO text property of www.yahoo.com
Browser("creationtime:=1").GetROProperty("text")

'Highlights microsoft.com
Browser("creationtime:=2").Highlight
```

QTP – Child Objects

The objects (text box, combo box, links) contained in the frame or window is known as child objects. Sometimes, we would be in a situation to get the properties of all the links in a webpage or to get the values of all radio buttons in a window.

In these circumstances, if we want to work on the child objects, we need to use description of objects using which we will be able to work on all the objects in a particular window/page. Descriptive programming will be dealt in detail in the upcoming chapter but the significance of this chapter is to understand child objects and its usage.
The following script gets the name of the links from the website "www.easycalculation.com"

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

'Find all the Links
Set obj = Browser("Math Calculator").Page("Math Calculator").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
```

The Result is printed in the output window as shown below:

![Output screenshot]

Feedback
Kids Math
Advertise
Create Calculator
Create Event Calendar
Ask Questions
Download Calculator
Sign In
Sign Up
Basic/Scientific calculator
Budget Calculators
Finance Calculators
Mortgage Calculator
Tax Calculators
Algebra Calculators
Matrix Calculations
Actions helps testers to divide scripts into groups of QTP statements. Actions are similar to functions in VBScript; however, there are a few differences. By default, QTP creates a test with one action.

<table>
<thead>
<tr>
<th>Actions</th>
<th>Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Action is an in-built feature of QTP.</td>
<td>VBScript Functions are supported by both VBScript and QTP.</td>
</tr>
<tr>
<td>Actions parameters are passed by value only.</td>
<td>Function parameters are passed either by by value or by ref.</td>
</tr>
<tr>
<td>Actions have extension .mts</td>
<td>Functions are saved as .vbs or .qfl</td>
</tr>
<tr>
<td>Actions may or may not be reusable.</td>
<td>Functions are always reusable.</td>
</tr>
</tbody>
</table>

The properties of the action can be accessed by right clicking on the Script Editor Window and selecting "Properties".
Action properties contains the 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.

<table>
<thead>
<tr>
<th>Action Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Insert Call to New Action</td>
<td>Inserts a New Action from the existing action</td>
</tr>
<tr>
<td>Insert Call to Copy of Action</td>
<td>Inserts a copy of an existing action</td>
</tr>
<tr>
<td>Insert Call to Existing Action</td>
<td>Inserts a call to existing re-usable action</td>
</tr>
</tbody>
</table>

**QTP – Call to New Action**

**Inserting Call to New Action**

Testers can insert a new action at any point of the script by performing the following steps:

**Step: 1** - Right click on the scripting area and select "Call to New Action".
Step: 2 - In the "Insert Call to New Action" Window, give the test name, description, and also specify if it is a reusable action or not. Most importantly, select the location of the action to be inserted.
Step: 3 - You can check the changes, graphically, in the test Name Tab as shown below:

You can also use QTP commands to call the Action at any point in the script.

```vbscript
RunAction "Calculate", oneIteration 'Executes Calculate Action for one iteration.
```

Action can be called with Parameters as shown below:

```vbscript
'Input to the action
num1 = 5
num2 = 10
Dim value1
'Run the action with parameters
OutputValue = RunAction("Calculate", oneIteration, num1, num2, value1)

'Display the output
print OutputValue
```

---

**QTP – Call to Copy of Action**

**Inserting Call to Copy of Action**

Testers can insert a copy of an existing action at any point of the script by performing the following steps:

**Step 1** - Right click on the scripting area and select "Call to Copy of Action"
Step 2 - In the "Insert Call to Copy of Action" Window, Select "Test Name", "Action Name" and also select the location of the action to be inserted.
Step 3 – Immediately, the script is auto-generated to show that the copy of an action is inserted.

```
RunAction "Copy of Calculate", oneIteration
```

Step 4 - You can check the changes graphically in the test Name Tab as shown below:
QTP – Call to Existing Action

Inserting call to Existing Action

Testers can insert an Existing action at any point of the script by performing the following steps-
Step 1 - Right click on the Scripting area and select "Call to Existing Action"

Step 2 - In the "Select Action" Window, give the test name, Action name, description and also specify the location of the action to be inserted.
Step 3 - Once inserted, the following script is generated exactly at the location where the action was inserted.

RunAction "Action1", oneIteration

Step 4 – You can check the changes graphically in the test Name Tab as shown below:
A DataTable, similar to Microsoft Excel, helps the testers to create data-driven test cases that can be used to run an Action multiple times. There are two types of Datatables-

- **Local DataTable** - Each action has its own private datable, also known as local datable, which is can also be accessed across actions.

- **Global DataTable** - 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 datatable in the Test Settings dialog, that can be accessed using File -> Settings -> Run(Tab) as shown below:
Example

For example, if a user wants 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.

<table>
<thead>
<tr>
<th>A3</th>
<th>1322</th>
</tr>
</thead>
<tbody>
<tr>
<td>Principal</td>
<td>Rate</td>
</tr>
<tr>
<td>1</td>
<td>232</td>
</tr>
<tr>
<td>2</td>
<td>2556</td>
</tr>
<tr>
<td>3</td>
<td>1322</td>
</tr>
<tr>
<td>4</td>
<td>32.21</td>
</tr>
<tr>
<td>5</td>
<td></td>
</tr>
</tbody>
</table>

Data Table Operations

There are three types of objects to access DataTable. DataTable operations can be well understood by traversing through the following-

<table>
<thead>
<tr>
<th>Object Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Table Methods</td>
<td>Gives detailed information about the data table methods.</td>
</tr>
<tr>
<td>DTParameter Object Methods</td>
<td>Gives detailed information about the DTParameter methods.</td>
</tr>
<tr>
<td>DTSheet Object Methods</td>
<td>Gives detailed information about the DTSheet methods.</td>
</tr>
</tbody>
</table>

QTP – DataTable Object Methods

<table>
<thead>
<tr>
<th>Method Name</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>AddSheet</td>
<td>Adds the specified sheet to the run-time data table</td>
<td>DataTable.AddSheet(SheetName)</td>
</tr>
<tr>
<td>DeleteSheet</td>
<td>Deletes the specified sheet from the run-time data table</td>
<td>DataTable.DeleteSheet SheetID</td>
</tr>
<tr>
<td>Export</td>
<td>Exports the Datatable to a new file in the specified location</td>
<td>DataTable.Export(FileName)</td>
</tr>
<tr>
<td>ExportSheet</td>
<td>Exports a Specific Sheet of the Datatable in run-time</td>
<td>DataTable.ExportSheet(FileName, SheetName)</td>
</tr>
<tr>
<td>GetCurrentRow</td>
<td>Returns the active row of the run-time data table of global sheet</td>
<td>DataTable.GetCurrentRow</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
<td>Syntax</td>
</tr>
<tr>
<td>-----------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>-----------------------------</td>
</tr>
<tr>
<td>GetParameterCount</td>
<td>Returns the number of columns in the run-time data table of Global Sheet.</td>
<td>DataTable.GetParameterCount</td>
</tr>
<tr>
<td>GetRowCount</td>
<td>Returns the number of rows in the run-time data table of Global Sheet.</td>
<td>DataTable.GetRowCount</td>
</tr>
<tr>
<td>GetSheet</td>
<td>Returns the specified sheet from the run-time data table.</td>
<td>DataTable.GetSheet(SheetID)</td>
</tr>
<tr>
<td>GetSheetCount</td>
<td>Returns the total number of sheets in the run-time data table.</td>
<td>DataTable.GetSheetCount</td>
</tr>
<tr>
<td>Import</td>
<td>Imports a specific external Excel file to the run-time data table.</td>
<td>DataTable.Import(FileName)</td>
</tr>
<tr>
<td>ImportSheet</td>
<td>Imports the specified sheet of the specific excel file to the destination sheet.</td>
<td>DataTable.ImportSheet(FileName, SheetSource, SheetDest)</td>
</tr>
<tr>
<td>SetCurrentRow</td>
<td>Sets the Focus of the Current row to the Specified Row Number.</td>
<td>DataTable.SetCurrentRow(RowNumber)</td>
</tr>
<tr>
<td>SetNextRow</td>
<td>Sets the focus of the next row in the run-time data table.</td>
<td>DataTable.SetNextRow</td>
</tr>
<tr>
<td>SetPreviousRow</td>
<td>Sets the focus of the previous row in the run-time data table.</td>
<td>DataTable.SetPrevRow</td>
</tr>
</tbody>
</table>

**DataTable Object Properties**

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>GlobalSheet</td>
<td>Returns the first sheet of the run-time data table.</td>
<td>DataTable.GlobalSheet</td>
</tr>
<tr>
<td>LocalSheet</td>
<td>Returns the Active local sheet of the run-time data table.</td>
<td>DataTable.LocalSheet</td>
</tr>
<tr>
<td>RawValue</td>
<td>Retrieves the raw value of the cell</td>
<td>DataTable.RawValue</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ParameterID, [SheetID]</td>
</tr>
<tr>
<td>Value</td>
<td>Retrieves the value of the cell in the specified parameter.</td>
<td>DataTable.Value(ParameterID, [SheetID])</td>
</tr>
</tbody>
</table>
Example
Consider the following DataTable:

<table>
<thead>
<tr>
<th>A3</th>
<th>1322</th>
</tr>
</thead>
<tbody>
<tr>
<td>Principal</td>
<td>Rate</td>
</tr>
<tr>
<td>1</td>
<td>232</td>
</tr>
<tr>
<td>2</td>
<td>2556</td>
</tr>
<tr>
<td>3</td>
<td>1322</td>
</tr>
<tr>
<td>4</td>
<td>32.21</td>
</tr>
<tr>
<td>5</td>
<td>Global</td>
</tr>
</tbody>
</table>

'Accessing Datatable to get Row Count and Column Count
rowcount = DataTable.GetSheet("Global").GetRowCount
msgbox rowcount    ' Displays 4

colcount = DataTable.GetSheet("Global").GetParameterCount
msgbox colcount    ' Displays 3

DataTable.SetCurrentRow(2)
val_rate = DataTable.Value("Rate","Global")
print val_rate    ' Displays 7%

val_ppl = DataTable.Value("Principal","Global")
print val_ppl    ' Displays 2556

val_Time = DataTable.Value("Time","Global")
print val_Time    ' Displays 5

QTP – DTParameter Object Properties

<table>
<thead>
<tr>
<th>Method Name</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Returns the name of the parameter in the run-time datatable.</td>
<td>DTParameter.Name</td>
</tr>
<tr>
<td>RawValue</td>
<td>Returns the raw value of the cell in the current row of the run-time datatable.</td>
<td>DTParameter.RawValue</td>
</tr>
<tr>
<td>Value</td>
<td>Retrieves or sets the value of the cell in the Active row of the parameter in the run-time datatable.</td>
<td>DTParameter.Value</td>
</tr>
<tr>
<td>ValueByRow</td>
<td>Retrieves the value of the cell in the specified row of the</td>
<td>DTParameter.ValueByRow(RowNum)</td>
</tr>
</tbody>
</table>
parameter in the run-time datatable.

Example

Consider the following DataTable:

<table>
<thead>
<tr>
<th>Principal</th>
<th>Rate</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>232</td>
<td>3.26</td>
</tr>
<tr>
<td>2</td>
<td>2556</td>
<td>7%</td>
</tr>
<tr>
<td>3</td>
<td>1322</td>
<td>6.50%</td>
</tr>
<tr>
<td>4</td>
<td>32.21</td>
<td>3.32%</td>
</tr>
</tbody>
</table>

Example code:

```python
Val = DataTable.GetSheet("Global").GetParameter("Principal").ValueByRow(2)
print Val  # Val Displays 2556

DataTable.SetCurrentRow(1)
Val1 = DataTable.GetSheet("Global").GetParameter("Principal").Value
print Val1  # Val1 displays 232
```

**QTP – DTSheet Methods**

<table>
<thead>
<tr>
<th>Method Name</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>AddParameter</td>
<td>Adds the specified column to the sheet in the run-time data table.</td>
<td>DTSheet.AddParameter(ParameterName, Value)</td>
</tr>
<tr>
<td>DeleteParameter</td>
<td>Deletes the specified parameter from the run-time data table.</td>
<td>DTSheet.DeleteParameter(ParameterID)</td>
</tr>
<tr>
<td>GetCurrentRow</td>
<td>Returns the row number of the active row in the run-time Data Table.</td>
<td>DTSheet.GetCurrentRow</td>
</tr>
<tr>
<td>GetParameter</td>
<td>Returns the specified parameter from the run-time Data Table.</td>
<td>DTSheet.GetParameter(ParameterID)</td>
</tr>
<tr>
<td>GetParameterCount</td>
<td>Returns the total number of Columns</td>
<td>DTSheet.GetParameterCount</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
<td>Code Example</td>
</tr>
<tr>
<td>------------------</td>
<td>------------------------------------------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>GetRowCount</td>
<td>Returns the total number of rows in the run-time Data Table.</td>
<td>DTSheet.GetRowCount</td>
</tr>
<tr>
<td>SetCurrentRow</td>
<td>Sets the Focus on the specified Row of the Data Table</td>
<td>DTSheet.SetCurrentRow(RowNumber)</td>
</tr>
<tr>
<td>SetNextRow</td>
<td>Shifts the Focus to the next Row of the Data Table</td>
<td>DTSheet.SetNextRow</td>
</tr>
<tr>
<td>SetPrevRow</td>
<td>Shifts the Focus to the Previous Row of the Data Table.</td>
<td>DTSheet.SetPrevRow</td>
</tr>
</tbody>
</table>

**Example**

```vbs```
' Accessing Datatable to get Row Count and Column Count
rowcount = DataTable.GetSheet("Global").GetRowCount
msgbox rowcount  ' Displays 4

colcount = DataTable.GetSheet("Global").GetParameterCount
msgbox colcount  ' Displays 3

DataTable.SetCurrentRow(2)
val_rate = DataTable.Value("Rate","Global")
print val_rate  ' Displays 7%

val_ppl = DataTable.Value("Principal","Global")
print val_ppl  ' Displays 2556

val_Time = DataTable.Value("Time","Global")
print val_Time  ' Displays 5
```
Checkpoints, as the name says it all, refer to a validation point that compares the current value of 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 of Checkpoints

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Standard Checkpoint</td>
<td>Verifies the property values of an object in application under test and supported by all add-in environments.</td>
</tr>
<tr>
<td>Bitmap Checkpoint</td>
<td>Verifies an area of your application as a bitmap.</td>
</tr>
<tr>
<td>File Content Checkpoint</td>
<td>Verifies the text in a dynamically generated or accessed file such as .txt,.pdf</td>
</tr>
<tr>
<td>Table Checkpoint</td>
<td>Verifies the information within a table. Not all environments are supported.</td>
</tr>
<tr>
<td>Text Checkpoint</td>
<td>Verify if the text that is displayed within a defined area in a Windows-based application, according to specified criteria.</td>
</tr>
<tr>
<td>Text Area Checkpoint</td>
<td>Verifies if the text string is displayed within a defined area in a Windows-based application, according to specified criteria.</td>
</tr>
<tr>
<td>Accessibility Checkpoint</td>
<td>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</td>
</tr>
<tr>
<td>Page Checkpoint</td>
<td>Verifies the characteristics of a Web page. It can also check for broken links.</td>
</tr>
<tr>
<td>Database Checkpoint</td>
<td>Verifies the contents of a database accessed by the application under test.</td>
</tr>
<tr>
<td>XML Checkpoint</td>
<td>Verifies the content of the .xml documents or .xml documents in Web pages and frames.</td>
</tr>
</tbody>
</table>

### 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.
Given below is the checkpoint menu, when the user is NOT in the recording mode.

![Checkpoint Menu](image1)

Given below is the checkpoint menu, when the user is in the recording mode.

![Checkpoint Menu](image2)

**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("Numbers")

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("French")

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

**Viewing Checkpoint Properties**

After inserting, in case a tester wants to change the values, we can do so by right clicking on the keyword 'checkpoint' of the script and navigating to "Checkpoint Properties" as shown below:
You can locate the same checkpoints in object repository, as well, as shown below. It exactly shows what type of checkpoint is used and what are the expected values, and timeout values.
Synchronization point is the time interface between Tool and Application under test. Synchronization point is a feature to specify the delay time between two steps of the test script.

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

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 in "www.easycalculation.com". We will now take a look at all the five 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.

```vbs
' 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

```
' 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**: Go to "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:

![Object Selection Window]

**Step 4**: Click OK; the "Add Synchronization Window" opens. Select the Property, Value and Time out value and click OK as shown below:

![Add Synchronization Point Window]

**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:

```javascript
Browser("Math Calculator").Page("Math Calculator").Link("Numbers").Click
```
Default Synchronization

When the user has not used any of the above sync methods, still QTP has an in-built Object synchronization timeout which can be adjusted by the user.

Navigate to "File" >> "Settings" >> Run Tab >> Object Synchronization Time out as shown below.
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 choose the base and Optional Filter Properties.

**Step 4**: Add Properties in Base Properties apart from the default one and also add/remove Optional Filter Properties. Please note that same properties cannot be a part of both Mandatory and Assistive Properties and click "OK".
**Step 5**: Verify 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 do not want to enable Smart Identification.
**Step 6**: We can even disable a test Level by applying at test script level under "Settings" of "File" Menu as shown below:

**Step 7**: If the Smart Identification is disabled as per Step# 6 then it will not apply smart identification for any object during the script execution.

**Step 8**: In case the objects are added with Smart Identification as Off, QTP will not use Smart Identification for recognizing in future, even though we have enabled it later.
Debugging, in automation testing context, is a systematic process of spotting and fixing the coding issues in the automation script 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 Colored" Dot and the line will be highlighted in RED as shown below:

<table>
<thead>
<tr>
<th>Method</th>
<th>ShortCut</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step Into</td>
<td>F11</td>
<td>Used to execute each and every Step. Steps into the Function/Action and executes line by line. It pauses on each line after execution.</td>
</tr>
<tr>
<td>Step Over</td>
<td>F10</td>
<td>Used to Step over the Function. Step Over runs only the current step in the active document.</td>
</tr>
<tr>
<td>Step Out</td>
<td>Shift+F11</td>
<td>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.</td>
</tr>
</tbody>
</table>

Options in Break Point

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

<table>
<thead>
<tr>
<th>ShortCut</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>F9</td>
<td>Insert/Remove BreakPoint</td>
</tr>
<tr>
<td>Ctrl+F9</td>
<td>Enable/Disable BreakPoint</td>
</tr>
<tr>
<td>Ctrl+Shift+F9</td>
<td>Clear All BreakPoint</td>
</tr>
<tr>
<td>Use Only Menu</td>
<td>Enable/Disable All BreakPoints</td>
</tr>
</tbody>
</table>
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:
There are various ways of handling errors in QTP. There are three possible types of errors, one would encounter, while working with QTP. They are-

- 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, use the keyboard shortcut 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, then it is known as a Logical error. 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 misunderstandings 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 verify the QTP output file/result file to ensure that the tool has performed the way it was supposed 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 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-

- File NOT found but the script trying to read the file
- Object NOT found but the script is trying to act on that particular object
- Dividing a number by Zero
- 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 in the Test Settings by navigating to "File" >> "Settings" >> "Run" Tab as shown below. We can select any of the specified settings and click "OK".

![Test Settings](image)

2. **Using On Error Statement** - The ‘On Error’ statement is used to notify the VBScript engine of intentions to handle the run-time errors by a 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 process 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 in-built object within VBScript that captures the run-time error number and error description with which we are 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 has 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

```vbnet
'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 Go To 0
End function
```

4. **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.

```vbnet
'Example
Reporter.ReportEvent micFail, "Login", "User is unable to Login."
```
13. QTP – Recovery Scenarios

Recovery Scenarios
While executing the QTP scripts, we might get some unexpected errors. In order to recover the tests 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: Click "New" Recovery Scenario button; the Recovery Scenario Wizard opens as shown below:
**Step 2**: Choose the Trigger Event. It corresponds to event, which can arise in any of the following four events-

- Pop-Up Window
- Object State
- Test Run Error
- Application Crash

**Step 3**: The Recovery Operations Window opens. Recovery Operation can perform any of the following Operation as shown in the screenshot below:
**Step 4**: After specifying the appropriate Recovery Operation, we need to specify the Post Recovery Operation as well, as shown below:

![Post-Recovery Test Run Options](image)

**Step 5**: After specifying the Post Recovery Operation, the recovery scenario should be named and added to the Test so that it can be activated.

![Name and Description](image)
**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".
Step 7: The Added Recovery Scenario will be as shown below and click the "Close" Button to continue.

![Recovery Scenario Manager]

Step 8: After clicking the Close button, QTP would prompt the user to save the created Recovery scenario. It will be saved with the extension .qrs and the wizard would close.

![Save Recovery Scenario]
Verification

The Created Recovery Scenario should be a part of the test now and can be verified by navigating to "File" → "Settings" → "Recovery" Tab.
QTP environment variables are special types of variables that can be accessed by all actions, function libraries, and recovery scenarios. There are in-built environment variables for Windows that are available to all the applications running on that particular system, but QTP environment variables are only available to that particular 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 an external file as a .xml file and can be loaded into the test as shown in the figure given below. It 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
```

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

```
<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>116</td>
<td>117</td>
<td>118</td>
</tr>
<tr>
<td>x = Environment.ExternalFileName</td>
<td>print x</td>
<td></td>
</tr>
</tbody>
</table>
```

Output

```
Print Log   Locate
Config.xml
```
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
```

1. Environment Variable XML File
2. Dynamically Loaded the Env File and accessed the Browser Parameter which has value "IE"
3. It has Printed "IE" as Expected in the Output Window
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
```

Output

```
Microsoft Windows 7 Workstation
Action1
IE
```
In order 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 the "+" button to add Function Library file and add it using the actual path or relative path as shown below:

![Test Settings](image)

**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.

```vbs
'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 the execution of the test (as per default settings). The following information is displayed:

- 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:

![Export Run Results Dialog Box](image)

We can choose what type of report is to be exported. It can be short results, detailed results or even, we can select nodes. After selecting the File Name and exporting it, the file is saved as .HTML File.

Filtering the Results

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

![Filter Dialog Box](image)
Raising Defects

Defects can be logged into QC directly from the Test Results Window pane by accessing "Tools" → "Add Defect" which open's the connection to ALM as shown below:

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"
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 the following three conditions:

- For Errors
- Always
- For Errors and Warnings
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**

Following 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**

```plaintext
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
```

17. QTP – Working with GUI Objects
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 will not 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

```python
Set Obj = Browser("Calculator").Page("Gmail").WebCheckBox("PersistentCookie")
Obj.Set "ON"
```

'To UnCheck the Check Box

```python
Obj.Set "OFF"
```

'Verifies the Existance of the Check box and returns Boolean Value

```python
val = Obj.Exist
print val
```

'Fetches the Name of the CheckBox

```python
a= Obj.GetROProperty("name")
print a
```

'Verifies the visible property and returns the boolean value.

```python
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 cannot 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 Existance 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**

```vbscript
'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**

```vbscript
'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")
```
Working with webTables

In Today's web based application, webTables have become very common and testers need to understand how WebTables work and how to perform an action on webTables. This topic will help you to work with the webTables effectively.

<table>
<thead>
<tr>
<th>Statement</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>if statement</td>
<td>An if statement consists of a boolean expression followed by one or more statements.</td>
</tr>
<tr>
<td>if..else statement</td>
<td>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.</td>
</tr>
<tr>
<td>if..elseif..else statement</td>
<td>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.</td>
</tr>
<tr>
<td>nested if statements</td>
<td>An if or elseif statement inside another if or elseif statement(s).</td>
</tr>
<tr>
<td>switch statement</td>
<td>A switch statement allows a variable to be tested for equality against a list of values.</td>
</tr>
</tbody>
</table>

- **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**

```plaintext
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.GetChildData(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.GetChildItemCount(2,1,"Link")
print z
**What are Virtual Objects?**

Sometimes, an 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, on the Ribbon, Insert tab is clicked and then the user clicks the "Picture" button. A button is recognized as WinObject; hence, importance of virtual objects is pronounced.

```plaintext
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 the "Next" button.
**Step 2**: Map the Object against the Class Type and click "Next".

Step 2 Image

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

Step 3 Image
Step 4: Select 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 does not support virtual objects for analog or low-level recording.
- Checkpoints cannot be added on Virtual Objects.
- Virtual Objects 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.
As such, QTP does not 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. They are-

- **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 differs in the way we connect to them. However, the connection strings can be built with the help of [www.connectionstrings.com](http://www.connectionstrings.com)

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
'Set 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;UserId=sa;Password=Password123;Database=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**

On executing the above script, the output is shown in the message box as shown below:
XML is a markup language designed to store data in a format that can be both readable by human and machine. Using XML, data can also be easily exchanged between computer and database systems.

Sample XML and their key elements are represented below-

Accessing XML

```vbscript
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 XMLs-

```
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
```
QTP scripts can execute only if the objects are present in the Object Repository. The descriptions of the Objects are created using Descriptive programming-

- When the 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, it results 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 scripting using Descriptive Programming technique. They are-

- Description Objects
- Description Strings

**Description Objects**

Script is developed using Description Objects that depend 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"
btcnalc("name").value = "calculate"
btcnalc("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 is developed using the properties and values as strings as shown below.
Child Objects

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

```vbscript
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 write the script based on ordinal identifiers, which will enable QTP to act on those objects when two or more objects have the same properties.

```vbscript
' 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
```
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 the testers to control the configuration settings and execute the scripts using the QTP interface. The Key Configurations/actions that can be controlled (but not limited to) are listed below-

- Loads all the required add-ins for a test
- Makes QTP visible while execution
- Opens the Test using the specified location
- Associates Function Libraries
- Specifies the Common Object Sync Time out
- Start and End Iteration
- Enable/Disable Smart Identification
- On Error Settings
- Data Table Path
- Recovery Scenario Settings
- 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

A tester can generate AOM script from QTP itself, using the "Generate Script" option. Navigate to "Run" >> "Settings" >> "Properties" Tab >> "Generate Script" as shown below:
Example

' A Sample Script to Demostrate 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").CloseOnExit = True

App.Test.Settings.Launchers("Windows Applications").Active = False
App.Test.Settings.Launchers("Windows Applications").RecordOnQTDescendants = True
App.Test.Settings.Launchers("Windows Applications").RecordOnExplorerDescendants = False

App.Test.Settings.Run.IterationMode = "rngAll"

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


App.Test.Settings.Recovery.Add "D:\GUITest2\recover_app_crash.qrs", "Recover_Application_Crash", 1

' System Local Monitoring settings
' 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
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 is 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, the testers can test how the application handles various inputs effectively. It can be any of the following 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.

![Hybrid Framework 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.

In the next chapter, we will learn how to design a simple framework.
Let us design a simple framework by taking a sample application. We will automate a few scenarios of the application under test and write reusable functions.

The sample application under test is "Calculator", a default application that is available as a part of Windows. Let us now create different components of a framework. Here, we will develop a hybrid framework and use Object Repository as it is fairly a simple application. However, this framework can be scaled to support a complex application as well.

The folder structure of the Framework is as shown below-

![Folder Structure Diagram]

**Explanation of the folder structure**-

- **Master Driver Script** - The Script that drives the entire execution. It performs prerequisite and initial settings that are required for the execution.

- **Library Files** - The Associated Functions that forms the Function Library.

- **Data Table** - The Test Data that is required for the Execution.

- **Object Repository** - The Objects and its properties that enable QTP to recognize the objects seamlessly.

- **Execution Logs** - The Folder contains the execution log file with user functions and function execution history.

**Master Driver Script**

```plaintext
MASTER DRIVER SCRIPT NAME : Calculator
DESCRIPTION : Drivers Script to Execute Automated Test for the Calculator
AUTHOR : Tutorials Point
DATE CREATED : 30-Dec-2013
OBJECT REPOSITORY ASSOCIATED : Calc.tsr
LIBRARY FILES ASSOCIATED : calculator.qfl, Common.qfl
MODIFICATION LOG
```
Option Explicit

Public ExecDrive

' Get the Root folder of the Test so that we can make use of relative paths.
Dim x : x=Instr(Environment.Value("TestDir"),"Driver")-2
ExecDrive = mid(Environment.Value("TestDir"),1,x)

' Get the path of Libraries using relative to the current Drive
Dim LibPath : LibPath = ExecDrive+"\Libraries\"

' Dynamically Load the Function Libraries
LoadFunctionLibrary LibPath + "Calculator.qfl", LibPath + "common_utils.qfl"

' Capturing the Start Time
' clscommon is the class object created in common.qfl library file
clscommon.StartTime = Time()

' Launching the Application
SystemUtil.Run "C:\Windows\System32\Calc.exe" : wait (2)

' Initialize the Data Table Path
Dim FileName : FileName = ExecDrive+"\TestData\Calculator.xls"
Dim SheetSource : SheetSource = "Calc_test"
Dim SheetDest : SheetDest = "Global"

' Import the DataTable into the QTP Script
DataTable.ImportSheet FileName , SheetSource , SheetDest

' Object Repository Path
Dim RepPath : RepPath = ExecDrive+"\Object_Repository\Calc.tsr"
RepositoriesCollection.RemoveAll()
RepositoriesCollection.Add(RepPath)

' To Keep a Count on iteration
Dim InttestIteration
Dim InttestRows : InttestRows = datatable.GetRowCount

' Fetching Date-TimeStamp which will be unique for Naming the Execution Log File
clscommon.StrDateFormatted = day(date()) & "_" & MonthName(Month(date()),true) &
"_" & YEAR(date())& "_"&hour(now)&"_"&minute(now)

' Name the LogFile
clscommon.StrLogFile = ExecDrive & "\Execution Logs\" &
clscommon.StrDateFormatted & ".txt"

' Create the Execution LogFile which captures the result
clscommon.Fn_FileCreate(clscommon.StrLogFile)

' Initialize the Parameters and all the relevant Test Details
Call Fn_InitializeLogFile()

' Kill all the previous calculator process
Call fn_Kill_Process("calc.exe")

For InttestIteration=1 to InttestRows
  datatable.SetCurrentRow InttestIteration
  Dim StrExecute : StrExecute = Ucase(Trim(datatable.Value("Run","Global")))

  If StrExecute = "Y" Then
    clscommon.Number1 = Trim(datatable.Value("Number_1","Global"))
    clscommon.Number2 = Trim(datatable.Value("Number_2","Global"))
    clscommon.Number3 = Trim(datatable.Value("Number_3","Global"))

    clscommon.Number4 = Trim(datatable.Value("Number_4","Global"))
    clscommon.Number5 = Trim(datatable.Value("Number_5","Global"))
    clscommon.Number6 = Trim(datatable.Value("Number_6","Global"))

    clscommon.Test_Case_ID = Trim(datatable.Value("Test_Case_ID","Global"))
    : clscommon.LogWrite "The Test Case Data is Located at:: " & tcDataPath
    clscommon.tcScenario = Trim(datatable.Value("Scenario","Global"))
    : clscommon.LogWrite "The Test Case Data is Located at:: " & tcDataPath
  End If
Next
Dim Expected_Val = Trim(datatable.Value("Expected_Val","Global"))'

: clscommon.LogWrite "The Test Case Data is Located at :: " & tcDataPath

Select case clscommon.tcScenario
    Case "Add"
        clscommon.LogWrite "=== Inside the Test Set :: " & clscommon.tcScenario & " ==="
        Call fnCalculate("+",Expected_Val)
    Case "Subtract"
        clscommon.LogWrite "=== Inside the Test Set :: " & clscommon.tcScenario & " ==="
        Call fnCalculate("-",Expected_Val)
    Case "Multiply"
        clscommon.LogWrite "=== Inside the Test Set :: " & clscommon.tcScenario & " ==="
        Call fnCalculate("*",Expected_Val)
    Case "Divide"
        clscommon.LogWrite "=== Inside the Test Set :: " & clscommon.tcScenario & " ==="
        Call fnCalculate("/",Expected_Val)
    Case "Sqrt"
        clscommon.LogWrite "=== Inside the Test Set :: " & clscommon.tcScenario & " ==="
        Call fnCalculate("sqt",Expected_Val)
End Select
End If
Next

' Calling the End Test to Add the result Footer in exec log file.
Call fn_End_test()

' =====================  End of Master Driver Script =========

114
Library Files

The Calculator Functions are written in a separate function file saved with the extension .qfl or .vbs. These functions are reusable across actions.

'  Calculator. Qfl File :: Associated Function Library for Calculator Master Driver

'  FUNCTION NAME      : Fn_InitializeLogFile
'  DESCRIPTION        : Function to Write the Initial Values in the Log File
'  INPUT PARAMETERS   : varExecDrive,StrDB,StrUId,Strpwd,StrNewDB
'  OUTPUT PARAMETERS  : NIL
'  RETURN VALUE       : Pass or Fail message
'  DATE CREATED       : 30-Dec-2013

Public Function Fn_InitializeLogFile()
    clscommon.LogWrite "********************
    clscommon.LogWrite "Calc Automation Started"
End Function

'  FUNCTION NAME      : fnCalculate
'  DESCRIPTION        : Function to perform Arithmetic Calculations
'  INPUT PARAMETERS   : operator,Expected_Val
'  OUTPUT PARAMETERS  : NIL
'  RETURN VALUE       : Pass or Fail message
'  DATE CREATED       : 30-Dec-2013

Function fnCalculate(operator,Expected_Val)
    clscommon.LogWrite "Executing the Function 'fnCalculate' 

        Window("Calculator").Activate

        If Trim(clscommon.Number1) <> "" Then
            Window("Calculator").WinButton(clscommon.Number1).Click
        End If

        If Trim(clscommon.Number2) <> "" Then
            Window("Calculator").WinButton(clscommon.Number2).Click
        End If

        If Trim(clscommon.Number3) <> "" Then
            Window("Calculator").WinButton(clscommon.Number3).Click
        End If

        Window("Calculator").WinButton(operator).Click
If Trim(clscommon.Number4) <> "" Then
Window("Calculator").WinButton(clscommon.Number4).Click

If Trim(clscommon.Number5) <> "" Then
Window("Calculator").WinButton(clscommon.Number5).Click

If Trim(clscommon.Number6) <> "" Then
Window("Calculator").WinButton(clscommon.Number6).Click

Window("Calculator").WinButton("=").Click
Dim ActualVal : ActualVal = Window("Calculator").WinEdit("Edit").GetROProperty("text")
clscommon.LogWrite "The Actual Value after the Math Operation is \\
ActualVal

If Trim(ActualVal) = Trim(Expected_Val) Then
    clscommon.WriteResult "Pass", clscommon.Test_Case_ID, \\
classcommon.tcScenario, " Expected Value matches with Actual Value :: \\
ActualVal
Else
    clscommon.WriteResult "Fail", clscommon.Test_Case_ID, \\
classcommon.tcScenario, " Expected Value - " & Expected_Val & " Does NOT matches\\nwith Actual Value :: " & ActualVal
End If

Window("Calculator").WinButton("C").Click

If Err.Number <> 0 Then
    clscommon.LogWrite "Execution Error : The Error Number is :: " & \\
Err.Number & " The Error Description is " & Err.Description
    Err.Clear
End If

clscommon.LogWrite "Exiting the Function 'fnCalculate' \\
End Function

' = = = = = = = = = = = = = = = = = = = = = = = = = = = =
' FUNCTION NAME : fn_Kill_Process
' DESCRIPTION : Function to Kill the process by name
' INPUT PARAMETERS : Application name to be killed
' OUTPUT PARAMETERS : NIL
' RETURN VALUE : NIL
Function fn_Kill_Process(process)
    Dim strComputer, strProcessToKill, objWMIService, colProcesses
    strComputer = "."
    strProcessToKill = process

    Set objWMIService = GetObject("winmgmts:" & 
    "\{impersonationLevel=impersonate}!\" 
    & strComputer & "\root\cimv2"

    Set colProcess = objWMIService.ExecQuery _("Select * from Win32_Process
Where Name = " & strProcessToKill & "")

    count = 0
    For Each objProcess in colProcess
        objProcess.Terminate()
        count = count + 1
    Next
End Function

Function fn_End_test()
    clscommon.LogWrite "Status Message - Executing the Function 'fn_End_test' "

    Window("Calculator").Close
    On Error Resume Next

    clscommon.StopTime = Time()
    clscommon.ElapsedTime = DateDiff("n",clscommon.StartTime,clscommon.StopTime)
    Dim TotalTests
    TotalTests = clscommon gintPassCount + clscommon gintFailCount
clscommon.LogWrite "## The Execution Start Time :: " & clscommon.StartTime
clscommon.LogWrite "## The Execution End Time :: " & clscommon.StopTime
clscommon.LogWrite "## The Time Elapsed :: " & clscommon.ElapsedTime & " Minutes"
clscommon.LogWrite "## The OS :: " & Environment.Value("OS")
clscommon.LogWrite "## The Total No of Test Cases Executed :: " & Totaltests
clscommon.LogWrite "## The No. of Test Case Passed :: " & clscommon.gintPassCount
clscommon.LogWrite "## The No. of Test Case Failed :: " & clscommon.gintFailCount
clscommon.LogWrite "## The OS :: " & Environment.Value("OS")
SystemUtil.CloseDescendentProcesses
End Function

' ===============     End of Calculator. Qfl   ============================= '

The other library file, which is 'common_utils.qfl' that contains the functions, which enables us to write the output to a text file.

Set clscommon=New OS_clsUtils

'Creating a class file to handle global variables.
Class OS_clsUtils
    Dim StrLogFile
    Dim StrDateFormatted
    Dim Result

    Dim Number1, Number2 , Number3
    Dim Number4, Number5 , Number6
    Dim Test_Case_ID , tcScenario
    Dim StartTime, StopTime, ElapsedTime

    Dim gintPassCount , gintFailCount , gintWarningCount , gintdoneCount, gintinfoCount

    Function Fn_FileCreate(strFileName)
        Dim objFSO: Set objFSO = CreateObject("Scripting.FileSystemObject")
        On Error Resume Next
    End Function

118
Dim objTextFile : Set objTextFile = objFSO.CreateTextFile(strFileName)
objTextFile.Close
Set objTextFile = Nothing
Set objFSO = Nothing
End Function

Function LogWrite(sMsg)
    Const ForAppending = 8
    Dim objFSO : Set objFSO = CreateObject("scripting.FileSystemObject")
    Dim objTextFile : Set objTextFile = objFSO.OpenTextFile
    (clscommon.StrLogFile, ForAppending, True)

    objTextFile.WriteLine day(date()) & "/" & MonthName(Month(date()),true)
    & "/" & YEAR(date()) & "/" & time & ": " & sMsg
    objTextFile.Close

    Set objTextFile = Nothing
    Set objFSO = Nothing
End Function

Function WriteResult(strStatus,functionName,functionDescription,Result)
    Const ForAppending = 8
    Dim objFSO : Set objFSO = CreateObject("scripting.FileSystemObject")
    Dim objTextFile : Set objTextFile = objFSO.OpenTextFile
    (clscommon.StrLogFile, ForAppending, True)

    objTextFile.WriteLine day(date()) & "/" & MonthName(Month(date()),true)
    & "/" & YEAR(date()) & "/" & time & ": " & "* * * * * * Test Case Exec Details * * * * * * *
    objTextFile.WriteLine day(date()) & "/" & MonthName(Month(date()),true)
    & "/" & YEAR(date()) & "/" & time & ": " & "Test status :: " & strStatus
    objTextFile.WriteLine day(date()) & "/" & MonthName(Month(date()),true)
    & "/" & YEAR(date()) & "/" & time & ": " & "Test ID :: " & functionName
    objTextFile.WriteLine day(date()) & "/" & MonthName(Month(date()),true)
    & "/" & YEAR(date()) & "/" & time & ": " & "Test Description :: " & functionDescription
    objTextFile.WriteLine day(date()) & "/" & MonthName(Month(date()),true)
    & "/" & YEAR(date()) & "/" & time & ": " & "Test Result Details :: " & Result

    objTextFile.Close
Set objTextFile = Nothing
Set objFSO = Nothing
End Function
Object Repository

Object Repository has got all the objects that the user would be acting upon. The image given below shows the list of all objects added into the repository with the name calc.tsr
DataTable

DataTable contains the keywords, which drive the tests and also Test the data with which QTP will act on the objects.
The Execution Log

The Execution log file or output file contains user actions and function log, which will enable the testers to debug upon script failures.

8/Jan/2014 5:09:16 PM: **************************************************
8/Jan/2014 5:09:16 PM: Calc Automation Started
8/Jan/2014 5:09:16 PM: === Inside the Test Set :: Add ===
8/Jan/2014 5:09:16 PM: Executing the Function 'fnCalculate'
8/Jan/2014 5:09:17 PM: * * * * * * Test Case Exec Details * * * * *
8/Jan/2014 5:09:17 PM: Test status :: Pass
8/Jan/2014 5:09:17 PM: Test ID :: TC_001
8/Jan/2014 5:09:17 PM: Test Description :: Add
8/Jan/2014 5:09:17 PM: Test Result Details :: Expected Value matches with Actual Value :: 949.
8/Jan/2014 5:09:17 PM: * * * * * * * * * * * * * * * * * * * * * * *
8/Jan/2014 5:09:17 PM: Exiting the Function 'fnCalculate'
8/Jan/2014 5:09:17 PM: === Inside the Test Set :: Subtract ===
8/Jan/2014 5:09:17 PM: Executing the Function 'fnCalculate'
8/Jan/2014 5:09:17 PM: * * * * * * Test Case Exec Details * * * * *
8/Jan/2014 5:09:17 PM: Test status :: Pass
8/Jan/2014 5:09:17 PM: Test ID :: TC_002
8/Jan/2014 5:09:17 PM: Test Description :: Subtract
8/Jan/2014 5:09:17 PM: Test Result Details :: Expected Value matches with Actual Value :: 415.
8/Jan/2014 5:09:17 PM: * * * * * * * * * * * * * * * * * * * * * *
8/Jan/2014 5:09:17 PM: Exiting the Function 'fnCalculate'
8/Jan/2014 5:09:17 PM: === Inside the Test Set :: Multiply ===
8/Jan/2014 5:09:17 PM: Executing the Function 'fnCalculate'
8/Jan/2014 5:09:18 PM: Test status :: Pass
8/Jan/2014 5:09:18 PM: Test ID :: TC_003
8/Jan/2014 5:09:18 PM: Test Description :: Multiply
8/Jan/2014 5:09:18 PM: Test Result Details :: Expected Value matches with Actual Value :: 278883.

8/Jan/2014 5:09:18 PM: Exiting the Function 'fnCalculate'
8/Jan/2014 5:09:18 PM: === Inside the Test Set :: Divide ===
8/Jan/2014 5:09:18 PM: Executing the Function 'fnCalculate'
8/Jan/2014 5:09:19 PM: Test status :: Pass
8/Jan/2014 5:09:19 PM: Test ID :: TC_004
8/Jan/2014 5:09:19 PM: Test Description :: Divide
8/Jan/2014 5:09:19 PM: Test Result Details :: Expected Value matches with Actual Value :: 3.

8/Jan/2014 5:09:19 PM: Exiting the Function 'fnCalculate'
8/Jan/2014 5:09:19 PM: === Inside the Test Set :: Sqrt ===
8/Jan/2014 5:09:19 PM: Executing the Function 'fnCalculate'
8/Jan/2014 5:09:20 PM: Test status :: Pass
8/Jan/2014 5:09:20 PM: Test ID :: TC_005
8/Jan/2014 5:09:20 PM: Test Description :: Sqrt
8/Jan/2014 5:09:20 PM: Exiting the Function 'fnCalculate'
8/Jan/2014 5:09:20 PM: Status Message - Executing the Function 'fn_Finish_test'
8/Jan/2014 5:09:20 PM: ## # # # # # # # # # # # # # # # # # # # # # # # # # #
8/Jan/2014 5:09:20 PM: ##  The Execution Start Time  ::  5:09:14 PM
8/Jan/2014 5:09:20 PM: ##  The Execution End Time  ::  5:09:20 PM
8/Jan/2014 5:09:20 PM: ##  The Time Elapsed ::   0 Minutes
8/Jan/2014 5:09:20 PM: ##  The OS ::  Microsoft Windows Vista Server
8/Jan/2014 5:09:20 PM: ##  The Total No of Test Cases Executed  ::  25
8/Jan/2014 5:09:20 PM: ##  The No. of Test Case Passed ::  25
8/Jan/2014 5:09:20 PM: ##  The No. of Test Case Failed ::
8/Jan/2014 5:09:20 PM: ## # # # # # # # # # # # # # # # # # # # # # # # # # #