WAAT – Web Analytics Automation Testing Framework

Anand Bagmar

abagmar@gmail.com

http://www.essenceoftesting.blogspot.com

CONTENTS

1. What is Web Analytics

Web Analytics is the measurement, collection, analysis and reporting of internet data for purposes of understanding and optimizing web usage. Web Analytics is not just a tool for measuring website traffic, but can also be used as a tool for business research and market research.

Web Analytics applications can also help companies measure the results of traditional print advertising campaigns. It helps one to estimate how the traffic to the web site changed after the launch of a new advertising campaign.

Web Analytics provides data on the number of visitors, page views, etc. to gauge the traffic popularity trends which helps doing the market research.

Official definition of Web Analytics from Wikipedia

2. What is WAAT?

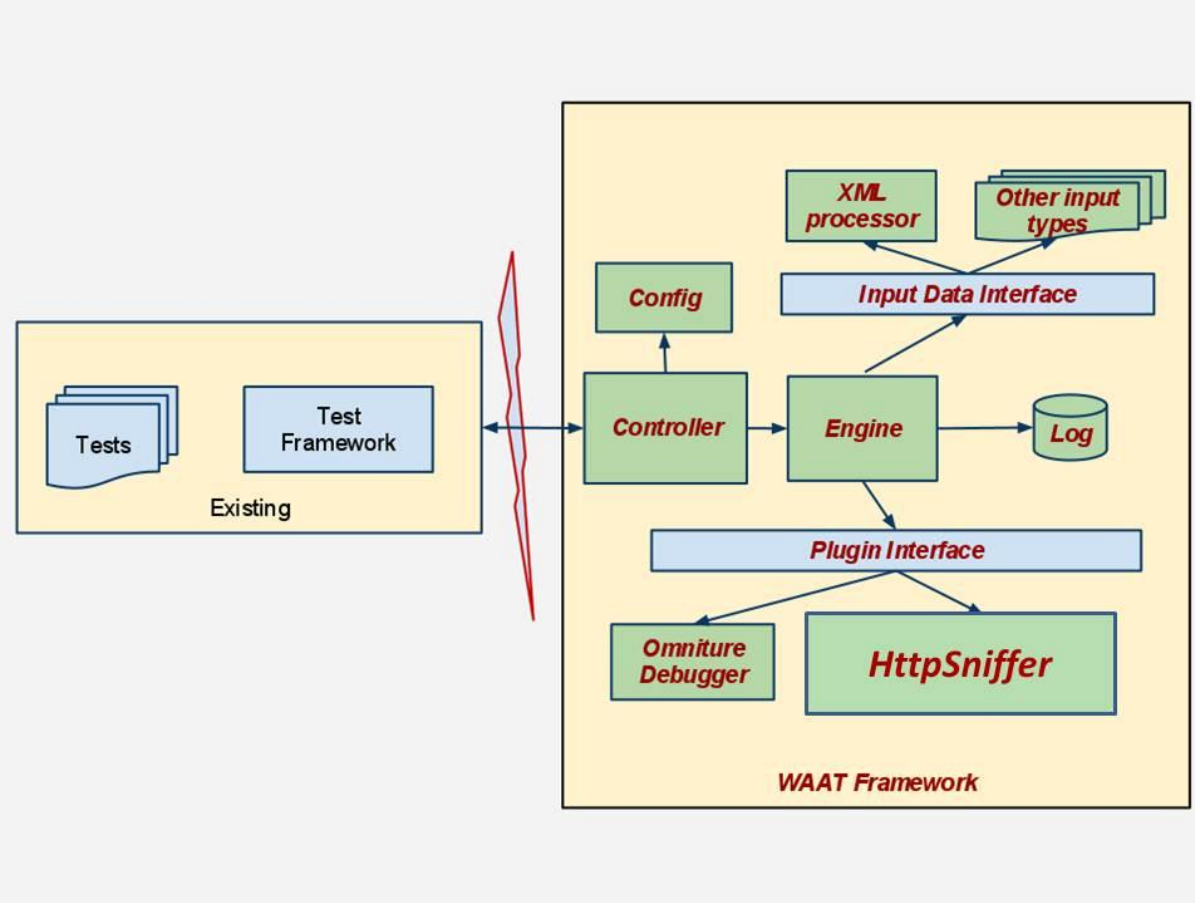
WAAT stands for Web Analytics Automation Testing.

3. Need for WAAT

- Manual testing for Web Analytics is tedious and time consuming (and also boring).
- It is possible to miss out on some of the tags for testing that will provide a feed to the Web Analytic system.

This framework provides a way to automate the verification of name-value pair properties / tags being reported to a Web Analytics System.

4. WAAT Architecture



5. WAAT Setup

- Download WAAT for the platform of your choice (Windows / Linux / Unix / MacOS) .
- Unzip the downloaded file in the lib folder of your existing project (say, c:\myProject\lib).
- Add c:\myProject\lib\WAAT\lib to your library classpath.

6. How to use WAAT?

WAAT can be used in 2 ways – Omniture Debugger, or, HttpSniffer.

6.1 WAAT using Omniture Debugger

6.1.1 Approach

The Omniture Debugger works by executing a javascript (provided by Omniture SiteCatalyst) in the same browser where your UI test is executing. This action opens a new window and shows the list of name/value pair properties reported to SiteCatalyst.

This approach makes used of the UI framework being used.

To use WAAT in this approach, you need to follow the steps mentioned below.

6.1.2 Create an implementation of the ScriptRunner interface

Your test framework could be using any of the available UI drivers (example: Selenium, WebDriver, Sahi, etc.).

To work with WAAT, you will first need to implement a ScriptRunner class. This class provides a hook for the WAAT framework to execute the javascript in the currently executing browser instance to open, and capture the data being sent to SiteCatalyst.

Selenium and WebDriver sample script runners are provided with the WAAT distribution. See the SeleniumScriptRunner.java and WebDriverScriptRunner.java files in folder - \samplescripts\com\thoughtworks\webanalyticsautomation\scriptrunner in the downloaded zip distribution for reference.

Copy the relevant ScriptRunner file in your test framework, update its namespace and your ScriptRunner class is ready for use.

6.1.3 Selenium ScriptRunner

- Copy the SeleniumScriptRunner.java from \samplescripts\com\thoughtworks\webanalyticsautomation\scriptrunner in the downloaded zip file to your test framework,
- Update its namespace
- Your ScriptRunner class is ready for use.

6.1.4 WebDriver ScriptRunner

- Copy the WebDriverScriptRunner.java from \samplescripts\com\thoughtworks\webanalyticsautomation\scriptrunner in the downloaded zip file to your test framework,
- Update its namespace
- Your ScriptRunner class is ready for use.

6.1.5 Other ScriptRunner

If you are using any other UI testing framework, you can create a class similar to the SeleniumScriptRunner.java or WebDriverScriptRunner.jar by implementing the com.thoughtworks.webanalyticsautomation.scriptrunner.ScriptRunner interface.

6.1.6 Define Test data

Provide the test data to be validated for each action in an xml file. Sample XML file is provided here.

6.1.7 Test code changes

You will need to make changes in your existing / new tests.

See the sections below for samples of Selenium and WebDriver-based changes.

6.1.7.1 Selenium-based test changes

- Import relevant packages
- Define & provide parameter values
- Initialize Engine
- Enable Web Analytics Testing
- Verify Web Analytics reporting
- Sample validations

- Disable Web Analytics reporting

```
package com.thoughtworks.webanalyticsautomation;

/**
 * Created by: Anand Bagmar
 * Email: abagmar@gmail.com
 * Date: Dec 29, 2010
 * Time: 9:34:02 AM
 */

/*
Import packages
*/
import com.thoughtworks.webanalyticsautomation.common.BROWSER;
import com.thoughtworks.webanalyticsautomation.plugins.WebAnalyticTool;
import com.thoughtworks.webanalyticsautomation.scriptrunner.SeleniumScriptRunner;
import static com.thoughtworks.webanalyticsautomation.Controller.getInstance;
import static com.thoughtworks.webanalyticsautomation.common.Utils.currentDirectory;
import static com.thoughtworks.webanalyticsautomation.common.Utils.fileSeparator;

import org.apache.log4j.Logger;
import com.thoughtworks.webanalyticsautomation.utils.SeleniumScriptRunnerHelper;

import org.testng.annotations.AfterMethod;
import org.testng.annotations.Test;
import static org.testng.Assert.assertEquals;
import static org.testng.Assert.assertNotNull;

import static com.thoughtworks.selenium.grid.tools.ThreadSafeSeleniumSessionStorage.session;
import com.thoughtworks.selenium.Selenium;

public class OmnitureDebuggerWithSeleniumTest {
   private SeleniumScriptRunnerHelper seleniumScriptRunnerHelper;
   private Logger logger = Logger.getLogger(getClass());

   /*
   Define & provide parameter values
   */
   private Engine engine;
   private WebAnalyticTool webAnalyticTool = WebAnalyticTool.OMNITURE_DEBUGGER;
   private InputFileType inputFileType = InputFileType.XML;
   private boolean keepLoadedFileInMemory = true;
   private String log4jPropertiesAbsoluteFilePath = currentDirectory() + fileSeparator() + "resources"  +
fileSeparator() + "log4j.properties";
   private String inputDataFileName = currentDirectory() + fileSeparator() + "test"  + fileSeparator() +
"sampledata"  + fileSeparator() + "TestData.xml";
   private String actionName = "OpenUpcomingPage_OmnitureDebugger_Selenium";

   @Test
   public void captureAndVerifyDataReportedToWebAnalytics_Selenium_IE() throws Exception {

      /*
      Initialize Engine
      */
     engine = Controller.getInstance (
                  webAnalyticTool,
                  inputFileType,
                  keepLoadedFileInMemory,
```

```
                    log4jPropertiesAbsoluteFilePath
                    );

        /*
        Enable Web Analytics Testing
        */
        engine.enableWebAnalyticsTesting();

        /*
        Existing test / test framework code
        */
        session().open(SeleniumScriptRunnerHelper.BASE_URL + "/upcoming");

        /*
        Verify Web Analytics Reporting
        */
        Result verificationResult = engine.verifyWebAnalyticsData (inputDataFileName, actionName, new
SeleniumScriptRunner(session()));

        /*
        Sample validations
        */
        assertNotNull(verificationResult.getVerificationStatus(), "Verification status should NOT be NULL");
        assertNotNull(verificationResult.getListOfErrors(), "Failure details should NOT be NULL");
        logVerificationErrors(verificationResult);
        assertEquals(verificationResult.getVerificationStatus(), Status.PASS, "Verification status should be PASS");
        assertEquals(verificationResult.getListOfErrors().size(), 0, "Failure details should be empty");
    }
}


    /*
    Disable Web Analytics Testing
    */
    /*
    *       If using TestNG, override the @AfterMethod annotation
    *       If using JUnit, override the @After annotation
    *       and add the following line in it:
    *                       engine.disableWebAnalyticsTesting();
    *       See the example shown below
    */
    @AfterMethod()
    public void tearDown() throws Exception {
        engine.disableWebAnalyticsTesting();
        seleniumScriptRunnerHelper.stopSeleniumDriver();
    }
}
```

6.1.7.2 WebDriver-based test changes

- Import relevant packages
- Define & provide parameter values
- Initialize Engine
- Enable Web Analytics Testing
- Verify Web Analytics reporting using WAAT
- Sample Validations
- Disable Web Analytics reporting

```
package com.thoughtworks.webanalyticsautomation;

/*
Import packages
*/
import com.thoughtworks.webanalyticsautomation.scriptrunner.WebDriverScriptRunner;
import com.thoughtworks.webanalyticsautomation.common.BROWSER;
import com.thoughtworks.webanalyticsautomation.inputdata.InputFileType;
import com.thoughtworks.webanalyticsautomation.plugins.WebAnalyticTool;
import static com.thoughtworks.webanalyticsautomation.Controller.getInstance;
import static com.thoughtworks.webanalyticsautomation.common.Utils.currentDirectory;
import static com.thoughtworks.webanalyticsautomation.common.Utils.fileSeparator;

import com.thoughtworks.webanalyticsautomation.scriptrunner.helper.WebDriverScriptRunnerHelper;
import org.openqa.selenium.WebDriver;

import org.testng.annotations.AfterMethod;
import org.testng.annotations.BeforeMethod;
import org.testng.annotations.Test;
import static org.testng.Assert.assertEquals;
import static org.testng.Assert.assertNotNull;

/**
 * Created by: Anand Bagmar
 * Email: abagmar@gmail.com
 * Date: Jan 4, 2011
 * Time: 10:36:28 AM
 */

public class OmnitureDebuggerWithWebDriverTest {
   private WebDriverScriptRunnerHelper webDriverScriptRunnerHelper;
   private WebDriver driverInstance;

   /*
   Define & provide parameter values
   */
   private Engine engine;
   private WebAnalyticTool webAnalyticTool = WebAnalyticTool.OMNITURE_DEBUGGER;
   private InputFileType inputFileType = InputFileType.XML;
   private boolean keepLoadedFileInMemory = true;
   private String log4jPropertiesAbsoluteFilePath = currentDirectory() + fileSeparator() + "resources"  +
fileSeparator() + "log4j.properties";
   private String inputDataFileName = currentDirectory() + fileSeparator() + "test"  + fileSeparator() +
"sampledata"  + fileSeparator() + "TestData.xml";
   private String actionName = "OpenUpcomingPage_OmnitureDebugger_WebDriver";

   @Test
   public void captureAndVerifyDataReportedToWebAnalytics_WebDriver_IE() throws Exception {
```

```
        /*
        Initialize Engine
        */
      engine = Controller.getInstance (
                        webAnalyticTool,
                        inputFileType,
                        keepLoadedFileInMemory,
                        log4jPropertiesAbsoluteFilePath
                        );

        /*
        Enable Web Analytics Testing
        */
      engine.enableWebAnalyticsTesting();

        /*
        *   Existing test / test framework code
        */
      driverInstance.get(WebDriverScriptRunnerHelper.BASE_URL + "/upcoming");

        /*
        Verify Web Analytics Reporting using WAAT
        */
      Result verificationResult = engine.verifyWebAnalyticsData (inputDataFileName, actionName, new
WebDriverScriptRunner(driverInstance));

        /*
        Sample validations
        */
      assertNotNull(verificationResult.getVerificationStatus(), "Verification status should NOT be NULL");
      assertNotNull(verificationResult.getListOfErrors(), "Failure details should NOT be NULL");
      logVerificationErrors(verificationResult);
      assertEquals(verificationResult.getVerificationStatus(), Status.PASS, "Verification status should be PASS");
      assertEquals(verificationResult.getListOfErrors().size(), 0, "Failure details should be empty");
   }

   /*
   Disable Web Analytics Testing
   */
   /*
   *        If using TestNG, override the @AfterMethod annotation
   *        If using JUnit, override the @After annotation
   *        and add the following line in it:
   *                        engine.disableWebAnalyticsTesting();
   *        See the example shown below
   */
   @AfterMethod()
   public void tearDown() throws Exception {
      engine.disableWebAnalyticsTesting();
      webDriverScriptRunnerHelper.stopDriver();
   }
}
```

6.2 WAAT using HttpSniffer

6.2.1 Approach

This is a generic approach to Web Analytics Test Automation. Regardless of the end-product doing Web Analytics reporting and analysis (example: Omniture, Google Analytics, etc.), you can use this approach to test, in an automated way, if correct name/value properties are being reported by the product / system under test.

This approach captures TCP packets on the network layer, from all the network devices available. Based on the criteria specified in your tests, only those relevant packets are filtered out, and used for analysis and verifications.

6.2.2 Setup 3rd-party libraries

In order to enable packet capturing, the WAAT framework needs helps of some 3-rd party open-source libraries. These libraries are available for various different operating systems in the download section of the WAAT project.

Refer to the Jpcap Setup section for platform specific install instructions.

6.2.3 Define Test data

Provide the test data to be validated for each action in an xml file. Sample XML file is provided here.

6.2.4 Test code changes

You will need to make the following changes in your existing / new tests.

6.2.4.1 Import relevant packages
6.2.4.2 Define & provide parameter values
6.2.4.3 Initialize Engine
6.2.4.4 Enable Web Analytics Testing
6.2.4.5 Verify Web Analytics reporting using WAAT
6.2.4.6 Sample validations
6.2.4.7 Disable Web Analytics reporting

```
package com.thoughtworks.webanalyticsautomation;

/*
Import packages
*/
import com.thoughtworks.webanalyticsautomation.common.BROWSER;
import com.thoughtworks.webanalyticsautomation.plugins.WebAnalyticTool;
import static com.thoughtworks.webanalyticsautomation.Controller.getInstance;
import static com.thoughtworks.webanalyticsautomation.common.Utils.currentDirectory;
import static com.thoughtworks.webanalyticsautomation.common.Utils.fileSeparator;

import com.thoughtworks.selenium.Selenium;
import org.testng.annotations.AfterMethod;
import org.testng.annotations.BeforeMethod;
import org.testng.annotations.Test;
import static org.testng.Assert.assertEquals;
import static org.testng.Assert.assertNotNull;

/**
 * Created by: Anand Bagmar
```

```
 * Email: abagmar@gmail.com
 * Date: Jan 31, 2011
 * Time: 12:14:04 PM
 */

public class HttpSnifferTest {
   private Selenium selenium;
     /*
     Define & provide parameter values
     */
   private Engine engine;
       private String baseURL = "http://essenceoftesting.blogspot.com";
       private String navigateToURL = baseURL + "/2010/12/waat-web-analytics-automation-testing.html";
       private String[] urlPatterns = new String[] {"GET /ps/ifr?container=friendconnect&mid=0"};
       private int minimumNumberOfPackets = 1;

       private String actionName = " OpenWAATArticleOnBlog_HttpSniffer";
       private WebAnalyticTool webAnalyticTool = WebAnalyticTool.HTTP_SNIFFER;
       private IputFileType inputFileType = InputFileType.XML;
       private boolean keepLoadedFileInMemory = true;
       private String log4jPropertiesAbsoluteFilePath = currentDirectory() + fileSeparator() + "resources"  +
fileSeparator() + "log4j.properties";
       private String inputDataFileName = currentDirectory() + fileSeparator() + "test"  + fileSeparator() +
"sampledata"  + fileSeparator() + "TestData.xml";

   @Test
   public void captureAndVerifyDataReportedToWebAnalytics_HTTPSniffer_GoogleAnalytics_Selenium_IE()
throws Exception {

       /*
       Initialize Engine
       */
       engine = getInstance(webAnalyticTool, inputFileType, keepLoadedFileInMemory,
log4jPropertiesAbsoluteFilePath);

       /*
       Enable Web Analytics Testing
       */
       engine.enableWebAnalyticsTesting();

       /*
       *   Existing test / test framework code
       */
       selenium.open(navigateToURL);

       /*
       Verify Web Analytics Reporting using WAAT
       */
       Result verificationResult = engine.verifyWebAnalyticsData (inputDataFileName, actionName, urlPatterns,
minimumNumberOfPackets);

       /*
       Sample validations
       */
       assertNotNull(verificationResult.getVerificationStatus(), "Verification status should NOT be NULL");
       assertNotNull(verificationResult.getListOfErrors(), "Failure details should NOT be NULL");
       logVerificationErrors(verificationResult);
       assertEquals(verificationResult.getVerificationStatus(), Status.PASS, "Verification status should be PASS");
       assertEquals(verificationResult.getListOfErrors().size(), 0, "Failure details should be empty");
```

```
    }

    /*
    Disable Web Analytics Testing
    */
    /*
    *       If using TestNG, override the @AfterMethod annotation
    *       If using JUnit, override the @After annotation
    *       and add the following line in it:
    *                       engine.disableWebAnalyticsTesting();
    *       See the example shown below
    */
   @AfterMethod
   public void tearDown() throws Exception {
      engine.disableWebAnalyticsTesting();
      seleniumScriptRunnerHelper.stopDriver();
   }
}
```

7. Jpcap setup

7.1.1 What is Jpcap?

Jpcap © is a Java library for capturing and sending network packets.
For information specific to this library, please contact Keita Fujii, kfujii@uci.edu.

7.1.2 Custom installation

You can use the binaries packaged (and tested) with WAAT using the instructions provided below.

7.1.2.1 Windows

You can use one of the following approaches to setup Jpcap and its dependencies on Windows OS:

7.1.2.1.1 Copy dependent files

- Copy all dlls from \WAAT\lib\httpSniffer\Windows\System32 to C:\Windows\System32 directory.
- Copy npf.sys from \WAAT\lib\httpSniffer\Windows\System32\drivers to C:\Windows\System32\drivers directory.

7.1.2.1.2 Install dependent files

- Install WinPcap_4_1_2.exe from \WAAT\lib\httpSniffer\Windows directory.
- Install JpcapSetup-0.7.exe from \WAAT\lib\httpSniffer\Windows directory.

7.1.2.2 Fedora, RedHat

Install the Jpcap RPM package available in \WAAT\lib\httpSniffer\Linux\jpcap-0.7-1.i386.rpm

7.1.2.3 Ubuntu, GNU/Debian

Install the Jpcap RPM package available in \WAAT\lib\httpSniffer\Linux\jpcap-0.7.deb

7.1.3 Default installation

Install Jpcap and its dependencies directly based on the instructions provided on the Jpcap website or a local copy of that installation document available here.

8. FAQs

- **What languages are supported by WAAT?**
  WAAT is available as a Java framework. There are plans to make this available as a Ruby gem file and a .NET dll.

- **What UI Testing frameworks are supported by WAAT?**

  WAAT using Omniture Debugger approach:
  WAAT is not dependent on any particular UI testing framework.
  This has been tested with Selenium and WebDriver. However, if you are using any other framework, all you need to do is implement the ScriptRunner interface in your existing framework, and you will be good to go.

  WAAT using HttpSniffer approach:
  This is UI framework independent.

- **What Web Analytic systems are supported by WAAT?**
  WAAT has been tested with sites reporting to Omniture SiteCatalyst and Google Analytics. This framework can be used by any site sending name-value pair of properties / tags as parameters in the request URL for any Web Analytic tool.

- **What Browsers are supported by WAAT?**
  WAAT is Browser independent.

- **What Operating Systems are supported by WAAT?**
  It has been tested on Windows 7, Windows XP, Ubuntu Linux and Mac OS.