



## Web Robot User Guide

1	Introduction .....	3
2	Configuring Service Tester for Web Robots .....	3
2.1	Schedule a Web Robot with Service Tester .....	3
2.1.1	Test Parameters.....	3
2.1.2	Robot Parameters .....	4
2.1.3	Browser Parameters.....	5
2.1.4	User Credentials.....	5
2.2	Test the Web Robot using the Service Tester .....	6
2.2.1	Test result.....	6
2.2.2	Robot Transactions .....	6
2.2.3	Event Log.....	7
2.3	Debugging a robot.....	8
3	BrowserTest.....	8
3.1	Command Line Syntax.....	8
3.2	Command Line Options.....	8
3.3	Configuration File .....	9
3.4	Setting up Mozilla Firefox.....	10
3.5	Manual Test.....	10
3.6	Log Files.....	10
4	Test Case Files .....	11
4.1	OneView Test Case Commands.....	11
5	Test Case Example.....	12
5.1	Test Case Example in Selenium IDE .....	12
5.2	Test Case Example in HTML format.....	13
5.3	Manual Execution of Test Case Script .....	14
5.4	Log file output.....	15
6	Appendix 1 - OneView Selenium Commands .....	17
7	Appendix 2 – Supported Selenium Commands in OneView .....	18
7.1	List of Selenium “target” and “value” entries that BrowserTest supports .....	22

## 1 Introduction

OneView can be scheduled to run a predefined test scenario to make sure that for example an application is working. The test is set up using a script designed in Selenium IDE and run via the OneView Service Tester extender.

This document provides you with details on setting up a test in the Service Tester, an introduction to creating a test script in Selenium IDE, and finally how to test the script using the Service Tester.

A Web robot is a good tool of providing information on a workflow in an application, and verifying that an application is working during the night and weekends when you cannot extract the data from the log files.

Be aware that Web Robots can require some work to maintain – it is therefore important to keep them as simple as possible.

## 2 Configuring Service Tester for Web Robots

For a Service Tester to be able to schedule robots, a Web Robot directory must already exist. A Web Robot directory is an installation directory containing the executable BrowserTest.jar and corresponding configuration files.

BrowserTest must be installed and configured correctly before they can be used with Service Tester. This includes setting up a web browser and a browser profile. See chapters on setting up WebRobot and BrowserTest.

### 2.1 Schedule a Web Robot with Service Tester

Login to OneView as an administrator and go to the Extenders tab. Choose a Service Tester with a local Web Robot directory.

#### 2.1.1 Test Parameters

Click on the button “New Test” and choose “Web robot” from the list of service types. Type a descriptive name for your robot. The name will be part of the source name in Oneview.

<b>Version:</b> 4.0.1001.680 (64-bit)		<b>Build date:</b> 2015-05-13 15:11		<b>Started at:</b> 2015-05-13 15:29:54		<b>Current time:</b> 2015-05-21 09:20:40	
<b>Server:</b> http://127.0.0.1:1234		<b>ID:</b> 15		<b>Last Delivery:</b> 13 seconds ago		<b>Transactions Last Hour:</b> 33126	
<b>Avg. Wait Time:</b> 0 ms							
<input type="button" value="Stop"/>	<input type="button" value="Start"/>	<input type="button" value="Start Auto Refresh"/>	<b>Name:</b> <input type="text"/>		<b>Type:</b> <input type="text" value="All"/>		<input type="button" value="New Test"/>
<input type="button" value="All"/> <input type="button" value="Enabled"/> <input type="button" value="Disabled"/> <input type="button" value="Running"/> <input type="button" value="Failed"/>							
#	Actions		Name				

### New Test

Type	Web Robot - Executes a Web Robot ▼
Name	My new robot

Next

The first section on the next page should look like this:

### Test Parameters

Test Name ?	My new robot
Test Interval ?	300 seconds.
Robot Timeout ?	600 seconds.
Options	<input type="checkbox"/> Enabled: <input type="checkbox"/> Debug Info:

Save

**Test Name:** The name of the test (as entered on the first page). The name will be a part of the source name in Oneview.

**Test Interval:** This is how often the robot is scheduled to run in seconds.

**Robot Timeout:** If the robot runs for longer than this, the robot and all sub processes are terminated.

**Options:** Mark Enabled to enable scheduling of the robot.

## 2.1.2 Robot Parameters

The section on robot parameters contains information on the test cases in the OneView test case directory.

### Robot Parameters

Source Name ?	ServiceTester@CT116596 My new robot
Robot Directory ?	c:\test\install\BrowserTest
Test Cases ?	Add test case... ▼
Seed Test ?	False ▼
Iterations ?	1
Users ?	1
Test Case Step Delay ?	0 milliseconds.
Test Case Step Timeout ?	40 seconds.
Test Case Timeout ?	600 seconds.

**Source Name:** This value is read-only. It is the combination of ServiceTester, name of machine and the test

name.

**Robot Directory:** The path to the directory where BrowserTest.jar and the “testcases” subdirectory exist.

**Test Cases:** If a valid robot directory exists, you can choose any number of test cases from the list of test cases found in that directory.

**Seed Test:** If true, the first test case in the list is considered a seed test. The seed is executed before any of the other test cases in the test file.

**Iterations:** How many times the test cases in the test file is executed.

**Users:** If more than one user, the robot will be started in more instances running in parallel.

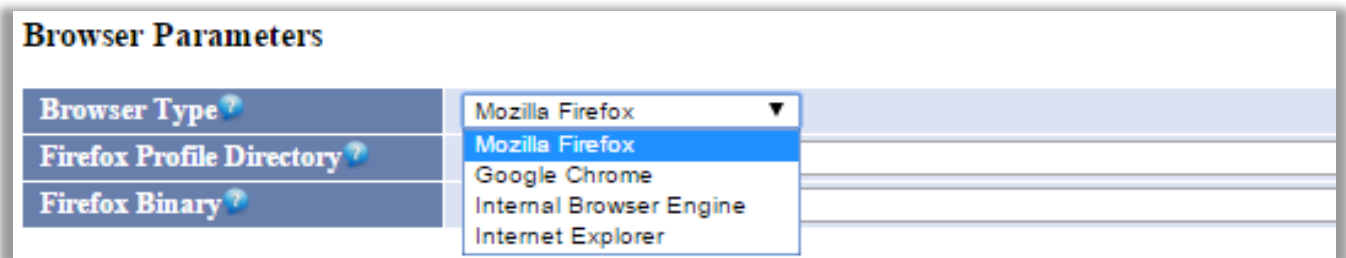
**Test Case Step Delay:** Introduces an artificial delay after each execution of a test case step.

**Test Case Step Timeout:** Enter maximum number of seconds to wait for a test case step to finish. On detection of a timeout, the current transaction and test case will fail.

**Test Case Timeout:** Enter maximum number of seconds to wait for a test case to finish. On detection of a timeout, the current test case will fail.

### 2.1.3 Browser Parameters

The section on browser parameters contains information on which browser engine the Service Tester should use.



Browser Parameters	
Browser Type ?	Mozilla Firefox ▼
Firefox Profile Directory ?	
Firefox Binary ?	

**Browser Type:** The robot may use different browser engines. Select the one that is suitable for your test.

**Profile Directory:** For some browsers you may choose which user profile to use. Please enter the path to a local profile directory here.

**Binary:** For some browsers you may choose which binary executable the robot must execute. Leave blank for default binary.

### 2.1.4 User Credentials

The section on user credentials contains information on the user credentials for the scheduled windows task.

### User Credentials

If you fill in a username and a password, the robot will be executed as scheduled windows task using the supplied user credentials.

Domain name ?	<input type="text"/>
User name ?	<input type="text"/>
Password ?	<input type="password"/>

**Domain name:** Enter the Windows domain name belonging to the user. If this is a local user, you must enter the name of the local machine.

**User name:** Enter the Windows user name of the user that should execute the robot.

**Password:** Enter the Windows password for the user that should execute the robot.

As the final step - Click 'Save'.

## 2.2 Test the Web Robot using the Service Tester

The Service Tester has a built-in test of web robots. The test button is located at the top of the test definition once the test has been saved.

### OneView Service Tester (Lenovo-BCH): Edit Test Web Robot

[Home](#)
[Test](#)
[Copy](#)

Task was saved!

Click 'Test' and the Service tester will start the robot script. Once the web robot is executing it will show both the test result and the event log in two separate tables – click 'refresh' to get the latest updates on transactions and steps performed in the web robot.

### 2.2.1 Test result

The test result contains information on the transactions coming from the web robot test. These transactions are the transaction that is sent to the OneView server as well. The result indicates whether the transaction is successful or failed and the value indicates the transaction time.

Timestamp	Source	Type	Name	Result	Value
2015-11-19 15:14:07	ServiceTester@JGO eksempel	Transaction	Test eksempel (A) - søg google	Success	4,364 ms
2015-11-19 15:14:10	ServiceTester@JGO eksempel	Transaction	Test eksempel (B) - Monsalta rundt	Success	2,488 ms
2015-11-19 15:14:10	ServiceTester@JGO eksempel	Transaction	Test eksempel Complete Test Case	Success	6,904 ms
2015-11-19 15:14:10	ServiceTester@JGO eksempel	Transaction	Test eksempel Test Case Execution	Success	6,904 ms

### 2.2.2 Robot Transactions

If the web robot is scheduled and running the results will be logged to Oneview as transactions. The 'Complete Test Case' and 'Test Case Execution' are OneView specific transactions whereas the other

transactions are defined in the test script in Selenium IDE.

The OneView specific transactions are defined as follows:

Transaction Name	Success	Response Time
<test case name> Complete Test Case	Depends on result of test case	Test Case Execution Time
<test case name> Test Case Execution	Yes	Test Case Execution Time

## 2.2.3 Event Log

The event log is a collection of all the steps taken when testing the web robot using the OneView Service Tester. Please note that the execution starts from the end of the table below and works its way up. The example is an extract of a full event log. Use the 'refresh' button at the top of the page to update the table when the test is running.

Timestamp	Component	Level	Text
2015-11-19 15:14:09.636	eksempel.html -	Info	Step 13: command=waitForElementPresent, target=link=Teknik
2015-11-19 15:14:09.630	eksempel.html -	Info	-----
2015-11-19 15:14:09.629	eksempel.html -	Info	Step 12: Status=CONTINUE
2015-11-19 15:14:09.046	eksempel.html -	Info	Step 12: command=click, target=id=Page_Layout_Elm2566965_Image
2015-11-19 15:14:09.045	eksempel.html -	Info	-----
2015-11-19 15:14:09.044	eksempel.html -	Info	Step 11: Status=CONTINUE
2015-11-19 15:14:08.897	eksempel.html -	Info	Step 11: command=waitForElementPresent, target=id=Page_Layout_Elm2566965_Image
2015-11-19 15:14:08.891	eksempel.html -	Info	-----
2015-11-19 15:14:08.890	eksempel.html -	Info	Step 10: Status=CONTINUE
2015-11-19 15:14:07.961	eksempel.html -	Info	Step 10: Loading http://monsalta.dk
2015-11-19 15:14:07.960	eksempel.html -	Info	Step 10: command=open, target=http://monsalta.dk
2015-11-19 15:14:07.959	eksempel.html -	Info	-----
2015-11-19 15:14:07.958	eksempel.html -	Info	Step 9: Status=CONTINUE
2015-11-19 15:14:07.955	eksempel.html -	Info	Step 9: Transaction "Test eksempel (B) - Monsalta rundt" start
2015-11-19 15:14:07.954	eksempel.html -	Info	Step 9: Label="(B) - Monsalta rundt"
2015-11-19 15:14:07.953	eksempel.html -	Info	-----
2015-11-19 15:14:07.952	eksempel.html -	Info	Step 8: Status=CONTINUE
2015-11-19 15:14:07.951	eksempel.html -	Info	Step 8: Transaction timer reset
2015-11-19 15:14:07.950	eksempel.html -	Info	Step 8: Transaction "(A) - søg google" logged as successful. Execution time was 4.364 milliseconds.
2015-11-19 15:14:07.913	eksempel.html -	Info	Step 8: Action=LogStamp, ResetTimer
*****			
2015-11-19 15:14:03.391	BrowserTest	Info	Firefox is started. Process Id = 13448
2015-11-19 15:14:03.273	BrowserTest	Info	Browser Info: firefox 42.0
2015-11-19 15:13:55.448	BrowserTest	Info	Starting Firefox using FirefoxBinary(C:\Program Files (x86)\Mozilla Firefox\firefox.exe)
2015-11-19 15:13:55.438	BrowserTest	Info	Setting firefox profile preference plugin.state.java = 2
2015-11-19 15:13:55.406	BrowserTest	Info	Loading Firefox profile = E:\BrowserTest\Firefox
2015-11-19 15:13:55.405	BrowserTest	Info	Firefox profile lock acquired.
*****			
2015-11-19 15:13:54.555	ServiceTester	Info	Creating task Monsalta\eksempel_Thread-1 for user jan_2
2015-11-19 15:13:54.554	ServiceTester	Info	Created temporary task xml file E:\OneViewServiceTester\temp\task163212653853930798.xml
2015-11-19 15:13:54.553	ServiceTester	Info	SUCCESS: The scheduled task "Monsalta\eksempel_Thread-1" was successfully deleted.
2015-11-19 15:13:54.496	ServiceTester	Info	Deleting task Monsalta\eksempel_Thread-1
2015-11-19 15:13:54.495	ServiceTester	Info	Executing command: "E:\jdk1.8.0_60\jre\bin\java.exe" -XX:-UsePerfData -jar BrowserTest.jar /control /proxy "http://127.0.0.1:1235/proxy" /instance "eksempel_Thread-1" /source "ServiceTester@JGO eksempel" /testcfg "E:\OneViewServiceTester\conf\webrobot_20151119_135427.test" eksempel.html

## 2.3 Debugging a robot

When the robot enabled and executing it will log its progress to local log files. Please see information about BrowserTest log files section [5.4](#)

## 3 BrowserTest

BrowserTest is another way to execute web robots. BrowserTest starts up a browser and executes a number of test case scripts. BrowserTest is designed to be started by the Service Tester in a scheduled task.

The script execution status and execution time are reported to the Service Tester, if the `/control` and `/proxy` parameters are set.

### 3.1 Command Line Syntax

```
java -jar BrowserTest.jar
    /instance <instance-name>
    /source <source-name>
    [/config <filename>]
    [/proxy <url>]
    [/control]
    [/seedtest]
    [/iterations <iterations>]
    [/testcfg <filename>]
    [/testfile <filename>]
    [testcase1] [testcase2] ... [testcaseN]
```

### 3.2 Command Line Options

#### **/instance <instance-name>**

A required parameter that uniquely identifies this running BrowserTest instance. Used in log files and task names and for controlling execution from ServiceTester.

#### **/source <source-name>**

A required parameter identifying the OneView source. BrowserTest passes the value unmodified to OneView. The source name should be descriptive, like a computer name or robot name or any combination thereof.

#### **/config <filename>**

An optional parameter specifying a BrowserTest configuration file. Default configuration is located in *conf/config.properties*

#### **/proxy <url>**

An optional parameter specifying a OneView proxy server where the test results should be logged. Used by ServiceTester.

#### **/control**

When this optional parameter is set, test case execution is controlled by an external process via values of Windows registry keys. This is used by OneView ServiceTester.



### **/seedtest**

When this optional parameter is set, the first test case in the test must be used as a seed test for the rest of the test cases. If iterations>1 the seed test case will only be executed once.

### **/iterations**

An optional parameter specifying how many times the sequence of use cases should be executed. Default number of iterations is 1.

### **/testcfg <filename>**

An optional parameter specifying a local ServiceTester .test definition file containing a full Web Robot definition.

### **/testfile <filename>**

An optional parameter specifying a local text file containing a list of test cases. The file must contain one test case filename per line.

## **3.3 Configuration File**

When BrowserTest starts, it reads its configuration from a properties file conf/config.properties

Parameter	Default value	Description
testcases.dir	testcases	Directory where test case files are located
logfile.days	7	Delete OneView log files after this number of days
logfile.dir	logs	Directory for OneView transaction log files
temp.dir	temp	Directory for temporary files
step.millis	0	Delay in milliseconds between each step in a test case
timeout.secs	40	Interrupt test case step after this number of seconds
case.timeout.secs	600	Interrupt test case after this number of seconds
driver.type	firefox	Selenium driver type (firefox, chrome, htmlunit)
firefox.profile.dir		Directory containing firefox profile
firefox.binary		Path to firefox binary executable
chrome.profile.dir		Directory containing chrome profile
chrome.binary		Path to chrome binary executable
api.key		Monsalta SMS gateway API-key
api.host	sms.oneview.eu	Monsalta SMS gateway API server
proxy.host		Proxy server hostname
proxy.port		Proxy server port number
proxy.user		Proxy server username
proxy.domain		Proxy server domain name
proxy.pass		Proxy server password

### 3.4 Setting up Mozilla Firefox

The most commonly used browser when running web robots is Mozilla Firefox. It is important that automatic updates are turned off in order to prevent the system from upgrading to versions not compatible with BrowserTest. The latest supported version is 46.0.1 and a portable version can be downloaded and installed from the Monsalta documentation website under 'Download'.

When installing the software perform a custom installation. You should install only the Mozilla Firefox browser software – the component 'Mozilla Maintenance Service' should not be installed. If this service is installed, please remove it by using the 'Add/Remove Programs' option in Windows.

Install Mozilla Firefox as you would normally do on the same machine as BrowserTest is installed.

Startup Firefox to create a Firefox user profile. Default location for user profile directories on Windows is `%APPDATA%\Mozilla\Firefox\Profiles`.

Locate your user profile directory and set value of the BrowserTest configuration parameter `firefox.profile.dir` to the directory path. Backslashes must be followed by an extra backslash. It is not necessary to specify the location of the Firefox binaries unless you want to run a specific instance/version of Firefox.

Example configuration for Mozilla Firefox:

`driver.type=firefox`

`firefox.profile.dir=C:\\Users\\monview\\AppData\\Roaming\\Mozilla\\Firefox\\Profiles\\xy755xj8.default`  
`logfile.days=3`

### 3.5 Manual Test

When creating or debugging test cases it is useful to manually execute a test case and view the output.

To test a single test case file, these are the minimum options required:

```
java -jar BrowserTest.jar /instance "test" /source "test" <filename>
```

To test a list of test cases, these are the minimum options required:

```
java -jar BrowserTest.jar /instance "test" /source "test" /testfile  
<filename>
```

### 3.6 Log Files

BrowserTest creates log files in the `logs` sub directory named like this

`BrowserTest_<instance>_<YYYY_MM_DD>.log.`

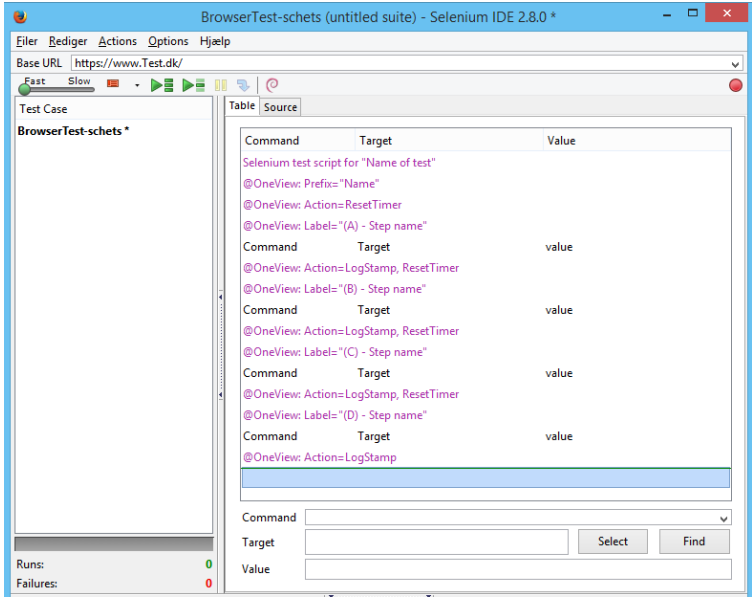
## 4 Test Case Files

Test case files are Selenium script files in HTML format. A test case is defined by a list of test case steps. Each step consists of a Selenium command, target and value. In HTML this becomes a table with a row of 3 cells for each step.

You can create test files using any text or HTML editor. However using the freely available Selenium IDE Firefox plugin is much easier. Selenium IDE enables you to record, edit and test a test case in real time.

```

<!--Selenium test script for "Name of test"-->
<!--@OneView: Prefix="Name" -->
<!--@OneView: Action=ResetTimer -->
<!--@OneView: Label="(A) - Step name"-->
<tr>
    <td>Command</td>
    <td>Target</td>
    <td>value</td>
</tr>
<!--@OneView: Action=LogStamp, ResetTimer -->
<!--@OneView: Label="(B) - Step name"-->
<tr>
    <td>Command</td>
    <td>Target</td>
    <td>value</td>
</tr>
<!--@OneView: Action=LogStamp, ResetTimer -->
<!--@OneView: Label="(C) - Step name"-->
<tr>
    <td>Command</td>
    <td>Target</td>
    <td>value</td>
</tr>
<!--@OneView: Action=LogStamp, ResetTimer -->
<!--@OneView: Label="(D) - Step name"-->
<tr>
    <td>Command</td>
    <td>Target</td>
    <td>value</td>
</tr>
<!--@OneView: Action=LogStamp -->
        
```



The screenshot shows the Selenium IDE 2.8.0 interface. On the left, there's a 'Test Case' pane with the title 'BrowserTest-schets \*'. On the right, there's a 'Table' pane showing a table with 3 columns: 'Command', 'Target', and 'Value'. The table contains 4 rows of test steps. Below the table, there are input fields for 'Command', 'Target', and 'Value', along with 'Select' and 'Find' buttons.

Command	Target	Value
Selenium test script for "Name of test"		
@OneView: Prefix="Name"		
@OneView: Action=ResetTimer		
@OneView: Label="(A) - Step name"		value
@OneView: Action=LogStamp, ResetTimer		
@OneView: Label="(B) - Step name"		value
@OneView: Action=LogStamp, ResetTimer		
@OneView: Label="(C) - Step name"		value
@OneView: Action=LogStamp, ResetTimer		
@OneView: Label="(D) - Step name"		value
@OneView: Action=LogStamp		

Test case files must be located in the test cases subdirectory when used by BrowserTest.

### 4.1 OneView Test Case Commands

You may add OneView commands to Selenium test case files that has special meaning when executing the test case from BrowserTest.

The OneView commands are implemented as XML/HTML-comments. This means that Selenium IDE is still able to execute the script even if it contains customized Oneview commands.

The syntax for imbedding an OneView command in HTML is as follows

```
<!--@OneView: Command1=Value1 Command2=Value2 ... -->
```

In Selenium IDE you should insert a comment like this

```
@OneView: Command1=Value1 Command2=Value2 ...
```

For the complete list of OneView Selenium commands and the allowed values please refer to [Appendix 1](#).

BrowserTest only supports a subset of Selenium commands – for the complete list please refer to [Appendix 2](#)

## 5 Test Case Example

The following test case opens Google search. Waits for the element containing the text “Monsalta ApS” to be loaded. Opens <http://monsalta.dk> and waits until the element “Samler” is loaded. It then clicks at the link “Samler” and waits for the link “Teknik”.

### 5.1 Test Case Example in Selenium IDE

The following screenshot is from Selenium IDE and shows you the above mentioned scenario shown in table form:

eksempel (untitled suite) - Selenium IDE 2.9.0

File Edit Actions Options Help

Base URL <http://www.monsalta.dk/>

Fast Slow

Test Case

eksempel

Command	Target	Value
Selenium test script for Test		
@OneView: Prefix=	"Test eksempel"	
@OneView: Action=	ResetTimer	
@OneView: Label=	"(A) - søg google"	
open	<a href="http://google.dk">http://google.dk</a>	
type	id=lst-ib	monsalta
waitForElementPres...	//div[@id='rhs_block']/ol/li/div/div/...	
verifyText	//div[@id='rhs_block']/ol/li/div/div/...	Monsalta ApS
@OneView: Action=	LogStamp, ResetTimer	
@OneView: Label=	"(B) - Monsalta rundt"	
open	<a href="http://monsalta.dk">http://monsalta.dk</a>	
waitForElementPres...	id=Page_Layout_Elm2566965_Image	
click	id=Page_Layout_Elm2566965_Image	
waitForElementPres...	link=Teknik	
verifyText	link=Teknik	Teknik
@OneView: Action=	LogStamp	

Command Selenium test script for Test

Target

Cancel Find

Select an element by clicking on it in the browser or click Cancel to cancel.

Value

Runs: 1

Failures: 0

Log	Reference	Expert	UI-Element	Rollup	Info	Clear
[info]	Executing:	open	<a href="http://monsalta.dk">http://monsalta.dk</a>			
[info]	Executing:	waitForElementPresent	id=Page_Layout_Elm2566965_Image			
[info]	Executing:	click	id=Page_Layout_Elm2566965_Image			
[info]	Executing:	waitForElementPresent	link=Teknik			
[info]	Executing:	verifyText	link=Teknik		Teknik	
[info]	Test case passed					

Please note that the commands in purple are the specific OneView commands providing you with information transaction names, prefix information and time reset to clear the timing between transactions.

## 5.2 Test Case Example in HTML format

The following shows you the Selenium script in HTML format:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head profile="http://selenium-ide.openqa.org/profiles/test-case">
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<link rel="selenium.base" href="http://www.monsalta.dk/" />
<title>START virksomhed</title>
</head>
<body>
<table cellpadding="1" cellspacing="1" border="1">
<thead>
<tr><td rowspan="1" colspan="3">START virksomhed</td></tr>
</thead><tbody>
<!--Selenium test script for Test-->
<!--@OneView: Prefix="Test eksempel" -->
<!--@OneView: Action=ResetTimer -->
<!--@OneView: Label="(A) - søg google"-->
<tr>
<td>open</td>
<td>http://google.dk</td>
<td></td>
</tr>
<tr>
<td>type</td>
<td>id=lst-ib</td>
<td>monsalta</td>
</tr>
<tr>
<td>waitForElementPresent</td>
<td>//div[@id='rhs_block']/ol/li/div/div/div/ol/div[2]/div/div</td>
<td></td>
</tr>
<tr>
<td>verifyText</td>
<td>//div[@id='rhs_block']/ol/li/div/div/div/ol/div[2]/div/div</td>
<td>Monsalta ApS</td>
</tr>
<!--@OneView: Action=LogStamp, ResetTimer-->
<!--@OneView: Label="(B) - Monsalta rundt"-->
<tr>
<td>open</td>
<td>http://monsalta.dk</td>
<td></td>
</tr>
<tr>
<td>waitForElementPresent</td>
<td>id=Page_Layout_Elm2566965_Image</td>
<td></td>
</tr>
<tr>
<td>click</td>
<td>id=Page_Layout_Elm2566965_Image</td>
<td></td>
</tr>
<tr>
<td>waitForElementPresent</td>
<td>link=Teknik</td>
<td></td>
</tr>
<tr>
<td>verifyText</td>
<td>link=Teknik</td>
<td>Teknik</td>
</tr>
<!--@OneView: Action=LogStamp-->
</tbody></table>
</body>
</html>
```

## 5.3 Manual Execution of Test Case Script

You can execute the test case script manually to verify that everything is working according to expectations. Go to the BrowserTest directory using a command prompt and issue the following command:

```
E:\BrowserTest>E:\jdk1.8.0_60\jre\bin\java -jar BrowserTest.jar /instance test /source  
test testcases\eksempel.html
```

Remember to locate your Java directory in the beginning of the manual execution before adding the path to the Selenium script.

You should see the following command output:

```
BrowserTest v4.0.1256.860 (64-bit, Selenium v2.45.0, built 2015-11-09 13:14).  
Java(TM) SE Runtime Environment (1.8.0_60-b27, 64-bit, E:\jdk1.8.0_60\jre)
```

Your browser window should open with Google search, and end with the Monsalta homepage:



## 5.4 Log file output

The output from the test can be found in the BrowserTest logfile in the BrowserTest directory. The format for the log file is BrowserTest\_*instance\_yyyy\_mm\_dd*.log.

The logfile looks like below – please note that this is only a subset of the content of the file:

```
23 nov. 2015 14:03:35 [test] INFO test - [test]
*****

23 nov. 2015 14:03:35 [test] INFO test - [test] BrowserTest v4.0.1256.860 (64-bit, Selenium v2.45.0, built 2015-11-09 13:14).
23 nov. 2015 14:03:35 [test] INFO test - [test] Java(TM) SE Runtime Environment (1.8.0_60-b27, 64-bit, E:\jdk1.8.0_60\jre)
23 nov. 2015 14:03:35 [test] INFO test - [test]
*****

23 nov. 2015 14:03:35 [test] INFO test - [test] Reading BrowserTest configuration file conf/config.properties
23 nov. 2015 14:03:35 [test] INFO test - [test] Loading test cases from directory testcases
23 nov. 2015 14:03:35 [test] INFO test - [test] Initializing XML parser
23 nov. 2015 14:03:35 [test] INFO test - [test] Loading test case E:\BrowserTest\testcases\eksempel.html
23 nov. 2015 14:03:35 [test] INFO test - [test] Using temporary directory E:\BrowserTest\temp\test
23 nov. 2015 14:03:35 [test] INFO test - [test] Cleaning directory E:\BrowserTest\temp\test
23 nov. 2015 14:03:35 [test] INFO test - [test] Using log directory E:\BrowserTest\logs
23 nov. 2015 14:03:35 [test] INFO test - [test] Cleaning up log files older than 3 days.
23 nov. 2015 14:03:35 [test] INFO test - [test] Initializing Process Util
23 nov. 2015 14:03:35 [test] INFO test - [test] BrowserTest process id = 12180
23 nov. 2015 14:03:35 [test] INFO test - [test] Initializing firefox driver
23 nov. 2015 14:03:35 [test] INFO test - [test] Acquiring Firefox profile lock.
23 nov. 2015 14:03:35 [test] INFO test - [test] Firefox profile lock acquired.
23 nov. 2015 14:03:35 [test] INFO test - [test] Loading Firefox profile = E:\BrowserTest\Firefox
23 nov. 2015 14:03:35 [test] INFO test - [test] Setting firefox profile preference plugin.state.java = 2
23 nov. 2015 14:03:35 [test] INFO test - [test] Starting Firefox using FirefoxBinary(C:\Program Files (x86)\Mozilla Firefox\firefox.exe)
23 nov. 2015 14:03:43 [test] INFO test - [test] Browser Info: firefox 42.0
23 nov. 2015 14:03:43 [test] INFO test - Looking for child process "firefox.exe" of parentPID=12180
23 nov. 2015 14:03:43 [test] INFO test - Child: name="firefox.exe" childPID=16368 parentPID=12180
23 nov. 2015 14:03:43 [test] INFO test - Found child process: name="firefox.exe", childPID=16368, parentPID=12180
23 nov. 2015 14:03:43 [test] INFO test - [test] Firefox is started. Process Id = 16368
23 nov. 2015 14:03:43 [test] INFO test - [test] Releasing Firefox profile lock.
23 nov. 2015 14:03:43 [test] INFO test - [test] Initializing Selenium backed driver
23 nov. 2015 14:03:43 [test] INFO test - [test] Iteration #1, Test case #1 eksempel.html
23 nov. 2015 14:03:43 [test] INFO test - [test] Maximizing browser window
23 nov. 2015 14:03:43 [test] INFO test - [test] eksempel.html - Step 0: -----
23 nov. 2015 14:03:43 [test] INFO test - [test] eksempel.html - Step 0: Setting test case timer to default 600 seconds.
23 nov. 2015 14:03:43 [test] INFO test - [test] eksempel.html - Step 0: -----
23 nov. 2015 14:03:43 [test] INFO test - [test] eksempel.html - Step 1: Prefix="Test eksempel"
23 nov. 2015 14:03:43 [test] INFO test - [test] eksempel.html - Step 1: Status=CONTINUE
23 nov. 2015 14:03:43 [test] INFO test - [test] eksempel.html - Step 0: -----
23 nov. 2015 14:03:43 [test] INFO test - [test] eksempel.html - Step 2: Action=ResetTimer
23 nov. 2015 14:03:43 [test] INFO test - [test] eksempel.html - Step 2: Transaction timer reset
23 nov. 2015 14:03:43 [test] INFO test - [test] eksempel.html - Step 2: Status=CONTINUE
23 nov. 2015 14:03:43 [test] INFO test - [test] eksempel.html - Step 0: -----
23 nov. 2015 14:03:43 [test] INFO test - [test] eksempel.html - Step 3: Label="(A) - søg google"
23 nov. 2015 14:03:43 [test] INFO test - [test] eksempel.html - Step 3: Transaction "Test eksempel (A) - søg google" start
23 nov. 2015 14:03:43 [test] INFO test - [test] eksempel.html - Step 3: Status=CONTINUE
23 nov. 2015 14:03:43 [test] INFO test - [test] eksempel.html - Step 0: -----
23 nov. 2015 14:03:43 [test] INFO test - [test] eksempel.html - Step 4: command=open, target=http://google.dk
23 nov. 2015 14:03:43 [test] INFO test - [test] eksempel.html - Step 4: Loading http://google.dk
23 nov. 2015 14:03:44 [test] INFO test - [test] eksempel.html - Step 4: Status=CONTINUE
```

\*\*\*\*\*

```

23 nov. 2015 14:03:49 [test] INFO test - [test] eksempel.html - Step 0: -----
23 nov. 2015 14:03:49 [test] INFO test - [test] eksempel.html - Step 13: command=waitForElementPresent, target=link=Teknik
23 nov. 2015 14:03:49 [test] INFO test - [test] eksempel.html - Step 13: Locator: By.linkText: Teknik
23 nov. 2015 14:03:49 [test] INFO test - [test] eksempel.html - Step 13: Status=CONTINUE
23 nov. 2015 14:03:49 [test] INFO test - [test] eksempel.html - Step 0: -----
23 nov. 2015 14:03:49 [test] INFO test - [test] eksempel.html - Step 14: command=verifyText, target=link=Teknik, value=Teknik
23 nov. 2015 14:03:49 [test] INFO test - [test] eksempel.html - Step 14: Locator: By.linkText: Teknik
23 nov. 2015 14:03:50 [test] INFO test - [test] eksempel.html - Step 14: Verification of element text Teknik was successful
23 nov. 2015 14:03:50 [test] INFO test - [test] eksempel.html - Step 14: Status=CONTINUE
23 nov. 2015 14:03:50 [test] INFO test - [test] eksempel.html - Step 0: -----
23 nov. 2015 14:03:50 [test] INFO test - [test] eksempel.html - Step 15: Action=LogStamp
23 nov. 2015 14:03:50 [test] INFO test - [test] eksempel.html - Step 15: Transaction "(B) - Monsalta rundt" logged as successful. Execution
time was 2.468 milliseconds.
23 nov. 2015 14:03:50 [test] INFO test - [test] eksempel.html - Step 15: Status=CONTINUE
23 nov. 2015 14:03:50 [test] INFO test - [test] eksempel.html - Step 0: -----
23 nov. 2015 14:03:50 [test] INFO test - [test] eksempel.html - Step 0: Test case logged as successful.
23 nov. 2015 14:03:50 [test] INFO test - [test] eksempel.html - Step 0: Test case execution time was 6874 milliseconds.
23 nov. 2015 14:03:50 [test] INFO test - [test] Killing all children of Firefox (pid=16368)
23 nov. 2015 14:03:50 [test] INFO test - Killing all children of parent process id 16368
23 nov. 2015 14:03:50 [test] INFO test - [test] Tearing down web driver.
23 nov. 2015 14:03:53 [test] INFO test - [test] Killing all children of BrowserTest (pid=12180)
23 nov. 2015 14:03:53 [test] INFO test - Killing all children of parent process id 12180
23 nov. 2015 14:03:53 [test] INFO test - [test] Cleaning directory E:\BrowserTest\temp\test
23 nov. 2015 14:03:53 [test] WARN test - [test] Could not delete E:\BrowserTest\temp\test\javasysmon67414616741211735227.dll: Unable to delete
file: temp\test\javasysmon67414616741211735227.dll
23 nov. 2015 14:03:53 [test] INFO test - [test] Exiting with exit code = 0

```



## 6 Appendix 1 - OneView Selenium Commands

Command	Syntax	Description
Prefix	Prefix=<string>	This string will be prefixed all transaction names in the test case before being sent to OneView.
CaseTimeoutSecs	CaseTimeoutsecs=<secs>	Number of seconds before test case execution is interrupted.
Action	Action=[LogStamp ResetTimer]+	LogStamp ends current transaction and logs status and execution time to Oneview.  ResetTimer resets timer for current transaction.
Label	Label=<string>	Sets the name of current transaction
ExecuteCommand	ExecuteCommand=<executable> (<parameter>=<value>)*  Possible variables: {FIREFOX_PID} – Firefox Process ID {CHROME_PID} – Chrome Process ID {CHILD_PID} – Child process ID from last waitForChildProcess {INSTANCE} – Instance name {USECASE} – Test case name	Executes a command in an OS shell
LockAcquire	LockAcquire=<Name of critical section>	Enter critical section synchronized across all robots
LockRelease	LockRelease=<Name of critical section>	Leave critical section synchronized across all robots
Include	Include=<filename>[;<parameter>=<value>]*	Includes another test case into this test case.  Included test case can accept parameters. All occurrences of parameter names in the included test case are replaced with the assigned values by simple string substitution.
WaitForProcess	WaitforProcess=<Name of process to wait for>	Wait for a process to start with this name
WaitForChildProcess	WaitForChildProcess=<Name of process to wait for>	Wait for a child process to start with this name.
SendMail	SendMail=<server> to=<to_email> [subject=<subject>] [body=<message>] [useGateway=true/false]	Sends an email using preconfigured email server properties in a file in conf directory called <server>.properties  If useGateway is true, the email is sent through Monsalta email gateway web service.
ReceiveSMS	ReceiveSMS=<regexp> [sender=<sender>] [receiver=<receiver>] [waitSecs=<secs>]	Waits for an SMS sent to Monsalta SMS gateway using the following filters: <regexp> is a required regular expression that must match the text contents of the SMS. Match groups are allowed. <receiver> must be a mobile number supported by Monsalta SMS gateway (optional) <sender> can be any mobile number with + prefix and country code. (optional) <secs> defines the maximum number of seconds to wait for the SMS. (optional) If an SMS is received that matches the above criteria, the following variables are set:  smsSender smsReceiver smsMessage smsMatchGroup1 .. smsMatchGroupN
SendSMS	SendSMS=<message> receiver=<receiver>	Sends an SMS through Monsalta SMS gateway.

## 7 Appendix 2 – Supported Selenium Commands in OneView

Command	Value	Description
assertAlert		<p>The message of the most recent JavaScript alert.</p> <p>Retrieves the message of a JavaScript alert generated during the previous action, or fail if there were no alerts.</p> <p>Getting an alert has the same effect as manually clicking OK. If an alert is generated but you do not consume it with getAlert, the next Selenium action will fail.</p> <p>Under Selenium, JavaScript alerts will NOT pop up a visible alert dialog.</p> <p>Selenium does NOT support JavaScript alerts that are generated in a page's onload() event handler. In this case a visible dialog WILL be generated and Selenium will hang until someone manually clicks OK.</p>
assertEval	<p>Arguments:</p> <ul style="list-style-type: none"> <li>script - the JavaScript snippet to run</li> </ul>	<p>Returns: The results of evaluating the snippet</p> <p>Gets the result of evaluating the specified JavaScript snippet. The snippet may have multiple lines, but only the result of the last line will be returned.</p> <p>Note that, by default, the snippet will run in the context of the "selenium" object itself, so this will refer to the Selenium object. Use window to refer to the window of your application, e.g. window.document.getElementById('foo')</p> <p>If you need to use a locator to refer to a single element in your application page, you can use this.browserbot.findElement("id=foo") where "id=foo" is your locator.</p>
assertText	<p>Arguments:</p> <ul style="list-style-type: none"> <li>locator - an element locator</li> </ul>	<p>Returns: The text of the element</p> <p>Gets the text of an element. This works for any element that contains text. This command uses either the textContent (Mozilla-like browsers) or the innerText (IE-like browsers) of the element, which is the rendered text shown to the user.</p>
answerOnNextPrompt	<p>Arguments:</p> <ul style="list-style-type: none"> <li>answer - the answer to give in response to the prompt pop-up</li> </ul>	<p>Instructs Selenium to return the specified answer string in response to the next JavaScript prompt [window.prompt()].</p>
clickAndWait	<p>Arguments:</p> <ul style="list-style-type: none"> <li>locator - an element locator</li> </ul>	<p>Clicks on a link, button, checkbox or radio button. If the click action causes a new page to load (like a link usually does), call waitForPageToLoad.</p>
click	<p>Arguments:</p> <ul style="list-style-type: none"> <li>locator - an element locator</li> </ul>	<p>Clicks on a link, button, checkbox or radio button. If the click action causes a new page to load (like a link usually does), call waitForPageToLoad.</p>
echo	<p>Arguments:</p> <ul style="list-style-type: none"> <li>message - the message to print</li> </ul>	<p>Prints the specified message into the third table cell in your Selenese tables. Useful for debugging.</p>
endWhile		
goBack		<p>Simulates the user clicking the "back" button on their browser.</p>

Command	Value	Description
goBackAndWait		Simulates the user clicking the "back" button on their browser.
gotof		
label		
open	Arguments: <ul style="list-style-type: none"> <li>url - the URL to open; may be relative or absolute</li> </ul>	Opens an URL in the test frame. This accepts both relative and absolute URLs. The "open" command waits for the page to load before proceeding, ie. the "AndWait" suffix is implicit. Note: The URL must be on the same domain as the runner HTML due to security restrictions in the browser (Same Origin Policy). If you need to open an URL on another domain, use the Selenium Server to start a new browser session on that domain.
pause	Arguments: <ul style="list-style-type: none"> <li>waitTime - the amount of time to sleep (in milliseconds)</li> </ul>	Wait for the specified amount of time (in milliseconds)
runScript	Arguments: <ul style="list-style-type: none"> <li>script - the JavaScript snippet to run</li> </ul>	Creates a new "script" tag in the body of the current test window, and adds the specified text into the body of the command. Scripts run in this way can often be debugged more easily than scripts executed using Selenium's "getEval" command. Beware that JS exceptions thrown in these script tags aren't managed by Selenium, so you should probably wrap your script in try/catch blocks if there is any chance that the script will throw an exception.
select	Arguments: <ul style="list-style-type: none"> <li>selectLocator - an element locator identifying a drop-down menu</li> <li>optionLocator - an option locator (a label by default)</li> </ul> Select an option from a drop-down using an option locator.	Option locators provide different ways of specifying options of an HTML Select element (e.g. for selecting a specific option, or for asserting that the selected option satisfies a specification). There are several forms of Select Option Locator.  label=labelPattern: matches options based on their labels, i.e. the visible text. (This is the default.) label=regex:^(^ Oo)ther value=valuePattern: matches options based on their values. value=other id=id: matches options based on their ids. id=option1 index=index: matches an option based on its index (offset from zero). index=2 If no option locator prefix is provided, the default behaviour is to match on label.
close		Simulates the user clicking the "close" button in the titlebar of a popup window or tab.
selectWindow	<ul style="list-style-type: none"> <li>Arguments: windowID - the JavaScript window ID of the window to select</li> </ul>	Selects a popup window using a window locator; once a popup window has been selected, all commands go to that window. To select the main window again, use null as the target.
selectFrame	Arguments: <ul style="list-style-type: none"> <li>locator - an element locator identifying a frame or iframe</li> </ul>	Selects a frame within the current window. (You may invoke this command multiple times to select nested frames.) To select the parent frame, use "relative=parent" as a locator; to select the top frame, use "relative=top". You can also select a frame by its 0-based index number; select the first frame with "index=0", or the third frame with "index=2".  You may also use a DOM expression to identify the frame you want directly, like this: dom=frames["main"].frames["subframe"]

Command	Value	Description
store storeEval storeExpression	Arguments: <ul style="list-style-type: none"> <li>expression - the value to store</li> <li>variableName - the name of a variable in which the result is to be stored.</li> </ul>	Evaluates an expression and stores the result in a variable.
storeAttribute	Arguments: <ul style="list-style-type: none"> <li>attributeLocator - an element locator followed by an @ sign and then the name of the attribute, e.g. "foo@bar"</li> </ul>	Returns: The value of the specified attribute. Gets the value of an element attribute. The value of the attribute may differ across browsers (this is the case for the "style" attribute, for example).
storeElementPresent	Arguments: <ul style="list-style-type: none"> <li>locator - an element locator</li> </ul>	Returns: true if the element is present, false otherwise. Verifies that the specified element is somewhere on the page.
storeLocation		Returns: The absolute URL of the current page. Gets the absolute URL of the current page.
storeText	Arguments: <ul style="list-style-type: none"> <li>locator - an element locator</li> </ul>	Returns: The text of the element. Gets the text of an element. This works for any element that contains text. This command uses either the textContent (Mozilla-like browsers) or the innerText (IE-like browsers) of the element, which is the rendered text shown to the user.
storeConfirmation		Returns: The message of the most recent JavaScript confirmation dialog.  Retrieves the message of a JavaScript confirmation dialog generated during the previous action. By default, the confirm function will return true, having the same effect as manually clicking OK. This can be changed by prior execution of the chooseCancelOnNextConfirmation command. If an confirmation is generated but you do not consume it with getConfirmation, the next Selenium action will fail.
sendKeys	Arguments: <ul style="list-style-type: none"> <li>locator - an element locator</li> <li>value - the value to type</li> </ul>	Simulates keystroke events on the specified element, as though you typed the value key-by-key.  This simulates a real user typing every character in the specified string; it is also bound by the limitations of a real user, like not being able to type into a invisible or read only elements. This is useful for dynamic UI widgets (like auto-completing combo boxes) that require explicit key events.  Unlike the simple "type" command, which forces the specified value into the page directly, this command will not replace the existing content. If you want to replace the existing contents, you need to use the simple "type"
type	Arguments: <ul style="list-style-type: none"> <li>locator - an element locator</li> <li>value - the value to type</li> </ul>	Sets the value of an input field, as though you typed it in.  Will replace the existing content  Can also be used to set the value of combo boxes, check boxes, etc. In these cases, value should be the value of the option selected, not the visible text.

Command	Value	Description
verifyEval	Arguments: <ul style="list-style-type: none"> <li>script - the JavaScript snippet to run</li> </ul>	<p>Returns: the results of evaluating the snippet</p> <p>Gets the result of evaluating the specified JavaScript snippet. The snippet may have multiple lines, but only the result of the last line will be returned.</p> <p>Note that, by default, the snippet will run in the context of the "selenium" object itself, so this will refer to the Selenium object. Use window to refer to the window of your application, e.g. window.document.getElementById('foo')</p> <p>If you need to use a locator to refer to a single element in your application page, you can use this.browserbot.findElement("id=foo") where "id=foo" is your locator.</p>
verifyElementPresent	Arguments: <ul style="list-style-type: none"> <li>locator - an element locator</li> </ul>	<p>Returns: true if the element is present, false otherwise.</p> <p>Verifies that the specified element is somewhere on the page.</p>
verifyElementNotPresent	Arguments: <ul style="list-style-type: none"> <li>locator - an element locator</li> </ul>	<p>Returns: true if the element is present, false otherwise</p> <p>Verifies that the specified element is somewhere on the page.</p>
verifyText	Arguments: <ul style="list-style-type: none"> <li>locator - an element locator</li> </ul>	<p>Returns: the text of the element.</p> <p>Gets the text of an element. This works for any element that contains text. This command uses either the textContent (Mozilla-like browsers) or the innerText (IE-like browsers) of the element, which is the rendered text shown to the user.</p>
verifyTextNotPresent	Arguments: <ul style="list-style-type: none"> <li>pattern - a pattern to match with the text of the page</li> </ul>	<p>Returns: true if the pattern matches the text, false otherwise.</p> <p>Verifies that the specified text pattern appears somewhere on the rendered page shown to the user.</p>
waitForElementPresent	Arguments: <ul style="list-style-type: none"> <li>locator - an element locator</li> </ul>	<p>Returns: true if the element is present, false otherwise.</p> <p>Verifies that the specified element is somewhere on the page.</p>
waitForElementNotPresent	Arguments: <ul style="list-style-type: none"> <li>locator - an element locator</li> </ul>	<p>Returns: true if the element is present, false otherwise.</p> <p>Verifies that the specified element is somewhere on the page.</p>
waitForText	Arguments: <ul style="list-style-type: none"> <li>locator - an element locator</li> </ul>	<p>Returns: the text of the element.</p> <p>Gets the text of an element. This works for any element that contains text. This command uses either the textContent (Mozilla-like browsers) or the innerText (IE-like browsers) of the element, which is the rendered text shown to the user.</p>
waitForFrameToLoad	Arguments: <ul style="list-style-type: none"> <li>frameAddress - FrameAddress from the server side</li> <li>timeout - a timeout in milliseconds, after which this command will return with an error</li> </ul>	<p>Waits for a new frame to load.</p> <p>Selenium constantly keeps track of new pages and frames loading, and sets a "newPageLoaded" flag when it first notices a page load.</p> <p>See waitForPageToLoad for more information.</p>

Command	Value	Description
waitForVisible	Arguments: <ul style="list-style-type: none"> <li>locator - an element locator</li> </ul>	Returns: true if the specified element is visible, false otherwise. Determines if the specified element is visible. An element can be rendered invisible by setting the CSS "visibility" property to "hidden", or the "display" property to "none", either for the element itself or one of its ancestors. This method will fail if the element is not present.
waitForPopUp	Arguments: <ul style="list-style-type: none"> <li>windowID - the JavaScript window "name" of the window that will appear (not the text of the title bar) If unspecified, or specified as "null", this command will wait for the first non-top window to appear (don't rely on this if you are working with multiple popups simultaneously).</li> <li>timeout - a timeout in milliseconds, after which the action will return with an error. If this value is not specified, the default Selenium timeout will be used. See the setTimeout() command.</li> </ul>	Waits for a popup window to appear and load up.
waitForPageToLoad	Arguments: <ul style="list-style-type: none"> <li>timeout - a timeout in milliseconds, after which this command will return with an error</li> </ul>	Waits for a new page to load. You can use this command instead of the "And-Wait" suffixes, "clickAndWait", "selectAndWait", "typeAndWait" etc. (which are only available in the JS API).
while		

## 7.1 List of Selenium “target” and “value” entries that BrowserTest supports

Target	Description	Value	Description
css		Regexp:	Regular expression
link		glob:	String matching pattern
id		exact:	TRUE if they are exactly the same 'case-sensitive'
xpath			
name			