



QEngine White Paper

Functional Testing of Dynamic Web Applications

AdventNet, Inc.
5200 Franklin Dr, Suite 115
Pleasanton, CA 94588, USA
Phone: +1-925-924-9500
Fax: +1-925-924-9600
info@adventnet.com

<http://www.adventnet.com>
<http://www.qengine.com>
qengine-support@adventnet.com

Copyright 2006 AdventNet Inc, All Rights Reserved.

An automated testing tool for functional testing, load/performance testing of web applications and web services testing.

Table of Contents

ABSTRACT	2
INTRODUCTION.....	3
Prerequisites	3
Audience	3
TROUBLE SPOTS IN AUTOMATION	4
Handling Applications with varying Response Speeds	4
Handling Application Window Title / Size / Content Changes.....	4
Handling Application Window Element Changes	5
Handling UnExpected Application Popup during playback	6
Handling Application Server Host and Port Changes	6
SOLUTIONS	7
Adjusting Playback Speed	7
Using Regular Expressions for Window Title	7
Configuring Element Property Changes in Map File	7
Configuring Element Type Changes using Learn Mode	8
Using Dynamic Functions of QEngine	9
Handling Exceptions in Playback.....	9
Handling Host Port Changes during Playback	9
Configuring Element Property Changes based on User Input	10
CONCLUSION	11

Abstract

Web Sites and Applications are becoming crucial to the success of businesses and hence required to be tested *thoroughly* and *frequently*. However these applications change and they change constantly and the challenge of test automation projects is in dealing with these changes.

Successful automation is possible only if the automation tool can handle the changes and allow reuse of the automation project. In this paper, we present an overview of how does QEngine allow you to reuse the test automation project even after the application has changed.

Introduction

The purpose of this paper is to outline the provisions of QEngine product that allows users to reuse the test scripts when the application has changes. Reuse of scripts offers tremendous productivity gains in terms of time and effort.

For further information about QEngine, visit the URL:
<http://www.adventnet.com/products/qengine/index.html>

You can also try out the online demo version of QEngine using the URL: <http://demo.qengine.com>

Prerequisites

- Working knowledge of the web application to be tested.
- Basic working knowledge of using AdventNet QEngine Web Functional Testing Tool.
- Understanding of testing concepts.

Audience

- Quality Assurance Engineers.
- Users of QEngine who need to automate their Web Application functionality testing.

Trouble Spots in Automation

Applications change and they change constantly and the challenge of test automation projects is dealing with these changes. Successful automation is possible only if the automation tool can handle all of the scenarios described below:

- Handling Application with varying response speeds based on the system in which it runs.
- Handling Application Window changes in terms of size, title content based on the user input.
- Handle Application Window Element changes based on user input or for a new release.
- Handling Application Windows and Elements that vary dynamically during playback.
- Handling Applications with dynamic Windows & Elements.
- Handling Unexpected popup windows at runtime for unattended test execution.
- Handling Application location changes - server host and port changes during play.

Handling Applications with varying Response Speeds

Application's response speed is dependent on the system in which it runs. The expected window might appear immediately on a faster system or might appear after a significant delay when running on slower systems.

QEngine playback can be tuned to wait for the window in either of the scenarios. QEngine waits for window to appear for 60 seconds by default. This can be increased or decreased at runtime to accommodate varying response speeds of the application.

To know about the configuration details, please refer to **Adjusting Playback Speed** in the **Solutions** sub topic given below.

Handling Application Window Title / Size / Content Changes

Application window title and size are other aspects of application that might undergo change for one version of application to another.

Applications window size changes will not affect QEngine playback as it does not depend on screen coordinates for locating Window elements.

Inserting additional elements or removing elements into or from application after recording will also not affect playback.

However, if the window title changes then regular expressions can be used to handle these changes for a successful playback.

To know about the configuration details, please refer to **Using Regular Expressions for Window Title** in the **Solutions** sub topic given below.

Handling Application Window Element Changes

A good automated testing solution should ensure that tests can run unattended regardless of the application behavior during test execution. During a play session, a set of property values of a Window Element which are used for identifying the element might change. If the identification property values of an element differ from the property values stored during recording session then the tests may fail. The change in window element may fall in one of the categories described below.

Category 1 : Element Property Changes after Recording

When the property values of objects in your application has changed to another value after recording, QEngine allows you to modify the corresponding test object property values in the GUI map (or Object repository) so that you can continue to use your existing test scripts.

To know about the configuration details, please refer to **Configure Element Property Changes in Map File** in the **Solutions** sub topic given below.

Category 2: Element Location Changes

The location of Window Elements usually will not change, but could change between one release of your application and another Application basically to improve the aesthetics of the screen. QEngine playback does not require any configuration as it uses element identification which is not dependent on the order of the elements in the screen.

Category 3: Element Type Changes

Another possibility of a web application change is element type itself changing from one to another such as link to a button / button to image etc , after recording. In this case, the element properties are required to be added to the GUI Map file using learn mode and the script action for this element should be inserted into the script from learn mode view.

To know about the configuration details, please refer to **Configuring Element Type Changes using Learn Mode** in the **Solutions** sub topic given below.

Category 4 : Element Property Changes based on User Input

When web application's element property value changes at runtime based on user inputs, QEngine allows you to override the recorded values at runtime.

Let us consider an example where auto generated ids are used for elements in the page, then each time it is played the element identification property needs to be overwritten. QEngine provides property overriding functions to identify the changed element.

To know about the configuration details, please refer to **Configuring Element Property Changes based on User Input** in the **Solutions** sub topic given below.

Handling Applications with Dynamic Windows & Element

Applications can sometimes be designed to have varying window elements which cannot be predicted at the time of playback. Let us take a case of web page with images rotating at a set interval, clickable, navigating to different urls opening in separate windows.

In the above case, we have two unknowns in playback, one is the image appearing at the time of play and the other is the window that will appear after click of that image.

QEngine has built-in functions to fire event on such dynamic elements & windows. You need not record all the image clicks and the window that comes after the click. You can record just one image and then handle all possibilities with QEngine's built-in functions.

To know about the configuration details, please refer to **Using Dynamic Functions of QEngine** in the **Solutions** sub topic given below.

Handling Unexpected Application Popup during playback

One of the hardest things when developing automation scripts is handling the case where an unexpected window, like an ALERT box, pops up. Let us take a case of an application that generates error message if user input data exceeds allowed boundary, and If, during the next test run your input data will be valid and this error message will not appear.

Exception Handling mechanism of QEngine allows you to stop / continue playback after taking screenshot or close the unexpected popup window and thus allows you to perform unattended test execution.

To know about the configuration details, please refer to **Handling Exceptions in Playback** in the **Solutions** sub topic given below.

Handling Application Server Host and Port Changes

Another aspect of the application that will change after recording or anytime during playback is the host in which the application runs and the port it uses. QEngine allows configuring of modified host and port without changing the scripts or GUI map file. This also allows tester to run the automation project against any number of servers by just a point and a click.

To know about the configuration details, please refer to **Handling Host Port Changes during Playback** in the **Solutions** sub topic given below.

Solutions

Adjusting Playback Speed

To synchronize QEngine playback with the speed with which the application window appears, click on the **Settings** tab in Web Functional Home Page UI. From the Settings page, increase or decrease the value for “Maximum wait time for document ready” field. This will affect the overall playback speed of QEngine as it applies for all the setWindow() calls in the script. To dynamically vary the speed for specific setWindow() calls in the script, use the setDownloadTimeLimit(**seconds**).

Using Regular Expressions for Window Title

If your window title or parent title is dynamically changing and assume only the starting or ending word of the title remains the same across the pages, others are differing. To playback the scripts that includes such pages, you can specify regular expressions for window or parent title in the GUI Map file. The steps to achieve it are as follows:

- From the tree view of Web Functional Home Page UI, select a script file and right-click.
- A Popup menu is displayed. Choose **Edit GUI Map** menu item from the popup or choose the **View Map File** option in the script editor toolbar. The **GUI Map File Configuration** screen displays all the window and element object properties of a web page in a tree-view. In the left pane, you can view the parent node being the window node and the child nodes being the objects belonging to that window node.
- Click on the required element node from the left pane. You can view the property name and values displayed in a table in the right-pane.
- In the right pane, from the **Condition** column for the required window title or parent title, choose the specific condition such as **equals**, **starts with** or **ends with** to use regular expressions. In the **Value** column, enter the starting or ending word of the title that will not change.
- This will playback the script searching for the specified starting or ending word in the window or parent title. This eliminates the need to re-record the script.

Configuring Element Property Changes in Map File

To configure the element property changes in the Map file, follow the steps given below:

- From the tree view of Web Functional Home Page UI, select a script file and right-click.
- A Popup menu is displayed. Choose **Edit GUI Map** menu item from the popup or choose the **View Map File** option in the script editor toolbar. The **GUI Map File Configuration** screen displays all the window and element object properties of a web page in a tree-view. In the left pane, you can view the parent node being the window node and the child nodes being the objects belonging to that window node.
- Click on the required element node from the left pane. You can view the property name and values displayed in a table in the right-pane.
- Edit the property values for the selected node and click the **Update** button. After editing all the required properties, finally click the **Save** button to save the changed values in the appropriate GUI map file.
- This will successfully playback the script. During playback, QEngine reads the changed object description (window and element properties) in the GUI Map file and then looks for an HTML object with the same properties in the web application being tested.

Configuring Element Type Changes using Learn Mode

To add the properties of the changed element type to the GUI Map file and to insert the relevant function or user action for the element type using learn mode, follow the steps given below:

1. Select the recorded script from the tree-view of Web Functional Home Page UI.
2. From QEngine Toolbar, click **Launch Browser** option. In the launched browser, type the URL of the html file which includes the changed element type.
3. From QEngine Toolbar, click the **Start Record** option and then click on the **Switch To Learn Mode** option. This will display the **Learn Mode Settings** screen.
4. From the **Learn Mode Settings** screen, choose the map file mode in which the learned object properties should be stored. The Map File mode includes:
 - **Local** - This map file is local to the script created. The learned object properties will be stored in **test-scr1.map** file in `<QEngine_Home>/projects/<Suite_Name>/webscripts/<Script_Name>` directory.
 - **Global** - This map file is global to the entire suite. The learned object properties will be stored in **global.map** file in `<QEngine_Home>/conf` directory.
 - **Shared** - This map file is a shared map file wherein the object properties can be shared by any script within the suite using the **Import** option. The learned object properties will be stored in the specified shared map file in `<QEngine_Home>/projects/<Suite_Name>/conf` directory and also in the local script map file in `<QEngine_Home>/projects/<Suite_Name>/webscripts/<Script_Name>` directory.
5. In the **Repository Name** field, enter the name of the file in which the learned object properties should be stored.
6. After choosing the map file mode, choose the learn mode options from the **Learn Options** pane. To learn a specific element type, choose the radio option **Click element to learn** and then click on the **Learn** button.
7. Click on the required element type whose properties has to be learned from the web page launched in the web browser. This will learn the selected element object properties.
8. Click **Stop Record** to stop the recording.
9. Select the script in the tree-view and right-click.
10. From the popup menu, click **Orchestrate**.
11. The left pane of the screen will display the Object IDs of the learned elements in a icon-based tree view.
12. Select the object ID of the changed element type.
13. The right pane of the screen will display the following:
 - Category and Functions to be inserted in the script.
 - From the **Category** combo, select the category of the function to be inserted in the script.
 - From the **Functions** combo, select the function to be inserted in the script.
 - The arguments to be passed to the selected function will be displayed as Property, Type and Value columns in the **Configure Argument Values** pane.
 - You can also parameterize the argument values by choosing the **Data Configuration** option placed next to each of the argument fields.
 - The **Functions Preview** screen placed below the **Configure Argument Values** pane will display the selected function with the arguments to be passed.
 - Below the **Functions Preview** screen is the **Script Editor** screen. In this screen, place the cursor in the required position and insert the selected function at the desired location. This will insert the function in the Script Editor screen.
 - Click on the **Save** image placed above the script editor to save the changes. This will save the script.

Now, you can successfully playback the script which includes the user actions for the changed element type.

Using Dynamic Functions of QEngine

To fire event on element whose properties change dynamically during playback. Assume a image is rotating whose src property changes, to fire event on such an element use the following function.

fireEventOnElement() - To fire the specified event for the given HTML element.

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

Example: fireEventOnElement("img", "src", "c:\test\image001.gif", 1, "click", "", "false")

The above function will click the gif image element in the HTML page.

Now to dynamically identify the window that appears after the click of gif image, use the following functions:

setDynamicWindow - To set browser windows that does not appear during recording but appears during playback. These windows will not have any map entries.

Syntax: 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:

setDynamicWindow("My_parent_window_title",1,"frame_window_title","My_frame_window_title",1)

This will dynamically identify the window with parent title as "My_parent_window_title" and frame window title as "My_frame_window_title" so that further actions on any element in this window can be achieved using fireEventOnElement.

Handling Exceptions in Playback

To automatically handle unexpected popups during playback, follow the steps given below:

1. Choose the **Settings** tab in the Web Functional Home Page UI.
2. In the Settings page, from the **Exception Handling** block, under **Popup Exceptions** choose any of the check box such as **Report & Continue Play**, **Stop Play**, **Capture Screen & Continue Play**, **Close & Continue Play** or **Call Script** to handle the exception during playback.
3. Click the **Apply** button to save the changes.

Handling Host Port Changes during Playback

To configure the server and port changes, follow the steps given below:

1. Click the Settings tab from Web Functional Home Page UI.
2. From the Settings page, select the Use Host-port Configuration check box and then click on the Host-Port Editor link. This will display the Host-Port Configuration UI.
3. In the Host-Port Configuration UI, click the **New** button placed next to the **File Name** combo. This will display a dialog that prompts to specify a property file name. Enter a file name in which the configured host and port details has to be stored and click the **OK** button. This will populate the newly created property file name in the combo.
4. Click the '+' sign placed before the **Host** label. This will add a new row in the table.
5. From the **Old Protocol** field, select the existing protocol (HTTP or HTTPS) used in the recorded URL.
6. From the **New Protocol** field, select the new protocol (HTTP or HTTPS) using which the scripts has to be replayed.
7. In the **Old Host** field, enter the existing host name in the recorded URL.

8. In the **New Host** field, enter the new host name to be fetched dynamically and replayed.
9. In the **Old Port** field, enter the existing port number in the recorded URL.
10. In the **New Port** field , enter the new port number to be fetched dynamically and replayed.
11. To activate the configured host-port details, choose the **Activate this Host-Port Configuration** check box. This will activate the currently selected host-port property file name.
12. Click the **Apply** button to save the changes. You can add more configurations by clicking the '+' sign placed before the **Host** label.

Configuring Element Property Changes based on User Input

Assume a online ticket booking system where a user fills in a form to book a ticket. Once a ticket is booked, assume a message is displayed in the HTML page as "New Ticket ID: T1001 has been successfully generated" and a dynamic link is created to view the booked ticket details. Now, to click on this dynamic link whose id and innertext properties include the Ticket ID, you need to fetch the TicketID from the message that is displayed using the `webGetText()` and pass the fetched TicketID value as the fourth argument in the `clickLink()` as shown below:

Example script which uses the `webGetText()` and `clickLink()` with four arguments for a link whose innertext and id properties are dynamically generated during playback:

```
useLocalMapFile()
```

```
launchApplication("about:blank")
```

```
changeURL("http://localhost:4444/booktickets/jsp/ticketbooking.jsp",1)
```

```
ticketIDVal =webGetText("Ticket ID: ", "has")
```

```
displayMessage(ticketIDVal)
```

```
clickLink("BookedTicketDetails",1,"innertext",ticketIDVal)
```

Here, the **webGetText()** will search and fetch the value (TicketID) placed inbetween the given prefix and suffix text and stores it in the **ticketIDVal** variable. This variable is then used as the fourth argument for the **clickLink()** to click on the dynamically generated link.

Similarly, there are a variety of other four argument functions available to handle specific HTML elements such as Table, Textfield, Button, Image, Select, etc. whose property value changes dynamically.

For details, refer to the help documentation in <http://www.adventnet.com/products/qengine/help.html>

Conclusion

Thank you. This concludes the paper on handling dynamic changes of application in your automation project with QEngine. We hope that you now have a better understanding of the features and capabilities of AdventNet QEngine. If you have further questions, please write to qengine-support@adventnet.com and we will happily answer any question you have.

If you would like to try AdventNet QEngine , we have a fully-functional 15-day trial available that you may download and experience for yourself how automated software testing with QEngine can improve the quality of your software.