AUTOnomyV User's Guide

DiJohn IC

www.dijohn-ic.com

"Making Innovation The Standard"

Table of Contents

1.0 Introduction

AUTOnomyV is an automated test tool which uses the VBScript scripting language (Hence the "V" in "AUTOnomyV") to automate Internet Explorer HTML-based applications. It derives its name from the words "automation" (the capital A-U-T-O) and "autonomy", which means "freedom" and/or "the state of free". AUTOnomyV provides web test automation freedom from overly expensive commercial tools, and is totally free of charge!

1.1 Why AUTOnomyV

There are several reasons to use AUTOnomyV for your automation needs, including the fact that it:

- Is a free Open Source tool. There are no costs to use the tool.
- Uses VBScript, a full-featured object-oriented scripting language, rather than a proprietary vendorscript.
- Can be used in conjunction with standard VBScript syntax, to increase the power of your automated tests.
- Provides utilities and functions that make it easier for you to identify and access Internet browser objects. Once these objects have been accessed, you can use standard VBScript and/or Document Object Model properties, methods, collections and events (visit http://msdn.microsoft.com/ for more information).
- Is powerful and easy to use.
- Offers key **features** that similar open source frameworks neglect to offer including:
    - Record & Playback! We all know some of the reasons to not rely solely on record and playback, but having such a feature is key in accessing object properties, and just helping to quickly learn the tool language, as well as quickly learning some VBScripting Basics.
    - You can perform automation on an already open browser window! Many similar frameworks limit your automation activities to browser windows open during the script run, which severely limits your automation and debugging capabilities. AUTOnomyV allows you the freedom of automating actions in already open browser windows.
    - AUTOnomyV offers several utilities that will be useful whether you choose to use AUTOnomyV syntax, basic VBScripting syntax, or another scripting language syntax (i.e. GetElementAttributes.vbs utility, which generates a file with the object attributes of all open Internet windows)

1.2 AUTOnomyV Compatibility

AUTOnomyV will drive Internet applications that are generated as HTML pages in an Internet Explorer browser. AUTOnomyV will not work with Java Applets, Macromedia Flash, or other plug-in applications. To determine whether AUTOnomyV can be used to automate a part of a web application, right click on the object and see if the View Source menu option is available. If you can view the HTML source, that object can be automated using AUTOnomyV.

1.3 Prerequisites

To use the tool, you should have a basic understanding of:

- HTML: A fundamental understanding of HTML will be necessary for effectively automating Internet applications.

- Programming: You should understand programming basics, including variables and simple control flow functions like “for” loops and “if-then-else” statements.
- VBScript: VBScript is not a prerequisite for getting started with AUTOnomyV, but learning VBScript is important for maximizing the effectiveness of the tool.

2.0 Getting Started

AUTOnomyV runs on Windows machines and works with the Internet Explorer web browser. Development and testing of the framework has taken place on Windows 2000 and XP.

2.1 AUTOnomyV Installation

Installing AUTOnomyV for use is simply a matter of unzipping the downloaded AUTOnomyV zip file, and running a sample test to help ensure AUTOnomyV is compatible with your system.

2.1.1 DOWNLOAD THE ZIP FILE

Upon downloading the AUTOnomyV zip file from the www.dijohn-ic.com website, and unzipping the file to the C drive (i.e.: C:\), you will find an AUTOnomyV folder on your C drive containing the AUTOnomyV library, utilities, samples and documentation.

2.1.2 RUNNING THE AUTONOMYV SAMPLE TEST

To run a sample test to ensure AUTOnomyV and VBScript will work on your system:

1. Open the C:\AUTOnomyV\Samples folder.
2. Double click on the Sample1_VerifyDefaults.vbs script.
3. Ensure that all pop-up messages indicate a check has passed.

Note: If you are experiencing a problem, visit the Active Content Blocked portion of the 8.0 Trouble Shooting section.

Note: You can also execute the script from the command line by opening up DOS, changing the directory to C:\AUTOnomyV\Samples, and then typing the cscript <filename>. This may appear as follows:

C:\>cd C:\AUTOnomyV\Samples

C:\AUTOnomyV\Samples>cscript Sample1_VerifyDefaults.vbs

This will run the script and cause the messages to be generated in the DOS window as opposed to pop-up windows.

3.0 Editing and Executing Scripts

Since AUTOnomyV is built using standard VBScript, there are some VBScript editing basics that will also apply for editing and executing your AUTOnomyV scripts.

3.1 Choose an Editor

Scripts may be edited in a regular text editor, such as Notepad or WordPad. In addition to Notepad or WordPad, you can use a text editor with more user friendly features such as the SciTE Editor by Scintilla. Once the script is saved with the .vbs extension, it is recognized by your system as a VBScript file. The 4.0 AUTOnomyV Test Scripting section provides more information on how to make the script take advantage of the AUTOnomyV Library.

3.2 Executing Scripts

A script may be executed in one of the following ways:

- Running it from the command line (As described in section 2.1.2).
- Double-clicking on the icon in the file system.
- Using an editor such as the SciTE editor to execute the script (refer to section 3.3).

3.3 The SciTE Editor

The SciTE editor offers some syntax coloring and other features. The editor is also included with the AUTOnomyV packet courtesy of Scintilla (Copyright 1998-2003 by Neil Hodgson <neilh@scintilla.org>). Below you will find information for the installation and use of this editor.

3.3.1 SCITE INSTALLATION

Use the following procedures to install the SciTE editor.

1. Open the C:\AUTOnomyV\Editor folder.
2. Double click on the scite-1.66-setup-1.exe file.
3. Follow the prompts to install the editor.

3.3.2 EDITING SCRIPTS

Open files by using the File → Open menu item. Notice the syntax coloring. If you are creating a new file, you should immediately save it with the .vbs extension, and then you will see the appropriate syntax coloring for VBScript syntax.

3.3.3 RUNNING SCRIPTS

To execute files as if from the command line (with messages appearing in an output pane as opposed to a pop-up window) use the Tools → Build menu item. This will cause all message to be generated in the SciTE output pane (unless generated with the 'msgbox' statement). To execute a file as if it was double-clicked on use the Tools → Go menu item. This will cause all messages to appear in a pop-up window.

For more information on the Scintilla Editor visit http://www.scintilla.org/SciTE.html.

4.0 AUTOnomyV Test Scripting

This section provides you with the basic steps involved in creating AUTOnomyV test scripts for automating Internet applications. Actual AUTOnomyV syntax is presented in section 5.0.

4.1 Learning the Application

Prior to creating your automated tests, it is important to first understand the application under test (AUT), and the tests that must be performed in that application. Be sure to study the application and answer questions about the following:

- Object Referencing – How might you uniquely reference an object (i.e. – you may reference a button by the text written on the button)?
- Inputs – What fields require inputs?
- Outputs – What are some of the values that may need to be obtained from the application in order to verify an action was successful or may need to be used later?
- Selections – What fields require values to be selected from it?

- Presses – What buttons and links may need to be pressed?

You may obtain detailed information about application objects by viewing the HTML source code of the Internet page (in Internet Explorer, right click on the web page and select View Source from the menu that appears). For example, the HTML source code for the www.google.com home page has the following line of HTML in it:

```
<input type=submit value="Google Search" name=btnG>
```

The "type" attribute with a value equal to "submit" tells us that it is a button, the caption or value that you see on the button is "Google Search", and the name of the button is "btnG". AUTOnomyV also provides a more user friendly way of obtaining pertinent object attributes. In the AUTOnomyV\Utilities\ folder, there is a utility script named "GetElementAttributes.vbs" that may be used for obtaining object attributes of open Internet pages (refer to section 8.1).

4.2 Developing the Scripts

There are two approaches for developing an AUTOnomyV test:

- Manual Script Creation
- Record & Playback Script Generation

4.2.1 APPROACH 1 – MANUALLY CREATE THE SCRIPT

The steps for manually creating the automated script are as follows:

1. Open your text editor.
2. Name your test file with a **.vbs** (VBScript) extension.
3. Provide your new test file with access to the AUTOnomyV tool by loading the AUTOnomyV library.
4. Use AUTOnomyV syntax to invoke the application.
5. Use AUTOnomyV syntax to automate actions in the application.
6. Use AUTOnomyV syntax to add test verification.
7. Execute and verify the test script results (add wait times and other statements to make the script more robust).

Note: Remember, you can use standard VBScript syntax in conjunction with AUTOnomyV syntax to enhance your automated test scripts.

4.2.2 APPROACH 2 – USE THE RECORD & PLAYBACK FEATURE

1. Execute the C:\AUTOnomyV\Utilities\record.vbs script, and follow the prompts (be sure to name the test script with a **.vbs** (VBScript) extension.
2. While the script is recording, manually execute the test steps in the application.
3. Stop recording, then modify the generated test script wherever necessary to ensure the generated test matches your test design.
4. Use AUTOnomyV syntax to add test verification.
5. Execute and verify the test script results (add wait times and other statements to make the script more robust).

Note: Remember, you can use standard VBScript syntax in conjunction with AUTOnomyV syntax to enhance your automated test scripts.

5.0 AUTOnomyV Syntax

This section provides information on AUTOnomyV syntax. To illustrate the use of the AUTOnomyV syntax, most statements are accompanied, wherever appropriate, by helpful remarks, examples, a graphic of the Internet object that the statement may be performed on, along with the object in the form of HTML code (as it may be seen when viewing the source of the Internet page).

5.1 Loading the AUTOnomyV Library

To use the AUTOnomyV tool, be sure you first enter the following in your test script:

```
Dim fso, f, file, sStrm
file = "C:\AUTOnomyV\AutonomyLib.vbs"
Set fso = CreateObject("Scripting.FileSystemObject")
Set f = fso.OpenTextFile(file, 1)
sStrm = f.ReadAll
executeGlobal(sStrm)
```

This loads the AUTOnomyV library, allowing your test script to use the tool's functionality.

5.2 Invoking an Internet Explorer Application

To invoke an Internet Explorer application and navigate to your application use the following syntax:

5.2.1 SYNTAX

InvokeIE *URL*, *waitTime*

5.2.2 VARIABLES

URL	Required. The URL of the Internet Application Under Test.
waitTime	Required. Sets the maximum amount of time in seconds (in addition to the default time set by the AUTOnomyV library of about 10 seconds) that the script will wait for the browser to completely load the web page prior to performing any subsequent actions. If the page completely loads in less time than specified by this parameter, the script will move on before the specified amount of time passes.

5.2.3 ADDITIONAL REMARKS

None

5.2.4 EXAMPLES

In the following example, the script will invoke the Internet Explorer browser, and navigate to the www.dijohn-ic.com website, and wait up to 5 seconds (plus AUTOnomyV's default sync time) for the initial www.dijohn-ic.com page to load prior to moving on to any subsequent statements.

```
InvokeIE "www.dijohn-ic.com", 5
```

5.3 Performing Actions on Web Page Objects

AUTOnomyV supports actions on the following types of browser objects:

- WebEdits
- WebTextAreas
- WebButtons
- WebCheckButtons
- WebRadioButtons
- WebLinks
- WebImages
- WebLists

Each of these object types are defined in the following sections, and provided with AUTOnomyV syntax that allow these objects to be automated.

You will find that the AUTOnomyV syntax statements have a standard structure, based on the hierarchy of web pages. To successfully automate objects in a webpage, you must first reference the window, then the element, followed by the element's property or method that you'd like to manipulate. Based on this hierarchical structure, VBScript statements and AUTOnomyV statements have a structure that resembles the following statement:

```
Window.Element.Property/Method
```

The above statement is not actual syntax, but it reveals the basic structure of most statements you'll be using to automate your applications. The first part of the statement, which reads 'Window', is explained in section 5.4. The rest of the statement, which involves actions on web objects, is explained in later sections.

5.4 Window

Many browser windows may be open at a single point in time, and each window has an integer that represents its ordinal position based on the order in which the window was opened. So to access an Internet Explorer window you must specify which window you wish to access, and two attributes allow you to do that: window title, and the window's ordinal index.

5.4.1 SYNTAX

Window(*title*, *index, syncTime*).*element.property/method*

5.4.2 VARIABLES

title	The title of the Internet Explorer window. Not required, but to indicate that you don't wish to use a title, use quotation marks that enclose nothing.
index	Required. Integer that specifies the Internet Explorer Window to access. If the title parameter is used, this index will access the open Internet Explorer window with the title specified in the *title* variable and at the given ordinal position, where the first window that has a title equal to the value in the *title* variable has an index of 0, the

	second has an index of 1, and so on. If the title parameter is **not** used, this index will access the open Internet Explorer window with any title at the given ordinal position, where the first window (with any title) has an index of 0, the second has an index of 1, and so on.
syncTime	Required. Integer that specifies the maximum amount of time (in addition to the default time set by the AUTOnomyV library of about 10 seconds) to wait for the page to load before moving on and attempting to perform an action in that window. If the page loads in less time than is specified by this parameter, it will move on without waiting the full amount of time.

5.4.3 ADDITIONAL REMARKS

You can choose to ignore the title attribute even if the Internet Explorer window has a title.

5.4.4 EXAMPLES

Example 1:

In the following example, the script will access the first open browser window (index of 0) with a title of "Welcome". It then clicks on the second "read more" link it finds on the Welcome screen (this will be discussed in greater detail later). The script will wait up-to 5 seconds (plus AUTOnomyV's default sync time) for the "Welcome" screen to load before attempting to click the link.

```
Window("Welcome",0,5).all(WebLink("INNERTEXT", "read more",1)).click
```

The title may be found at the top of the browser window or in the HTML source code of the webpage in-between the title tags as illustrated below.

<title> **Welcome** </title>

Example 2:

In the following example, the script will access the *second* open browser window (index of 1) with a title of "Welcome" (there must be more than one open browser windows with a title of "Welcome"). It then clicks on the second "read more" link it finds on the Welcome screen (this will be discussed in greater detail later). The script will wait up-to 0 seconds for the "Welcome" screen to load before attempting to click the link.

```
Window("Welcome",1,0).all(WebLink("INNERTEXT", "read more",1)).click
```

Example 3:

In the final example, the script will access the *third* open browser window (index of 2) regardless of what the title is, and whether or not the window has a title at all (there must be at least three open browser windows). It then clicks on the second "read more" link it finds on the Welcome screen (this will be discussed in greater detail later). The script will wait up-to 10 seconds for the "Welcome" screen to load before attempting to click the link.

```
Window("",2,10).all(WebLink("INNERTEXT", "read more",1)).click
```

5.5 Frames

Many Internet applications use frames, which essentially allow for multiple Internet pages to exist inside of a single browser window. If you are having trouble accessing objects on a web page, be sure to check to see if the application is using frames. To find out, view the HTML source code of the page.

Frames may be defined in the HTML source code as follows:

```
<frameset cols="20%,80%">
<frame src="preview.html" name="preview" >
<frame src="main.html" name="main" >
</frameset>
```

In the HTML above, we have a preview frame on the left that takes up 20% of the window, and a main frame to the right that takes up the remaining 80% of the window. Sometime frames are easy to spot, but sometimes it looks as though the Internet browser only contains a single window.

In addition to viewing the source, you can use the AUTOnomyV utility called "GetElementAttributes.vbs" to determine whether or not the window has frames. See the Utilities section for more information on this utility's use.

AUTOnomyV allows access to frame objects by using the following syntax.

5.5.1 SYNTAX

window.frames(name(*frameName*,*index*)).document.*element.property/method*

5.5.2 VARIABLES

frameName	The name of the frame. Not required, but to indicate that you don't wish to use a title, use quotation marks that enclose nothing.
index	Required. Integer that specifies the frame to access. If the frameName parameter is used, this index will access the frame with the name specified in the *frameName* variable and at the given ordinal position, where the first frame that has a name equal to the value in the *frameName* variable has an index of 0, the second has an index of 1, and so on. If the frameName parameter is **not** used, this index will access the frame with any name at the given ordinal position, where the first frame (with any title) has an index of 0, the second has an index of 1, and so on.

5.5.3 ADDITIONAL REMARKS

After referencing a frame, be sure to include the word "document" or the frame won't be successfully accessed.

Frames can also have nested frames inside of them. To access such a frame, use the following syntax:

window.frames(name(*frameName*,*index*)).document.frames(name(*frameName*,*index*)).document.*element. property/method*

5.5.4 EXAMPLES

Example 1:

In the following example, the script will access the first open browser window (index of 0) with a title of "Welcome", and the frame inside of that Window named "preview". It then clicks on the second "read more" link it finds on the screen (this will be discussed in greater detail later). The script will wait up-to 5 seconds for the "Welcome" screen to load before attempting to click the link.

```
Window("Welcome",0,5).frames(name("preview",0)).all(WebLink("INNERTEXT", "read more",1)).click
```

5.6 CloseWindow

To close a window, use the following syntax.

5.6.1 SYNTAX

CloseWindow *title*, *index*, *syncTime*

Variables

title	The title of the Internet Explorer window. Not required, but to indicate that you don't wish to use a title, use quotation marks that enclose nothing.
index	Required. Integer that specifies the Internet Explorer Window to access. If the title parameter is used, this index will access the open Internet Explorer window with the title specified in the *title* variable and at the given ordinal position, where the first window that has a title equal to the value in the *title* variable has an index of 0, the second has an index of 1, and so on. If the title parameter is **not** used, this index will access the open Internet Explorer window with any title at the given ordinal position, where the first window (with any title) has an index of 0, the second has an index of 1, and so on.
syncTime	Required. Integer that specifies the maximum amount of time to wait for the page to load before moving on to the next line in the script. If the page loads in less time than is specified by this parameter, it will move on to the next line in the script without waiting the full amount of time.

5.6.2 ADDITIONAL REMARKS

You can choose to ignore the title attribute even if the Internet Explorer window has a title.

Also, note that no parentheses are used to enclose the variables. Using parenthesis may cause a VBScript error.

5.6.3 EXAMPLES

Example 1:

In the following example, the script will close the first open browser window (index of 0) with a title of "Welcome". The script will wait up-to 5 seconds for the "Welcome" screen to load before attempting to close the window.

```
CloseWindow "Welcome",0,5
```

The title may be found at the top of the browser window or in the HTML source code of the webpage in-between the title tags as illustrated below.

<title> **Welcome** </title>

5.7 Adding Wait Times

Wait times cause the script to wait a specific amount of time before proceeding to the next line. These are especially good to use when the sync times in the Window statement don't seem to be effective.

5.7.1 SYNTAX

wscript.sleep *timeAmount*

5.7.2 VARIABLES

timeAmount	The amount of time in milliseconds that the script will wait before proceeding to the next line.

5.7.3 ADDITIONAL REMARKS

This is different from the sync times in the Window statement, because the sleep time will cause the script to wait the entire time specified, not just the amount of time it takes a window to load.

5.7.4 EXAMPLES

In the following example, the script will access the first open browser window (index of 0) with a title of "Welcome". It then clicks on the second "read more" link it finds on the Welcome screen (this will be discussed in greater detail later). The script will wait up-to 5 seconds for the "Welcome" screen to load before attempting to click the link. The script then waits 2 seconds (2000 milliseconds) before moving on to the next line in the script, which accesses the Information screen and clicks on the first "Return" link.

```
Window("Welcome",0,5).all(WebLink("INNERTEXT", "read more",1)).click
wscript.sleep 2000
Window("Information",0,5).all(WebLink("INNERTEXT", "Return",0)).click
```

5.8 WebEdit

A WebEdit is another name for a textbox. You can input a value into a textbox or get a value from a textbox. On screen, the textbox may appear as follows:

WebEdit Box: 

5.8.1 SYNTAX

window.all(WebEdit(*type*, *attribute*, *attValue*, *index*)).value = *inputValue*

- Or -

outputVar = *window*.WebEdit(*type*, *attribute*, *attValue*, *index*).value

5.8.2 VARIABLES

type	Required. The type of textbox. There are two types of textboxes that AUTOnomyV supports: TEXT and PASSWORD.
attribute	The textbox attribute that will be used to identify the textbox. Three different attributes may be used in this parameter: NAME, ID, and

	VALUE. To not specify an attribute, use quotations that enclose nothing.
attValue	The value of the attribute specified in the above parameter. To not specify an attribute value, use quotations that enclose nothing. If no attribute is identified in the previous parameter, this parameter is ignored.
index	Required. Integer that specifies the textbox to access. If the attribute parameter is used, this index will access the textbox with the type specified in the *type* variable and with the attribute value specified in the *attValue* variable and at the given ordinal position, where the first textbox with the given *type* and *attValue* variable has an index of 0, the second has an index of 1, and so on. If the attribute variable is **not** used, this index will just use the type parameter to access the textbox with the type specified in the *type* variable (and with any attribute values) at the given ordinal position, where the first textbox has an index of 0, the second has an index of 1, and so on.
inputValue	The value that you want entered into the textbox. Note: to remove a value from the textbox, use quotations that enclose nothing.
outputVar	The variable that you'd like to use for capturing the value already in the textbox.

5.8.3 ADDITIONAL REMARKS

It is not recommended that you use VALUE in the attribute variable, because the property will change whenever the text in the textbox changes.

5.8.4 EXAMPLES

Example 1: Name Property

In the following example, the script will access the first open browser window (index of 0) with a title of "Login". It then inputs the word "Tony" into the first textbox (index of 0) with a type of "TEXT", and a "Name" property that is equal to "UserName". The script will wait up-to 5 seconds for the "Welcome" screen to load before attempting to input the value.

```
Window("Login",0,5).all(WebEdit("TEXT", "NAME", "UserName",0)).value = "Tony"
```

This textbox may appear in the HTML source code as follows:

```
<input type="text" name="UserName" id="UserID">
```

Example 2: ID Property

In the following example, the script will access the first open browser window (index of 0) with a title of "Login". It then inputs the word "Tony" into the first textbox (index of 0) with a type of "TEXT", and an "ID" property that is equal to "UserID". The script will wait up-to 5 seconds for the "Welcome" screen to load before attempting to input the value.

```
Window("Login",0,5).all(WebEdit("TEXT", "ID", "UserID",0)).value = "Tony"
```

This textbox may appear in the HTML source code as follows:

```
<input type="text" id="UserID">
```

Example 3:

In the following example, the script will access the first open browser window (index of 0) with a title of "Login". It then inputs the word "Secret" into the first textbox (index of 0) with a type of "Password", and an "ID" property that is equal to "UserPass". The script will wait up-to 5 seconds for the "Welcome" screen to load before attempting to input the value.

```
Window("Login",0,5).all(WebEdit("PASSWORD", "ID", "UserPass",0)).value = "Secret"
```

This textbox may appear in the HTML source code as follows:

```
<input type="password" id="UserName">
```

Example 4:

In the following example, the script will access the first open browser window (index of 0) with a title of "Login". It then inputs the word "Secret" into the first textbox (index of 0) with a type of "Password". The script will wait up-to 5 seconds for the "Welcome" screen to load before attempting to input the value.

```
Window("Login",0,5).all(WebEdit("PASSWORD", "", "",0)).value = "Secret"
```

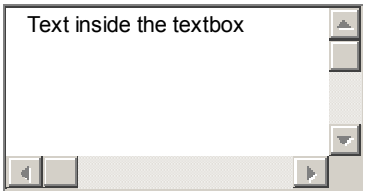
This textbox may appear in the HTML source code as follows:

```
<input type="password" id="UserName">
```

5.9 WebTextArea

You can input a value into a WebTextArea or get a value from a WebTextArea. On screen, the WebTextArea may appear as follows:

Enter Description in WebTextArea:



5.9.1 SYNTAX

window.all(WebTextArea(*attribute*, *attValue*, *index*)).value = *inputValue*

- Or -

outputVar = *window*.WebTextArea(*attribute*, *attValue*, *index*).value

5.9.2 VARIABLES

attribute	The WebTextArea attribute that will be used to identify the textbox.

	Three different attributes may be used in this parameter: NAME, ID, and VALUE. To not specify an attribute, use quotations that enclose nothing.
attValue	The value of the attribute specified in the above parameter. To not specify an attribute value, use quotations that enclose nothing. If no attribute is identified in the above parameter, this parameter is ignored.
index	Required. Integer that specifies the textbox to access. If the attribute parameter is used, this index will access the WebTextArea with the attribute value specified in the *attValue* variable and at the given ordinal position, where the first WebTextArea with the *attValue* variable has an index of 0, the second has an index of 1, and so on. If the attribute variable is **not** used, this index will just use the type parameter to access the WebTextArea (with any attribute values) at the given ordinal position, where the first WebTextArea has an index of 0, the second has an index of 1, and so on.
inputValue	The value that you want entered into the WebTextArea. Note: to remove a value from the textbox, use quotations that enclose nothing.
outputVar	The variable that you'd like to use for capturing the value already in the WebTextArea.

5.9.3 ADDITIONAL REMARKS

It is not recommended that you use VALUE in the attribute variable, because the property will change whenever the text in the WebTextArea changes.

5.9.4 EXAMPLES

Example 1: NAME Property

In the following example, the script will access the first open browser window (index of 0) with a title of "Home". It then inputs the words "Text Description" into the first WebTextArea (index of 0) with a "Name" property that is equal to "txtArea". The script will wait up-to 5 seconds for the "Home" screen to load before attempting to input the value.

```
Window("Home",0,5).all(WebTextArea("NAME", "txtArea",0)).value = "Text Description"
```

This textbox may appear in the HTML source code as follows:

```
<textarea name="txtArea">Text Description</textarea>
```

Example 2: ID Property

In the following example, the script will access the first open browser window (index of 0) with a title of "Home". It then inputs the words "Text Description" into the second WebTextArea (index of 1) with an "ID" property that is equal to "desc". The script will wait up-to 5 seconds for the "Home" screen to load before attempting to input the value.

```
Window("Home",0,5).all(WebTextArea("ID", "desc",1)).value = "Text Description"
```

This textbox may appear in the HTML source code as follows:

```
<textarea id="UserName">Text Description</textarea>
```

Example 3: No Property (Index Only)

In the following example, the script will access the first open browser window (index of 0) with a title of "Home". It then inputs the words "Text Description" into the first WebTextArea (index of 0) on the page. The script will wait up-to 5 seconds for the "Home" screen to load before attempting to input the value.

```
Window("Home",0,5).all(WebTextArea("", "",0)).value = "Text Description"
```

This textbox may appear in the HTML source code as follows:

```
<textarea name= "UserName" id="UserID">Text Description</textarea>
```

5.10 WebButton

A WebButton is an element that you click in an effort to have some actions executed by the application. On screen, the WebButton may appear as follows:

5.10.1 SYNTAX

window.all(WebButton(type, attribute, attValue, index)).click

5.10.2 VARIABLES

type	Required. The type of WebButton. There are three types of WebButtons that AUTOnomyV supports: BUTTON, SUBMIT and RESET.
attribute	The attribute that will be used to identify the WebButton. Three different attributes may be used in this parameter: NAME, ID, and VALUE. To not specify an attribute, use quotations that enclose nothing.
attValue	The value of the attribute specified in the above parameter. To not specify an attribute value, use quotations that enclose nothing. If no attribute is identified in the previous parameter, this parameter is ignored.
index	Required. Integer that specifies the WebButton to access. If the attribute parameter is used, this index will access the WebButton with the type specified in the *type* variable and with the attribute value specified in the *attValue* variable and at the given ordinal position, where the first textbox with the given *type* and *attValue* variable has an index of 0, the second has an index of 1, and so on. If the attribute variable is **not** used, this index will just use the type parameter to access the WebButton with the type specified in the *type* variable (and with any attribute values) at the given ordinal position, where the first textbox has an index of 0, the second has an index of 1, and so on.

5.10.3 ADDITIONAL REMARKS

The VALUE property for buttons is normally the text that is visible on the button, so typically you won't need to view the source code for this property.

5.10.4 EXAMPLES

Example 1: BUTTON Type and NAME Property

In the following example, the script will access the first open browser window (index of 0) with a title of "Orders". It then clicks on the first WebButton (index of 0) with a type of "BUTTON", and a "Name" property that is equal to "pOrder". The script will wait up-to 5 seconds for the "Orders" screen to load before attempting to input the value.

```
Window("Orders",0,5).all(WebEdit("BUTTON","NAME","pOrder",0)).click
```

This WebButton with a type of "BUTTON" may appear in the HTML source code with an "Input" tag or "Button" tag as shown below:

```
<input type="button" name="pOrder" value = "Click Me">
```

- Or -

```
<button type="button" name="pOrder" >Click Me</button>
```

Example 2: SUBMIT Type and ID Property

In the following example, the script will access the first open browser window (index of 0) with a title of "Orders". It then clicks on the first WebButton (index of 0) with a type of "SUBMIT", and an "ID" property that is equal to "OrderID". The script will wait up-to 5 seconds for the "Orders" screen to load before attempting to input the value.

```
Window("Orders",0,5).all(WebEdit("SUBMIT","ID","OrderID",0)).click
```

This WebButton with a type of "SUBMIT" may appear in the HTML source code with an "Input" tag or "Button" tag as shown below:

```
<input type="submit" id="OrderID" value = "Click Me">
```

- Or -

```
<button type="submit" id="OrderID" >Click Me</button>
```

Example 3: RESET Type and VALUE Property

In the following example, the script will access the first open browser window (index of 0) with a title of "Orders". It then clicks on the first WebButton (index of 0) with a type of "RESET", and a "VALUE" property that is equal to "Click Me". The script will wait up-to 5 seconds for the "Orders" screen to load before attempting to input the value.

```
Window("Orders",0,5).all(WebEdit("RESET","VALUE","Click Me",0)).click
```

This WebButton with a type of "RESET" may appear in the HTML source code with an "Input" tag or "Button" tag as shown below:

```
<input type="reset" id="OrderID" value = "Click Me">
```

- Or -

```
<button type="reset" id="OrderID" >Click Me</button>
```

Example 4: No Property (Index Only)

In the following example, the script will access the first open browser window (index of 0) with a title of "Orders". It then clicks on the first textbox (index of 0) with a type of "SUBMIT". The script will wait up-to 5 seconds for the "Welcome" screen to load before attempting to input the value.

```
Window("Orders",0,5).all(WebButton ("SUBMIT","","",0)).click
```

This WebButton may appear in the HTML source code with an "Input" tag or "Button" tag as shown below:

```
<input type="reset" id="OrderID" value = "Click Me">
```

- Or -

```
<button type="reset" id="OrderID" >Click Me</button>
```

5.11 WebCheckButton

The WebCheckButton is a Boolean element that can be checked on or off. On screen, the WebCheckButton may appear as follows:

☑ Check Option 1

5.11.1 SYNTAX

window.all(WebCheckButton(*attribute*, a*ttValue*, *index*)).click

5.11.2 VARIABLES

attribute	The WebCheckButton attribute that will be used to identify the WebCheckButton. Three different attributes may be used in this parameter: NAME, ID, and VALUE. To not specify an attribute, use quotations that enclose nothing.
attValue	The value of the attribute specified in the above parameter. To not specify an attribute value, use quotations that enclose nothing. If no attribute is identified in the above parameter, this parameter is ignored.
index	Required. Integer that specifies the textbox to access. If the attribute parameter is used, this index will access the WebCheckButton with the attribute value specified in the *attValue* variable and at the given ordinal position, where the first WebCheckButton with the *attValue* variable has an index of 0, the second has an index of 1, and so on. If the attribute variable is **not** used, this index will just use the type parameter to access the WebCheckButton (with any attribute values) at the given ordinal position, where the first WebCheckButton has an index of 0, the second has an index of 1, and so on.

5.11.3 ADDITIONAL REMARKS

None.

5.11.4 EXAMPLES

Example 1: NAME Property

In the following example, the script will access the first open browser window (index of 0) with a title of "Home". It then clicks the first WebCheckButton (index of 0) with a "Name" property that is equal to "option". The script will wait up-to 5 seconds for the "Home" screen to load before attempting to input the value.

```
Window("Home",0,5).all(WebCheckButton("NAME","option",0)).click
```

This WebCheckButton may appear in the HTML source code as follows:

```
<input type="checkbox" name="option"> Check Option 1
```

Example 2: ID Property

In the following example, the script will access the first open browser window (index of 0) with a title of "Home". It then clicks the third WebCheckButton (index of 2) with an "ID" property that is equal to "opt1". The script will wait up-to 5 seconds for the "Home" screen to load before attempting to input the value.

```
Window("Home",0,5).all(WebCheckButton("ID","opt1",2)).click
```

This WebCheckButton may appear in the HTML source code as follows:

```
<input type="checkbox" id="opt1"> Check Option 1
```

Example 3: VALUE Property

In the following example, the script will access the first open browser window (index of 0) with a title of "Home". It then clicks the first WebCheckButton (index of 0) with a "VALUE" property that is equal to "firstOption". The script will wait up-to 5 seconds for the "Home" screen to load before attempting to input the value.

```
Window("Home",0,5).all(WebCheckButton("VALUE","firstOption",0)).click
```

This WebCheckButton may appear in the HTML source code as follows:

```
<input type="checkbox" id="option" value="firstOption"> Check Option 1
```

Example 4: No Property (Index Only)

In the following example, the script will access the first open browser window (index of 0) with a title of "Home". It then clicks the second WebCheckButton (index of 1) on the page. The script will wait up-to 5 seconds for the "Home" screen to load before attempting to input the value.

```
Window("Home",0,5).all(WebCheckButton("", "",1)).click
```

This WebCheckButton may appear in the HTML source code as follows:

```
<input type="checkbox" id="option" value="firstOption"> Check Option 1
```

5.12 WebRadioButton

The WebRadioButton is a Boolean element that allows users to select one option from a set of options. On screen, the WebRadioButton may appear as follows:

 Radio Option 1

5.12.1 SYNTAX

window.all(WebRadioButton(*attribute*, a*ttValue*, *index*)).click

5.12.2 VARIABLES

attribute	The WebRadioButton attribute that will be used to identify the WebRadioButton. Three different attributes may be used in this parameter: NAME, ID, and VALUE. To not specify an attribute, use quotations that enclose nothing.
attValue	The value of the attribute specified in the above parameter. To not specify an attribute value, use quotations that enclose nothing. If no attribute is identified in the above parameter, this parameter is ignored.
index	Required. Integer that specifies the textbox to access. If the attribute parameter is used, this index will access the WebRadioButton with the attribute value specified in the *attValue* variable and at the given ordinal position, where the first WebRadioButton with the *attValue* variable has an index of 0, the second has an index of 1, and so on. If the attribute variable is **not** used, this index will just use the type parameter to access the WebRadioButton (with any attribute values) at the given ordinal position, where the first WebRadioButton has an index of 0, the second has an index of 1, and so on.

5.12.3 ADDITIONAL REMARKS

None.

5.12.4 EXAMPLES

Example 1: NAME Property

In the following example, the script will access the first open browser window (index of 0) with a title of "Home". It then clicks the first WebRadioButton (index of 0) with a "Name" property that is equal to "option". The script will wait up-to 5 seconds for the "Home" screen to load before attempting to input the value.

```
Window("Home",0,5).all(WebRadioButton("NAME","option",0)).click
```

This WebRadioButton may appear in the HTML source code as follows:

```
<input type="radio" name="option"> Check Option 1
```

Example 2: ID Property

In the following example, the script will access the first open browser window (index of 0) with a title of "Home". It then clicks the third WebRadioButton (index of 2) with an "ID" property that is equal to "opt1". The script will wait up-to 5 seconds for the "Home" screen to load before attempting to input the value.

```
Window("Home",0,5).all(WebRadioButton("ID","opt1",2)).click
```

This WebRadioButton may appear in the HTML source code as follows:

```
<input type="radio" id="opt1"> Check Option 1
```

Example 3: VALUE Property

In the following example, the script will access the first open browser window (index of 0) with a title of "Home". It then clicks the first WebRadioButton (index of 0) with a "VALUE" property that is equal to "firstOption". The script will wait up-to 5 seconds for the "Home" screen to load before attempting to input the value.

```
Window("Home",0,5).all(WebRadioButton("VALUE","firstOption",0)).click
```

This WebRadioButton may appear in the HTML source code as follows:

```
<input type="radio" id="option" value="firstOption"> Check Option 1
```

Example 4: No Property (Index Only)

In the following example, the script will access the first open browser window (index of 0) with a title of "Home". It then clicks the second WebRadioButton (index of 1) on the page. The script will wait up-to 5 seconds for the "Home" screen to load before attempting to input the value.

```
Window("Home",0,5).all(WebRadioButton("", "",1)).click
```

This WebRadioButton may appear in the HTML source code as follows:

```
<input type="radio" id="option" value="firstOption"> Check Option 1
```

5.13 WebLink

A WebLink is an element that allows users navigate to internet content. On screen, the WebRadioButton may appear as follows:

Click To Enter DiJohn IC Site

5.13.1 SYNTAX

window.all(WebLink(*attribute*, *attValue*, *index*)).click

5.13.2 VARIABLES

attribute	The WebLink attribute that will be used to identify the WebLink. Two different attributes may be used in this parameter: INNERTEXT and HREF. To not specify an attribute, use quotations that enclose nothing.
attValue	The value of the attribute specified in the above parameter. To not specify an attribute value, use quotations that enclose nothing. If no

	attribute is identified in the above parameter, this parameter is ignored.
index	Required. Integer that specifies the textbox to access. If the attribute parameter is used, this index will access the WebLink with the attribute value specified in the *attValue* variable and at the given ordinal position, where the first WebLink with the *attValue* variable has an index of 0, the second has an index of 1, and so on. If the attribute variable is **not** used, this index will just use the type parameter to access the WebLink (with any attribute values) at the given ordinal position, where the first WebLink has an index of 0, the second has an index of 1, and so on.

5.13.3 ADDITIONAL REMARKS

The INNERTEXT property is normally the text of a link as it is shown on the screen, so typically you won't need to view the source code for this property.

The HREF property is the URL that the links navigates to.

5.13.4 EXAMPLES

Example 1: INNERTEXT Property

In the following example, the script will access the first open browser window (index of 0) with a title of "Home". It then clicks the first WebLink (index of 0) with an "INNERTEXT" property that is equal to "Click to Enter DiJohn IC Website". The script will wait up-to 5 seconds for the "Home" screen to load before attempting to input the value.

```
Window("Home",0,5).all(WebRadioButton("INNERTEXT","Click To Enter DiJohn IC Website",0)).click
```

This WebLink may appear in the HTML source code with an 'A' tag as follows:

```
<a href="www.dijohn-ic.html">Click To Enter DiJohn IC Website</A>
```

Example 2: ID Property

In the following example, the script will access the first open browser window (index of 0) with a title of "Home". It then clicks the first WebLink (index of 0) with an "HREF" property that is equal to "Click to Enter Site". The script will wait up-to 5 seconds for the "Home" screen to load before attempting to input the value.

```
Window("Home",0,5).all(WebRadioButton("HREF","www.dijohn-ic.com",0)).click
```

This WebLink may appear in the HTML source code with an 'A' tag as follows:

```
<a href="www.dijohn-ic.html">Click To Enter DiJohn IC Website</A>
```

Example 3: No Property (Index Only)

In the following example, the script will access the first open browser window (index of 0) with a title of "Home". It then clicks the second WebLink (index of 1) on the page. The script will wait up-to 5 seconds for the "Home" screen to load before attempting to input the value.

```
Window("Home",0,5).all(WebRadioButton("", "",1)).click
```

This WebLink may appear in the HTML source code with an 'A' tag as follows:

```
<a href="www.dijohn-ic.html">Click To Enter DiJohn IC Website</A>
```

5.14 WebImage

A WebImage is an element that defines an image on the screen. On screen, the WebImage may appear as a picture that when clicked functions as a link or button.

5.14.1 SYNTAX

window.all(WebImage(*attribute*, *attValue*, *index*)).click

5.14.2 VARIABLES

attribute	The WebImage attribute that will be used to identify the WebImage. Three different attributes may be used in this parameter: NAME, ID, and VALUE. To not specify an attribute, use quotations that enclose nothing.
attValue	The value of the attribute specified in the above parameter. To not specify an attribute value, use quotations that enclose nothing. If no attribute is identified in the above parameter, this parameter is ignored.
index	Required. Integer that specifies the textbox to access. If the attribute parameter is used, this index will access the WebImage with the attribute value specified in the *attValue* variable and at the given ordinal position, where the first WebImage with the *attValue* variable has an index of 0, the second has an index of 1, and so on. If the attribute variable is **not** used, this index will just use the type parameter to access the WebImage (with any attribute values) at the given ordinal position, where the first WebImage has an index of 0, the second has an index of 1, and so on.

5.14.3 ADDITIONAL REMARKS

None.

5.14.4 EXAMPLES

Example 1: NAME Property

In the following example, the script will access the first open browser window (index of 0) with a title of "Home". It then clicks the first WebImage (index of 0) with a "Name" property that is equal to "Picture". The script will wait up-to 5 seconds for the "Home" screen to load before attempting to input the value.

```
Window("Home",0,5).all(WebImage("NAME","Picture",0)).click
```

This WebImage may appear in the HTML source code with an 'INPUT tag or 'IMG' tag as shown below:

```
<img src="face.gif" name="Picture" />
```

- Or -

```
<input type="image" src="face.gif" name="Picture" />
```

Example 2: ID Property

In the following example, the script will access the first open browser window (index of 0) with a title of "Home". It then clicks the first WebImage (index of 0) with an "ID" property that is equal to "Picture". The script will wait up-to 5 seconds for the "Home" screen to load before attempting to input the value.

```
Window("Home",0,5).all(WebImage("ID","pic1",0)).click
```

This WebImage may appear in the HTML source code with an 'INPUT tag or 'IMG' tag as shown below:

```
<img src="face.gif" id="pic1" />
```

- Or -

```
<input type="image" src="face.gif" id="pic1" />
```

Example 3: SRC Property

In the following example, the script will access the first open browser window (index of 0) with a title of "Home". It then clicks the first WebImage (index of 0) with a "SRC" property that is equal to "face.gif". The script will wait up-to 5 seconds for the "Home" screen to load before attempting to input the value.

```
Window("Home",0,5).all(WebImage("SRC","face.gif",0)).click
```

This WebImage may appear in the HTML source code with an 'INPUT tag or 'IMG' tag as shown below:

```
<img src="face.gif" name="Picture" />
```

- Or -

```
<input type="image" src="face.gif" name="Picture" />
```

Example 4: No Property (Index Only)

In the following example, the script will access the first open browser window (index of 0) with a title of "Home". It then clicks the fourth WebImage (index of 3) on the page. The script will wait up-to 5 seconds for the "Home" screen to load before attempting to input the value.

```
Window("Home",0,5).all(WebImage("", "",3)).click
```

This WebImage may appear in the HTML source code with an 'INPUT tag or 'IMG' tag as shown below:

```
<img src="face.gif" name="Picture" />
```

- Or -

```
<input type="image" src="face.gif" name="Picture" />
```

5.15 WebList

You can select a value from a list of items provided in a WebList or get index of the value that is currently selected. On screen, the WebList may appear as follows:

Selection: 

5.15.1 SYNTAX

window.all(WebList(*attribute*, *attValue*, *index*)).selectedIndex = *intSelection*

- Or -

outputVar = *window*.all(WebList(*attribute*, a*ttValue*, *index*)).selectedIndex

5.15.2 VARIABLES

attribute	The WebList attribute that will be used to identify the textbox. Two different attributes may be used in this parameter: NAME and ID. To not specify an attribute, use quotations that enclose nothing.
attValue	The value of the attribute specified in the above parameter. To not specify an attribute value, use quotations that enclose nothing. If no attribute is identified in the above parameter, this parameter is ignored.
index	Required. Integer that specifies the textbox to access. If the attribute parameter is used, this index will access the WebList with the attribute value specified in the *attValue* variable and at the given ordinal position, where the first WebList with the *attValue* variable has an index of 0, the second has an index of 1, and so on. If the attribute variable is **not** used, this index will just use the type parameter to access the WebList (with any attribute values) at the given ordinal position, where the first WebList has an index of 0, the second has an index of 1, and so on.
intSelection	The index of the option that you want selected in the WebList. Options in a select object are indexed in the order in which they are defined, starting with an index of 0. Note: to remove a value from the textbox, use quotations that enclose nothing.
outputVar	The variable that you'd like to use for capturing the value already in the WebTextArea.

5.15.3 ADDITIONAL REMARKS

None.

5.15.4 EXAMPLES

Example 1: NAME Property

In the following example, the script will access the first open browser window (index of 0) with a title of "Book List". It then selects the 3rd option (selected index of 2) in the second WebList (index of 1) with a "Name" property that is equal to "books". The script will wait up-to 5 seconds for the "Book List" screen to load before attempting to select the option.

```
Window("Book List",0,5).all(WebTextArea("NAME","books",1)).selectedIndex = 2
```

This WebList may appear in the HTML source code as follows (in which "A Tail of Two Cities" would be the selected option:

```
<select name = "books">
  <option >Pride and Prejudice</option>
  <option >A Tail of Two Cities</option>
  <option >Song of Solomon</option>
</select>
```

Example 2: ID Property

In the following example, the script will access the first open browser window (index of 0) with a title of "Book List". It then selects the 3rd option (selected index of 2) in the second WebList (index of 1) with an "ID" property that is equal to "bkList". The script will wait up-to 5 seconds for the "Book List" screen to load before attempting to select the option.

```
Window("Book List",0,5).all(WebTextArea("NAME","bkList",1)).selectedIndex = 2
```

This WebList may appear in the HTML source code as follows (in which "A Tail of Two Cities" would be the selected option:

```
<select id = "bkList">
  <option >Pride and Prejudice</option>
  <option >A Tail of Two Cities</option>
  <option >Song of Solomon</option>
</select>
```

Example 3: No Property (Index Only)

In the following example, the script will access the first open browser window (index of 0) with a title of "Book List". It then selects the 3rd option (selected index of 2) in the fourth WebList (index of 3) on the page. The script will wait up-to 5 seconds for the "Book List" screen to load before attempting to select the option.

```
Window("Book List",0,5).all(WebTextArea("", "",3)).value = "Text Description"
```

This WebList may appear in the HTML source code as follows (in which "A Tail of Two Cities" would be the selected option:

```
<select id = "bkList">
  <option >Pride and Prejudice</option>
  <option >A Tail of Two Cities</option>
  <option >Song of Solomon</option>
</select>
```

6.0 Test Verification

Just because a script successfully runs without error, doesn't necessarily mean the test associated with that script passed. To create a valid test case, we need to use some scripting code to verify the test accomplished all expected results. Using AUTOnomyV in conjunction with VBScript/Document Object Model syntax (see http://msdn.microsoft.com/ for more information on the Document Object Model) you

can create the necessary checks to verify a test. For example, you may have a test that needs to verify a WebEdit has a certain default value. This may be accomplished as follows:

```
output = Window("Login",0,5).all(WebEdit("TEXT","NAME","UserName",0)).value
If output = “Tony” then
        wscript.echo “Login Default Test Passed”
Else
        wscript.echo “Login Default Test Failed”
End If
```

This example does the following:

1. Assign the value inside of the “UserName” textbox to the variable called “output”
2. If the textbox contains “Tony” a message is printed that reads, “Login Default Test Passed”
3. If the textbox doesn’t contain “Tony” a message is printed that reads, “Login Default Test Failed”

Note: running a test from the command line will cause wscript.echo to print messages to the DOS window. Running the test by double clicking on the script icon will cause wscript.echo to print messages in a pop-up message box.

7.0 Batch Execution

There are many ways to accomplish batch script execution, but the recommended approach is to use a Windows Batch file. Batch files are files that allow MS-DOS and Microsoft Windows users to create a list of command-line statements to run upon execution of the batch file.

7.1 Creating a Batch File

You can use the same editor that you use for creating your .vbs files, except you will need to save the batch file with a .bat extension.

7.2 Calling Scripts from a Batch File

Section 2.1.2 revealed the syntax for executing a script from the command line. The same syntax is used within the batch file: cscript <filename>. An example is as follows:

cscript C:\AUTOnomyV\Samples\Sample1_VerifyDefaults.vbs

This statement placed inside of the batch script will execute the Sample1_VerifyDefaults.vbs script, as if executed from the command line. Refer to the Samples folder for an actual batch file that is used for batch script execution.

8.0 Utilities

Two utilities offered as part of the AUTOnomyV framework include:

- GetElementAttributes.vbs
- record.vbs

8.1 GetElementAttributes Utility

This utility prints all open internet explorer windows, frames, elements, and attributes to a file you specify. If you file you specify already exists, the data will be appended to the file. Before executing the script, close all Windows Explorer windows. Open Windows Explorer windows may thwart results.

8.2 Record Utility

This utility allows users to record actions in an Internet Explorer window. Constraints of this utility include:

- You can only record on an Internet Explorer window opened by the script. Actions in other Internet Explorer windows will be ignored.
- If an action in the Internet Explorer window causes a new window to open, no actions will be recorded in the new window. You will need to manually add AUTOnomyV syntax after the recording is completed to handle the new window.
- Every time you start a new recording, the syntax for loading the AUTOnomyV library is added to the script. If you are recording to an existing script, this syntax will need to be deleted manually before you can execute the recorded test script.

9.0 AUTOnomyV Constraints

9.1 Popup Windows

JavaScript generated pop-ups boxes such as Security Alert windows are not contained within an Internet browser are not accessible the same way HTML pages are. Currently, AUTOnomyV will not support performing any automated actions on pop-up windows.

10.0 Trouble Shooting

10.1 Active Content Blocked

If you are running on Windows XP with Service Pack 2, you may not be able to run the tests unless you enable active content. This is because if the website you are testing has active content, the IE browser will "block" the content. To enable active content perform the following:

1. From the browser, select Tools > Internet Options
2. Select the Advanced tab
3. Under Security, check "Allow active content to run in files on My Computer"
4. Click OK

10.2 Window Not Found Error

The Window Not Found Error may be caused by one of several issues including the following:

1. The title in the Window statement is incorrect. You may want to double check the HTML source code to ensure you have the correct title.
2. The page hasn't finished loading. Sync problems are often the root cause of this error. You may want to increase the syncTime variable in the Window statement.
3. The application has a defect, and didn't load the correct screen

10.3 Element Not Found Error

The Element Not Found Error (WebEdit error, WebImage error, etc.) may be caused by one of several issues including the following:

1. The syntax for the statement is incorrect.
2. The attributes used for identifying an object are incorrect.
3. The page hasn't finished loading. Sync problems are often the root cause of this error. You may want to increase the syncTime variable in the Window statement.

4. The application has a defect, and didn't load the correct screen.

11.0 Future Enhancements

Be on the look out for the following future enhancements:

- Being able to record on already open browser windows
- Being able to select a value from a list box by it's option name as opposed to just its index
- AutonomyV syntax that simplifies data-driven testing.
- Increased documentation.

12.0 Questions, Comments, Concerns

If you have any questions, comments or concerns, please contact the AUTOnomyV development team at autonomyv@dijohn-ic.com.

If you are interested in training on Scripting Language Automation in general or specifically on AUTOnomyV, please contact DiJohn IC at services@dijohn-ic.com.