



QEngine Web Funtional Evaluation Guide

ZOHO Corp.
4900 Hopyard Rd., Suite 310
Pleasanton, CA 94588, USA
Phone: +1-925-924-9500
Fax: +1-925-924-9600
info@manageengine.com

<http://www.qengine.com>
qengine-suppot@manageengine.com

Table of Contents

WELCOME TO QENGINE	4
About this Guide	4
About QEngine.....	4
Online Resources	4
STARTING THE TESTING PROCESS.....	5
QEngine Testing Process	5
Installing/Uninstalling QEngine	6
Starting/Shutting down QEngine Server	6
Installing QEngine Toolbar.....	6
Connecting and Logging into QEngine Server.....	7
UNDERSTANDING THE BASICS	8
Exploring the Suite Manager UI	8
Creating Test Suite.....	8
QEngine Suite Structure.....	8
Selecting Test Type and Creating Webscripts.....	9
Exploring the Main Web Functional UI Exploring QEngine Toolbar Options	9
Exploring QEngine Toolbar Options.....	10
Exploring Quick Links Options	10
Exploring Script Editor Toolbar Options	10
HOW TOS.....	11
Working with Test Suites.....	11
What is a Suite ?	11
How do I create a new Suite ?	11
How do I import a Suite ?	11
How do I export a Suite ?	11
Creating/Editing Scripts	11
How are scripts created ?.....	11
How do I record events ?	11
Can I edit a Script ?.....	12
Can I add looping and conditional statements ?	12
Can I call one script from another ?	12

Working with GUI Map	12
What is a GUI Map ?	12
How does QEngine identify a GUI object ?	12
How do I view/edit the GUI object properties ?	12
How do I decide on which GUI Map File mode to use ?	12
How do I use a Global Map File for my script ?	13
How do I merge my local map files ?	13
Can I unmerge a global map file to local map file ?	13
Can I manually add an object to a GUI Map file ?	13
Creating Checkpoints / Test Cases.....	13
What are checkpoints ?	13
How do I insert a single checkpoint ?	13
Can I check for a single property ?	14
Can I check for a single object ?	14
Can I check two or more objects in a Window ?	14
Can I include expressions or specify conditions in a testcase ?	14
How do I search for a text in a HTML page ?	14
How do I search for the non-existence of a text in a HTML page ?	14
How do I check all the table properties ?	15
Can I create a database checkpoint ?	15
Can I edit a test case ?	15
Inserting Built-in Functions.....	15
How to insert functions ?	15
How do I get the number of rows and columns in a Table ?	16
How do I get the table cell value of a HTML Table element ?	16
How do I test a dynamically changing link element ?	16
How to capture the property value and store it in a variable ?	16
Can I programmatically look for objects of a certain type on a web page ?	16
How do I check the color or font of text links ?	17
Can I add custom Jython functions in my script ?	17
Can I set and get values across scripts ?	17
How do I wait for objects or windows to appear ?	17
Can I use regular expressions in functions ?	17
How to print a message in the log file ?	17
How to start other applications from test script ?	18
How do I compare two files ?	18

Data-Driven Test Scripts	18
What are data-driven scripts ?	18
How to create data-driven test scripts?	18
How to import data from a database ?	18
Exception Handling for Unattended Execution.....	18
How do I handle web exceptions ?.....	18
How do I handle script and object exceptions ?	18
How do I handle unexpected popups during playback ?.....	19
Running Scripts.....	19
How do I run a test script to check my application ?	19
Can I test my scripts in any locale without re-recording ?.....	19
How do I play the recorded script with any server?	19
How do I run my test script against other databases like oracle or sql server ?	19
Is there any settings to be done before running a test ?	19
Can I run scripts from command line mode ?	19
Can I run scripts automatically at scheduled intervals ?	20
Can I execute specific test scripts ?	20
Bug Tracking	20
How do I track product features and bugs ?	20
Can I integrate third party bug tracking system ?.....	20
APPENDIX.....	21
Using Built In Functions	21
Window Functions	22
STRING FUNCTIONS.....	28
HTML ELEMENT FUNCTIONS	31
HTML Get Functions	32
HTML Check Functions	38
HTML Table Functions.....	44
HTML General Functions	48
Database Related Functions.....	50
File Operations.....	52
General Functions.....	54

Welcome to QEngine

About this Guide

This evaluation guide is provided to help you get started with QEngine Web Functional Test Tool. Provides a quick overview of QEngine and its features, online resources available, how to start the testing process, installation, startup steps and an overview of QEngine Web interfaces. Most of the general queries raised during the evaluation phase of QEngine are given in the **How Tos** section. If you have any specific queries/feedbacks regarding QEngine please send us your queries or feedbacks to qengine-support@adventnet.com or post your queries in the forums.

About QEngine

ManageEngine QEngine is a powerful tool for automated functional and performance testing of your web applications and web services. QEngine Web Functional test tool offers the following:

- Automatic recording of any Web Browser events and translate it into a clear and editable scripting language.
- Easy-to-use script editor to edit the Python scripts.
- Application Map Editor to view and edit the map object properties (window and element properties) of a web page.
- Ability to add checkpoints (GUI/Database/File checkpoints) to compare the expected outcome with the actual ones from the test run.
- Requires no proficiency in Scripting Language. Point and click User Interface to create scripts.
- Provision for Custom Plug-ins offers great flexibility to handle unique user test requirements.
- Data driven testing makes it possible to automate large number of tests with minimum effort.
- Support for testing multiple applications without too much additional resources.
- Unattended test execution allows test execution overnight, with accurate, repeatable results.
- Comprehensive HTML-based reports to indicate the status of the test execution. Summary reports for test scripts and test cases which links to the passed and failed test cases. This helps you to effectively review failures.
- Automated error recovery during testing.

Online Resources

QEngine includes the following online resources:

What's New: Provides the latest features and enhancements in the current version of the product.

Product Documentation: Provides the complete user guide for all the test tools in QEngine in HTML, ZIP and PDF format.

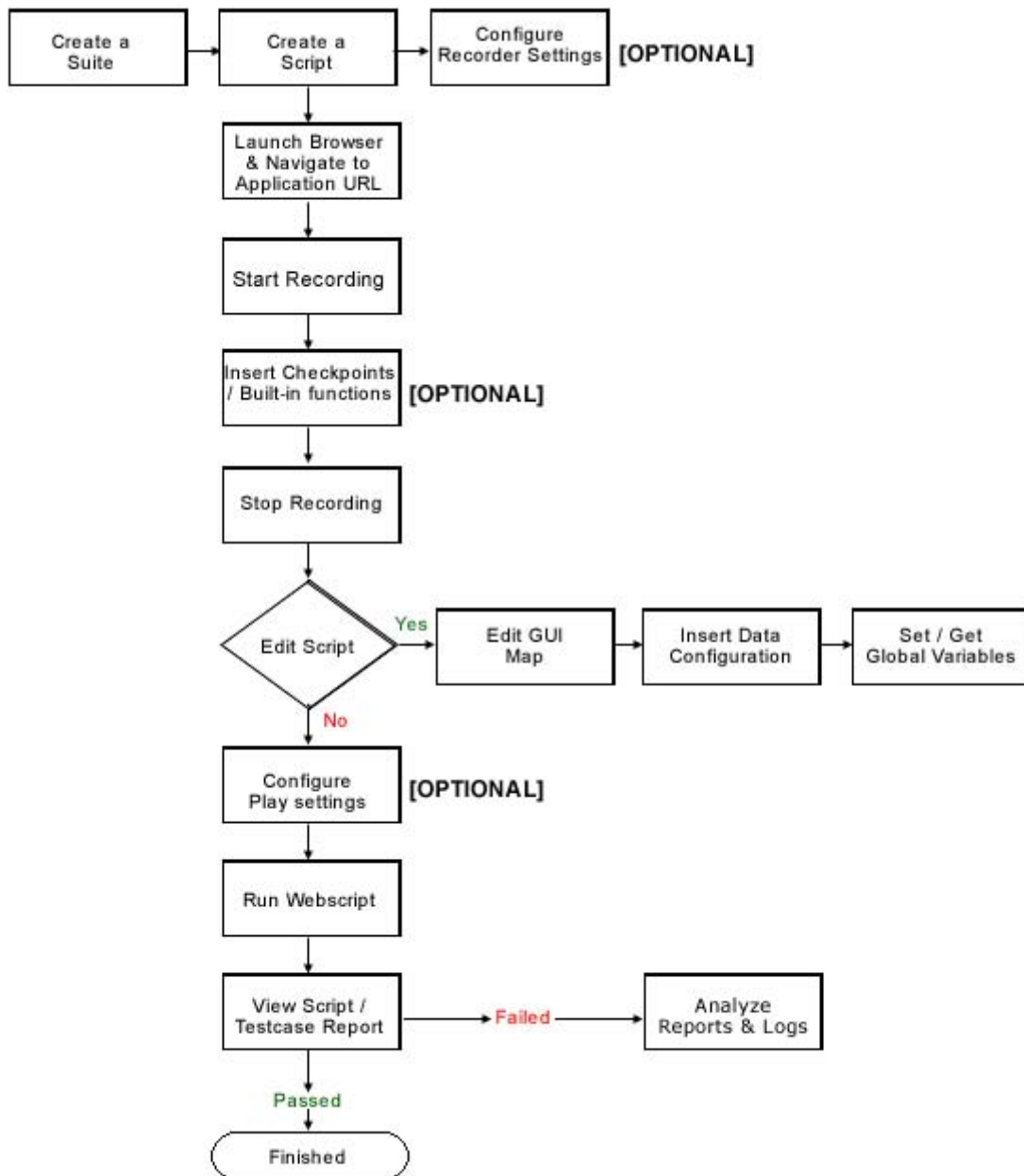
Datasheet: Provides the complete list of features with screenshots for all the test tools in QEngine.

Comparison Documents: Provides the comparison documents for Web Functional, Web Performance and Web Services test tools of QEngine.

Starting the Testing Process

QEngine Testing Process

Testing a web application may vary from one organization to another. Some organizations may need to automate their web application to perform some simple tests while others may need to automate the testing of complex web applications from early in development through application deployment. The flowchart given below illustrates the complete flow of how to test your web application using QEngine.



Sample Application

A sample **Payroll application** named **AdventNetPayrollSystem** has been bundled with QEngine in `<QEngine_Home>/projects/AdventNetPayrollSystem` directory to illustrate the various features of Web Functional Test tool. To understand the sample application and how to execute the sample scripts, please refer to Help->Tutorials->Adding/Executing Web Functional Test Cases topic by opening the `<QEngine_Home>/help/index.html` file or view the online help in <http://www.manageengine.com/products/qengine/help/index.html>.

Installing/Uninstalling QEngine

Installing QEngine

Installation in Windows

To install QEngine in Windows, download the EXE file **AdventNetQEngine_6_0_0.exe** from the site www.manageengine.com and install it by executing the EXE file.

Installation in Linux

To install QEngine in Linux, download the BIN file **AdventNetQEngine_6_0_0_Linux.bin** from the site www.manageengine.com and execute it.

Uninstalling QEngine

In Windows - ManageEngine QEngine can be uninstalled through the OS Control Panel Add / Remove Programs options.

In Linux - ManageEngine QEngine can be uninstalled by removing the directory containing the files and directories extracted during installation. At the command prompt, type **rm -rf AdventNet/QEngine** to successfully uninstall QEngine.

Starting/Shutting down QEngine Server

Starting QEngine Server

After installing QEngine, execute the **StartWebTestServer.bat/.sh** file in `<QEngine_Home>/bin` directory to start the QEngine server. The server will be started in the default server port 4444.

Shutting down QEngine Server

To shutdown QEngine server, execute the **ShutDownWebTestServer.bat/.sh** file in `<QEngine_Home>/bin` directory.

Installing QEngine Toolbar

After starting the QEngine server, the first step is to install/download the QEngine toolbar. As soon as you start the QEngine server, a page pops up to help you install the QEngine toolbar. The details for first time users and existing users are given in this page.

Connecting and Logging into QEngine Server

To connect to the server, open the browser (IE, Mozilla or FireFox) in which you have installed the QEngine Toolbar and click on the **Connect** option in QEngine Toolbar. This will bring up the **Connection Details** screen. In this screen, specify the following details:

1. Server Host - The host name where QEngine server is running.
2. Server Port - The port where the QEngine server is running.
3. Choose the connection type -**Direct Connection** or **Use Proxy**. If the connection has to be done through proxy, then specify the Proxy host name, Proxy port where the proxy server is running and the username and password to authenticate and connect to the proxy server.
4. After specifying the QEngine server details and LAN Connection Details, click the **OK** button to connect to the server. This will display the login page.
5. Enter the appropriate username and password and click the **Login** button to authenticate and login. This will display the SuiteManager page which allows you to create suites and perform functionality testing of your web applications.

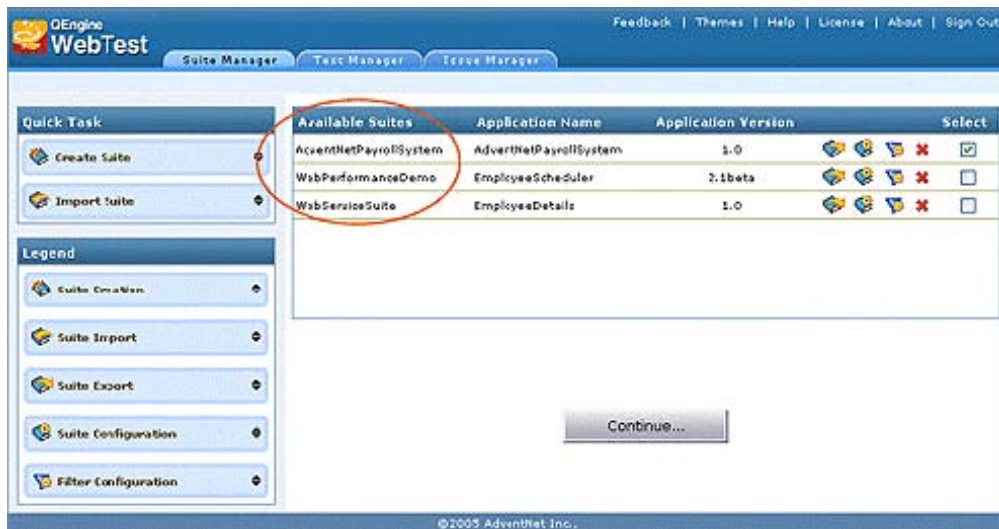
Understanding the Basics

Exploring the Suite Manager UI

The Suite Manager UI is displayed after connecting and logging-into QEngine server.

Creating Test Suite

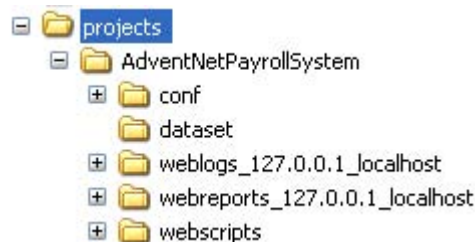
In the SuiteManager page as shown below, select the required suite from the **Available Suites** table or create a new suite by clicking the **Create Suite** option in **Quick Task** from the left pane. The **Quick Task** section also provides the **Import Suite** option to import a suite.



For each suite in the **Available Suites** table, you can use the export option to export a suite as a .ged file, configure the suite details such as, proxy details, severity, bug tracker, mail server details and reports using the Suite Configuration UI, set filters to execute specific scripts and delete a suite.

QEngine Suite Structure

A webscript is created under a defined test suite. QEngine test suite has the following structure:

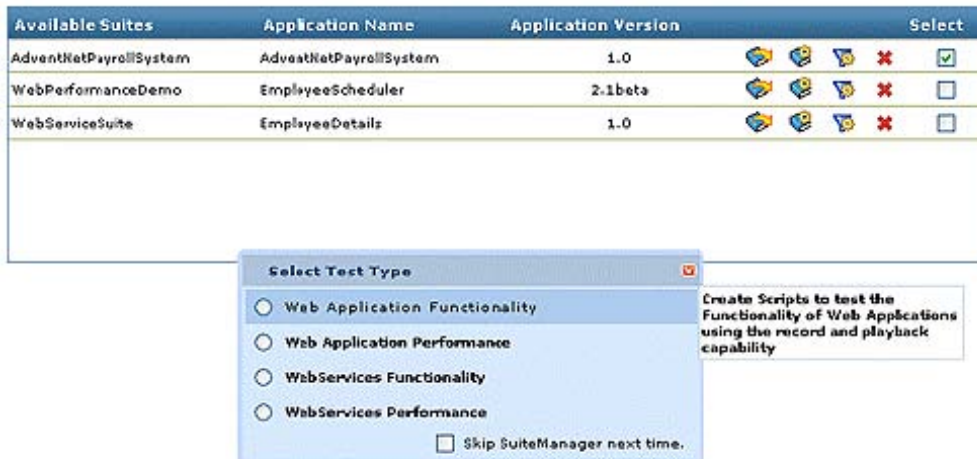


AdventNetPayrollSystem	The Suite name.
conf	Contains configuration files required to execute the Suite.
dataset	Contains the CSV files and datasource files to include data-driven test scripts wherein the values are dynamically fetched from a CSV, database or using variable substitution option.
webreports	Contains the reports generated for web functional test scripts.

weblogs	Contains the logs created while executing the web functional test scripts.
webscripts	Contains scripts recorded for web functional testing.

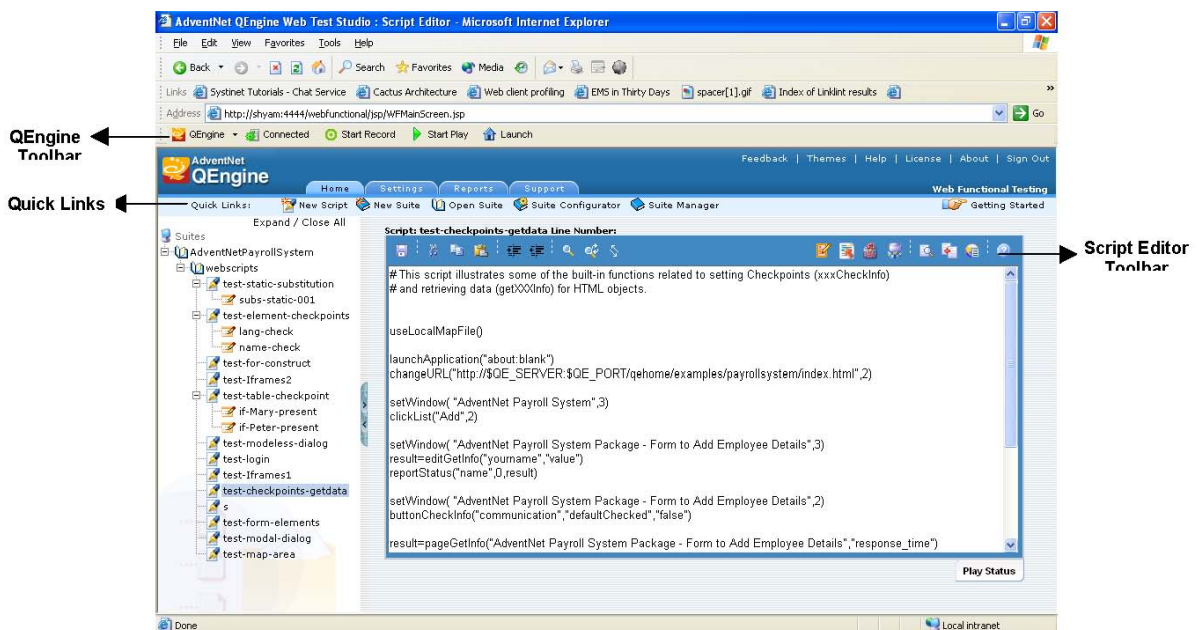
Selecting Test Type and Creating Webscripts

Click on the **Continue** button to select the test type (Web Application Functionality) as shown in the below image:



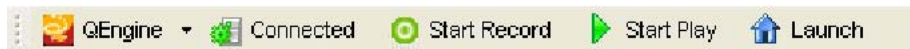
This will bring up the Web Functional Main Screen page to test the functionality of your Web applications. In the main screen page that gets displayed based on the selected test type, click **New Script** from the **Quick Links** placed below the tabs to create webscripts.

Exploring the Main Web Functional UI Exploring QEngine Toolbar Options



Exploring QEngine Toolbar Options

The function of each option available in QEngine Tool Bar are explained in the table below:



Toolbar Option	Description
	Provides options to go to manageengine website, forums, hide toolbar and uninstall toolbar.
	Easy way to connect to the QEngine server. Connected indicates that you have already connected to the QEngine server.
	To start/stop the recording of web functional test scripts.
	To start/stop the playback of the recorded web functional test scripts.
	To launch the default browser and load the web application URL to be tested.

Exploring Quick Links Options

The function of each option available in Quick Links are explained in the table below:

Quick Links Option	Description
	Invokes the Script Creation screen to create a new script.
	Invokes the Suite Creation screen to create a new Suite.
	Opens the selected test suite.
i	Invokes the Suite Configuration UI to configure the proxy details, suite details, severity, bug tracker, mail server details and test reports.
	To move to the Suite Manager page.

Exploring Script Editor Toolbar Options

The function of each toolbar option available in the script editor are explained in the table below:

Script Editor Toolbar Options	Description
	Invokes the Single Checkpoint screen to create a single checkpoint.
	Invokes the Testcase Configuration screen to create a test case.
	Invokes the Function Generator tool to insert built-in functions.
	Invokes the Data Configuration screen to fetch data from CSV, database or using variable substitution option.
	To view the GUI Map file and edit the physical properties of the GUI objects.
	To merge local map files to a global map file.
	Invokes the Global Variables Editor screen to set global variables.
	Invokes the help documentation for QEngine.

How Tos

Working with Test Suites

What is a Suite ?

Suite is a collection of test scripts, test data, environment data and configuration data. Suite helps you to store test information in an organized manner and share configuration and environment data amongst the different test types it contains.

How do I create a new Suite ?

To create a new Suite, from the Suite Manager UI , click the **Create Suite** option in the **Quick Task** section in the left pane.

How do I import a Suite ?

Use the **Import Suite** option in the Suite Manager page to import a Suite. The Suite Manager page gets displayed after connecting to the server and after logging-in. From the Suite Manager page, choose the **Import Suite** option in the **Quick Task** section to browse and specify the .qed file and click on the **Import** button to import the suite.

How do I export a Suite ?

A Suite can be exported by clicking the icon next to each of the suite in the **Suite Manager** page. The Suite Manager gets displayed after connecting to the server and after logging-in. Choose the icon or the **Export Suite** option in the **Quick Task** section in the Suite Manager page to export the suite. The exported Suite will be available as a .qed file under < QEngine Home > directory.

Creating/Editing Scripts

How are scripts created ?

To create scripts, choose **New Script** option from **Quick Links** placed below the tabs in the top frame or right-click the **webscripts** node from the Web Functional Main UI and choose the **Create New Script** option from the popup menu. In the dialog, specify the name of the new script file and click the **OK** button. The script file is by default saved in <QEngine_Home>/webscripts/<script_name> directory. You can also save the script file in any other directory under <QEngine_Home>/webscripts/ directory (by creating a new directory).

How do I record events ?

To record events:

1. Create a New Script by choosing the **New Script** option from **Quick Links** placed below the tabs in the top frame. Or, right-click the webscripts node from the Web Functional Main UI and choose the **Create New Script** option from the popup menu. In the dialog, specify the name of the new script file and click the **OK** button. This will create a new script.
2. Launch a browser using the **Launch Browser** option in QEngine toolbar. The default browser will be launched. Specify a valid URL in the launched browser.
3. To record the events, choose the **Start Record** toolbar option from the QEngine toolbar. The list of actions performed in the Web page are recorded in the newly created script file. The script file can be viewed in the script editor.

Can I edit a Script ?

Yes, you can edit a script to insert functions, create data-driven test scripts, add single checkpoints or multiple checkpoints, add custom functions, include looping constructs, conditional statements, set <QEngine_Home>/projects/<Suite_Name>/conf directory.

Can I add looping and conditional statements ?

Yes, you can add for loop constructs, if...else conditional statements supported in standard Jython language. Please refer to the link: <http://www.jython.org/j-jython1-ltr.pdf> to know the syntax of using Jython constructs.

Can I call one script from another ?

Yes, you can call one script from another using the callScript(). The web script to be called and its location are specified as arguments as

callScript("<Web Script Name>", "<Web Script Path>"). For example, assume the web script named **test.wcs** is present under **webscripts/case1/test** folder, then callScript is to be written **ascallScript("test", "case1/test")**.

Working with GUI Map

What is a GUI Map ?

A GUI Map file contains the window and element properties in a Web page. This file is created during a test run to represent the actual objects in your Web application. Each window or element node in this file has a list of properties that uniquely identifies the object in your Web application. During playback, QEngine reads the window and element properties from this GUI map file to identify and check the corresponding objects in your Web application.

How does QEngine identify a GUI object ?

QEngine has the **ID** attribute for each window and element node in the map file to uniquely identify each object in your Web application. This attribute is generated based on the physical properties of the object. This attribute is non-editable. For example, for a text box object which has the physical properties such as, name="loginName", formname="loginForm", type="text", disable="false", etc, QEngine generates the identification attribute for this text box object as ID="loginForm. The ID attribute and the physical properties of the object together ensure that each GUI object has its own unique identification.

How do I view/edit the GUI object properties ?

To edit the GUI object properties, select a script file from the Suites tree and right-click. A Popup menu is displayed. Choose **Edit Application Map** menu item or choose the **View Map File** option from the script editor toolbar. This will display the Application Map Editor. You can view the window and element nodes in the left pane and the property name and values in the right pane. Edit any property value for the selected node, except the **ID** attribute and click the **Update** button. After editing all the required properties, finally click the **Save** button to save the changed values in the appropriate application map file.

How do I decide on which GUI Map File mode to use ?

First time users or evaluation users can use the local map file mode. When you have a small number of scripts per Suite, you can go with the local map file mode. When you have large number of scripts per Suite and if you are editing the GUI map file for multiple scripts to suite specific needs then you can use the Global Map File mode. This mode will use a single **map file for the entire suite** of test scripts. The default option is local map file. The first line **useLocalMapFile()** in each script indicates that the script has a local map file. Note that in the case of global map file option, the **useLocalMapFile()** command is not embedded in the webscript.

How do I use a Global Map File for my script ?

To enable the Global map file mode, choose the **Settings** tab. In the Play Settings page, from the Recorder Settings options, choose the check box **Use Suite Level GUI Map**. This mode creates a global map file for the entire Suite. The global map file is placed in.

How do I merge my local map files ?

To merge the local map file into a global map file, choose the **Merge Map File** option



from the script editor toolbar option. A confirmation dialog is displayed. Clicking Yes, merges the local map file(s) to global map file and displays a message box to remove the **useLocalMapFile()** statement from the script.

Can I unmerge a global map file to local map file ?

You can unmerge the global map file by renaming the **<YouMap_File_Name>.map.merged** to **<YouMap_File_Name>.map**. After adding **useLocalMapFile()** in the first line of the script, it can run using the local map file.

Can I manually add an object to a GUI Map file ?

No, you cannot manually add an object in a GUI Map file. You can only edit the GUI Map file.

Creating Checkpoints / Test Cases

What are checkpoints ?

A checkpoint helps you in verifying the properties of an object in an HTML page. You can use a single property checkpoint to validate a single property value of an object or use test case type of checkpoint to validate multiple properties of GUI objects (such as HTML Element, Table or Text) in your Web application. Checkpoints can be inserted only in recording mode.

How do I insert a single checkpoint ?

The steps to insert a single checkpoints are:


1. In recording mode, choose the Insert Checkpoints option




in the script editor toolbar. This will display the Single Checkpoint Configuration tool.

1. The Category list box will display the HTML Check Functions category.
2. The Functions list box will display all the HTML Check functions.
3. Click on the hand icon present next to the **Functions** list box. From the browser in which you are recording the Web page, click on the required element whose property you want to check. Ensure that you choose the same type of element based on the function you have chosen, i.e., if you have chosen **editCheckInfo** function then choose a **text object**.
4. In the Single Checkpoint Configuration tool you will be able to view the element ID of the chosen object in the text field "**Element ID**". The properties of the object will be listed in the list box named "**Property**" and also the default value is listed in the text field named "**Value**". Choose the desired property that you want to verify and enter the value to be compared with.
5. Click on the button "**Paste**" which will insert the single checkpoint statement in the script where the cursor is placed.


Can I check for a single property ?

Yes, you can check for a single property of an object such as, a button is enabled or disabled or  whether an item in a list is selected or not using the **Insert Checkpoints** option in the script editor toolbar.

Can I check for a single object ?

Yes, you can check for a single object with its default properties or you can specify which properties to  check using the **Insert Testcase** option in the script editor toolbar. This will display the Test Case Configuration screen. Specify the test case details and choose the **HTML Check** radio option and then choose the **Element** radio option in the Test Case Configuration screen. From your Web page, select the object or HTML element whose properties need to be checked. The Test Case Configuration screen will display all the properties of the selected object. Select the required


Can I check two or more objects in a Window ?

Yes, you can check for two or more objects in a Window such as the button properties, link properties, check for a cell value in a table, etc. by adding a testcase with multiple checkpoints using the  **Insert Testcase** option in the script editor toolbar.

Can I include expressions or specify conditions in a testcase ?

Yes, you can add multiple checkpoints or a combination of checkpoints in a test case and specify conditions or expressions for the test case to pass. The Testcase Configuration screen provides the three radio options: Satisfy Any Criteria (OR Condition), Satisfy All Criteria (AND Condition) and Satisfy Condition (Use Expressions). For example, you can add a GUI checkpoint, Table Checkpoint and a Text checkpoint and specify the condition as Satisfy All Criteria (AND Condition) wherein during playback QEngine will check for the specified condition and reports the test case as Passed only if all the checkpoints return the status as Passed.


How do I search for a text in a HTML page ?

 in the script editor toolbar. This will display the Test Case Configuration screen. Specify the test case details and choose the **HTML Check** radio option and then choose the **Text** radio option in the Test Case Configuration screen. From your Web page, highlight the text to be searched. The Test Case Configuration screen will display the highlighted portion of the text in a textarea. Select the **Prefix text**, **String to Find** and the **Suffix text** to search for the given text. Click **Commit To List** and click **Apply**.


To search for a text in a HTML page, choose the **Insert Testcase** option

How do I search for the non-existence of a text in a HTML page ?

To search for the non-existence of a text in a HTML page, choose the **Insert Testcase** option


 in the script editor toolbar. This will display the Test Case Configuration screen. Specify the test case details and choose the **HTML Check** radio option and then choose the **Text** radio option in the Test Case Configuration screen. From your Web page, highlight the text which should not exist. The Test Case Configuration screen will display the highlighted portion of the text in a textarea. Select the **Prefix text**, **String to Find** and the **Suffix text** to search for the non-existence of the given text. Click the check box "**Search String Should Not Exist**" placed below the **Suffix text** and click **Commit To List** and click **Apply**. This will check for the non-existence of the text.

How do I check all the table properties ?

To check all the properties of a HTML table such as row count, column count and cell value at a particular row and column, choose the **Insert Testcase** option 

in the script editor toolbar. This will display the Test Case Configuration screen. Specify the test case details and choose the **HTML Check** radio option and then choose the **Table** radio option in the Test Case Configuration screen. From your Web page, click on the table whose properties has to be checked. The Test Case Configuration screen will display the **Property, Condition** and **Value** columns to check the row count, column count and cell value. Click **Add** button to add more properties in the selected table. Click **Apply**. In the Test Case Configuration screen, click **Commit To List** and click **Apply**. This will check for all the specified table properties.

Can I create a database checkpoint ?

 in the script editor toolbar. This will display the Test Case Configuration screen. Specify the test case details and choose the **DB Check** radio option. Here, you may check for Row Count, Column Count, Table Values or search for a table.

Yes, you can create a database checkpoint by choosing the **Insert Testcase** option


Can I edit a test case ?

Yes, you can edit a test case. Choose the script which contains the test case from the left tree-view of Web Functional Main UI. The test cases created for the script will be displayed as the sub-nodes. Click the test case to be edited. This will display the Test Case Configuration screen in the right pane. Edit the values and click **Commit To List** and then click **Apply**.

Inserting Built-in Functions

How to insert functions ?

To insert functions in your test script:

1. Choose the **Insert Built-In Function** option  from the script editor toolbar. This will display the Function Generator tool.
2. From the Category list box, choose the category to which the function belongs. Based on the selected category, the functions list will be populated.
3. From the Functions list box, choose the desired function to be inserted in the script.
4. If the selected function returns some data such as the value of a property then you may want to display it. To do so, check the **Log property retrieved checkbox which will insert the function "displayMessage(result)"** in the script to display the retrieved value.
5. Click on the hand icon present next to the **Functions** list box. From the browser in which you are recording the Web page, choose the required element whose property you want to check. Ensure that you choose the same type of element based on the function you have chosen, i.e., if you have chosen **editGetInfo** function then choose a **text object**.
6. Based on the object you have chosen, the appropriate properties will be listed in the list box labeled "**Property**". Select the required property to be verified.
7. Click on the button "**Paste**" which will insert the selected function statement at the cursor location.

How do I get the number of rows and columns in a Table ?

To get the number of rows and columns in a Table, you can use the `getHTMLTableRowCount()` and `getHTMLTableColumnCount()` in the HTML Table Functions category of Function Generator tool. To know the details of using these functions, please refer to Appendix->Using Built in Functions->HTML Element Functions->HTML Table Functions topic.

How do I get the table cell value of a HTML Table element ?

To get the table cell value in a particular row or column of a table, you can use the `getCellValueAt()` in the HTML Table Functions category of Function Generator tool. To know the details of using this function, please refer to `getCellValueAt()` in Appendix->Using Built in Functions->HTML Element Functions->HTML Table Functions topic.

How do I test a dynamically changing link element ?

Assume you have a link in a HTML table and a new link gets added everytime an action is performed. So, when you run the script for the second time, you will find a new link in the table with a different inner text. Here, it is assumed that the link id is the same for all the links. To test the functionality of such a dynamically changing link element, you can use the `fireEventOnCellElement()` in HTML Table Functions category of Function Generator tool. Please note that you have to insert this function in non-recording mode. To know the details of using this function, please refer to Appendix->Using Built in Functions->HTML Element Functions->HTML Table Functions topic.

How to capture the property value and store it in a variable ?

To capture the property value and store it in a variable, you can use the `getElementProperty()` in HTML Get Functions category of Function Generator tool. To know the details of using this function, please refer to Appendix ->Using Built in Functions->HTML Element Functions->HTML Get Functions topic.

Yes, you can display custom messages in the script report and test case report using the `reportStatus()` (for script report) and `reportTestCase()` (for test case report). To know the details of using this function, please refer to Appendix ->Using Built in Functions->General Functions topic.

Can I programmatically look for objects of a certain type on a web page ?

Yes, you can programmatically look for objects of a certain type on a web page and also fire events on the objects of specific type, by inserting the following functions in non-recording mode using the Function Generator tool:

`getElementProperty()` - To get the property value of a HTML element with the specified tag name.
`fireEventOnCellElement` - To invoke or fire the specified event for the given HTML Table Cell element.
`fireEventOnElement` - To invoke the specified event for the given HTML element.

To know the details of using these functions please refer to Appendix ->Using Built in Functions->HTML Element Functions topic.

How to work with windows that dynamically change its title?

Assume you have a window title such as "Acme_New_York" where the first part of the title does not change and the second part dynamically changes based on the selected location. To handle this case, you can edit the GUI Map file. Select the recorded script from the tree-view of Web Functional Main UI. Right-click and choose the **Edit Application Map File** option from the popup menu or choose **View Map File** option from the script editor toolbar. This will display the **Application Map Editor** screen. Select the window node from the tree-view for which the title is dynamically changing. In the right pane, for `windowtitle` and `parenttitle` Property, from the **Condition** column, choose the condition as "starts with" and in the **Value** column specify the value as "Acme". Click the **Update** button and then click on the **Save** button. Now, playback the script. This will only check for the starting word "Acme" during playback wherein your script will be successfully replayed without any errors.

How do I check the color or font of text links ?

You can check the style properties of a html element using the styleCheckInfo() in the HTML Check Functions category of Function Generator tool. The style properties that can be validated are cursor, color, background-color, font, font-family, font-size, font-style, text-decoration, width, height, background-image and text-align. To know the details of using this function, please refer to Appendix->Using Built in Functions->HTML Element Functions->HTML Check Functions topic.

Can I add custom Jython functions in my script ?

Yes, you can add custom Jython functions in your script. Create a python script <fileName>.py and save it under <QEngine Home>/jars/ directory, and define your functions in that file. Instead of saving the .py file under <QEngine_Home>/jars directory, you can also place it in any other directory and edit the file **setcommonenv.bat/sh** file to provide the path to the ".py" file for the variable PY_PATH. In your script, you can import the .py file using checkbox. This will insert the command " from "<filename>" import * This must be done in the first line of the script. Now, you can use the functions defined in the .py file.

Can I set and get values across scripts ?

Yes, you can set or get values in a global variable across scripts or from one script to another while using the callScript(). You can setGlobal("<variable_name>","<variable_value>") in script1, and in script10 you can get the global variable value using getGlobal("<variable_name>"). You can use the **Global Variables Editor** option in the script editor toolbar to configure the global variables. The syntax and usage for setting and getting global variable values are as follows:

Syntax:

```
setGlobal("<variable_name>","<variable_value>")
getGlobal("<variable_name>")
```

How do I wait for objects or windows to appear ?

To wait for objects or windows to appear, you can use the waitForPageDownload function to wait until the document ready state is reached or until the value specified for **Maximum waittime for document ready** in Play Settings option is timed out.

Can I use regular expressions in functions ?

Yes, you can use regular expressions in the following functions:

- getElementProperty()
- getHTMLElementCount()
- fireEventOnElement()
- doesElementExists()
- fireEventOnCellElement()

To know the details of using these functions, please refer to Appendix->Using Built in Functions->HTML Element Functions topic. getElementProperty() and getHTMLElementCount() in HTML Get Functions topic. fireEventOnElement() and doesElementExists() in HTML General Functions topic. fireEventOnCellElement() in HTML Table Functions topic.

How to print a message in the log file ?

To print a message in the log file, you can use the displayMessage() in the Function Generator tool. This will display the given message in the log file testout.txt in <QEngine_Home>/projects/<Suite_Name>/<webreports_ipname_hostname_currentdate_currenttime > directory.

How to start other applications from test script ?

To start other applications from your test script, you can use the `invokeApplication()` to invoke other applications or scripts from your test script. Use the Function Generator tool to insert this function in the script.

E.g., `invokeApplication("C:\test\testscript.sh")`

There are also other related functions such as `invokeApplicationInThread`, `invokeApplicationWithArgs` and `invokeApplicationInThreadWithArgs`.

How do I compare two files ?

To compare two files, you can use the `compareFile()` in the File Operations category of Function Generator tool. This function compares the given base file name with the given check file name. To know the details of using this function, please refer to Appendix->Using Built in Functions->File Operations topic.

Data-Driven Test Scripts

What are data-driven scripts ?

Data-Driven scripts allows you to add variables or parameterize lines in your test scripts. Test scripts can be created with varying input where the values are fetched at runtime from an external database, CSV file or from user-defined variables, or reserved variables.

How to create data-driven test scripts?

To create data-driven test scripts, invoke the Data Configuration screen by choosing the **Data Configuration** option from the script editor toolbar.

How to import data from a database ?

To know the details of how to import data from a database, please refer to the following URL in online help:

http://www.manageengine.com/products/qengine/help/context_sensitive_help/data_configuration.html

How to import data from a CSV file ?

To know the details of how to import data from a CSV file, please refer to the following URL in online help:

http://www.manageengine.com/products/qengine/help/context_sensitive_help/data_configuration.html

Exception Handling for Unattended Execution

How do I handle web exceptions ?

To handle web exceptions, from Web Functional Main UI, choose the **Settings** tab. This displays the Web Functional Settings screen. In this screen, from Exception Handling->Web Exceptions choose the required options: Stop Play, Report & Continue Play, and/or CallScript to handle web exceptions. Choose **Apply** to apply the changes.

How do I handle script and object exceptions ?


You need not handle these explicitly. Web Functional Test takes care of logging these in the log files and continue the test case execution as appropriate. You will be able to see these in the log files. In case of object exceptions, you can use the `callScript` function to call another script to perform some specific operation based on the exception received.

How do I handle unexpected popups during playback ?

To handle unexpected popups during playback, from Web Functional Main UI, choose the **Settings** tab. This displays the Web Functional Settings screen. In this screen, from Exception Handling->Popup Exceptions choose the required options: Report & Continue Play, Capture Screen & Continue Play, Stop Play, Close & Continue Play and/or CallScript to handle unexpected popups. Choose **Apply** to apply the changes.

Running Scripts

How do I run a test script to check my application ?

To run a test script, select the script from the Suites Tree and choose the toolbar icon  Start Play from the QEngine toolbar option. This executes the test cases (if any) under the selected script or executes the script (replays the browser events).

Can I test my scripts in any locale without re-recording ?

Yes, you can test the scripts in any locale without re-recording by configuring the options in Local Settings in Web Functional Settings screen. To view the Web Functional Settings screen, choose the **Settings** tab in Web Functional Main UI.

How do I play the recorded script with any server?

Assume you have scripts which has application/server name and port details in the recorded URLs. To playback the recorded script against any server or port without re-recording, configure the appropriate details in the **Host-Port** Settings screen in Web Functional Settings UI. To view the Web Functional Settings UI, choose the **Settings** tab in Web Functional Main UI.

QEngine can handle unexpected web exceptions, popup exceptions, script and object exceptions.

How do I run my test script against other databases like oracle or sql server ?

To run the test script against other databases like oracle or sql server, you need to set the appropriate classpath for the database driver by editing the DB_CLASSPATH variable in **setcommonenv.bat/sh** file in `<QEngine_Home>` directory. Also, you need to add the new database details in the Database Configuration screen. To invoke the Database Configuration screen, from Testcase Configuration Screen, choose the **DB Checkpoint** radio option and click the "Configure" button next to the **Database** field or invoke it from Data Configuration screen.

Is there any settings to be done before running a test ?

The settings are not mandatory. Based on your requirement, you can configure the appropriate values before running a test. From the Web Functional Main UI, choose the **Settings** tab. This will display the Web Functional Settings screen. In this screen, you can configure the replay speed, select options to handle exceptions, configure e-mail notification for failed test cases, locale settings to playback in the selected locale and host-port settings to dynamically change the host name and port number in the test scripts without re-recording.


Can I run scripts from command line mode ?

Yes, you can run test scripts from command line mode. Command Line Test Execution facilitates bulk test script execution (entire suite can be executed through command line) from a remote host. From the Suite Manager page, choose the **Test Manager** tab and download the command line toolkit to run the test suites from command line.

Can I run scripts automatically at scheduled intervals ?

Yes, QEngine provides test scheduler to run scripts automatically at scheduled intervals such as hourly, weekly, daily or monthly. From Web Functional Main UI, click the **Test Manager** tab to schedule test suites for unattended execution.

Can I execute specific test scripts ?

Yes, you can execute specific test scripts, by setting filters for a particular Suite. Filters allow you to execute selective scripts that match the specified filter criteria. You can configure filters for a particular suite, from the **Suite Manager** page. This page gets displayed after connecting to the server and after logging-in. In the **Suite Manager** page, from the right hand-side screen, click on the Filter icon  next to the respective suite for which the filter has to be configured.

Bug Tracking

How do I track product features and bugs ?

To track features and bugs, QEngine Issue Manager is by default bundled with QEngine. QEngine Issue Manager helps you to track product defects and manage product enhancement requests. It enables users to log in defects / requests from any geographic location and allows all the team members to access the tracking system from anywhere, anytime.

Can I integrate third party bug tracking system ?

Yes, you can integrate any third party bug tracking system by writing a custom class which should implement the interface **com.adventnet.testtools.server.ProblemReporterInterface**. Also, configure the custom class details in the Suite Configuration UI.

Appendix

Using Built In Functions

Web Functional Test provides Built-in functions for various operations. You can type these functions manually in the script but this may result in syntax error and you may find it difficult to debug. To facilitate non erroneous insertion, an utility called "Function Generator" is made available. To know the steps to use the Function Generator tool, refer to the context sensitive help. To know the complete list of functions, refer to the following links:

- Window Functions
- XML Validation Functions
- String Functions
 - HTML Element Functions
 - HTML Check Functions
 - HTML Get Functions
 - HTML Table Functions
 - HTML General Functions
- Database Functions
- File Operations
- General Functions
- Default Functions

Window Functions

- setDynamicWindow
- closeAllWindows
- doesWindowExists
- winCheckInfo
- winGetInfo

setDynamicWindow

Function Description:

To set browser windows that does not appear during recording but appears during playback. These windows will not have any map entries. Action on any element in this window can be achieved using fireEventOnElement. The parameters include:

1st argument - parent_window_title

2nd argument - parent_window_occurrence,

3rd argument - Title or Name as the property name. Here, Title is the frame_window_title and Name is the frame_window_name.

4th argument - Value of the Title or Name specified in the 3rd argument.

5th argument - frame_window_occurrence

How to Define:

setDynamicWindow

(parent_window_title,parent_window_index,frame_window_title or
frame_window_name,value_of_frame_title or value_of_name,frame_window_index)

Example:

Eg: setDynamicWindow("AdventNet",1,"frame_window_title","payroll",1)

Return Values:

0 for Success

1 for Failure

closeAllWindows Function Description:

To dynamically close all the windows opened during playback. No parameters required.

How to Define:

NA

Example: -

0 for Success 1 for Failure

doesWindowExists

Function Description:

To check if a browser window exists during playback. These windows will not have any map entries. If exists return 0(true), else 1(false). The parameters include:

- 1st argument - parent_window_title
- 2nd argument - parent_window_occurrence,
- 3rd argument - Title or Name as the property name. Here, Title is the frame_window_title and Name is the frame_window_name.
- 4th argument - Value of the Title or Name specified in the 3rd argument.
- 5th argument - frame_window_occurrence

How to Define:

doesWindowExists
 (parent_window_title,parent_window_index,frame_window_title or
 frame_window_name,value_of_frame_title or value_of_name,frame_window_index)

Example:

Eg: doesWindowExists("AdventNet",1,"frame_window_title","payroll",1)

Return Values:

- 0 for Success
- 1 for Failure

winCheckInfo Function Description:

To compare the value of the specified property of a window object with the given value.

How to Define:

winCheckInfo("Object ID", "property", "value")
 ObjectID = Generated by QEngine.
 Property = Required Property name to be checked.
 Value = Required Property value to be checked.

Example: winCheckInfo("The

Product Presentation
 Team","windowtitle","The
 Product Presentation
 Team")

Return Values:

- 0 for Success
- 1 for Failure

To retrieve the value of the specified property of the selected window.

How to Define:

```
result = winGetInfo("Object ID", "property")  
ObjectID = generated by QEngine.  
Property = Property to be retrieved.
```

Example:

```
result=winGetInfo("AdventNet Payroll System Package - Form to Add Employee  
Details","framecount")  
displayMessage(result)
```

Return Values:

Value of the specified property

XML Validation Functions

- doesNodeContainValue
- getNodeAttribute
- getNodeCount
- getNodeValue
- getNodeValues
- getTextNodeValues
- isNodePresent
- isIdentical
- isSimilar



Note: To know the appropriate xml expression to be used for xpath in the below functions, please refer to this URL
http://javaalmanac.com/egs/org.w3c.dom/xpath_GetChildElem.html

doesNodeContainValue

Function Description:

To check whether the specified node contains the given value in the XML file.

How to Define:

doesNodeContainValue(xmlfile,expression(xpath))

xml file = Valid XML file path.

xpath = Valid xpath.

Return Value:

0 for Success

1 for Failure

getNodeAttribute Function Description:

To get the node attribute from the specified XML file.

How to Define:

getNodeAttribute(xmlfile,expression(xpath)) xml file = Valid XML file path.

xmlpath = Valid xpath.

Return Value:

String

getNodeCount Function Description:

To get the node count from the specified XML file.

How to Define:

getNodeCount(xmlfile,expression(xpath)) xml file = Valid XML file path. xmlpath =

Valid xpath.

Return Value:

Integer

getNodeValue Function Description:

To get the node value from the specified XML file.

How to Define: getNodeValue(xmlfile,expression(xpath)) xml file = Valid XML file path.

xmlpath = Valid xpath.

Return Value:

String

getNodeValues Function Description:

To get the node values from the specified XML file.

How to Define: getNodeValues(xmlfile,expression(xpath)) xml file = Valid XML file path.

xmlpath = Valid xpath.

Return Value:

String array

getTextNodeValues Function Description:

To get the text node values from the specified XML file

How to Define: getTextNodeValues(xmlfile,expression(xpath)) xml file = Valid XML file path.

xmlpath = Valid xpath.

Return Value:

String array

isNodePresent Function Description:

To check whether the specified node is present in the XML file.

How to Define: `isNodePresent(xmlfile,expression(xpath))` xml file = xpath.alid

XML file path.

xmlpath = Valid xpath.

Return Value:

0 for Success

1 for Failure

isIdentical Function Description:

To check whether the two XML files has the same set of node values in the same order.

How to Define:

`isXMLIdentical(xmlfile,expression(xpath))` xml

file = xpath.alid XML file path.

xmlpath = Valid xpath.

Return Value:

0 for Success

1 for Failure

To check whether the two XML files has the same set of node values wherein the order can differ.

How to Define:

`isXMLSimilar(xmlfile,expression(xpath))`

xml file = Valid XML file path.

xmlpath = Valid xpath.

Return Value:

0 for Success

1 for Failure

String Functions

- endsWith
- indexOf
- length
- replaceAll
- replaceFirst
- split
- startsWith
- subString
- toLowerCase
- toUpperCase

endsWith Function Description: To check whether the specified String ("**String_Value**") ends with the specified end string value.

How to Define: endsWith("**String_Value**", "**End_String_Value**") Return

Values:

0 for Success
1 for Failure

indexOf Function Description:

To get the index within this string ("**String_Value**") of the first occurrence of the specified pattern string.

How to Define: indexOf("**String_Value**", "**Pattern_String**") Return Values:

Integer

length Function Description:

To get the length of the string.

How to Define: length("**String_Name**")

Integer

replaceAll Function Description:

To replace each substring of this string that matches the given search pattern with the given replace string.

How to Define:

replaceAll("String_Value", "Search_Pattern", "Replace_String")

Return Values:

String

replaceFirst Function Description:

To replace the first substring of this string that matches the given search pattern with the given replace string.

How to Define:

replaceFirst("String_Value", "Search_Pattern", "Replace_String")

Return Values:

String

split Function Description:

To split this string around matches of the given separator.

How to Define: split("String_Value", "Separator")

Return Values:

String Array

startsWith Function Description:

To check whether the string starts with the specified start string name.

How to Define: startsWith("String_Value", "Start_String_Value")

Return Values:

Values:

0 for Success

1 for Failure

substring Function Description:

To returns a new string that is a substring of this string.

How to Define: substring("String_Value", "Start_Index", "End_Index")

Return Values:

Return Values:

String

toLowerCase Function Description:

To convert the specified string to lowercase.

How to Define: toLowerCase("String_Value") **Return Values:**

String

toUpperCase Function Description:

To convert the specified string to uppercase.

How to Define: toUpperCase("String_Value") **Return Values:**

String

HTML Element Functions

HTML Element Functions: Introduction

HTML Element functions include the following categories:

- HTML Get Functions
- HTML Check Functions
- HTML Table Functions
- HTML General Functions

HTML Get Functions

- linkGetInfo
- editGetInfo
- imageGetInfo
- buttonGetInfo
- selectGetInfo
- listGetInfo
- pageGetInfo
- getElementWithFocus
- getElementIdentifier
- getHTMLElementCount
- webGetText

linkGetInfo Function Description:

To retrieve the value of the specified property of a link object.

How to Define:

```
result = linkGetInfo("Object ID", "property")
```

objectID = Generated by QEngine.
property = Property to be retrieved.

With Dynamic Property: If you need to overwrite (or dynamically change) the identification property for the HTML Element during playback then you can use the four arguments method as follows:

linkGetInfo("Object ID", "property", "<property_name>", "<property_value>") where property name is the identification property such as "id" or "innertext" and property value is the value to be overwritten during playback.

Example:

```
result=linkGetInfo("www.adventnet.com", "innertext")
displayMessage(result)
```

Return Values:

Value of the specified property

editGetInfo Function Description:

To retrieve the value of the specified property of a text object.

How to Define:

```
result = editGetInfo("Object ID", "property")
```

objectID = Generated by QEngine.
property = Property to be retrieved.

With Dynamic Property: If you need to overwrite (or dynamically change) the identification property for the HTML Element during playback then you can use the four arguments method as follows: editGetInfo("Object ID", "property", "<property_name>", "<property_value>")

where property name is the identification property such as "id" or "name" and property value is the value to be overwritten during playback.

Example:

```
result=editGetInfo("yourname","value")
displayMessage(result)
```

Return Values:

Value of the specified property

imageGetInfo Function Description:

To retrieve the value of the specified property of an image object.

How to Define:

```
result = imageGetInfo("Object ID", "property")
objectID = Generated by QEngine.
property = Property to be retrieved.
```

With Dynamic Property: If you need to overwrite (or dynamically change) the identification property for the HTML Element during playback then you can use the four arguments method as follows:

imageGetInfo("Object ID", "property", "<property_name>", "<property_value>") where property name is the identification property such as "id" or "src" and property value is the value to be overwritten during playback.

Example:

```
result=imageGetInfo("http://www.adventnet.com/images/free-edition-
appmanager.gif","source")

displayMessage(result)
```

Return Values:

Value of the specified property

buttonGetInfo Function Description:

To retrieve the value of the specified property of a button object. The object can be a push button, radio button, checkbox, submit and reset..

```
result = buttonGetInfo("Object ID", "property")
objectID = Generated by QEngine.
property = Property to be retrieved.
```

With Dynamic Property: If you need to overwrite (or dynamically change) the identification property for the HTML Element during playback then you can use the four arguments method

as follows:

`buttonGetInfo("Object ID", "property", "<property_name>","<property_value>")` where property name is the identification property such as "id" or "name" and property value is the value to be overwritten during playback.

Example:

```
result=buttonGetInfo("gender","defaultChecked")
displayMessage(result)
```

Return Values:

Value of the specified property

selectGetInfo Function Description:

To retrieve the value of the specified property of a single selection and multi selection list box (combo box) object.

How to Define:

```
result = selectGetInfo("Object ID", "property")
objectID = Generated by QEngine.
property = Property to be retrieved.
```

With Dynamic Property: If you need to overwrite (or dynamically change) the identification property for the HTML Element during playback then you can use the four arguments method as follows:

`selectGetInfo("Object ID", "property", "<property_name>","<property_value>")` where property name is the identification property such as "id" or "name" and property value is the value to be overwritten during playback.

Example:

```
result=selectGetInfo("department","selectedcount")
displayMessage(result)
```

Return Values:

Value of the specified property

To retrieve the value of the specified property of an element and <DIV> element.

How to Define:

```
result = listGetInfo("Object ID", "property")
objectID = Generated by QEngine.
property = Property to be retrieved.
```

With Dynamic Property: If you need to overwrite (or dynamically change) the identification property for the HTML Element during playback then you can use the four arguments method as follows:

listGetInfo("Object ID", "property", "<property_name>", "<property_value>") where property name is the identification property such as "id" or "name" and property value is the value to be overwritten during playback.

Example:

```
result=listGetInfo("list1","value")
displayMessage(result)
```

Return Values:

Value of the specified property

pageGetInfo Function Description:

To retrieve the value of the specified property of the selected page.

How to Define: result = pageGetInfo("Object

ID", "property") **Example:**

```
result=pageGetInfo("AdventNet Payroll
System Package - Form to Add
Employee Details","response_time")
```

```
displayMessage(result)
```

Return Values:

Value of the specified property

getElementWithFocus

Function Description:

To get the property value of the currently focussed HTML element for the given property name.

result = getElementWithFocus("<PropertyName>") **Example:**

```
result=getElementWithFocus("innertext")
```

```
displayMessage(result)
```

To get the innertext of the currently focussed element.

Return Values:

String

**getElementIdentifier(deprecated)
getElementProperty()****Function Description:**

To get the value for the specified property of a given HTML Element. Supports regular expressions.

How to Define:

```
getElementProperty("tagName", "propertyName",
"propertyValue", "iIndex", "propRequired", "useRegExp")
```

You can also use regular expressions for the 3rd parameter "**property value**" wherein **useRegExp** must be "true" to support regular expressions. By default, **useRegExp** is false. To know the details of using the regular expressions, refer to the following url:
<http://java.sun.com/developer/technicalArticles/releases/1.4regex/>

Example:

```
getElementProperty("A", "id", "linkid", 1, "innertext", "false")
```

The above function retrieves the innertext of a link element whose id="linkid".

Return Values:

Returns the property value as string.

getHTMLElementCount Function**Description:**

To get the count of the given HTML element with the specified property value. Supports regular expression

How to Define: getHTMLElementCount("tagName", "propertyName",

```
"propertyValue", useRegExp) Except tagname, all other parameters are optional.
```

You can also use regular expressions for the 3rd parameter "**property value**" wherein **useRegExp** must be "true" to support regular expressions. By default, **useRegExp** is false. To know the details of using the regular expressions, refer to the following url:
<http://java.sun.com/developer/technicalArticles/releases/1.4regex/>

Example:

getHTMLElementCount("A") - return the number of links in the HTML page or as
getHTMLElementCount("input","type ","password") - return the number of password input
elements in the HTML page.

Return Values:

Integer

webGetText Function Description:

To get the body text between the given prefix and suffix text from the HTML document.

How to Define:

webGetText(prefixstr, suffixstr)

Example:

```
result =webGetText("text1", "text2")
```

```
displayMessage(result)
```

Return Value:

String.

HTML Check Functions

- linkCheckInfo
- checkCellValueAt
- editCheckInfo
- imageCheckInfo
- buttonCheckInfo
- selectCheckInfo
- staticCheckInfo
- pageCheckInfo
- webCheckText
- styleCheckInfo
- listCheckInfo

linkCheckInfo Function Description:

To compare the value of the specified property of a link object with the given value.

How to Define:

linkCheckInfo("Object ID", "property","value")

ObjectID = Generated by QEngine.

Property = Required Property name to be checked.

Value = Required Property value to be checked.

With Dynamic Property: If you need to overwrite (or dynamically change) the identification property for the HTML Element during playback then you can use the five arguments method as follows:

linkGetInfo("Object ID", "property", "value", "<property_name>", "<property_value>")

where property name is the identification property such as "id" or "innertext" and property value is the value to be overwritten during playback.

Example:

```
linkCheckInfo("www.adventnet.com", "innertext","www.adventnet.com")
```

Return Values:

0 for Success

1 for Failure

checkCellValueAt Function Description:

To compare the value of the specified table cell property with the given value.

How to Define:

checkCellValueAt("Element ID","cellvalue","value_tobe_checked")

ElementID = Generated by QEngine.

cellvalue = The property name "cellvalue" to be checked.

With Dynamic Property: If you need to overwrite (or dynamically change) the identification property for the HTML Element during playback then you can use the five arguments method

as follows:

checkCellValueAt("Element ID", "cellvalue", "value_tobe_checked", "<property_name>", "<property_value>") where property name is the identification property such as "id" or "firstCellValue" and property value is the value to be overwritten during playback.

Example:

```
checkCellValueAt ("S.No", "cellvalue", "Product")
```

Return Values:

0 for Success
1 for Failure

editCheckInfo Function Description:

To compare the value of the specified property of a edit object with the given value.

How to Define:

```
editCheckInfo("Object ID", "property", "value")
```

ObjectID = Generated by QEngine.
Property = Required Property name to be checked.
Value = Required Property value to be checked.

With Dynamic Property: If you need to overwrite (or dynamically change) the identification property for the HTML Element during playback then you can use the five arguments method as follows:

```
editCheckInfo("Object ID", "property", "value", "<property_name>", "<property_value>")
```

where property name is the identification property such as "id" or "name" and property value is the value to be overwritten during playback.

Example:

```
editCheckInfo("yourname_1", "value", "Mary")
```

Return Values:

0 for Success
1 for Failure

Function Description:

To compare the value of the specified property of a image object with the given value.

How to Define:

```
imageCheckInfo("Object ID", "property", "value")
```

ObjectID = Generated by QEngine.
Property = Required Image Property name to be checked.

Value = Required Image Property value to be checked.

With Dynamic Property: If you need to overwrite (or dynamically change) the identification property for the HTML Element during playback then you can use the five arguments method as follows:

`imageCheckInfo("Object ID", "property", "value", "<property_name>", "<property_value>")`
 where property name is the identification property such as "id" or "src" and property value is the value to be overwritten during playback.

Example:

```
imageCheckInfo("http://www.adventnet.com/images/free-edition-
appmanager.gif", "source", "http://www.adventnet.com/images/free-edition-
appmanager.gif")
```

Return Values:

0 for Success
 1 for Failure

buttonCheckInfo

Function Description:

To compare the value of the specified property of a button object with the given value. The object can be a push button, radio button, checkbox, submit and reset

How to Define:

```
buttonCheckInfo("Object ID", "property", "value")
ObjectID = Generated by QEngine.
Property = Required Property name to be checked.
Value = Required Property value to be checked.
```

With Dynamic Property: If you need to overwrite (or dynamically change) the identification property for the HTML Element during playback then you can use the five arguments method as follows:

`buttonCheckInfo("Object ID", "property", "value", "<property_name>", "<property_value>")`
 where property name is the identification property such as "id" or "name" and property value is the value to be overwritten during playback.

Example:

```
buttonCheckInfo("language_3", "status", "true")

0 for Success 1 for Failure
```

selectCheckInfo Function Description:

To compare the value of the specified property of a single selection and multi selection list box (combo box) object with the given value.

How to Define:

```
selectCheckInfo("Object ID", "property", "value")
```

ObjectID = Generated by QEngine.
Property = Required Property name to be checked.
Value = Required Property value to be checked.

With Dynamic Property: If you need to overwrite (or dynamically change) the identification property for the HTML Element during playback then you can use the five arguments method as follows:

```
selectCheckInfo("Object ID", "property", "value", "<property_name>", "<property_value>")
```

where property name is the identification property such as "id" or "name" and property value is the value to be overwritten during playback.

Example:

```
selectCheckInfo("department_1", "selecteditems", "Sales")
```

Return Values:

0 for Success
1 for Failure

staticCheckInfo**Function Description:**

To compare the properties of the popups message with the specified value.

How to Define:

```
staticCheckInfo("Object ID", "property", "value")
```

ObjectID = Generated by QEngine.
Property = Required Property name to be checked.
Value = Required Property value to be checked.

With Dynamic Property: If you need to overwrite (or dynamically change) the identification property for the HTML Element during playback then you can use the five arguments method as follows:

```
staticCheckInfo("Object ID", "property", "value", "<property_name>", "<property_value>")
```

where property name is the identification property (i.e., classname) and property value is the value to be overwritten during playback.

Example:

```
staticCheckInfo("Static", "value", "Choose the communication type")
```

0 for Success
1 for Failure

pageCheckInfo Function Description:

To compare the value of the specified property of a page object with the given value.

How to Define:

pageCheckInfo("Object ID", "property", "value")
 ObjectID = Generated by QEngine.
 Property = Required Property name to be checked.
 Value = Required Property value to be checked.

Example:

pageCheckInfo("AdventNet Payroll System Package - Form to Add Employee
 Details", "windowtitle",
 "AdventNet Payroll System Package - Form to Add Employee Details")

Return Values:

0 for Success
 1 for Failure

webCheckText Function Description:

To check if a particular text exists on the web page under test. **How to Define:**

webCheckText("search text", "prefix", "suffix") **Example:**

webCheckText("Product", "The", "name") **Return Values:**

0 for Success
 1 for Failure

Function Description:

To validate the style properties of a html element. The style properties that can be validated are cursor, color, background-color, font, font-family, font-size, font-style, text-decoration, width, height, background-image and text-align.

How to Define:

styleCheckInfo("ElementID", "Property", "value")
 ElementID = Generated by QEngine.
 Property = Required Property name to be checked.
 Value = Required Property value to be checked.

With Dynamic Property: If you need to overwrite (or dynamically change) the identification property for the HTML Element during playback then you can use the five arguments method as follows:

styleCheckInfo("Object ID", "property", "value", "<property_name>", "<property_value>")
 where property name is the identification property such as "id", "name" or "src" and property value is the value to be overwritten during playback.

Example:

```
styleCheckInfo("Test Plans","font-style","normal")
```

Return Values:

0 for Success

1 for Failure

listCheckInfo

Function Description:

To compare the value of the specified property of an element or <DIV> element with the given value.

How to Define:

```
listCheckInfo("Object ID","property","value")
```

ObjectID = Generated by QEngine.

Property = Required Property name to be checked.

Value = Required Property value to be checked.

With Dynamic Property: If you need to overwrite (or dynamically change) the identification property for the HTML Element during playback then you can use the five arguments method as follows:

```
listCheckInfo("Object ID", "property", "value", "<property_name>", "<property_value>")
```

where property name is the identification property such as "id" or "name" and property value is the value to be overwritten during playback.

Example:

```
listCheckInfo("li1", "name", "listelement1")
```

0 for Success 1 for Failure

HTML Table Functions

- getAllHTMLTableCells
- getHTMLTableColumnValues
- getHTMLTableRowCount
- getHTMLTableColumnCount
- getCellElementProperty
- fireEventOnCellElement
- doesColumnContain
- getCellValueAt()

getAllHTMLTableCells Function

Description:

To retrieve all the table cell values for the selected table.

How to Define:

```
getAllHTMLTableCells("Element ID")
```

Element ID = Generated by QEngine **Example:**

```
result=getAllHTMLTableCells("element ")
```

You can verify the output of the function by iterating through the returned object.

```
E.g
for i in range(0,len(result)):
for j in range(0,len(result[i])):
displayMessage(str(result[i][j]))
```

Return Values:

Two dimensional array object containing the values of all the table cells

getHTMLTableColumnValues Function Description:

To retrieve the table column values for the given column index

How to Define:

```
getHTMLTableColumnValues("Element ID","Column Index")
```

Element ID = Generated by QEngine.

Column Index = Index of the column to be retrieved.

Example:

```
E.g.
result = getHTMLTableColumnValues("S.No_1",2)
for i in range(0,len(result)):
displayMessage(st(result[i]))
```

getHTMLTableRowCount Function Description:

To retrieve the number of the rows from the selected table.

How to Define:

result=getHTMLTableRowCount("Object ID") Object ID =Generated by QEngine.

Example:

result=getHTMLTableRowCount("Employee N") displayMessage(result)

Return Values:

Integer

getHTMLTableColumnCount Function Description:

To retrieve the number of the columns from the selected table

How to Define:

result=getHTMLTableColumnCount("Object ID") Object ID =Generated by QEngine.

Example:

result=getHTMLTableColumnCount("Employee N") displayMessage(result)

Return Values:

Integer

getCellElementProperty

Function Description:

To get the property value of a HTML Element inside a HTML Table Cell Element.

How to Define:

getCellElementProperty(tableid,rowindex, colindex, property needed, [tag], [propertyname], [propertyvalue], [index])

Mandatory Parameters:

1. tableid = Generated by QEngine.
2. rowindex = row no in the table.
3. colindex = column no in the table.
4. property needed = property for which the value need to be fetched.

Optional Parameters:

1. tag = tagname of the elements.
2. propertyname = property to be matched.
3. propertyvalue = property value to be matched.
4. index = occurrence of the elements.

Example:

getCellElementProperty("Employee N",1, 3, "value") Return Values:
String

fireEventOnCellElement

Function Description:

To invoke or fire the specified event for the given HTML Table Cell element.

How to Define:

fireEventOnCellElement(tableid,rowindex, colindex, actionname, [actionvalue], [tag], [propertyname], [propertyvalue], [index])

Mandatory Parameters :

1. tableid = Generated by QEngine.
2. rowindex = row no in the table.
3. colindex = column no in the table.
4. actionname = action or the event to be fired. (click, doubleclick, mouseover, select, settext, etc.)

Optional parameters :

1. actionvalue = Required only if actionname is settext or selected. For other actionnames, actionvalue is NONE.
2. tag = tagname of the element. Default NONE.
3. propertyname = property to be matched. Default NONE
4. propertyvalue = property value to be matched. Default NONE
5. index = occurrence of the element. Default value 0.

Example:

fireEventOnCellElement("TableID",2,3,"click","", "INPUT", "id", "buttonid",1)

The above statement in the script will click a input button whose id is "buttonid" which is present in the 2nd row, 3rd column of the table where "TableID" is the ID generated by QEngine.

-NA-

doesColumnContain Function Description:

Checks if the selected HTML Table cell value is found in the specified column of the resultset after query execution.

How to Define: **doesColumnContain("Element ID", "HTML Table Row Index", "HTML Table Column Index",**

"String sql query", "DB Column Index" or "DB Column Name") Example:

```
when ColumnName is selected
initDB("1.1")
doesColumnContain("S.No",3,4,"select name from emptable","name")
```

```
when ColumnIndex is selected
initDB("1.1")
doesColumnContain("S.No",10,4,"select name from emptable",1)
```

Return Values:

0 for Success
1 for Failure

getCellValueAt() Function Description:

To retrieve the value of the specified cell from the table.

How to Define: result=getCellValueAt("Object ID", row index, column index) **Mandatory**

Parameters :

1. tableid = Generated by QEngine.
2. rowindex = row no in the table.
3. colindex = column no in the table

Example:

```
result=getCellValueAt("Employee N",2,2)
displayMessage(result)
```

Return Values:

Value of the specified cell

HTML General Functions

- doesElementExists
- selectItemWithIndex
- fireEventOnElement

doesElementExists

Function Description:

To check whether the specified HTML Element with the given tagname and matching property name and value exists. Supports regular expressions.

How to Define:

doesElementExists(**tagName,propertyName,propertyValue,index,[regExpRequired]**)

Mandatory Parameters :

tagname = Tag name of the elements to be identified.

propertyname = Property to be matched.

propertyvalue = Property value to be matched,

index = Occurrence of the elements.

regExpRequired :

true - property value is considered as a regular expression.

false - property value will be matched for equal condition.

Example:

```
doesElementExists("input", "type", "text", 2, "false")
```

You can also use regular expressions for the 3rd parameter (PropertyValue).

regExpRequired accepts "true" or "false" to support regular expressions. To know the details of using the regular expressions, refer to the following url:

<http://java.sun.com/developer/technicalArticles/releases/1.4regex/>

Return Values:

0 for Success

1 for Failure

selectItemWithIndex Function Description:

To select the item in the dropdown list based on the specified index value. **How to Define:**

selectItemWithIndex(**"Element ID", "Element index", waittime**) **Example:**

```
result =selectItemWithIndex("department_1",5,1)) Return Values:
```

0 for Success

1 for Failure

To invoke the specified event for the given HTML element.

How to Define:

```
fireEventOnElement("tagName", "propertyName", "propertyValue", index, "actionName", "actionValue", "useRegExp")
```

tagName, propertyName, propertyValue, index and actionName are mandatory. For setText and selectedItem events, actionValue parameter is mandatory.

With Dynamic Property

```
fireEventOnElement("tagName", "propertyName", "propertyValue", index, "actionName",  
"actionValue", "useRegExp")
```

Here, **propertyValue** can be given as a dynamic property using regular expressions. **useRegExp** accepts "true" or "false" to support regular expressions. To know the details of using the regular expressions, refer to the following url:
<http://java.sun.com/developer/technicalArticles/releases/1.4regex/>

Example:

```
fireEventOnElement("INPUT", "type", "button", 2, "click", "", "false")
```

The above function will click the 2nd occurrence of the input button element in the HTML page.

Return Values:

0 for Success
1 for Failure

Database Related Functions

- getDBValues()
- getDBValueAt()
- getDBTableRowCount()
- writeToDB
- getDBTableColumnCount()

getDBValues() Function Description:

To retrieve the recordset from DB after executing the specified sql query.

How to Define: result=getDBValues("sqlquery")

Example:

To display all the values retrieved by the below query.

```
result=getDBValues("Select * from LoginTable")
for i in range (0,len (result)):
for j in range (0,len (result[i])):
displayMessage (result[i][j])
```

Return Values:

Two dimensional String array

getDBValueAt() Function Description:

To retrieve the value from the specified row and column from DB after executing the specified sql query.

How to Define:

result=getDBValueAt("sql query,rowIndex,columnIndex")

Example:

To display the value at the 1st row, 2nd column value of the query result.

```
result=getDBValueAt("Select * from LoginTable",1,2)

displayMessage(result)
```

Return Values:

String

getDBTableRowCount() Function Description:

To retrieve the Row count from DB after executing the specified sql query **How to Define:**

result=getDBTableRowCount("sql query") **Example:**

To display the row count of the query result.
result=getDBTableRowCount("Select * from LoginTable")
displayMessage(result)

Return Values:

String

writeToDB Function Description:

To execute the specified sql query and write the resultset to the database.

How to Define: result=writeToDB("sql") **Example:**

```
result=writeToDB("Insert into employee  
("emp_id","emp_name","dept")  
values("emp1","peter",payroll)")
```

Return Values:

0 for Success
1 for Failure

getDBTableColumnCount()

Function Description:

To retrieve the Column count from DB after executing the specified sql query.

How to Define:

```
result=getDBTableColumnCount("sql  
query")
```

Example:

```
result=getDBTableColumnCount("Select * from LoginTable")  
displayMessage(result)
```

Return Values:

String

File Operations

- compareFile
- getFileCount
- getFileSize
- getLinesInFile
- getStringCountInFile
- isFileInZip
- isStringInFile
- isStringInZip

compareFile Function Description:

Compares the given base file name with the given check file name.

How to Define: compareFile("baseFileName", "CheckFileName") **Return**

Value:

0 for Success
1 for Failure

getFileCount Function Description:

To get the file count in the given directory.

How to Define:

getFileCount("Directory_Path", "Include_Directory")

Directory_Path = The full path of the directory from which the file count has to be fetched.

Include_Directory = Accepts true or false. If true, counts even the directories in the path. Else, skips the directories.

Example:

```
getFileCount("c:\test\mytest\", "false")
```

Return Value:

Integer

To get the size of the given file.

How to Define: getFileSize("filename") **Return Value:**

Integer

getLinesInFile Function Description:

To get the number of lines in the given file.

How to Define: getLinesInFile("filename") **Return Value:**

Integer

getStringCountInFile

Function Description:

To get the number of occurrences of the specified search string in the given file name.

How to Define:

```
getStringCountInFile("fileName","search_string")
```

Return Value:
Integer

isFileInZip Function Description:

To check whether the given file name exists in the given zip file.

How to Define: isFileInZip("ZipFile","FileName")

Return Value:
0 for Success
1 for Failure

isStringInFile Function Description:

To check whether the given search string exists in the given file.

How to Define: isStringInFile("Search_String","FileName")

Return Value:
0 for Success
1 for Failure

isStringInZip Function Description:

To check whether the given search string exists in the given file name where the file exists in the given zip file.

How to Define:

```
isStringInZip("ZipFile","FileName","Search_String")
```

Return Value:

0 for Success
1 for Failure

General Functions

Script and TestCase Reporting Functions

- . • reportStatus
- . • reportTestCase

Chaining Scripts

- . • callScript
- . • callScriptUpto

Global Variables Related Functions

- . • setGlobal
- . • getGlobal

Popup Related Functions

- . • getPopupText
- . • getUnexpectedPopUpTitle
- . • closeUnexpectedPopUp

Locale Related Functions

- . • changeLocale
- . • getLocaleCountry
- . • getLocaleLanguage
- . • getLocalizedString

Invoking Other Scripts/Applications

- . • invokeApplication
- . • invokeApplicationInThread
- . • invokeApplicationWithArgs
- . • invokeApplicationInThreadWithArgs

Other General Functions

- . • saveLogs()
- . • displayMessage
- . • wait
- . • stop
- . • fireMouseOver
- . • getBrowserType
- . • getBrowserVersion
- . • getOSVersion
- . • getOSName
- . • writeToCSVFile
- . • getEncryptedValue
- . • getLastError
- . • saveScreenShot
- . • waitForPageDownload
- . • submitForm

Script and TestCase Reporting Functions

reportTestCase Function Description:

To display the test case status (passed or failed) in the test case report.

How to Define:

```
reportTestCase(caseid,result,remarks,severity)
caseid = ID to be displayed in teh reports.
result = 0-passed ; 1-failed
remarks = Remarks to be displayed in the reports.
severity = Severity of the case.
```

Example:

```
result=selectCheckInfo("department_1","selecteditems",
"Sales") if result == 0: reportTestStatus("tstStatus",0,
"SelectCheckInfo Passed") else:
reportTestStatus("tstStatus",1, "SelectCheckInfo
Failed","Critical")
```

Return Values:

-NA-

reportStatus

Function Description:

To display user defined status in the script reports.

Status Id is user defined; any value can be specified for identification. **result** can be 0 or 1 only. 0 for Success and 1 for Failure. **remarks** can be any textual comment to be displayed in the reports.

How to Define:

```
reportStatus("Status Id", result, remarks)
statusid = ID to be displayed in teh reports.
result = 0-passed ; 1-failed
remarks = Remarks to be displayed in the reports

result=getRowCount("Employee N")

if result == 5:
    reportStatus("rowcnt",0, "Row count is Correct") else:
    reportStatus("rowcnt",1, "Invalid Row count")
```

Return Values:

-NA-

Chaining Scripts

callScript Function Description:

Invoke another web script from the one currently being executed.

For more details refer to Web Functional Testing -> Web Script Development -> Creating Web Scripts -> Chaining the Scripts

How to Define:

`callScript("Web Script Name", "Web Script Path")`

Mandatory Parameters:

WebScriptName = Script name to be called.

WebScriptPath = Script path (relative to the web scripts directory).

Example: `callScript("test", "case1/test")` Return Values:

Whether script execution successfully completed or not

callScriptUpto Function Description:

Invoke another web script from the one currently being executed.

Execute up to the line number specified

For more details refer to Web Functional Testing -> Web Script Development -> Creating Web Scripts -> Chaining the Scripts

How to Define:

`callScriptUpto("Web Script Name", "Web Script Path", Line number)`

Example: `callScriptUpto("test", "case1/test", 75)` Return Values:

Whether script execution successfully completed or not.

Global Variable Functions

setGlobal Function Description:

To set the value for the global variable.

How to Define:

`setGlobal("<variable_name>", "<variable_value>")`

Example:

`setGlobal("qehome", "file:\\C:\test\WebTest\QEngineWebTest\examples\payrollsystem\in dex.html")`

Return Values:

-NA-

getGlobal Function Description:

To get the value for the global variable.

How to Define:

```
getGlobal("<variable_name>")
```

Example:

```
setGlobal("qehome","file:\\C:\\test\\WebTest\\QEngineWebTest\\examples\\payrollsystem\\index.html")
```

Return Values:

-NA-

getPopUpText Function Description:

To read the text message displayed in the popup.

How to Define:

```
result =getPopUpText("<window_name>",<wait_time>)) Example:
```

```
result =getPopUpText("AdventNet",6)
```

```
displayMessage(result)
```

Return Values:

String

getUnexpectedPopUpTitle**Function Description:**

During playback, if any popup that was not encountered during recording, appears then it is called unexpected popup and you can read the title of the popup using this function.

How to Define:

```
result =getUnexpectedPopUpTitle()
```

Example:

```
result =getUnExpectedPopUpTitle()
displayMessage(result)
```

Return Values:

String

closeUnExpectedPopUp

Function Description:

During playback, if any popup that was not encountered during recording, appears then it is called **unexpected popup** and you can close the same using this function. **How to Define:**

```
closeUnExpectedPopUp("Window Title",timeout)
```

Example: result =closeUnExpectedPopUp(result,2)

0 for Success 1 for Failure

Locale Related Functions

changeLocale Function Description:

To dynamically modify the localized language in the recorded script.

How to Define: changeLocale(language, country) **Example:**

```
changeLocale("ab(Abkhazian)","AF(Afghanistan)")
```

Return Values:

0 for Success
1 for Failure

getLocaleLanguage Function Description:

Returns the selected locale language.

How to Define:

```
getLocaleLanguage()
```

Return Value:

String indicating the selected language such as "ab(Abkhazian)"

getLocaleCountry Function Description:

Returns the selected locale country.

How to Define:

```
getLocaleCountry()
```

Return Value:

String indicating the selected country such as "AF(Afghanistan)"

getLocalizedString Function Description:

Returns the localized string for the given key. **How to Define:**

```
getLocalizedString("String").
```

 String is the property name in the map file.

Return Value:

String

Invoking Other Scripts/Applications

invokeApplication

Function Description:

Invoke any application (.exe, .bat etc) running in the system.

Current script execution will be suspended till the called Application is executed completely.

How to Define: invokeApplication("Command line") Example:

```
invokeApplication("C:\Mozilla\Mozilla.exe")
```

Return Values:

0 for Success

1 for Failure

invokeApplicationInThread Function

Description:

Invoke any application (.exe, .bat etc) running in the system.

Called Application will be started on new thread. Calling script's execution will continue

How to Define: invokeApplicationInThread("Command line") Example:

```
invokeApplicationInThread("C:\Mozilla\Mozilla.exe")
```

 Return Values:

0 for Success
1 for Failure

To invoke any application (.exe, .bat etc) running in the system with arguments to be passed (if any).

Current script execution will be suspended till the called Application is executed completely.

How to Define:

`invokeApplicationWithArgs("<fileName>","<arguments>")`

Return Values:

0 for Success
1 for Failure

invokeApplicationInThreadWithArgs

Function Description:

To invoke any application (.exe, .bat etc) running in the system with arguments to be passed (if any).

Called Application will be started on new thread. Calling script's execution will continue

How to Define:

`invokeApplicationInThreadWithArgs("<fileName>")`

Return Values:

0 for Success
1 for Failure

Other General Functions

saveLogs() Function Description:

Saves the test execution logs with the specified file name. Saves the test execution logs in <QEngine_ Home > / projects / <Suite Name> / weblogs /<Month>_<Date>_<Hour>_<Minutes> directory.

How to Define: `saveLogs("<LogFileName>")`

Example: `saveLogs("mymodulelogs")`

Return Values:

-NA-

displayMessage Function Description:

To display user defined message in the logs **How to Define:**

`displayMessage("Message to be displayed")`

Example:

```
displayMessage("Result = "+var)
```

Return Values:

-NA-

wait Function Description:

To pause the script execution for n seconds.

How to Define: wait(seconds)

Example: wait(5)

Return Values:

-NA-

saveScreenShot Function Description:

When any error is encountered during the execution of the webscript, the snapshot of the client screen can be taken using this function.

The screen shot will be saved in the file whose name is specified as the first argument. The file will be saved under **<QEngine Home/projects/<suite name>/webreports/screenshots** folder.

This will happen after the n seconds specified as waittime.

How to Define:

```
saveScreenShot("filename",waittime)
```

Example:

```
saveScreenShot("a",10) 0 for Success 1 for Failure
```

stop Function Description:

To stop the execution of the webscript. If this function is used in a child script which is getting called from a parent script using the callScript(), then stop() will stop the execution of both the child and the parent script.

How to Define:

```
stop()
```

Example:

```
stop()
```

Return Values:

0 for Success
1 for Failure

fireMouseOver Function Description:

To invoke the onMouseOver action on the HTML object over which the mouse pointer is placed.

How to Define:

`fireMouseOver("Object ID", timeout)`

With Dynamic Property

`fireMouseOver("Object ID", timeout, "Property", "value")`

Mandatory Parameters :

Object ID = Generated by QEngine.

Optional Parameters :

Property = Property name to be overwritten with dynamic value.

Value = Dynamic value of the property name specified.

Example:

`fireMouseOver("link1",2)`

With Dynamic Property

`fireMouseOver("link1",2,"innertext","link2")`

If the innertext of "link" has changed to "links", then you can use the same function by overwriting the value of innertext as above.

0 for Success 1 for Failure

getBrowserType Function Description:

To get the browser type (IE, Mozilla or FireFox) on which currently playing back.

How to Define:

No Parameters.

Example:

`getBrowserType()`

Return Values:

String - Browser type as string.

getBrowserVersion Function Description:

To get the browser version on which currently playing back.

How to Define:

No Parameters.

Example:

```
getBrowserVersion()
```

Return Values:

String - Browser version as string.

getOSVersion Function Description:

To get the OS version of the machine in which you are replaying the script.

How to Define:

No Parameters.

Example:

```
getOSVersion()
```

String - OS version as string.

getOSName Function Description:

To get the OS Name (Windows or Linux) of the machine in which you are replaying the script.

How to Define:

No Parameters.

Example:

```
getOSName()
```

Return Values:

String - OS name as string.

writeToCSVFile Function Description:

To write CSV values in the specified csv file.

How to Define:

```
writeToCSVFile("File Name","Data", "Append To File")
```

Here, "**File Name**" is the CSV file name (do not specify the path, since QEngine creates and stores the csv files in <QEngine_Home>/projects/<Suite_Name>/dataset directory), "**Data**" is a string array which contains the values to be written in the specified CSV file and "**Append To File**" is by default false which overwrites the CSV file, if already exists. If set to true, appends the contents in the existing csv file

Example:

```
writeToCSVFile("test.csv","result_array","true")
```

Return Values:

-NA-

getEncryptedValue**Function Description:**

To encrypt the input text. This will be useful in cases where you read the data from a CSV file and set it in the password field.

```
getEncryptedValue("text")
```

Example:

E.g. to set the encrypted password value retrieved from csv file in the loop.

```
initCSV ("C:\a.csv")
for i in range (0,4)
setwindow ("windowID",timeout)
username = getCSVValueAt(i,1)
password = getCSVValueAt(i,2)
encryptedPassword = getEncryptedValue (password)
setText ("usernamefieldID",username, timeout)
setText ("passwordfieldID", encryptedPassword, timeout)
clickButton ("buttonID",timeout)
setWindow ("windowID",timeout)
clickLink("logoutlinkID",timeout)
```

Return Values:

Encrypted text value as string.

getLastError Function Description:

To get the script error (if any) in the last script statement that was executed before calling the getLastError ().

How to Define:

```
getLastError()
```

Example:

```
result =getLastError()  
displayMessage(result)
```

Return Value:

String.

waitForPageDownload

Function Description:

This function waits until the document ready state is reached or until the value specified for **Maximum waittime for document ready** in Play Settings option is timed out.

How to Define:

```
waitForPageDownload() 0 for Success 1 for Failure
```

submitForm Function Description:

To submit a form for the given form name. **How to Define:**

```
submitForm(formName, waittime)
```

Example:

```
submitForm("payroll", 2)
```

Return Value:

0 for Success
1 for Failure
