Using Rapise® with Selenium®

Cross-Browser Testing Using Rapise & Selenium Together

Inflectra Corporation

Date: May 4th, 2017

Contents

Introduction

Rapise® is a next generation software test automation tool that leverages the power of open architecture to improve application quality and reduce time to market.

When developing and testing a web application you naturally need to test it with **different web browsers** and multiple version of each web browser. With Rapise natively you can record a test script using one browser and then play it back using **Mozilla Firefox, Google Chrome or Microsoft Internet Explorer**.

In addition, you can use Rapise with the open-source **Selenium WebDriver framework** to play back the same tests against other browsers such as **Apple Safari and Opera** (as well as IE, Firefox and Chome). You can also use Rapise to write native Selenium code for cases where you want to use existing Selenium WebDriver logic.

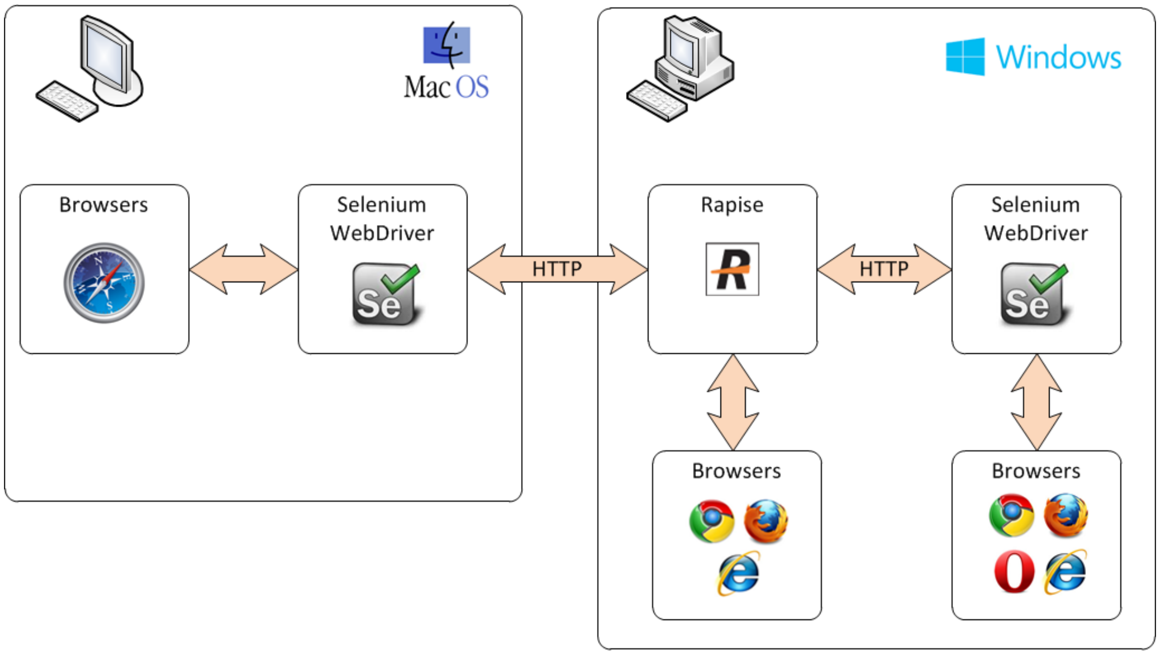
For information on using Rapise itself, please refer to the *Rapise User Guide*.

1. Setting up Selenium

This section describes the process for setting up Rapise to work with Selenium. Since Rapise is a Windows® application, you can use a single computer running Rapise to use the following web browsers:

- Internet Explorer
- Google Chrome
- Mozilla Firefox
- Opera
- Microsoft Edge

However because Safari only runs on Apple Mac computers, you will need to use two computers (a Mac running Safari) and a PC running Rapise to test using the Apple Safari web browser:

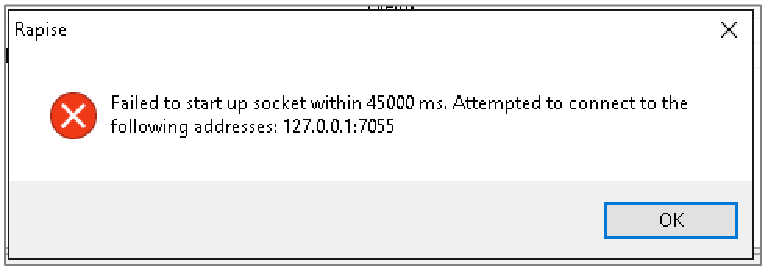


1.1. Configuring Selenium on a PC

Once you have installed Rapise on your local computer, you need to perform the following steps to configure each of the web browsers to use Selenium and Rapise:

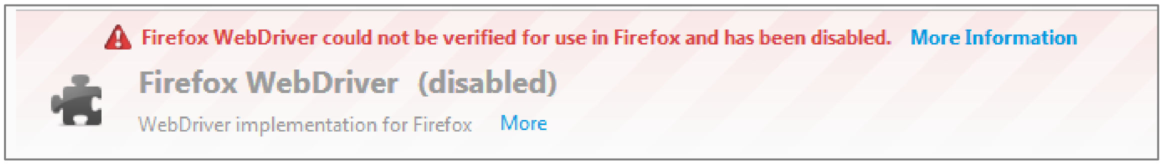
Firefox

Unlike the other web browsers, Firefox does not require anything special to be done, it already includes a built-in plugin for use by Selenium WebDriver. However if you start using Rapise with Firefox and you see the following issue when using Rapise with Firefox and Selenium:


Rapise

Failed to start up socket within 45000 ms. Attempted to connect to the following addresses: 127.0.0.1:7055

Then, look inside the instance of Firefox that was started by Selenium and you should see:

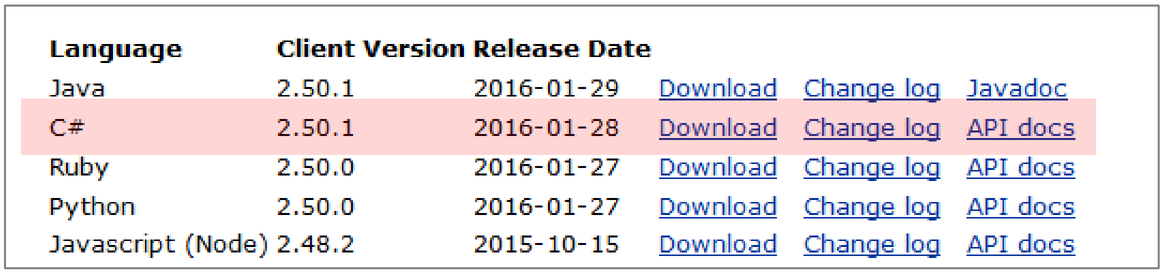


Firefox WebDriver could not be verified for use in Firefox and has been disabled. More Information

Firefox WebDriver (disabled)

WebDriver implementation for Firefox More

Then it means that the version of Selenium WebDriver that shipped with Rapise is no longer compatible with the installed version for Firefox. The solution is straightforward, just go to the main Selenium website: http://www.seleniumhq.org/download/ and then download the **C# WebDriver Bindings**:



Language	Client Version	Release Date			
Java	2.50.1	2016-01-29	Download	Change log	Javadoc
C#	2.50.1	2016-01-28	Download	Change log	API docs
Ruby	2.50.0	2016-01-27	Download	Change log	API docs
Python	2.50.0	2016-01-27	Download	Change log	API docs
Javascript (Node)	2.48.2	2015-10-15	Download	Change log	API docs

Download the **Selenium-dotnet-x.x.x.zip** file from the website. Proceed to unzip the archive and then look in the **net40** subfolder and extract the following two files and copy into the **C:\Program Files (x86)\Inflectra\Rapise\Bin** folder (or wherever you installed Rapise):

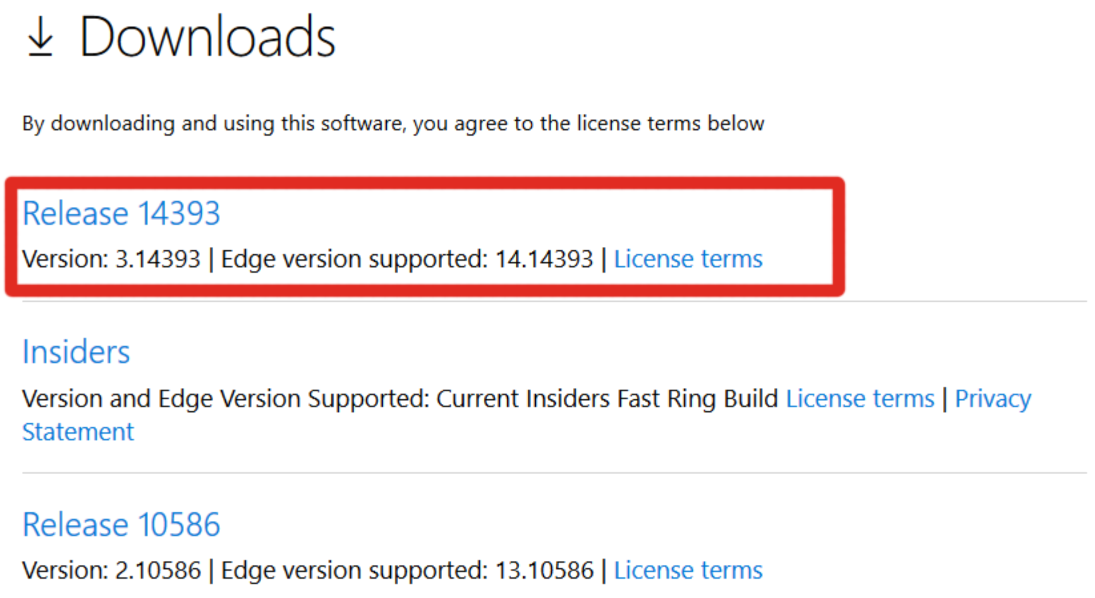
- WebDriver.dll
- WebDriver.Support.dll

Note: You will need to close Rapise before copying these files into the Bin folder.

Microsoft Edge

To use Selenium with Microsoft Edge, you will need to download the latest version of the Edge Driver from the Microsoft website:

https://developer.microsoft.com/en-us/microsoft-edge/tools/webdriver/#downloads



↓ Downloads

By downloading and using this software, you agree to the license terms below

Release 14393

Version: 3.14393 | Edge version supported: 14.14393 | License terms

Insiders

Version and Edge Version Supported: Current Insiders Fast Ring Build License terms | Privacy Statement

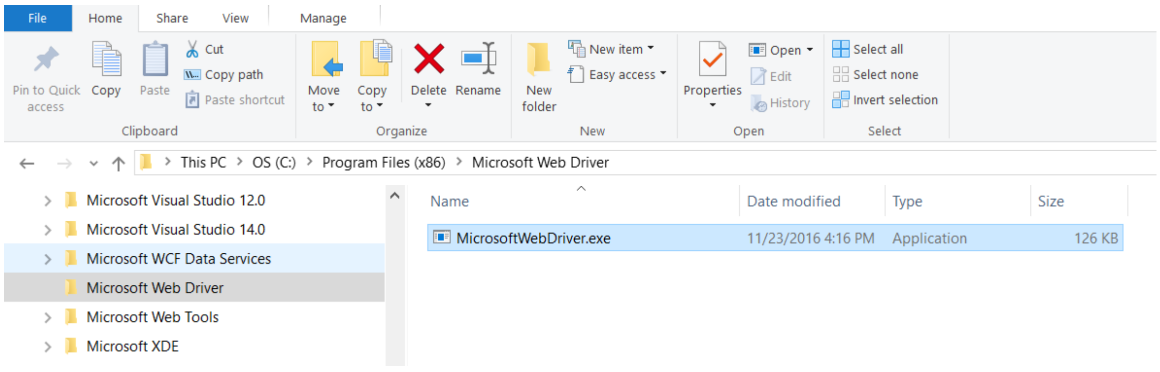
Release 10586

Version: 2.10586 | Edge version supported: 13.10586 | License terms

Create a new folder on your local PC called:

C:\Program Files (x86)\Microsoft Web Driver

Download the **MicrosoftWebDriver.exe** to this local folder you just created:



This PC › OS (C:) › Program Files (x86) › Microsoft Web Driver

- Microsoft Visual Studio 12.0
- Microsoft Visual Studio 14.0
- Microsoft WCF Data Services
- Microsoft Web Driver
- Microsoft Web Tools
- Microsoft XDE

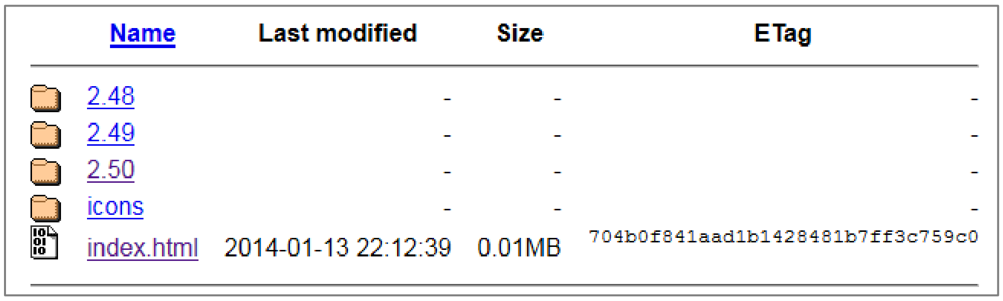
Name	Date modified	Type	Size
MicrosoftWebDriver.exe	11/23/2016 4:16 PM	Application	126 KB

Internet Explorer

To use Selenium with Internet Explorer, you will need to download the latest version of the Internet Explorer IE Driver:

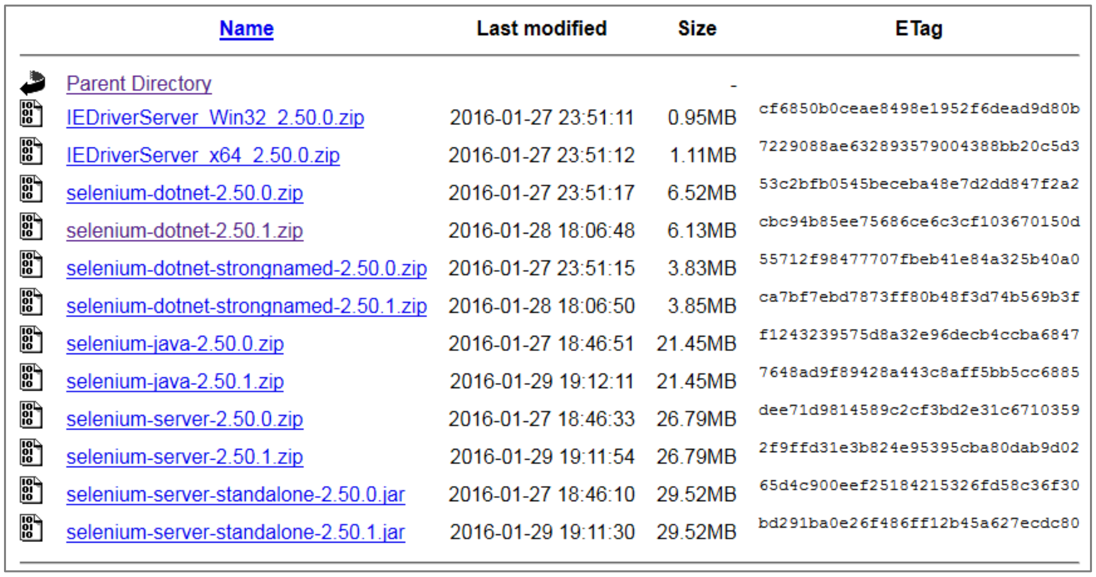
http://selenium-release.storage.googleapis.com/index.html

The list of versions at time of writing was:



Name	Last modified	Size	ETag
2.48	-	-	-
2.49	-	-	-
2.50	-	-	-
icons	-	-	-
index.html	2014-01-13 22:12:39	0.01MB	704b0f841aad1b1428481b7ff3c759c0

When you click on the folder for the latest version you will see the various files that can be downloaded:



Name	Last modified	Size	ETag
Parent Directory		-	
IEDriverServer_Win32_2.50.0.zip	2016-01-27 23:51:11	0.95MB	cf6850b0ceae8498e1952f6dead9d80b
IEDriverServer_x64_2.50.0.zip	2016-01-27 23:51:12	1.11MB	7229088ae63289357900438 8bb20c5d3
selenium-dotnet-2.50.0.zip	2016-01-27 23:51:17	6.52MB	53c2bfb0545beceba48e7d2dd847f2a2
selenium-dotnet-2.50.1.zip	2016-01-28 18:06:48	6.13MB	cbc94b85ee75686ce6c3cf103670150d
selenium-dotnet-strongnamed-2.50.0.zip	2016-01-27 23:51:15	3.83MB	55712f98477707fbeb41e84a325b40a0
selenium-dotnet-strongnamed-2.50.1.zip	2016-01-28 18:06:50	3.85MB	ca7bf7ebd7873ff80b48f3d74b569b3f
selenium-java-2.50.0.zip	2016-01-27 18:46:51	21.45MB	f1243239575d8a32e96decb4ccba6847
selenium-java-2.50.1.zip	2016-01-29 19:12:11	21.45MB	7648ad9f89428a443c8aff5bb5cc6885
selenium-server-2.50.0.zip	2016-01-27 18:46:33	26.79MB	dee71d9814589c2cf3bd2e31c6710359
selenium-server-2.50.1.zip	2016-01-29 19:11:54	26.79MB	2f9ffd31e3b824e95395cba80dab9d02
selenium-server-standalone-2.50.0.jar	2016-01-27 18:46:10	29.52MB	65d4c900eef25184215326fd58c36f30
selenium-server-standalone-2.50.1.jar	2016-01-29 19:11:30	29.52MB	bd291ba0e26f486ff12b45a627ecdc80

Download the **IEDriverServer_XXXX_X.X.X.zip** to your local PC:

- IEDriverServer_Win32_X.X.X.zip (for 32-bit Internet Explorer)
- IEDriverServer_x64_X.X.X.zip (for 64-bit Internet Explorer)

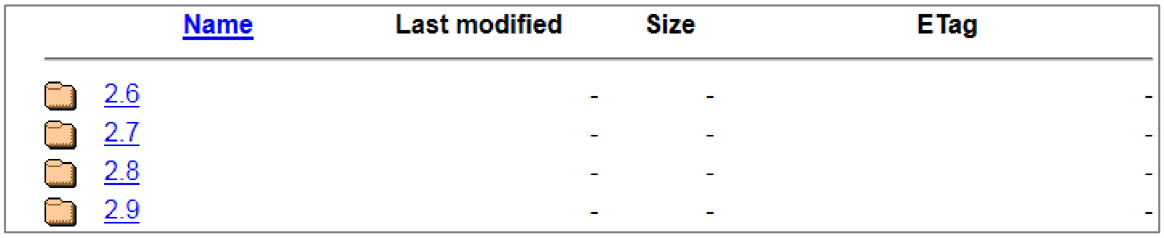
The file inside the zip archive is called **IEDriverServer.exe** and you need to copy it into the **C:\Program Files (x86)\Inflectra\Rapise\Bin** folder (or wherever you installed Rapise).

Chrome

To use Selenium with Google Chrome, you will need to download the latest version of the Chrome Driver:

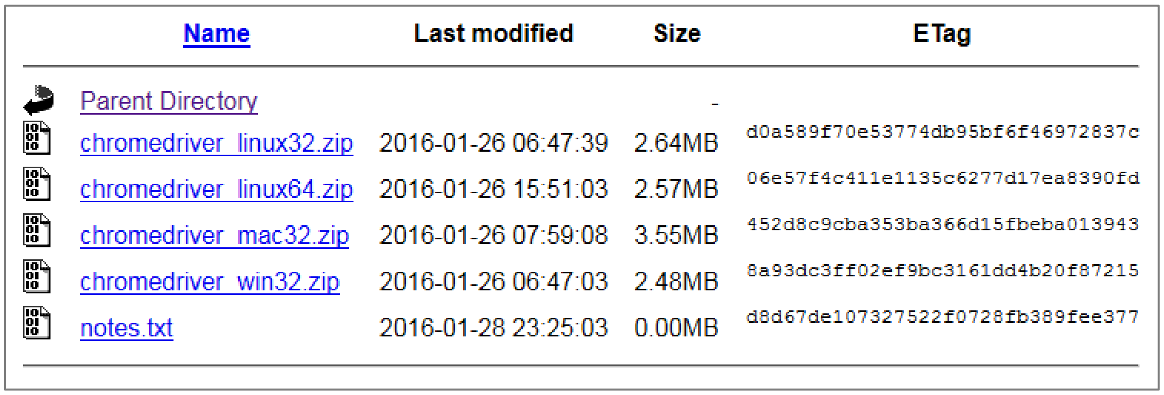
http://chromedriver.storage.googleapis.com/index.html

The list of versions at time of writing was:



Name	Last modified	Size	ETag
2.6	-	-	-
2.7	-	-	-
2.8	-	-	-
2.9	-	-	-

When you click on the folder for the latest version you will see the various files that can be downloaded:



Name	Last modified	Size	ETag
Parent Directory		-	
chromedriver_linux32.zip	2016-01-26 06:47:39	2.64MB	d0a589f70e53774db95bf6f46972837c
chromedriver_linux64.zip	2016-01-26 15:51:03	2.57MB	06e57f4c411e1135c6277d17ea8390fd
chromedriver_mac32.zip	2016-01-26 07:59:08	3.55MB	452d8c9cba353ba366d15fbeba013943
chromedriver_win32.zip	2016-01-26 06:47:03	2.48MB	8a93dc3ff02ef9bc3161dd4b20f87215
notes.txt	2016-01-28 23:25:03	0.00MB	d8d67de107327522f0728fb389fee377

Download the **chromedriver_win32.zip** to your local PC.

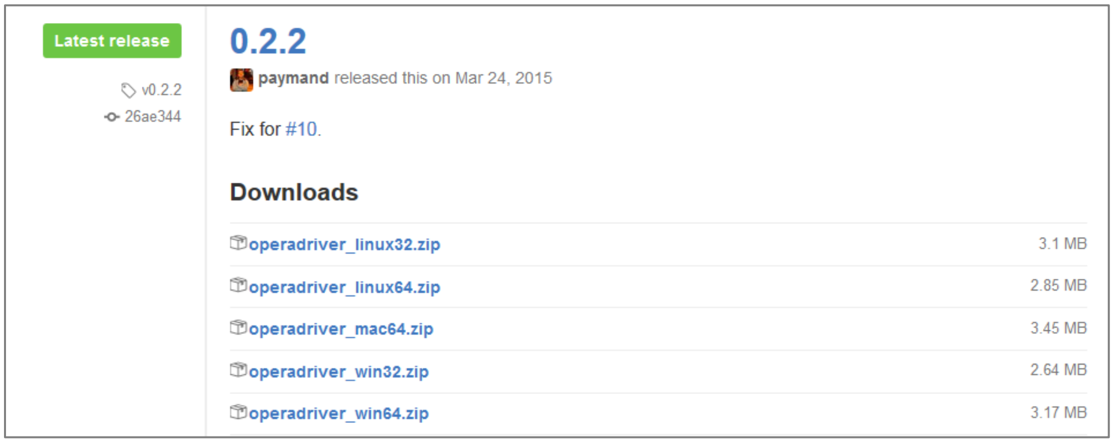
The file inside the zip archive is called **chromedriver.exe** and you need to copy it into the **C:\Program Files (x86)\Inflectra\Rapise\Bin** folder (or wherever you installed Rapise).

Opera

To use Selenium with Opera, you will need to download the latest version of the Opera Driver:

https://github.com/operasoftware/operachromiumdriver/releases

This page will list the latest version of the driver at the top of the page:


Latest release

v0.2.2

26ae344

0.2.2

paymand released this on Mar 24, 2015

Fix for #10.

Downloads

operadriver_linux32.zip	3.1 MB
operadriver_linux64.zip	2.85 MB
operadriver_mac64.zip	3.45 MB
operadriver_win32.zip	2.64 MB
operadriver_win64.zip	3.17 MB

Download the **operadriver_winXX.zip** to your local PC:

- operadriver_win32.zip (for 32-bit Opera)
- operadriver_win64.zip (for 64-bit Opera)

The file inside the zip archive is called **operadriver.exe** and you need to copy it into the **C:\Program Files (x86)\Inflectra\Rapise\Bin** folder (or wherever you installed Rapise).

1.2. Installing Selenium on a Mac

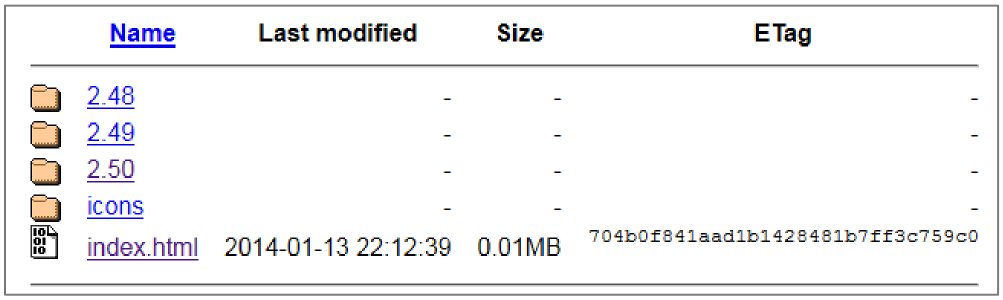
The reason for using Selenium running on a Mac is to be able to execute tests against the Safari web browser. So although you can also use the Mac to test with Firefox, Opera and Chrome, we do not recommend this as it adds needless complexity.

Safari

The first thing you need to do is download the latest version of the Selenium server for Apple Mac computers:

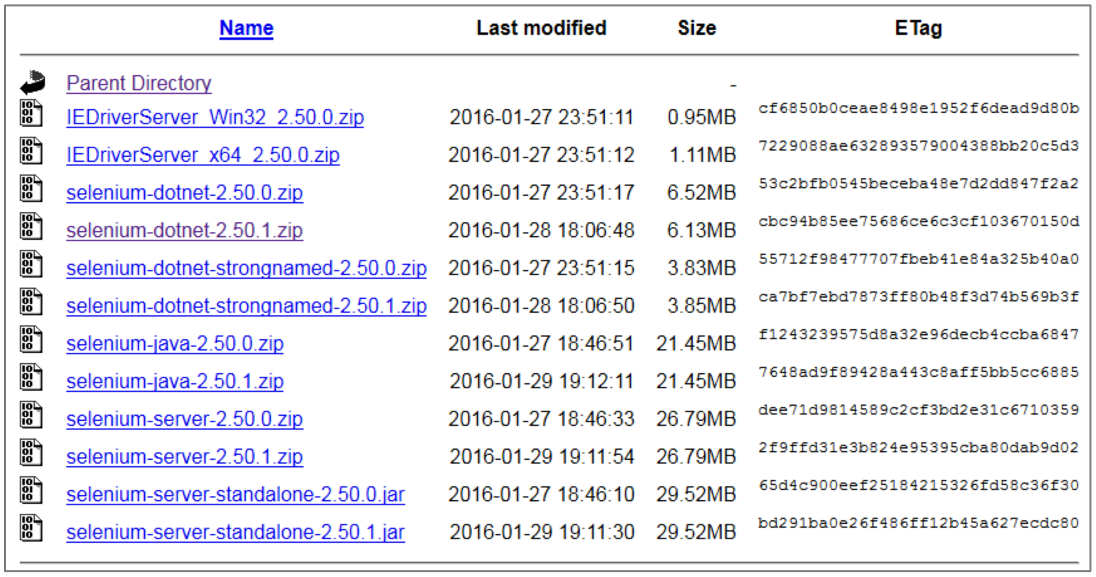
http://selenium-release.storage.googleapis.com/index.html

The list of versions at time of writing was:



Name	Last modified	Size	ETag
2.48	-	-	-
2.49	-	-	-
2.50	-	-	-
icons	-	-	-
index.html	2014-01-13 22:12:39	0.01MB	704b0f841aad1b1428481b7ff3c759c0

When you click on the folder for the latest version you will see the various files that can be downloaded:



Name	Last modified	Size	ETag
Parent Directory		-	
IEDriverServer_Win32_2.50.0.zip	2016-01-27 23:51:11	0.95MB	cf6850b0ceae8498e1952f6dead9d80b
IEDriverServer_x64_2.50.0.zip	2016-01-27 23:51:12	1.11MB	7229088ae63289357900438 8bb20c5d3
selenium-dotnet-2.50.0.zip	2016-01-27 23:51:17	6.52MB	53c2bfb0545beceba48e7d2dd847f2a2
selenium-dotnet-2.50.1.zip	2016-01-28 18:06:48	6.13MB	cbc94b85ee75686ce6c3cf103670150d
selenium-dotnet-strongnamed-2.50.0.zip	2016-01-27 23:51:15	3.83MB	55712f98477707fbeb41e84a325b40a0
selenium-dotnet-strongnamed-2.50.1.zip	2016-01-28 18:06:50	3.85MB	ca7bf7ebd7873ff80b48f3d74b569b3f
selenium-java-2.50.0.zip	2016-01-27 18:46:51	21.45MB	f1243239575d8a32e96decb4ccba6847
selenium-java-2.50.1.zip	2016-01-29 19:12:11	21.45MB	7648ad9f89428a443c8aff5bb5cc6885
selenium-server-2.50.0.zip	2016-01-27 18:46:33	26.79MB	dee71d9814589c2cf3bd2e31c6710359
selenium-server-2.50.1.zip	2016-01-29 19:11:54	26.79MB	2f9ffd31e3b824e95395cba80dab9d02
selenium-server-standalone-2.50.0.jar	2016-01-27 18:46:10	29.52MB	65d4c900eef25184215326fd58c36f30
selenium-server-standalone-2.50.1.jar	2016-01-29 19:11:30	29.52MB	bd291ba0e26f486ff12b45a627ecdc80

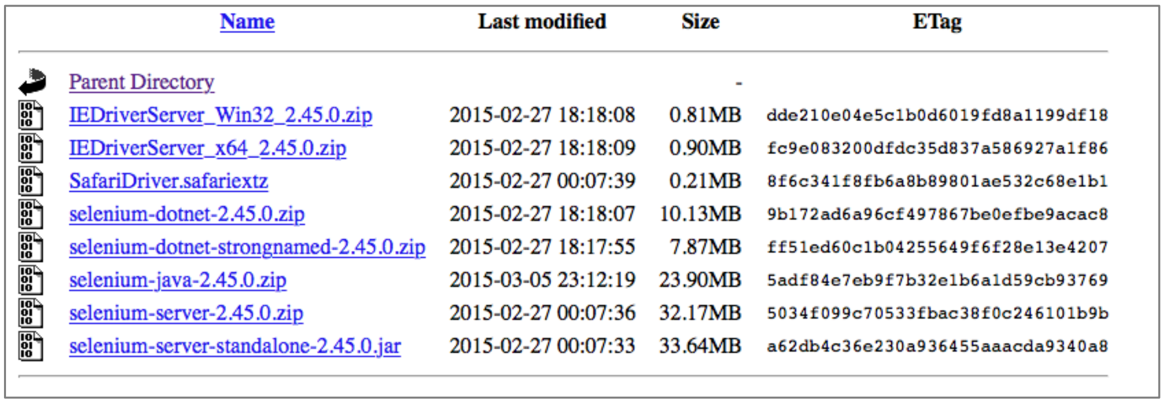
Download the **selenium-server-standalone-X.XX.X.jar** to the Mac.

Run this Java application by double clicking the downloaded .JAR file in Finder. This will startup the Selenium server.

Note: You will need to have the Java (ideally the latest version) installed on the Mac first.

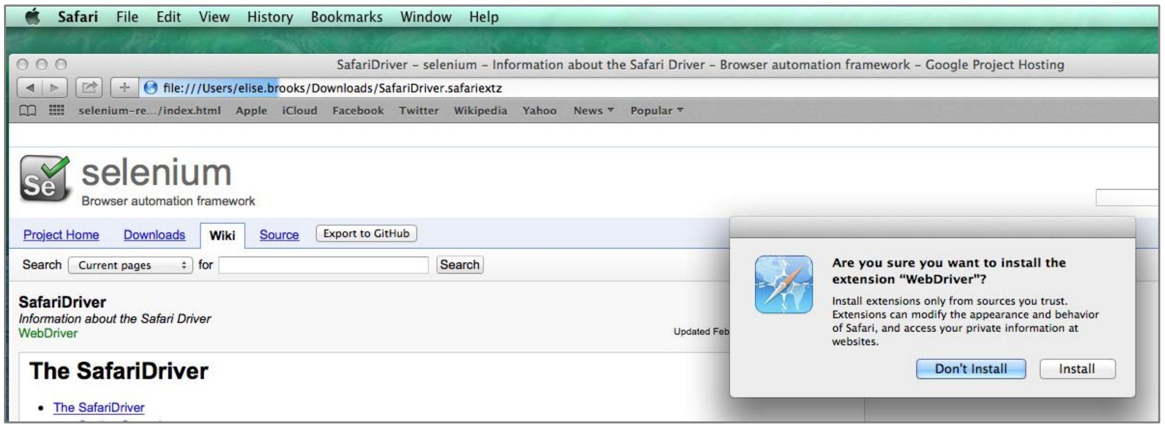
Once you have this running, you will need to then download the actual Safari WebDriver plugin. This can be found at the following location:

https://github.com/SeleniumHQ/selenium/wiki/SafariDriver



Name	Last modified	Size	ETag
Parent Directory		-	
IEDriverServer_Win32_2.45.0.zip	2015-02-27 18:18:08	0.81MB	dde210e04e5c1b0d6019fd8a1199df18
IEDriverServer_x64_2.45.0.zip	2015-02-27 18:18:09	0.90MB	fc9e083200dfdc35d837a586927a1f86
SafariDriver.safariextz	2015-02-27 00:07:39	0.21MB	8f6c341f8fb6a8b89801ae532c68e1b1
selenium-dotnet-2.45.0.zip	2015-02-27 18:18:07	10.13MB	9b172ad6a96cf497867be0efbe9acac8
selenium-dotnet-strongnamed-2.45.0.zip	2015-02-27 18:17:55	7.87MB	ff51ed60c1b04255649f6f28e13e4207
selenium-java-2.45.0.zip	2015-03-05 23:12:19	23.90MB	5adf84e7eb9f7b32e1b6a1d59cb93769
selenium-server-2.45.0.zip	2015-02-27 00:07:36	32.17MB	5034f099c70533fbac38f0c246101b9b
selenium-server-standalone-2.45.0.jar	2015-02-27 00:07:33	33.64MB	a62db4c36e230a936455aaacda9340a8

Download the **SafariDriver.safariextz** file to the local computer and the double-click to install in Safari:


Safari File Edit View History Bookmarks Window Help
SafariDriver – selenium – Information about the Safari Driver – Browser automation framework – Google Project Hosting
file:///Users/elise.brooks/Downloads/SafariDriver.safariextz
selenium-re.../index.html Apple iCloud Facebook Twitter Wikipedia Yahoo News Popular
selenium
Browser automation framework
Project Home Downloads Wiki Source Export to GitHub
Search Current pages for Search
Don't Install
Install

SafariDriver
Information about the Safari Driver
WebDriver

Updated Feb

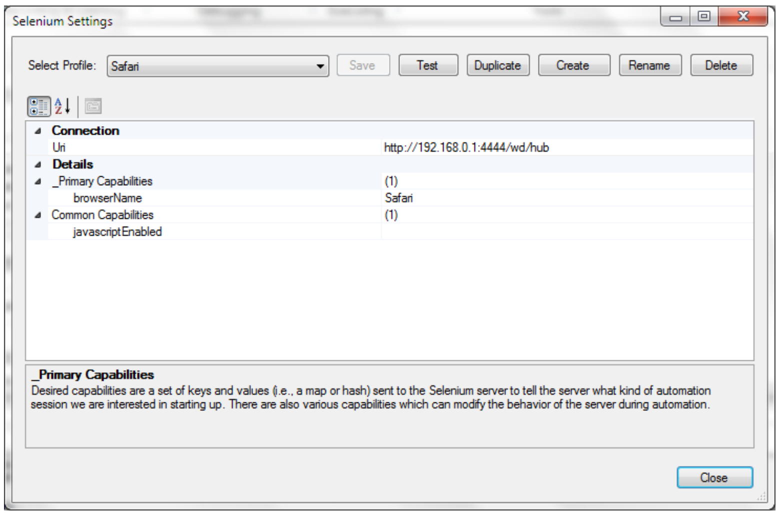
The SafariDriver

- The SafariDriver

Are you sure you want to install the extension "WebDriver"?

Install extensions only from sources you trust. Extensions can modify the appearance and behavior of Safari, and access your private information at websites.

Once that has been installed, you are now ready to test web applications running on Safari. The final step is to tell Rapise where it can find that instance of Selenium. To do that, open up Rapise (on your PC) and click on **Options > Tools** and then click on the **'Selenium Settings…'** entry:



Select Profile: Safari

Connection	
Uri	http://192.168.0.1:4444/wd/hub
Details	
_Primary Capabilities	(1)
browserName	Safari
Common Capabilities	(1)
javascriptEnabled	

_Primary Capabilities
Desired capabilities are a set of keys and values (i.e., a map or hash) sent to the Selenium server to tell the server what kind of automation session we are interested in starting up. There are also various capabilities which can modify the behavior of the server during automation.

Now you need to change the **Uri** field to point to your Mac. The format of the URI will be:

- `http://<IP or DNS name of MAC computer>:4444/wd/hub`

(for example it could be http://test-mac01.local:4444/wd/hub or http://192.168.0.52:4444/wd/hub)

2. Playing & Recording Tests

Now that you have installed and configured the integration between Rapise and Selenium, we shall discuss how to use Selenium with Rapise to record and play tests.

Now one of the important points is that there are some limitations as to the operations that can be performed using Selenium-based web browsers as opposed to the native browsers supported by Rapise:

Feature	Rapise Native Browser	Selenium Browser
Learn HTML Objects	Yes	**(Only in Web Spy)**
Record HTML Events	Yes	**No**
Playback HTML Events	Yes	Yes
Web Spy	Yes	Yes
Learn Java Applets	Yes	**No**
Learn Silverlight Controls	Yes	**No**
Manual Testing	Yes	**No**

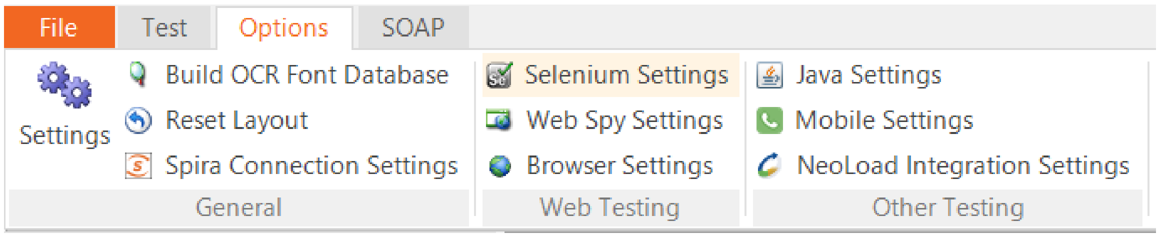
So if you are planning on using Rapise to record a test script by clicking HTML objects and having Rapise create the script using the learned objects and adding the events (DoClick, SetText, etc.) then you will need to use one of the native browsers (Chrome, IE, Firefox) to create the test script. You can then playback the same test in either the native or Selenium browsers.

If you are planning on using Rapise to learn objects using the Web Spy, and then create the test script from those objects by either dragging the object methods and properties from the Object Tree into the test script or just using Intellisense to type the methods (DoClick, SetText, etc.) then you can use either a native or Selenium web browser just as easily.

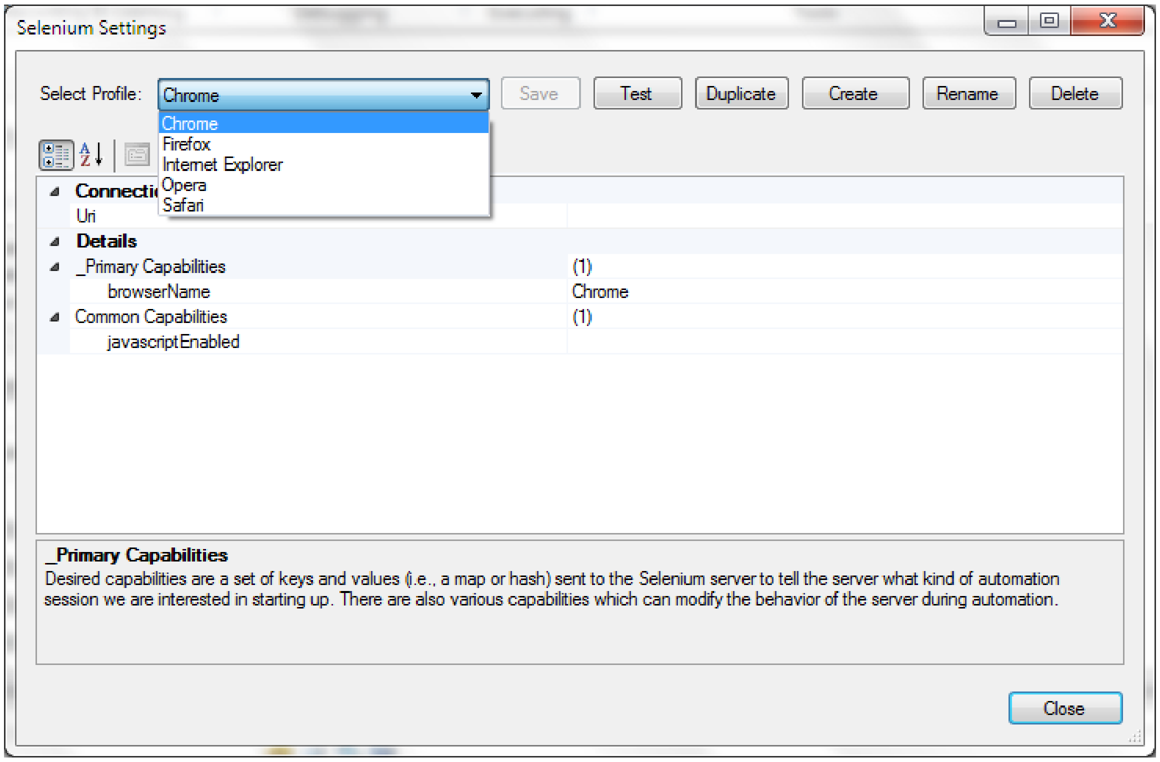
For most users, the primary reason for using the Selenium web browsers will be to playback their tests on a greater number of browsers or to leverage existing Selenium WebDriver scripts created outside of Rapise (see section 3).

2.1. Managing the Selenium Profiles

Rapise allows you to maintain different profiles for your different installed Selenium web browsers (both on the same machine as Rapise and also those running on a remote Selenium WebDriver server), To see the different Selenium profiles, go to the **Options** ribbon:



Click on the **Selenium Settings** option in the Web Testing section, this will bring up the Selenium profile manager:


Selenium Settings

Select Profile: Chrome

Chrome
Firefox
Internet Explorer
Opera
Safari

Connection	
Uri	
Details	
_Primary Capabilities	(1)
browserName	Chrome
Common Capabilities	(1)
javascriptEnabled	

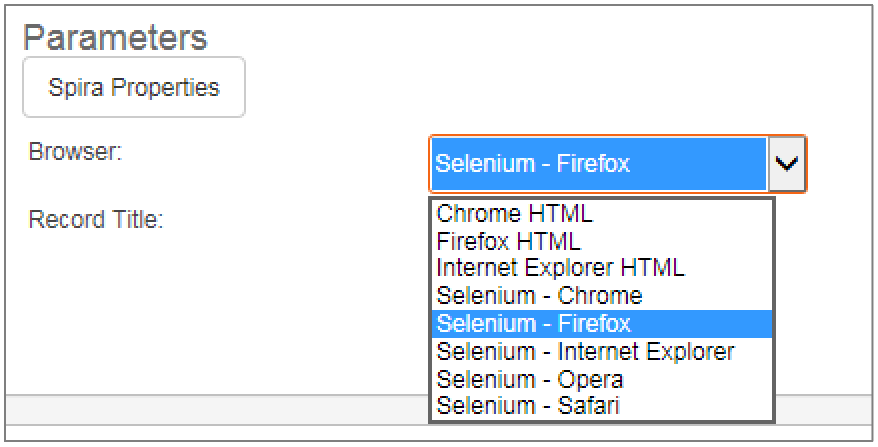
_Primary Capabilities
Desired capabilities are a set of keys and values (i.e., a map or hash) sent to the Selenium server to tell the server what kind of automation session we are interested in starting up. There are also various capabilities which can modify the behavior of the server during automation.

By default there is one profile for each of the Selenium WebDriver supported web browsers (Chrome, Firefox, Internet Explorer, Opera, Safari). However you can clone and change the profiles if you want to have different versions of the browsers (e.g. a local instance of Firefox and one running on a remote Selenium server).

Most users will only need to change the **Uri** field of the Safari web browser (see section 1.1) since the defaults are typically sufficient for most testing needs.

2.2. Recording using Selenium

To start recording a web testing using a Selenium WebDriver based browser, make sure you change the test's web browser parameter to one of the Selenium profiles:

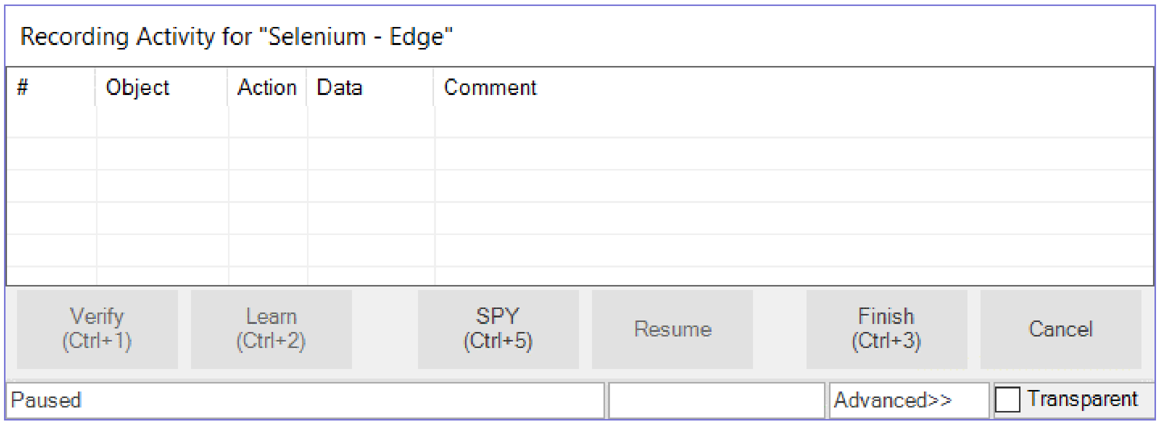

Parameters

Browser: Selenium - Firefox

Record Title:

Chrome HTML
Firefox HTML
Internet Explorer HTML
Selenium - Chrome
Selenium - Firefox
Selenium - Internet Explorer
Selenium - Opera
Selenium - Safari

When you click the **Record/Learn** button in the main Test ribbon you will see the following **Recording Activity Dialog**:

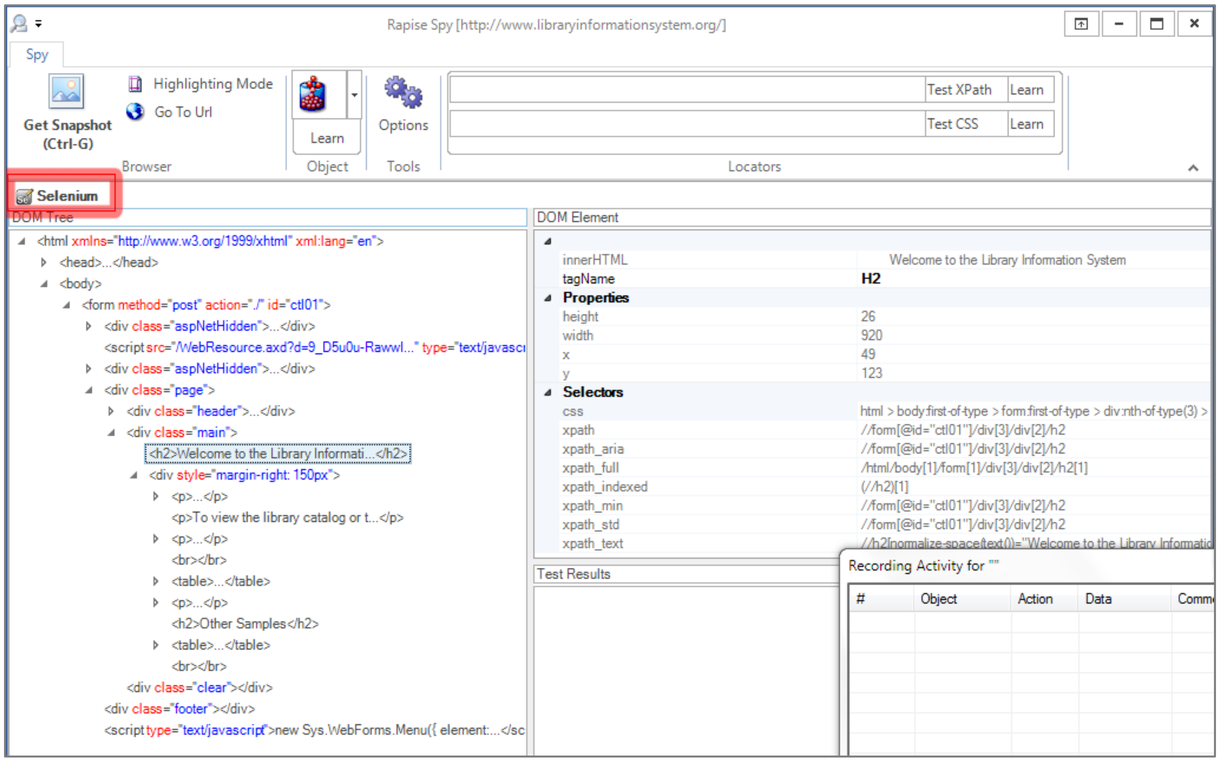


Recording Activity for "Selenium - Edge"

#	Object	Action	Data	Comment

You will notice that the **Verify**, and **Learn** options are not available. If you want to use these tools you will need to use a native browser (non-Selenium) instead.

When using a Selenium profile for recording, you will need to use the **Spy (Ctrl+5)** tool to do the learning of objects on the web page. This brings up the Web Spy:



Rapise Spy [http://www.libraryinformationsystem.org/]

```
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
  <head>...</head>
  <body>
    <form method="post" action="./" id="ctl01">
      <div class="aspNetHidden">...</div>
      <script src="/WebResource.axd?d=9_D5u0u-Rawwl..." type="text/javasci
      <div class="aspNetHidden">...</div>
      <div class="page">
        <div class="header">...</div>
        <div class="main">
          <h2>Welcome to the Library Informati...</h2>
          <div style="margin-right: 150px">
            <p>...</p>
            <p>To view the library catalog or t...</p>
            <p>...</p>
            <br></br>
            <table>...</table>
            <p>...</p>
            <h2>Other Samples</h2>
            <table>...</table>
            <br></br>
        <div class="clear"></div>
      <div class="footer"></div>
      <script type="text/javascript">new Sys.WebForms.Menu({ element:...</sc
```

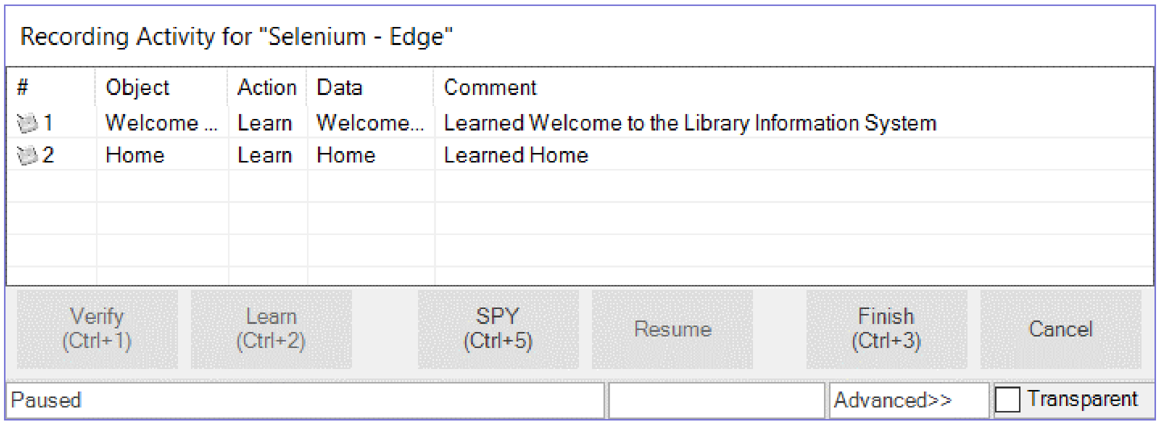
innerHTML	Welcome to the Library Information System
tagName	**H2**
Properties	
height	26
width	920
x	49
y	123
Selectors	
css	html > body:first-of-type > form:first-of-type > div:nth-of-type(3) >
xpath	//form[@id="ctl01"]/div[3]/div[2]/h2
xpath_aria	//form[@id="ctl01"]/div[3]/div[2]/h2
xpath_full	/html/body[1]/form[1]/div[3]/div[2]/h2[1]
xpath_indexed	(//h2)[1]
xpath_min	//form[@id="ctl01"]/div[3]/div[2]/h2
xpath_std	//form[@id="ctl01"]/div[3]/div[2]/h2
xpath_text	//h2[normalize-space(text())="Welcome to the Library Informatio

Recording Activity for ""

#	Object	Action	Data	Comm

When using the Web Spy with a Selenium profile you will notice that the web browser icon / name shows “Selenium” rather than the browser name and the option to Track an item (CTRL+T) is not present. That means you need to select the HTML DOM object in the DOM Tree and learn it from there (rather than clicking on the web page itself which is possible when using a native browser profile).

When you choose to Learn an object in the DOM tree it will be displayed in the Recording Activity Dialog as a new Learned Object:


Recording Activity for "Selenium - Edge"

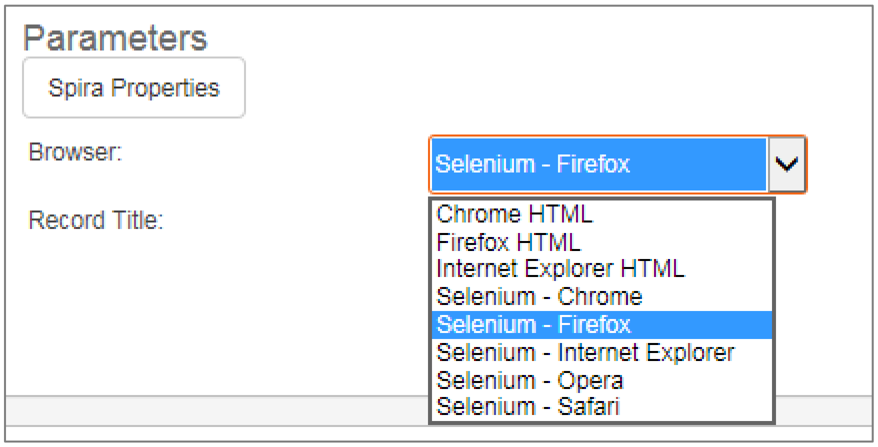
#	Object	Action	Data	Comment
1	Welcome ...	Learn	Welcome...	Learned Welcome to the Library Information System
2	Home	Learn	Home	Learned Home

Objects Learned using a Selenium profile will be added to the Rapise Object Tree in the usual way and as is typical with Learning, you have the option to specify an Action in the Recording Activity Grid (e.g. change Learn to Click) in which case test script code is also generated.

***Tip:** Due to the inherent limitations in recording using a Selenium browser profile (vs. a native browser profile) most users will record their scripts using a native browser and then use Selenium primarily for debugging using the Web Spy and playback.*

2.3. Playback using Selenium

To playback a web test using a Selenium web browser profile, simply choose the appropriate profile in the test Start Page:

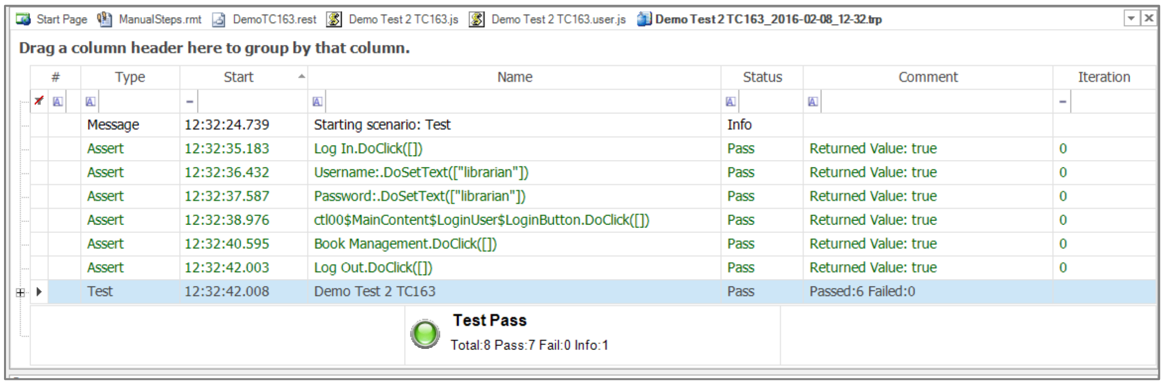

Parameters

Spira Properties

Browser: Selenium - Firefox

Record Title:

Then click the **Play** button on the main Test ribbon. The test will now start execution. Unlike recording there is nothing different in the way Rapise handles the playback of a Selenium test. The only difference will be that if the test uses non-HTML technologies such as Silverlight, Java, etc. those parts of the test will fail.



Start Page | ManualSteps.rmt | DemoTC163.rest | Demo Test 2 TC163.js | Demo Test 2 TC163.user.js | Demo Test 2 TC163_2016-02-08_12-32.trp

Drag a column header here to group by that column.

#	Type	Start	Name	Status	Comment	Iteration
	Message	12:32:24.739	Starting scenario: Test	Info		
	Assert	12:32:35.183	Log In.DoClick([])	Pass	Returned Value: true	0
	Assert	12:32:36.432	Username:.DoSetText(["librarian"])	Pass	Returned Value: true	0
	Assert	12:32:37.587	Password:.DoSetText(["librarian"])	Pass	Returned Value: true	0
	Assert	12:32:38.976	ctl00$MainContent$LoginUser$LoginButton.DoClick([])	Pass	Returned Value: true	0
	Assert	12:32:40.595	Book Management.DoClick([])	Pass	Returned Value: true	0
	Assert	12:32:42.003	Log Out.DoClick([])	Pass	Returned Value: true	0
	Test	12:32:42.008	Demo Test 2 TC163	Pass	Passed:6 Failed:0	

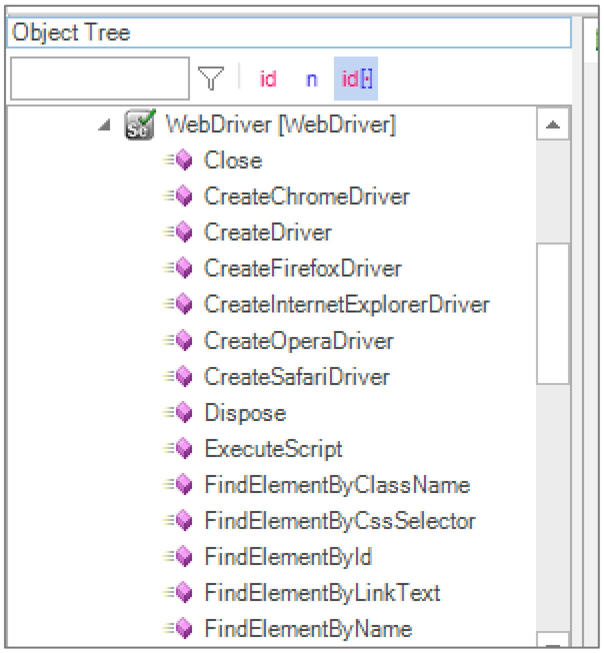
Test Pass
Total:8 Pass:7 Fail:0 Info:1

3. Using Native Selenium Code

When using the standard Rapise Record, Learn, and Web Spy tools you can create your web tests using the Rapise built-in Object Tree. This lets you design your tests using a higher level of abstraction than working directly with Selenium WebDriver.

For example, you can learn an object `SeS("EditButton")` that points to a dynamic XPATH or CSS query that the automation engineer knows will be accurate even if the data on the page changes (for example). The QA analyst can then simply drag and drop these Rapise objects from the Object Tree into the test script (e.g. `SeS("EditButton").DoClick()`) to perform the desired action.

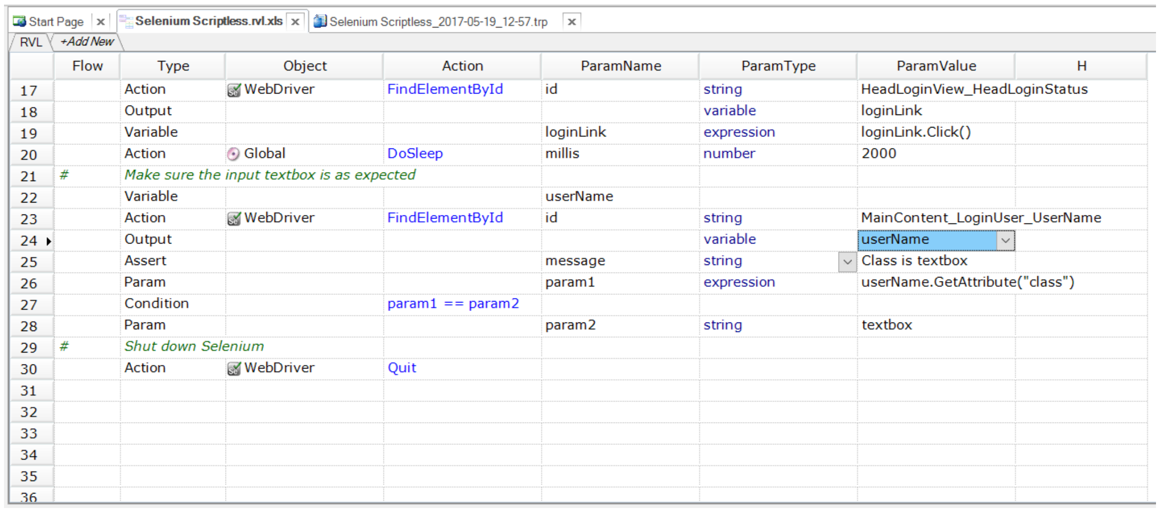
Sometimes however you will want to be able to run standard Selenium raw WebDriver code inside Rapise using the standard Selenium WebDriver API functions (http://www.seleniumhq.org/docs/03_webdriver.jsp). Rapise allows you to do this using the special **WebDriver** global object:



The **WebDriver** object implements the various standard Selenium WebDriver API calls for automating the web browser. There is a sample available for Rapise called "UsingSelenium" that illustrates using the WebDriver code directly, but for completeness, here is a sample that uses the **www.libraryinformationsystem.org** same web site and performs some simple actions:

3.1. Using the Rapise Visual Language (RVL)

An example test script using these objects and the Rapise Visual Language (RVL) scriptless option is shown below:


Start Page
Selenium Scriptless.rvl.xls
Selenium Scriptless_2017-05-19_12-57.trp
RVL
+Add New

	Flow	Type	Object	Action	ParamName	ParamType	ParamValue	H
17		Action	WebDriver	FindElementById	id	string	HeadLoginView_HeadLoginStatus	
18		Output				variable	loginLink	
19		Variable			loginLink	expression	loginLink.Click()	
20		Action	Global	DoSleep	millis	number	2000	
21	#	*Make sure the input textbox is as expected*						
22		Variable			userName			
23		Action	WebDriver	FindElementById	id	string	MainContent_LoginUser_UserName	
24		Output				variable	userName	
25		Assert			message	string	Class is textbox	
26		Param			param1	expression	userName.GetAttribute("class")	
27		Condition		param1 == param2				
28		Param			param2	string	textbox	
29	#	*Shut down Selenium*						
30		Action	WebDriver	Quit				
31								
32								
33								
34								
35								
36								

3.2. Using JavaScript Scripting

An example test script using these objects and the Rapise JavaScript scripting option is shown below:

```
//First create the Firefox driver
WebDriver.CreateFirefoxDriver();

//Open the URL for the www.libraryinformationsystem.org website:
WebDriver.SetUrl('http://www.libraryinformationsystem.org');

//Find the body element and verify the text in it
var el = WebDriver.FindElementByXPath("//body");
Tester.Assert("Text found in BODY", el.GetText().indexOf("Library Information System")
!= -1);

//Click on the login link
var logInLink = WebDriver.FindElementById('HeadLoginView_HeadLoginStatus');
logInLink.Click();

//Make sure the input textbox is as expected
var userName = WebDriver.FindElementByCssSelector("html > body > form > div:nth-of-
type(3) > div:nth-of-type(2) > div:nth-of-type(2) > fieldset > p:first-of-type >
input");
Tester.AssertEqual("class is 'textbox'", "textbox", userName.GetAttribute("class"));

//Go to a different URL (http://libraryinformationsystem.org/HtmlTest.htm)
WebDriver.SetUrl('http://www.libraryinformationsystem.org/HtmlTest.htm');

//Click on the Alert box
var alertBtn = WebDriver.FindElementById("btnAlert");
alertBtn.Click();

//Switch to this alert box and close
var alertElement = WebDriver.SwitchToAlert();
alertElement.Accept();

//Shut down Selenium
WebDriver.Quit()
```

Rapise provides full Intellisense and code-completion for the **WebDriver** global object:

WebDriver Method	Description

CreateFirefoxDriver	Initializes a new instance of the Firefox WebDriver
CreateInternetExplorerDriver	Initializes a new instance of the Microsoft IE WebDriver
CreateChromeDriver	Initializes a new instance of the Google Chrome WebDriver
CreateOperaDriver	Initializes a new instance of the Opera WebDriver
CreateSafariDriver	Initializes a new instance of the Apple Safari WebDriver
CreateDriver	This will create a WebDriver for currently selected Browser profile
GetUrl	Gets the URL the browser is currently displaying
SetUrl	Sets the URL the browser is currently displaying
GetCurrentWindowHandle	Gets the current window handle, which is an opaque handle to this window that uniquely identifies it within this driver instance.
GetPageSource	Gets the source of the page last loaded by the browser.
GetTitle	ets the title of the current browser window.
GetWindowHandles	Gets the window handles of open browser windows.
Quit	Closes the Browser
Close	Close the Browser and Dispose of WebDriver
Dispose	Dispose of WebDriver
ExecuteScript	Executes JavaScript in the context of the currently selected frame or window
FindElementByClassName	Finds the first element in the page that matches the CSS Class supplied
FindElementByCssSelector	Finds the first element matching the specified CSS selector.
FindElementById	Finds the first element in the page that matches the ID supplied
FindElementByLinkText	Finds the first of elements that match the link text supplied
FindElementByName	Finds the first of elements that match the name supplied
FindElementByPartialLinkText	Finds the first of elements that match the part of the link text supplied
FindElementByTagName	Finds the first of elements that match the DOM Tag supplied
FindElementByXPath	Finds the first of elements that match the XPath supplied
FindElementsByClassName	Finds a list of elements that match the class name supplied
FindElementsByCssSelector	Finds all elements matching the specified CSS selector.
FindElementsById	Finds the first element in the page that matches the ID supplied
FindElementsByLinkText	Finds a list of elements that match the link text supplied
FindElementsByName	Finds a list of elements that match the name supplied
FindElementsByPartialLinkText	Finds a list of elements that match the part of the link text supplied
FindElementsByTagName	Finds a list of elements that match the DOM Tag supplied
FindElementsByXPath	Finds a list of elements that match the XPath supplied
SwitchToAlert	Switches to the currently active modal dialog for this particular driver instance.
SwitchToFrame	Move to different frame using its name
SwitchToParentFrame	Select the parent frame of the currently selected frame.
SwitchToDefaultContent	Change the active frame to the default

When you click **'Play'** to playback your Selenium script, make sure you have selected one of the Selenium web browser profiles. If you have selected a native browser profile (e.g. "Firefox HTML" instead of "Selenium – Firefox") you will get the error message "WebDriver" is not defined.

When you are using functions such as `FindElementsById()` in your code, the returned object will be a Selenium **Web Element**. Such elements have a variety of supported functions:

WebElement Method	**Description**
Submit	Submits this element to the web server.
SendKeys	Simulates typing text into the element.
GetCssValue	Gets the value of a CSS property of this element.
GetAttribute	Gets the value of the specified attribute for this element.
FindElementsByXPath	*Same as WebDriver Method except only looks for children of this element*
FindElementsByTagName	*Same as WebDriver Method except only looks for children of this element*
FindElementsByPartialLinkText	*Same as WebDriver Method except only looks for children of this element*
FindElementsByName	*Same as WebDriver Method except only looks for children of this element*
FindElementsByLinkText	*Same as WebDriver Method except only looks for children of this element*
FindElementsById	*Same as WebDriver Method except only looks for children of this element*
FindElementsByCssSelector	*Same as WebDriver Method except only looks for children of this element*
FindElementsByClassName	*Same as WebDriver Method except only looks for children of this element*
FindElementByXPath	*Same as WebDriver Method except only looks for children of this element*
FindElementByTagName	*Same as WebDriver Method except only looks for children of this element*
FindElementByPartialLinkText	*Same as WebDriver Method except only looks for children of this element*
FindElementByName	*Same as WebDriver Method except only looks for children of this element*
FindElementByLinkText	*Same as WebDriver Method except only looks for children of this element*
FindElementById	*Same as WebDriver Method except only looks for children of this element*
FindElementByCssSelector	*Same as WebDriver Method except only looks for children of this element*
FindElementByClassName	*Same as WebDriver Method except only looks for children of this element*

ClickAt	Clicks this element at the specified location
Click	Clicks this element.
Clear	Clears the content of this element.
GetTagName	Gets the tag name of this element.
GetSize	Gets a 'Size' object containing the height and width of this element.
GetSelected	Gets a value indicating whether or not this element is selected.
GetLocation	Gets a 'Point' object containing the coordinates of the upper-left corner of this element relative to the upper-left corner of the page.
GetEnabled	Gets a value indicating whether or not this element is enabled.
GetDisplayed	Gets a value indicating whether or not this element is displayed.
GetCoordinates	Gets a 'Point' object containing the coordinates of the upper-left corner of this element relative to the upper-left corner of the page.
GetText	Gets the innerText of this element, without any leading or trailing whitespace, and with other whitespace collapsed.

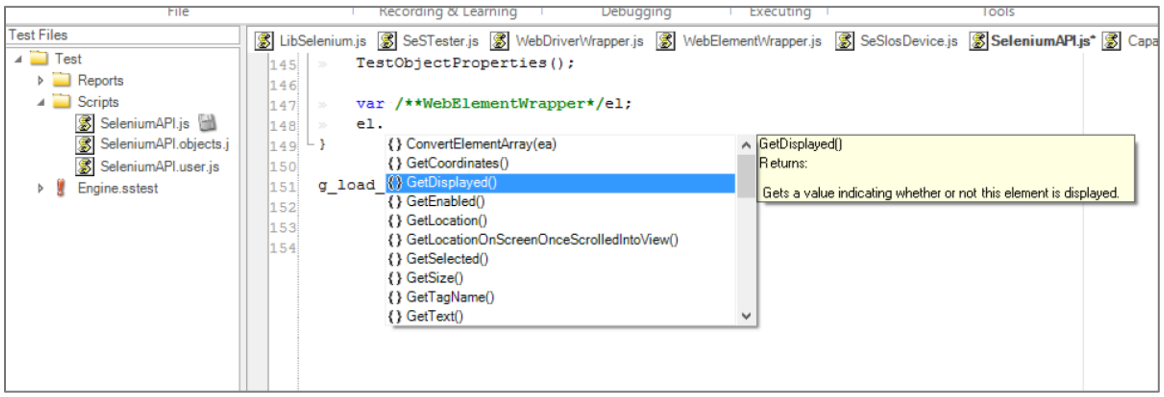
If you want to have intellisense and code-completion for the returned WebElement objects, you need to do two things:

- Click on CTRL+E to include the Rapise engine in your project.
- Prefix the variable with `/**WebElementWrapper*/`, so instead of just:

`var el = WebDriver.FindElementById(…)` you use:

`var /**WebElementWrapper*/el = WebDriver.FindElementById(…)` instead

For example:



```
    TestObjectProperties();

    var /**WebElementWrapper*/el;
    el.
}

g_load_
```

```
{} ConvertElementArray(ea)
{} GetCoordinates()
{} GetDisplayed()
{} GetEnabled()
{} GetLocation()
{} GetLocationOnScreenOnceScrolledIntoView()
{} GetSelected()
{} GetSize()
{} GetTagName()
{} GetText()
```

GetDisplayed()
Returns:

Gets a value indicating whether or not this element is displayed.

3.2. Interoperability with Rapise Objects

In addition to being able to use raw Selenium code on its own, you can also use a mixture of Rapise object-based code and raw Selenium WebDriver code.

For example, you are using the standard Rapise approach (using learned `SeS('object')` objects) for testing but at some point want to switch to Selenium API to call a couple of WebElement functions on a learned object, you can use the special **‘element’** property:

```
var webElement = SeS('MyObject').element;
```

If you want to the reverse and be able to create a Rapise `SeS('object')` ‘on the fly’ from a physical object on the web page, you can do the `MakeObjectForXPath(xpath)` function that returns a Rapise **SeSObject**, in the same way that `SeS(‘id’)` does normally:

```
var sesObj = MakeObjectForXPath("//body//div[@id='logArea']");
```

Legal Notices

Please send comments and questions to:

Technical Publications

Inflectra Corporation

8121 Georgia Ave, Suite 504

Silver Spring, MD 20910-4957

U.S.A.

support@inflectra.com