

# HP OpenView Internet Services

## Web Transaction Recorder Guide



**Manufacturing Part Number: J4511-90007**

**April 2005**

© Copyright 2004 - 2005 Hewlett-Packard Development Company, L.P.

## Legal Notices

### Warranty

*Hewlett-Packard makes no warranty of any kind with regard to this document, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. Hewlett-Packard shall not be held liable for errors contained herein or direct, indirect, special, incidental or consequential damages in connection with the furnishing, performance, or use of this material.*

A copy of the specific warranty terms applicable to your Hewlett-Packard product can be obtained from your local Sales and Service Office.

### Restricted Rights Legend

Use, duplication, or disclosure by the U.S. Government is subject to restrictions as set forth in subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause in DFARS 252.227-7013.

Hewlett-Packard Company  
United States of America

Rights for non-DOD U.S. Government Departments and Agencies are as set forth in FAR 52.227-19(c)(1,2).

### Copyright Notices

© Copyright 2004 - 2005 Hewlett-Packard Development Company, L.P.

No part of this document may be copied, reproduced, or translated into another language without the prior written consent of Hewlett-Packard Company. The information contained in this material is subject to change without notice.

### Trademark Notices

Java™ is a trademark of Sun Microsystems, Inc.

Microsoft Windows®, Windows NT®, MS Windows®, Windows 2000®, MS-DOS® and XP® are U.S. registered trademarks of Microsoft Corporation.

Netscape™ and Netscape Navigator™ are U.S. trademarks of Netscape Communications Corporation.

UNIX® is a registered trademark of The Open Group.

Adobe® and Acrobat® are registered trademarks of Adobe Inc.

All other product names are the property of their respective trademark or service mark holders and are hereby acknowledged.

## Support

Please visit the HP OpenView web site at:

<http://www.managementsoftware.hp.com/>

This web site provides contact information and details about the products, services, and support that HP OpenView offers.

You can also go directly to the support web site at:

<http://support.openview.hp.com/>

HP OpenView online software support provides customer self-solve capabilities. It provides a fast and efficient way to access interactive technical support tools needed to manage your business. As a valuable support customer, you can benefit by being able to:

- Search for knowledge documents of interest
- Submit and track progress on support cases
- Manage a support contract
- Look up HP support contacts
- Review information about available services
- Enter discussions with other software customers
- Research and register for software training

Most of the support areas require that you register as an HP Passport user and log in. Many also require a support contract.

To find more information about access levels, go to the following URL:

[http://support.openview.hp.com/access\\_level.jsp](http://support.openview.hp.com/access_level.jsp)

To register for an HP Passport ID, go to:

<https://passport.hp.com/hpp2/newuser.do>



<b>Chapter 1</b>	<b>OVIS Web Transaction Recorder</b>	<b>9</b>
	Getting Started	10
	Web Page Basics	10
	HTML	10
	JavaScript and Plug-In Examples	12
	Frames	13
	HTTP	13
	HTTPS and SSL	14
	Authentication and Digital Certificates	14
	Proxy Server	15
	Caching	15
	URL	15
	Cookies	16
	Web Transaction Recording Modes	19
	Quick Tips for Determining Which Mode to Use	21
	How the Web Transaction Recorder Works	23
	Recording	23
	IE Mode Recording	23
	URL Mode Recording	23
	Results of the Recording	24
	Simplified Example of Recording	24
	Playback	27
	URL Mode Playback	27
	IE Mode Playback	27
	How the HTTP_TRANS Probe Works	29

<b>Chapter 2</b>	<b>Using the Web Transaction Recorder</b>	<b>31</b>
	Configuring the HTTP_TRANS Probe	32
	Steps to Record a Web Transaction	34
	Basic Steps	35
	Specifying Global Transaction Properties	37
	Recording Options	39
	Playback Options	40
	Tracing Levels	42
	Proxy Settings	43
	Transaction Step Properties Basic	44
	Transaction Step Properties Advanced	46
	Advanced Usage	47
	Internet Explorer Security Dialog Boxes	48
	Dynamic URL Substitution	50
	External Scripts for Form Data Input (IE Mode)	55
	External Script for Availability Check (IE Mode)	56
	Dynamic Frame Names	57
	Changing the Log On Account for the Probe	58
	Cookie Settings	59
	Loading and Saving Session Cookies	59
	Authentication	59
	Browser Plug-Ins	59
	Complex Page Redirections	60
	Duplicate Input Variables	61
	Registry Settings that Influence Probe Results (IE mode)	62
	HTML Dialog and Pull-down Menu Internals	63
	Position Independent Navigation Points	72
	Transaction Script Reference	73
	IE Mode	73
	URL/Navigation Point Mode	89
	Web Transaction Recorder Tips	95
<b>Chapter 3</b>	<b>Troubleshooting</b>	<b>97</b>
	Recording and Playback Issues	98
	Recording Issues	98

Transaction Step Not Recorded (Menus, Plug-Ins) in IE Mode . . . . .	98
Pop-Up Windows in IE Mode . . . . .	99
Form Values are Not Recorded . . . . .	99
Playback Issues: . . . . .	100
Transaction Times Out . . . . .	100
What the HTTP_TRANS Probe Means to Your Resource Usage . . .	101
Dynamic Frame Names (ERROR: Unable to find frame x) . . . . .	102
Initialization Error Dialog Boxes . . . . .	103
Response Time and/or System Load Increases . . . . .	103
Java Applets Causing Abort or Incorrect Behavior . . . . .	104
Playback Does Not Detect End of a Transaction Step . . . . .	104
Empty Message Box during Playback/Stepping . . . . .	105
New Window is Not Loaded Correctly . . . . .	105
Content of a Selection Box Changes with Every Playback . . . . .	105
IE Mode Registry Settings . . . . .	106
Dynamic ID Fields . . . . .	107
Scripts Against Web Sites with Active-X Controls/Java Applets . . .	107
Limitations on Windows 2000 Playback . . . . .	108
HTTP_TRANS Probe Configuration for TIPs . . . . .	108
Annotated Trace File for HTTP_TRANS . . . . .	109





# OVIS Web Transaction Recorder

The HTTP\_TRANS probe is used to monitor multi-step web transactions, such as catalog lookup, login/logout, and shopping carts. When you create an HTTP\_TRANS service target, the Web Transaction Recorder is automatically launched. The Web Transaction Recorder allows you to specify the user actions that you want to track and record them for the probe to play back on a regular basis, simulating typical end-user activity and collecting important availability and response time data.

Using the Web Transaction Recorder for configuring the HTTP\_TRANS probe(s) alleviates the likelihood of errors and accelerates configuration steps. Instead of having you manually type numerous URLs or page references, the Web Transaction Recorder allows you to go through each step of a typical end-user transaction, while it automatically captures your actions and the sequence of accessed pages and links to which you navigate. Later, you can test and verify the transaction and make additional modifications on the recorded transaction steps.



Internet Explorer 5.5 or later is required for use with the Web Transaction Recorder. Internet Explorer 6.0 with the latest service pack is recommended because it provides the capability to intercept and log HTTP status codes.

You are allowed only one HTTP\_TRANS service target per service group. The maximum number of steps you can record is 100.

# Getting Started

Before recording web transactions, it is helpful to first understand how a web page or web site is constructed and what elements it might include. These things can affect how you record the transaction, and what options need to be used in the Web Transaction Recorder to be able to successfully play back the recording in the HTTP\_TRANS probe.

## Web Page Basics

The following basic web page information is described in the following sections:

- HTML
- JavaScript and Plug-In Examples
- Frames
- HTTP
- HTTPS and SSL
- Authentication and Digital Certificates
- Proxy Servers
- Caching
- URL
- Cookies

## HTML

HTML (Hypertext Markup Language) is a subset of SGML (Standardized General Markup Language). HTML is used to define the structure of a web document. In HTML, tags (keywords or elements) enclosed in angle brackets (< >) define content type. Markup is the act of inserting tags or other text into a document that is not visible to the reader or part of the content, but enhances the document. Markup may include format tags, structure tags, or hypertext linking capability. Tags can have attributes or options. This is an example of HTML tags:

```
<em>Hello</em>
```

```
<table border>

<h1>Hello, <em>world!</em></h1>
```

A simple web page contains two sections: head and body (see the example below). The title is shown in the title bar of the browser window. The .html file markup begins with `<html>` and ends with `</html>`.

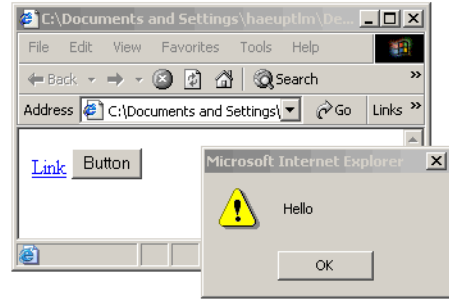
```
<html>
<head>
<title>A Simple Web Page</title>
</head>
<body>Hello World!</body>
</html>
```

Web pages can include the following:

- **Cascading style sheets (CSS).** A CSS defines document styles (that is, specific appearance: font use, colors, and so on, throughout the document).
- **JavaScript and Visual Basic Script.** Complex web pages may make use of scripts to provide a dynamic component in the web browser (that is, the script is executed in the web browser not on the server).
- **Java applets and other plug-ins.** Applets and plug-ins (such as Macromedia FLASH) are programs that are loaded by the browser. They usually have nothing to do with HTML and are independent programs that execute separately.

## JavaScript and Plug-In Examples

JavaScript is executed in the browser directly without going out to the server and back. The JavaScript code is enclosed in the HTML `<SCRIPT>` tag.



```
<html><body>
<SCRIPT LANGUAGE="JavaScript">
  function f1() {
    alert("Hello");
    return false;
  }
</SCRIPT>
<a href=alink.html onclick="f1()">Link</a>
<INPUT TYPE=button VALUE=Button
  onclick="f1()">
</body></html>
```

In the previous example, the function call in the script displays a message dialog box. The `f1` function is then referenced in the HTML `OnClick` handler for the link (`<a href>`) and the input button. Clicking on the link or the button shows the dialog box defined in the script section.

Other handlers, such as `mousemove` and `focus`, can be associated with JavaScript.

When using a plug-in, the HTML `<object>` tag defines the component to be used.

The example below shows the use of the Macromedia FLASH plug-in. Each HTML element is defined following the example.

```
<object classid="clsid:D27CDB6E-AE6D-11cf-96B8-444553540000"
codebase="http://download.macromedia.com/pub/shockwave/cabs/flash/
  swflash.cab#version=5,0,0,0" width="500" height="200" align="top">
<param name="movie" value="/images/tbird_new.swf">
<param name="SCALE" value="noborder">
<embed src="/images/tbird_new.swf" width="500" height="200" align="top"
```

```

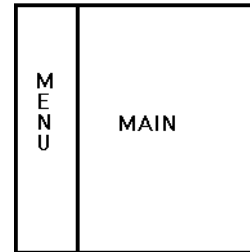
pluginspage="http://www.macromedia.com/go/getflashplayer"
type="application/x-shockwave-flash" scale="noborder"></embed>
</object>

```

The `codebase` parameter defines where to download the component.

The `param` parameter is passed to the component.

The `<embed>` tag defines the multimedia content to be viewed in Macromedia FLASH. Within this tag, in the example above, the `SRC` attribute defines the content, type and plug-in web page that defines the Shockwave component that should be used to render this content.



## Frames

Frames are used in web pages as a method of breaking up a browser window into multiple views. Each view holds a different document. Often clicking on a link in one view loads a new document in another view.

For example, many web pages use a two-pane design. Clicking on links in the navigation pane (MENU in the example), brings up different documents in the main pane.

Example:

```

<FRAMESET COLS="216,*">
  <FRAME SRC="menu.html" NAME="MENU">
  <FRAME SRC="main.html" NAME="MAIN">
</FRAMESET>

```

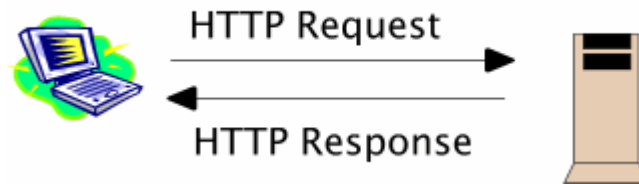
The browser loads the document with the frameset definition. The frameset tag has attributes that define the layout of the web page. The browser parses and loads the frames specified by the `SRC` attribute and divides the window according to the frameset attributes.

## HTTP

Hypertext Transfer Protocol (HTTP) is an application protocol (over TCP) that is used to request, send, and receive web objects over the Internet for display in a browser. The current version of HTTP is 1.1.

When you enter a URL in your browser, this sends an HTTP command to the web server requesting it to send the specified web page.

HTTP operates in a client/server model. On the client, the browser requests, receives, and displays web objects via HTTP requests. The web server sends objects in response to requests, via HTTP response.



## HTTPS and SSL

HTTPS is HTTP over SSL (Secure Sockets Layer). HTTPS protocol is used when data needs to be sent securely over the World Wide Web. HTTPS sends individual messages securely, and SSL establishes a secure connection between two computers. By convention, web pages that require an SSL connection start with https: instead of http:.

## Authentication and Digital Certificates

Authentication is the process of determining whether someone or something is, in fact, who or what it is declared to be. Authentication can be simply the use of logon username and password. But Internet business and many other transactions require a more stringent authentication process. For these types of transactions, digital certificates are used. Digital certificates are issued and verified by a Certification Authority (CA) and are used to perform authentication.

To be able to generate an SSL connection, a web server requires an SSL Certificate. The SSL Certificate authenticates the web server and provides keys for encryption. The SSL Certificate must be issued by a CA recognized by the browser. CAs achieve recognition by having their Root CA Certificate installed in browsers such as Internet Explorer and Netscape.

## Proxy Server

A proxy server is a system that is between a client application (such as a web browser) and a server. It intercepts all requests to the real server to see if it can fulfill the requests. If not, it forwards the request to the real server. Proxy servers can be used to improve performance or to filter requests.

Proxies are often used to protect an intranet (or private network), so that only the proxy is allowed to be on the Internet. A firewall is a system designed to prevent unauthorized access to or from a private network. One firewall technique uses proxy servers to intercept all messages entering and leaving the network. The proxy server effectively hides the true network addresses.

## Caching

A cache is a place where something is temporarily stored. The files that are automatically requested when you look at a web page are stored on your hard disk in a cache subdirectory under the directory for your browser. When you return to a page you recently looked at, the browser can get these files from the cache rather than from the original server, saving you time and saving the network some additional traffic. You can usually vary the size of your cache, depending on your particular browser.

Content delivery is the service of copying the pages of a web site to geographically dispersed servers and, when a page is requested, dynamically identifying and serving page content from the closest server to the user. Content delivery enables faster delivery. When you click a URL that is content-delivery enabled, the content delivery network reroutes your request away from the site's originating server to a cache server closer to you. The cache server determines what content in the request exists in the cache, serves that content, and retrieves any non-cached content from the originating server. Any new content is also cached locally. Other than faster loading times, the process is generally transparent to the user, except that the URL served may be different than the one requested (also called redirection).

## URL

Uniform Resource Locator (URL) is a way of referring uniquely to a document or other object on the Internet. When you want to display a web page in a browser, you enter a URL. The syntax is as follows:

```
protocol://domain_name[:port number]/<url or path name>
```

Example: **http://www.hp.com/index.html**

When using http, the browser performs the following steps to display a URL:

- 1 Parses the URL.
- 2 Obtains the domain name of the URL; for example, www.hp.com.
- 3 Asks the Domain Name Server (DNS) for translation of the domain name to the IP address.
- 4 Uses the IP address to connect to the server (TCP).
- 5 Sets up an HTTP connection to the server.
- 6 Passes the path name portion of the URL to the server.

In the case of a web proxy, the browser sends the complete URL to the proxy. Other protocols, such as HTTPS or FTP, may be used in a URL.

## Cookies

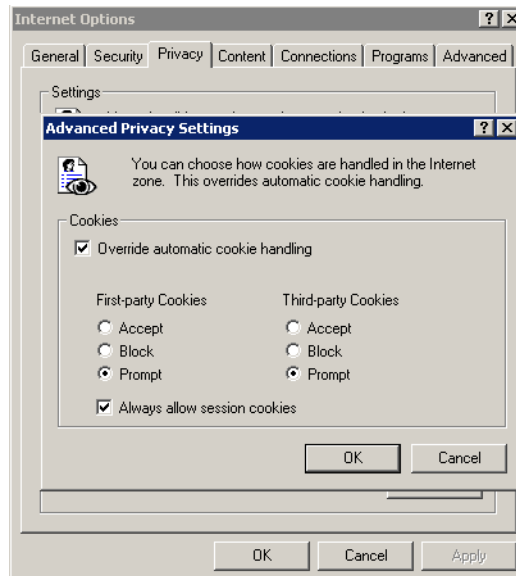
A cookie is information the server sends to a browser. In subsequent requests from the browser to this server, the browser presents the cookie in the header; for example, http request + Cookie: OVIS. The server then matches the presented cookie with server-stored information.

Cookies can be used to maintain state between web transactions. This can be useful for authentication, identifying a registered user without requiring log-in, remembering user preferences, and maintaining a shopping cart.

There are two types of cookies: session cookies and persistent cookies. Session cookies are only used during one Internet session. Persistent cookies are stored on the client system and invoked in subsequent Internet sessions. Persistent cookies usually contain expiration information.



You can watch cookies as they are sent. In Internet Explorer 6.0 or later, select **Tools** → **Internet Options** → **Privacy** → **Advanced** tab. Check **Override automatic cookie handling** and, under **First party cookies** and **Third party cookies**, select **Prompt**.



For each cookie received from a server, the following dialog box will be displayed to allow you to see the cookies being downloaded.



# Web Transaction Recording Modes

In the Web Transaction Recorder there are three possible recording modes. Before you begin recording a web transaction you need to select a recording mode (also called a navigation mode). The navigation modes are the following.

- **IE Mode Probing** – HTTP\_TRANS probes created with IE mode run only on Windows systems. They call the IE Browser Engine to execute the recorded transaction.

This probe uses the IE Browser Engine and as a result the probe is able to handle such things as JavaScript and screen rendering. Screen rendering draws the web page and executes scripts and embedded objects such as Java and other plug-ins. IE mode provides response time measurements that are very close to those experienced by an end-user.

- **URL Mode Probing** – The URL mode uses a custom HTTP-like probe that only downloads the URLs and does not attempt to render the content via the IE Browser Engine. This is good for verifying the availability of a web site but is less representative of the actual user experience in terms of response time, because a user will be using the IE browser.

The URL mode probe is platform-independent so that probes can be run on UNIX, Linux, or MS Windows. The URL mode also requires less memory resources than the IE mode.

- **Navigation Point Mode Probing** – The Navigation Point mode is obsolete and is provided as an option for backward compatibility. All recording can be handled with either of the other two modes.

The difference between the IE mode and URL mode is the technology used to do the probing and whether individual transaction steps are specified as a URL.

HTTP\_TRANS probes are more resource-intensive than URL mode probes, especially when there are a large number of steps per transaction sequence. This is because the steps are executed one after the other with little or no think-time delays between steps. When many such transactions are executed in parallel, CPU utilization can be high. In addition, each probe execution requires a process that needs anywhere from 3 to 20 Mbytes of memory, with the IE mode requiring more than the URL mode.



Non-IE mode can use more CPU time because the probe does not delay (no think time) between each step. A large number of steps can result in CPU spikes, especially when the target steps return results quickly. Memory usage for non-IE mode probes is approximately 3-5 K of resident memory per transaction. IE mode uses less CPU time for the probe itself because the IE mode probe pauses for 1.5 seconds between each step. IE mode can, however, use more CPU time when the rendered page contains many client-side browser controls and logic that executes in the browser itself (such controls would not actually execute in the non-IE mode probe). Memory usage for non-IE mode probes is approximately 10-20 K of resident memory per transaction and can vary depending on the type of content rendered in the downloaded page.

If a probe system is dedicated for OVIS probes, attention needs to be given to the number of parallel executions. If the number saturates system and memory resources such that the probes themselves are starved, the probes will time out and fail to execute to completion. If HTTP\_TRANS probes are executed on a probe system that is also used for other computing tasks and applications, equal attention needs to be given to the number of parallel executions to make sure that the additional resource usage of the probes do not negatively impact these other computing functions.

In general, the default concurrency value of 32 is high for HTTP\_TRANS probes having more than five steps in each sequence. In such environments with many HTTP\_TRANS targets, a concurrency value of 10 or less is more appropriate. You can control the concurrency of the probes by setting up a network connection with concurrency of between one and 10, or using the Target Priority option in the Probe Location dialog box. Each IE mode probe can then be assigned to this network. When in doubt, monitor the probe system with performance tools to assess the impact of the probes and whether or not the probes are able to complete in the allotted intervals. See the *OVIS User's Reference Guide*, chapter 7 section on Scalability for details. See [What the HTTP\\_TRANS Probe Means to Your Resource Usage on page 101](#) in chapter 3 of this manual for details on what concurrency to use.

The following table provides an overview of the supported features for the navigation modes.

**Table 1 Web Transaction Navigation Modes**

Navigation Mode	Java Script	ActiveX Java	Dynamic URLs	Plug-ins	Screen Rendering
IE Mode (default) probeHttpTrans2	Yes	Yes*	Yes	Yes**	Yes
URL Mode probeHttpTrans	No	No	Yes***	Yes**	No
Navigation Point probeHttpTrans	No	No	Yes	No	No

\* Some web pages have Active-X controls or Java applets where mouse clicks and keystrokes cannot be recorded through Internet Explorer. The Record Mouse and Keyboard Events option allows these raw clicks and keystrokes to be recorded and later played back (see Record Mouse and Keyboard Events (IE mode) in the section [Recording Options on page 39](#) for limitations).

\*\* Only plug-ins that load URLs

\*\*\* Through substitution rules



The executables for the probes are different: IE mode is probeHttpTrans2; URL mode is probeHttpTrans.

## Quick Tips for Determining Which Mode to Use

### Use IE Mode When the Following is True

- Web pages contain JavaScript or Java applets (such as online forms).
- Web transactions are complex.
- Web pages are generated dynamically each time the user visits.
- Input is coming from a program or script.
- You want your measurements to closely emulate a customer's browser.
- The web transaction cannot be recorded using the URL mode; for example, the page has JavaScript or requires screen rendering.

**Use By URL Mode When the Following is True**

- The probe needs to run from a UNIX or Linux system.
- The IE mode features are not necessary.

# How the Web Transaction Recorder Works

The Web Transaction Recorder provides for both recording and playback. The Web Transaction Recorder functions differently in the two recording modes: URL mode and IE mode.

## Recording

### IE Mode Recording

When you record in IE mode, the Web Transaction Recorder calls the IE Browser Engine (IE API). As you step through your web transaction, the IE Browser Engine fires notification events that are captured by the Web Transaction Recorder. Notification events deliver information about what is being loaded in each step. For example, the `BeforeNavigate2` event is fired before navigation occurs; the `DocumentComplete` event is fired when the entire document is finished loading; `OnClick` event is fired when you click an HTML element such as a button or link.

In IE mode recording, the Web Transaction Recorder records the actual HTML element you click. Recording the HTML element itself makes the playback more resilient when content changes, and also more closely duplicates the true user experience.

### URL Mode Recording

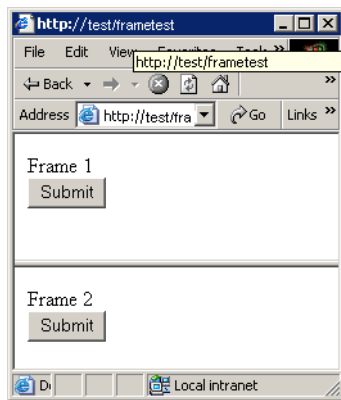
When you record in URL mode, the Web Transaction Recorder calls the IE Browser Engine. As you step through your web transaction, a new URL is loaded by Internet Explorer each time you click a link or press a button. The `BeforeNavigate2` event is fired every time Internet Explorer loads a new URL. The Web Transaction Recorder captures the URL that is reported by each `BeforeNavigate2` event that is fired. The `BeforeNavigate2` event passes all the URLs, but the Web Transaction Recorder only captures the first URL for a step. Note that this is for recording only. When the probe runs, it is the HTTP probe that executes.

## Results of the Recording

The `httptrans.dat` file contains configuration information for the HTTP\_TRANS probe as well as the instructions for the steps you recorded. Enough information must be recorded to allow for correct playback. For URL mode recording, the instructions just need to contain the URL. For IE mode, various parameters are recorded in order for the probe to be able to find the correct HTML element later during playback.

## Simplified Example of Recording

The simplified example below shows the HTML for the following dialog box.



There is a frameset document, and one HTML document each for Frame 1 and Frame 2. These frames each have a **Submit** button. If you press the **Submit** button in Frame 1, the following is recorded by the Web Transaction Recorder:

```
IENAVPNT=INPUT:button "f1" "5" "" "Submit" ""
```

See the sections following the example for an explanation of where this information comes from.

frameset.html:

```
<html>
<FRAMESET ROWS="100,*">
<FRAME name="f1" src="f1.html">
<FRAME name="f2" src="f2.html">
</FRAMESET>
```



```
<body>  
</body>  
</html>
```

**f1.html:**

```
<html>  
<title>Frame 1</title>  
<body>  
Frame 1  
<br>  
<INPUT TYPE=button VALUE=Submit>  
</body>  
</html>
```

**f2.html:**

```
<html>  
<title>Frame 2</title>  
<body>  
Frame 2  
<br>  
<INPUT TYPE=button VALUE=Submit>  
</body>  
</html>
```

The Web Transaction Recorder needs to record enough information when you press the Frame 1 **Submit** button so that it can play back the transaction exactly. The Web Transaction Recorder uses information from the DOM (Document Object Model) to identify the HTML element (the button you pressed) uniquely.

The following shows the DOM (Document Object Model) view of the three frames in the example above.

- Index – Each HTML element has an index in the DOM; for example, 0000, 0001.
- Frame – The frame name is shown; for example, \_top, f1, f2.
- html element – The DOM shows the HTML element; for example, HEAD, INPUT.
- Web Transaction Recorder format in brackets [] – The Web Transaction Recorder saves the various properties of each HTML element; for example, type, frame name, index. This information is needed by the Web Transaction Recorder in playback to find the element in the DOM again.
- html source – The DOM shows the source text of the element and all that it contains.

Frametest.html:

```
index|frame|html |Web Recorder|      html source      |
      element| format [ ] |
0000: '_top' HTML [# "top" "0"] '<HTML><HEAD></HEAD><FRAMESET rows=100,*><FRAME name=f1 src="'
0001: '_top' HEAD [# "_top" "1"] '<HEAD></HEAD>'
0002: '_top' TITLE [# "_top" "2"] ''
0003: '_top' FRAMESET [# "top" "3"] '<FRAMESET rows=100,*><FRAME name=f1 src="f1.html"><FRAME nam'
0004: '_top' FRAME [# "top" "4"] '<FRAME name=f1 src="f1.html">'
0005: '_top' FRAME [# "_top" "5"] '<FRAME name=f2 src="f2.html">'
```

f1.html:

```
0000: 'f1' HTML [# "f1" "0"] '<HTML><HEAD><TITLE>Frame 1</TITLE></HEAD> <BODY>Frame 1 <BR'
0001: 'f1' HEAD [# "f1" "1"] '<HEAD><TITLE>Frame 1</TITLE></HEAD>'
0002: 'f1' TITLE [# "f1" "2"] '<TITLE>Frame 1</TITLE>'
0003: 'f1' BODY [# "f1" "3"] ' <BODY>Frame 1 <BR><INPUT type=button value=Submit> </BODY>'
0004: 'f1' BR [# "f1" "4"] '<BR>'
0005: 'f1' INPUT [INPUT:button "f1" "5" "" "Submit" ""] '<INPUT type=button value=Submit>'
```

f2.html:

```
0000: 'f2' HTML [# "f2" "0"] '<HTML><HEAD><TITLE>Frame 2</TITLE></HEAD> <BODY>Frame 2 <BR'
0001: 'f2' HEAD [# "f2" "1"] '<HEAD><TITLE>Frame 2</TITLE></HEAD>'
0002: 'f2' TITLE [# "f2" "2"] '<TITLE>Frame 2</TITLE>'
0003: 'f2' BODY [# "f2" "3"] ' <BODY>Frame 2 <BR><INPUT type=button value=Submit> </BODY>'
0004: 'f2' BR [# "f2" "4"] '<BR>'
0005: 'f2' INPUT [INPUT:button "f2" "5" "" "Submit" ""] '<INPUT type=button value=Submit>'
```

Pressing the **Submit** button in Frame 1 in the example above results in the following being recorded by the Web Transaction Recorder:

```
IENAVPNT=INPUT:button "f1" "5" "" "Submit" ""
```

The `httptrans.dat` file contains the instructions for the steps you recorded. The information is in script format. The instructions `IESTEP` and `IENAVPNT` are used to identify each user action in the transaction. In this example, the user action consists of the Submit button being pressed in Frame 1.

During playback, the Web Transaction Recorder follows different procedures depending on whether the instruction is `IESTEP` or `IENAVPNT`. See the Playback section below for more information.

## Playback

### URL Mode Playback

When URL mode is used, the Web Transaction Recorder establishes a pipe to launch `probeHttpTrans.exe` (the `HTTP_TRANS` probe executable). `probeHttpTrans.exe` prints a dump *filename* for the current step in the recording to `stdout` and pauses. The Web Transaction Recorder detects the pause, reads the dump file, and displays it. Web Transaction Recorder then instructs `probeHttpTrans` through the pipe to continue.

### IE Mode Playback

The `httptrans.dat` file contains configuration information for the `HTTP_TRANS` probe as well as the instructions for the steps you recorded. The information is in script format. The instructions `IESTEP` and `IENAVPNT` are used to identify each user action in the transaction.

The following is an example.

```
[HTTP_TRANS]
CUSTOMER=hp
SERVICENAME=OVIS
FLAGS=cache=0,silent=0,noNewWindow=0,waitTime=1500,incWaitTime=0,runJava=1,about
BlankInit=1,captureWindow=0
IESTEP=www.hp.com
PATTERN="+Welcome to HP"
IENAVPNT=INPUT:image " top" "466" "submit" "" "http://welcome.hp-ww.com/img/
hpweb_1-2_arrw_sbmt.gif"
PATTERN="+Search HP US - Search results for 'Internet Services'"
FORMDATA=##FPHEGPHACOHDFGFBHCGDIEGGPHCGNCOHBHEDNEJGOHEGFHCGOGFHECAFDGFHCHGGJGDG
FHD
IENAVPNT=A " top" "231" "" "http://search.hp.com/gwuseng/
redirect.html?url=http%3A//www.openview.hp.com/solutions/
&qt=Internet+Services&pos=1&type=REG"
PATTERN="+Hewlett-Packard OpenView - solutions"
```

During playback, IESTEP instructs Internet Explorer to navigate to the defined URL. If the IENAVPNT instruction is found, the Web Transaction Recorder must do the following to play back the recording/step:

- 1 Parse the HTML element information.
- 2 Find the element in the Document Object Model (DOM).
- 3 Verify that the recorded element and the element found are the same.
- 4 Simulate a user clicking on the HTML element.
- 5 Detect the end of step.



You may have to set various options in order for the playback to correctly reflect what you want to measure in the web transaction.

## How the HTTP\_TRANS Probe Works

The HTTP\_TRANS probe is actually two probes:

- probeHttpTrans.exe – the HTTP\_TRANS probe created when you record a web transaction using URL mode
- probeHttpTrans2.exe – the HTTP\_TRANS probe created when you record a web transaction using IE mode

The URL mode probe essentially calls the HTTP probe code for each step (URL) in the list you recorded and maintains a per-process cookie cache between the steps. This is good for verifying the availability of a web site but is less representative of the response time of the actual user experience because a user will be using the IE browser.

The IE mode probe calls the IE Browser Engine and then simulates the user experience by stepping through the web transaction just as a user would within a browser window. Because this probe uses the IE Browser Engine, there is more overhead associated with the probe. However, the IE Browser Engine also does more. As a result, the probe is able to handle such things as JavaScript and screen rendering. See [What the HTTP\\_TRANS Probe Means to Your Resource Usage on page 101](#) for more information on how to determine how many HTTP\_TRANS probes you can run concurrently.



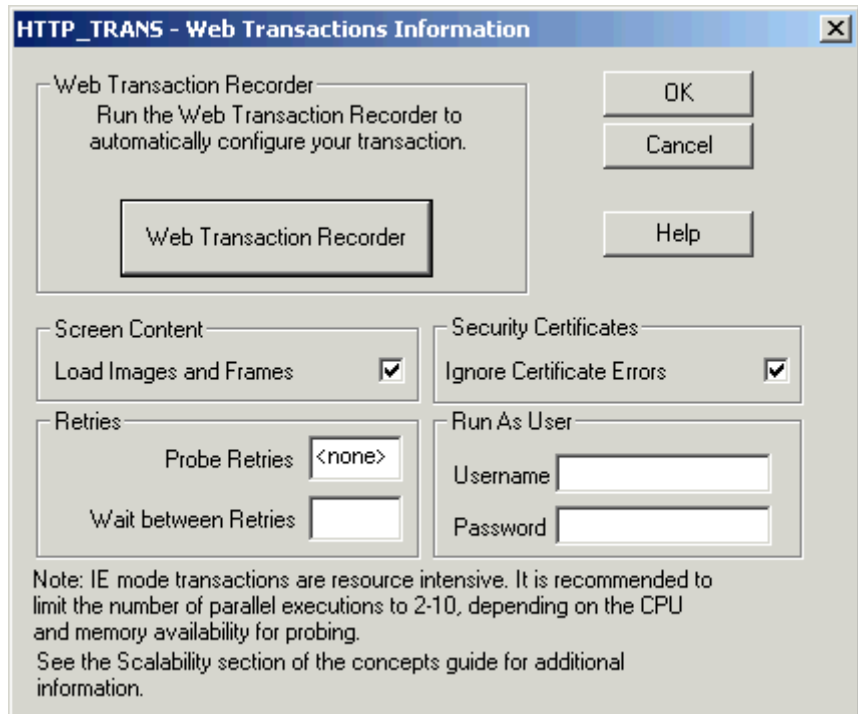
# Using the Web Transaction Recorder

This chapter covers the following topics:

- [Configuring the HTTP\\_TRANS Probe](#)
- [Steps to Record a Web Transaction](#)
  - [Basic Steps](#)
  - [Specifying Global Transaction Properties](#)
  - [Transaction Step Properties Basic](#)
  - [Transaction Step Properties Advanced](#)
- [Advanced Usage](#)
- [Transaction Script Reference](#)
- [Web Transaction Recorder Tips](#)

## Configuring the HTTP\_TRANS Probe

Before starting the Web Transaction Recorder, you must first configure an HTTP\_TRANS probe as you would any other probe, using the OVIS Configuration Manager. That is, you set up a customer, service group and a service target. When you begin to create a service target, the following dialog box is displayed.



In this dialog box you can define a number of options and then click the **Web Transaction Recorder** button to open the Web Transaction Recorder dialog.

You can configure HTTP\_TRANS probes to run as a specific user as opposed to the account that runs the OVIS scheduler. Add the Username and Password in the Run As User section of the HTTP\_TRANS Web Transactions Information dialog box.

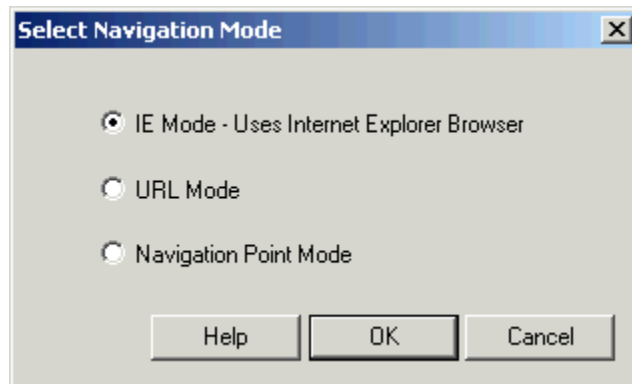


- ▶ The HTTP\_TRANS probe in IE mode requires significant CPU and memory resources that can limit the number of parallel executions of this probe type. Too many parallel executions can cause aborts of the probe program `probehttptrans2.exe`. To prevent this from happening, limit the concurrency in the Probe Location dialog box of the Configuration Manager to between 1-10. See [What the HTTP\\_TRANS Probe Means to Your Resource Usage on page 101](#) for details.

You are then prompted to select the navigation mode:

- IE mode
- URL mode
- Navigation Point mode (This mode is obsolete and is only provided for backward compatibility.)

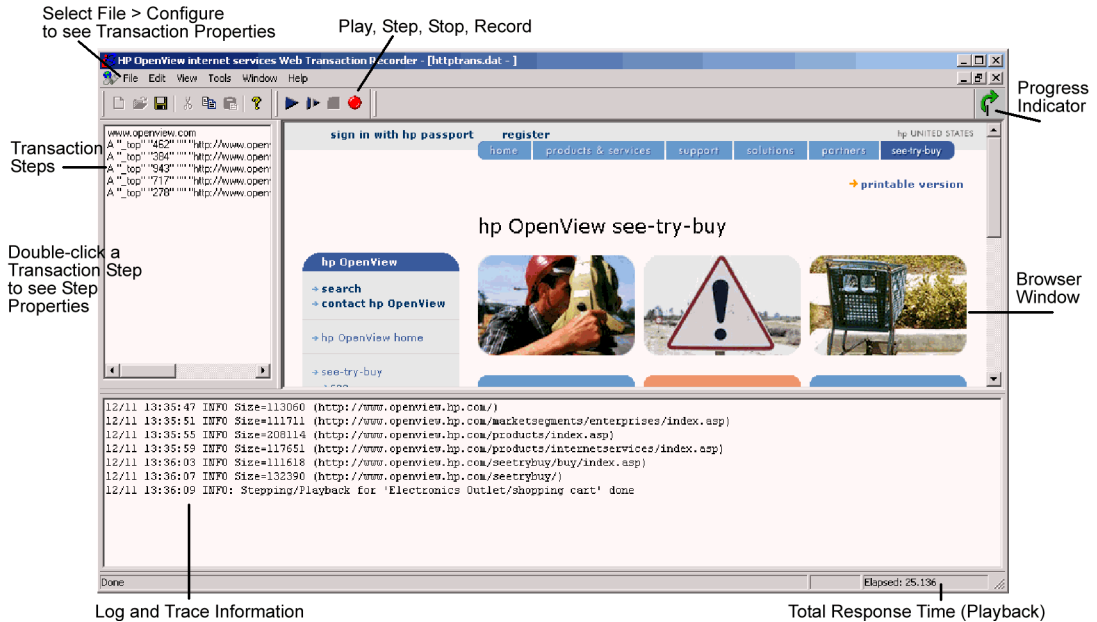
Refer to chapter 1 for more information on IE mode and URL mode and how to choose the best recording method for your web transaction.



In the Web Transaction Recorder, a transaction is composed of individual steps. A step can be a URL or the HTML element (like a link or button) the user clicked in the web page.

## Steps to Record a Web Transaction

The Web Transaction Recorder GUI is used to record a web transaction. It is launched automatically from the Configuration Manager when you configure an HTTP\_TRANS service target. It can also be run separately as an executable (<install dir>\bin\WebRecorder.exe). The following illustration shows an example of the GUI



In the Web Transaction Recorder, a transaction is composed of individual steps. You click or navigate from one link, button, or web page to another within a web application to perform a transaction. All your actions are recorded.

You can play back your recording to verify the transaction and make modifications as needed.

The HTTP\_TRANS probe uses the recording to monitor the web transaction. The probe collects availability and response time metrics for the transaction. This information is then displayed in the OVIS Dashboard.

## Basic Steps

The basic steps to recording a transaction are as follows:

- 1 Plan what web pages you want to record. Go through the steps of the web transaction in a browser first to be sure you know the sequence and selections to make for correct recording.
- 2 In the Web Transaction Recorder use the **Tools** → **Cache Viewer** to delete cookies before recording your transaction. You can set the Scrub Cookies flag to enable this during playback (see [Specifying Global Transaction Properties on page 37](#)).
- 3 Configure global transaction properties in the **File** → **Configure** → **Properties** dialog box. You can specify such things as how the recorder should handle pop-ups and error dialogs, timeout and wait time, error capture, proxy settings, and tracing levels.
- 4 In the main window press the **Record** button to start recording the transaction steps.
- 5 Enter the starting URL.

After the web page is completely loaded in the browser window in the right pane (indicated when the circular green arrow in the upper right corner stops turning), you can navigate through the transaction steps in the web page displayed.

The transaction steps are shown in the left pane. Log and trace information for the transaction steps is shown in the lower pane.

- 6 Press **Stop** to end the recording.
- 7 Play back the recording to test whether all the steps you wanted were included. Check the response time during playback, which is shown in the lower right corner.
- 8 Make any changes using the options in the Web Transaction Recorder; for example:
  - Modify transaction step properties in the **Step Properties** dialog box, which you access by right-clicking on a transaction step in the left pane and selecting **Properties**.

- Enter advanced scripting information in the **Advanced Step Properties** dialog box, which you access by right-clicking on a transaction step in the left pane, selecting **Properties**, and then selecting the **Advanced** tab.
- 9 Exit the Web Transaction Recorder and press the **OK** button in the HTTP\_TRANS Web Transaction Information dialog.
  - 10 Set up a probe location for the probe you created in the Web Transaction Recorder.
  - 11 Set up service level objectives for the probe you created in the Web Transaction Recorder.

You can specify Step Alarming. See the *OVIS User's Reference Guide*, chapter 3 section on "Configuring Service Level Objectives," for details on configuring alarms for an individual step in the HTTP\_TRANS probe transaction. The step alarming function is for alarms only. The step thresholds are not used for service level objectives.

You can only select a single step per set of alarms. To create alarms for several individual steps in a transaction, create a separate alarm definition for each. You can use the same metric in multiple sets of alarm definitions.

Step Alarming is for use with Response Time metrics, not Availability. This is because Availability is determined for the whole transaction. If any step is unavailable, the transaction is marked as unavailable.

The step number is displayed in the Dashboard and in the Status page drill-down in the Configuration Manager.

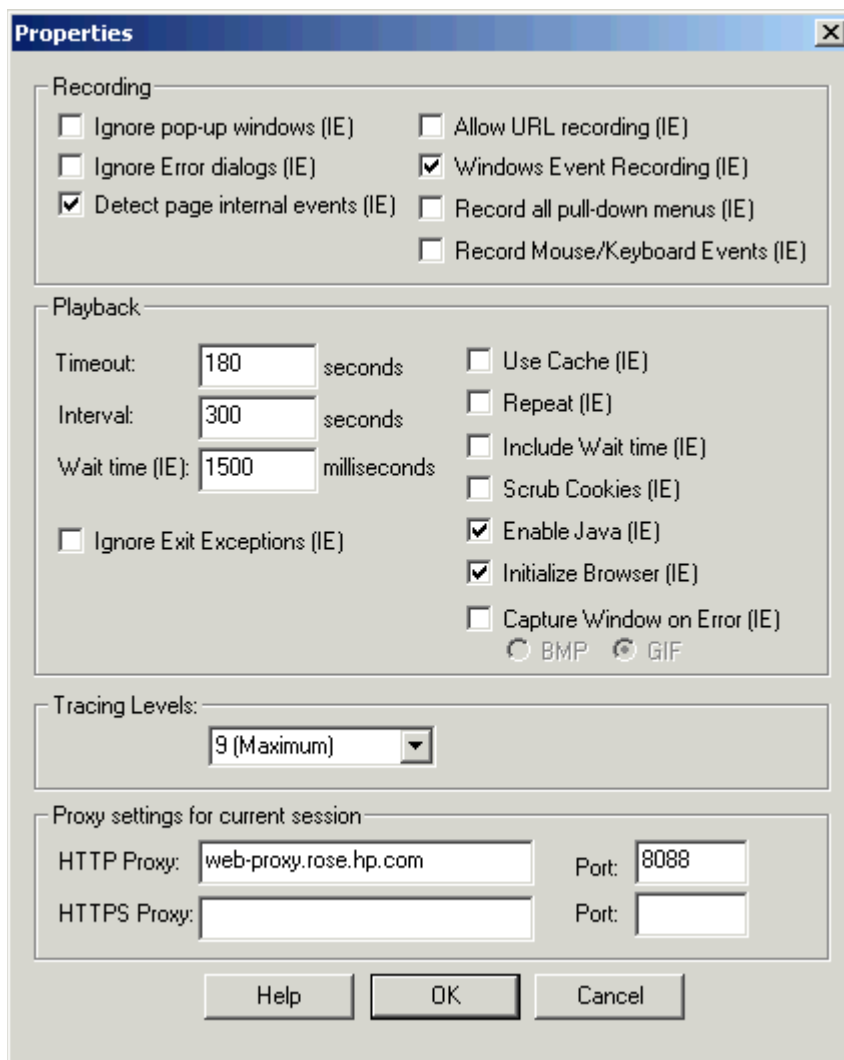
- 12 Save the probe changes in the Configuration Manager and exit.



Note that within the Configuration Manager, the **Test On Management Server** selection for IE mode HTTP\_TRANS probes, only works on Windows XP or higher.

## Specifying Global Transaction Properties

Selecting the **File** → **Configure** → **Properties** dialog box allows you to specify properties that should apply to the overall transaction. You can also specify properties that apply to an individual step in the transaction by double-clicking on the step to display the Step Properties dialog box. Each item is described below. Also see [Recording and Playback Issues on page 98](#) for some examples of when you might need to set a value for the overall transaction.



## Recording Options

**Ignore pop-up windows (IE mode).** Set this flag to have the recording ignore pop-up windows such as advertisements. This option is only available in IE mode. The script option `enableIEDialog` overrides this checkbox (see [enableIEDialog on page 85](#)).

**Ignore Error dialogs (IE mode).** Set this flag to have the recording ignore dialog boxes such as those displaying JavaScript errors. This option is only available in IE mode.

**Detect Page Internal Events (IE mode).** If this flag is set, the recorder automatically inserts `continueClick=1` flags for steps that do not load documents from the server. This option is only available in IE mode.

**Allow URL recording (IE mode).** Set this flag to allow the recorder to add URLs in case a Navigation Point could not be determined. Use this option for complex web applications that handle mouse events internally. This option is only available in IE mode.



If the recorded URL is dynamic, a substitution rule must be applied (see [Dynamic URL Substitution on page 50](#)). This option is only available in IE mode.

**Windows Event Recording (IE mode).** If this flag is not set, the recorder uses the pre-OVIS-5.0 method to determine the HTML element that was selected (which is using the `OnClick` notification event). This option is only available in IE mode.

When you set this flag, the Web Transaction Recorder uses a different algorithm (similar to that used by IE) to determine the HTML element that is under the mouse.

**Record all pull-down menus (IE mode).** If you have a web application that performs dynamic content loading based on pull-down menu selections, you need to set this flag right before selecting the pull-down menu item, during a recording. The Web Transaction Recorder does not normally record these pull-down menu selections. The assumption is that the pull-down menu items are just part of a form, and the Web Transaction Recorder waits until the form is completed to record the next step. If this flag is set, the pull-down menu selections can be recorded. It is not recommended to leave this option checked after you record the pull-down menu or set it at the beginning of a transaction recording. Doing so may result in duplicate navigation points and may cause other navigation points to not be recorded. See [HTML Dialog and Pull-down Menu Internals on page 63](#).

In addition, it may be necessary to remove some formdata statements that are recorded for the SELECT navigation point. Depending on the web application, only the formdata statement for the particular menu should be part of the navigation point.

Here is an example. Say that the web page has two menus, m1 and m2. When recording the second menu m2, it may contain the formdata parameter for m1:

```
ienavpnt=SELECT "_top" "169" "m2" "2"  
formData=_top.form.m1=value_for_m1  
formData=_top.form.m2=value_for_m2
```

This should be changed to:

```
ienavpnt=SELECT "_top" "169" "m2" "2"  
formData=_top.form.m2=value_for_m2
```

In addition, the Web Transaction Recorder needs to be instructed to not execute the submit function of the form, because this is the default behavior of a SELECT navigation point. This can be achieved with the nosubmit option on the SELECT navigation point and by adding continueClick=1 because no page navigation is performed:

```
ienavpnt=SELECT "_top" "169" "m2" "2" "nosubmit"  
formData=_top.form.m2=value_for_m2  
continueClick=1
```

**Record Mouse and Keyboard Events (IE mode).** Some web pages have Active-X controls or Java applets where mouse clicks and keystrokes cannot be recorded through IE. This option allows these raw clicks and keystrokes to be recorded and later played back. It is recommended that this option only be turned on for those specific steps that require mouse and keyboard event recording.

There are cases when the playback of these raw clicks will not work because the Active-X control or Java applet is expecting mouse and keyboard events to originate from a lower level of the operating system. These are cases that Web Transaction Recorder cannot support.

## Playback Options

**Timeout.** Adjust timeout for the playback. This is useful during the test of the transaction. Once a valid timeout value has been found, it needs to be specified in the Probe Location dialog box again.



**Interval.** Interval for repeated playback. This only applies when repeat is checked. This value is not applied to the interval setting in the Probe Location dialog box.

**Wait Time (IE mode).** The wait time the probe uses to detect redirections in IE mode. Adjust this time if you find that a transaction times out prematurely. This option is only available in IE mode.

**Ignore Exit Exceptions (IE mode).** This checkbox will force the Web Recorder probe to ignore runtime errors that cause exceptions after the transaction has already completed. Sometimes third party Internet Explorer plug-ins can trigger exceptions when attempting to clean up and exit a transaction. Checking this box will prevent an error message from appearing on the console every time the probe runs.

### Checkboxes

**Use Cache (IE mode).** This flag influences how the underlying IE browser will download objects (HTML, images, and so on). If not checked (default), the probe instructs IE to force the HTTP request through to the server and ignore the proxy's cache. (The objects are retrieved from the server and not from the proxy cache if the proxy supports caching.) In addition, the local IE cache is ignored and the server is always asked for updated information about the objects that should be downloaded. If the objects are up to date (HTTP 1.1 304 reply by the server), the object in the cache is used and not downloaded from the server. This option is only available in IE mode.

Certain IE registry keys influence the cache as well. See [Registry Settings that Influence Probe Results \(IE mode\)](#) on page 62 for more details.

**Repeat (IE mode).** If checked, the transaction is played back continuously. This option only controls repeat playback in recording mode within the Web Transaction Recorder application. It does not modify a probe's behavior because that is controlled by the probe target's interval setting. This option is only available in IE mode.

**Include Wait Time (IE mode).** When checked, includes the specified wait time (Wait Time (IE) field) in the response time calculation. This can be used to add user think time to each transaction step. This option is only available in IE mode.

**Scrub Cookies (IE mode).** When checked, all persistent cookies are deleted before the scheduler runs any probe. This only applies to IE mode. In URL mode, the probe does not store cookies.



Cookies are user specific and shared between all running transactions. Therefore, transactions that run at the same time and use the same cookie interfere with each other. Also, cookies are only deleted at the beginning of the largest interval (as specified in the Probe Location dialog box). For example, if one transaction runs every 300 seconds and the other transaction runs 140 seconds, cookies are only deleted every 300 seconds. In these cases, use a new probe system for probing or make sure that all transactions run at the same interval.

**Enable Java (IE mode).** If checked, Java applets are loaded and executed during playback. Interactions with Java are not supported.

**Initialize Browser (IE mode).** Uncheck if the transaction pops up a new window and playback fails in this new window; otherwise, JavaScript authentication errors are reported. This flag must be checked if the web server or proxy requires authentication. Alternatively, you can use `noAbout=1` to disable the initialization for a specific step. Set `noAbout` in the Advanced Step Properties dialog box. See `noAbout` in the Transaction Script Reference Table on page 84

**Capture Window on Error (IE mode).** This option takes a screen capture of the web page in case of a pattern-match failure or timeout. This can be useful in troubleshooting specific problems. The screen capture is saved under `<data dir>\data\tmp\probe\capture\<<Customer>_<Service Group>_<ProbeID>_<ServiceID>_<TargetID>.bmp` or `.gif`. Select whether to save the file as `.bmp` or `.gif`. Note that `.bmp` images are best viewed with the Imaging application. Depending on your screen resolution, MSPaint might not show a capture in `.bmp` correctly. Only the latest screen capture is available. If there is an existing screen capture, it is overwritten. You can override this default behavior using registry key settings (see [IE Mode Registry Settings on page 106](#)).

## Tracing Levels

**Tracing.** Select a level for tracing: 0=off, 1=minimum tracing, 5=high level of tracing, 9=maximum tracing. The trace is written to the message window in the recorder. For troubleshooting, it is recommended that tracing be set to level 9. See [Annotated Trace File for HTTP\\_TRANS on page 109](#) for an example.

## Proxy Settings

**Proxy Setting for Current Session.** Enter proxy setting and port as needed for this recording session. Use this to temporarily overwrite what is configured in the Probe Location dialog box for this HTTP\_TRANS probe.

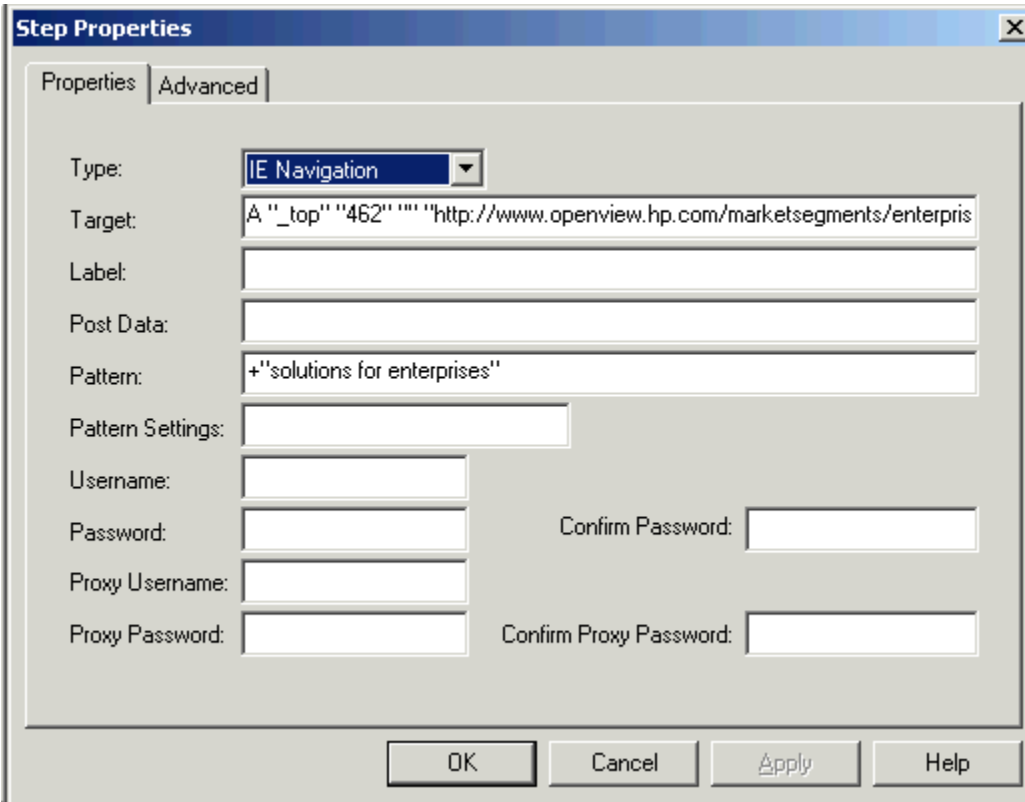
If you leave this blank, the probe will run using the Probe Location's proxy settings defined in the Configuration Manager.

If you have not configured the probe location yet, then leaving this blank means the probe will run with the IE connection settings for proxy.

To blank out or turn the proxy off, enter <no proxy> in this field.

## Transaction Step Properties Basic

Double-clicking on a transaction step displays the Step Properties dialog box that allows you to view the details for each step and specify additional information needed for the probe to correctly execute the step. See [Web Transaction Recorder Tips on page 95](#) and [Recording and Playback Issues on page 98](#) for some examples of when you would want to define these step properties to handle a problem with a particular step.



The screenshot shows the 'Step Properties' dialog box with the 'Advanced' tab selected. The 'Type' dropdown is set to 'IE Navigation'. The 'Target' field contains the XPath expression: 'A "\_top" "462" "" "http://www.openview.hp.com/marketsegments/enterpris'. The 'Pattern' field contains: '+solutions for enterprises'. The 'Pattern Settings' field is empty. The 'Username', 'Password', 'Proxy Username', and 'Proxy Password' fields are empty. The 'Confirm Password' and 'Confirm Proxy Password' fields are also empty. The dialog has 'OK', 'Cancel', 'Apply', and 'Help' buttons at the bottom.

**Type.** Select the transaction step type. The statement in the Target field must correspond with the type.

**Target.** Displays a statement describing the action the probe should take depending on the type. For IE mode, `iestep=URL`, `ienavpnt=HTML element`.

**Label.** Use this field to specify a label for this transaction step. The label is shown in the OVIS Dashboard drill-down instead of the URL, providing better readability.



Dynamic URLs could cause the Internet Services Dashboard to show multiple graphs for the same transaction step.

**Post Data.** HTTP Post data. In URL mode (or IEstep in IE mode), the probe does an HTTP POST operation and includes the data instead of a GET operation. There are two methods of sending data to the server: POST and GET with a query string. Checking this option sends HTTP POST data.

**Pattern.** Pattern matching. If pattern is not included in the text downloaded by the probe, the transaction is marked unavailable. This is preset in IE navigation mode with the title of the document. To look at the text downloaded by the recorder/probe, go to **View** → **View Source (HTML)**.



Certain HTML characters are encoded. For example, & is `&amp;` and " is `&quot;`; . When setting up a pattern, look at the HTML source to see whether the string displayed in the browser contains any encoded characters. If encoded characters are part of the string, adjust the pattern accordingly.

**Pattern Settings.** The default operator for word concatenation is AND. This can be changed here to OR. A word-compare is case sensitive by default. This can be changed to case insensitive by entering ICASE. For example, to enable case insensitive pattern-matching using a logical OR for your pattern, enter OR ICASE in this field.

**Username.** Username if transaction step requires HTTP authentication.

**Password.** Password if transaction step requires HTTP authentication.

**Confirm Password.** Retype the password you entered in the previous field.

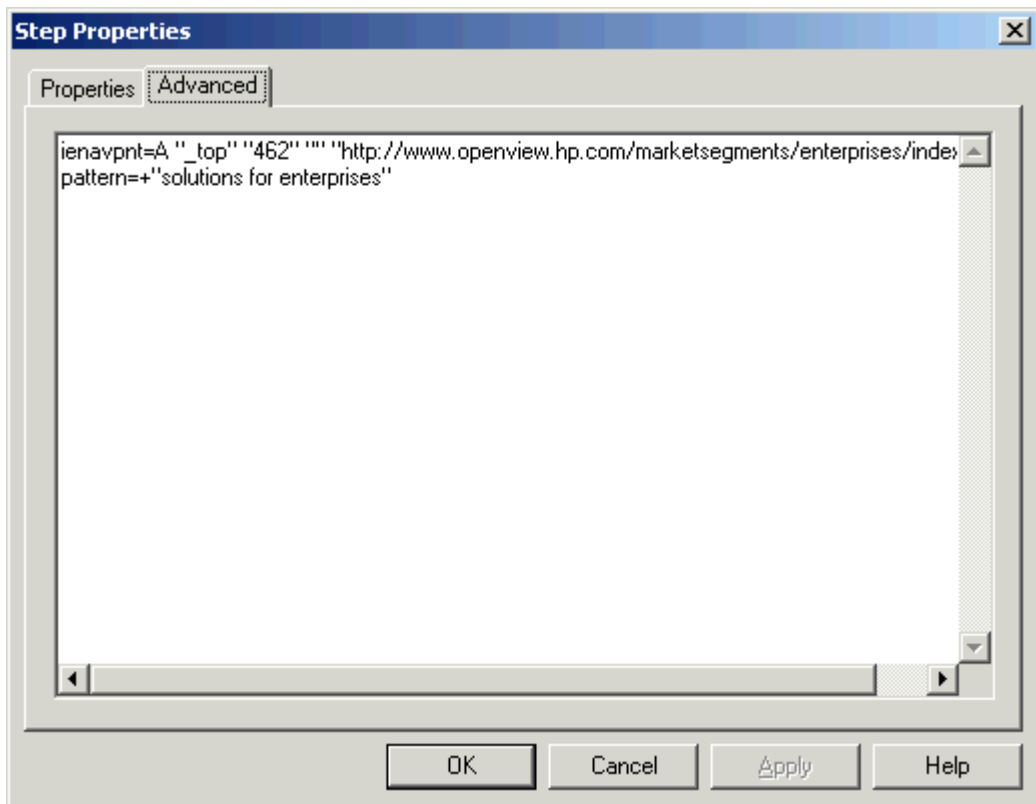
**Proxy Username.** Proxy Username if transaction step requires HTTP authentication by proxy.

**Proxy Password.** Proxy Password if transaction step requires HTTP authentication by proxy.

**Confirm Proxy Password.** Retype the password you entered in the previous field.

## Transaction Step Properties Advanced

Selecting the Advanced tab in the Step Properties dialog box displays a window that allows you to set up advanced scripting options for the transaction step. See [Advanced Usage on page 47](#) and [Transaction Script Reference on page 73](#) for information on the types of script statements available for handling complex web page recordings. Typically, you first record the transaction and then select Advanced Step Properties after the recording to add the specific script statements that are required to make the recording play back correctly.



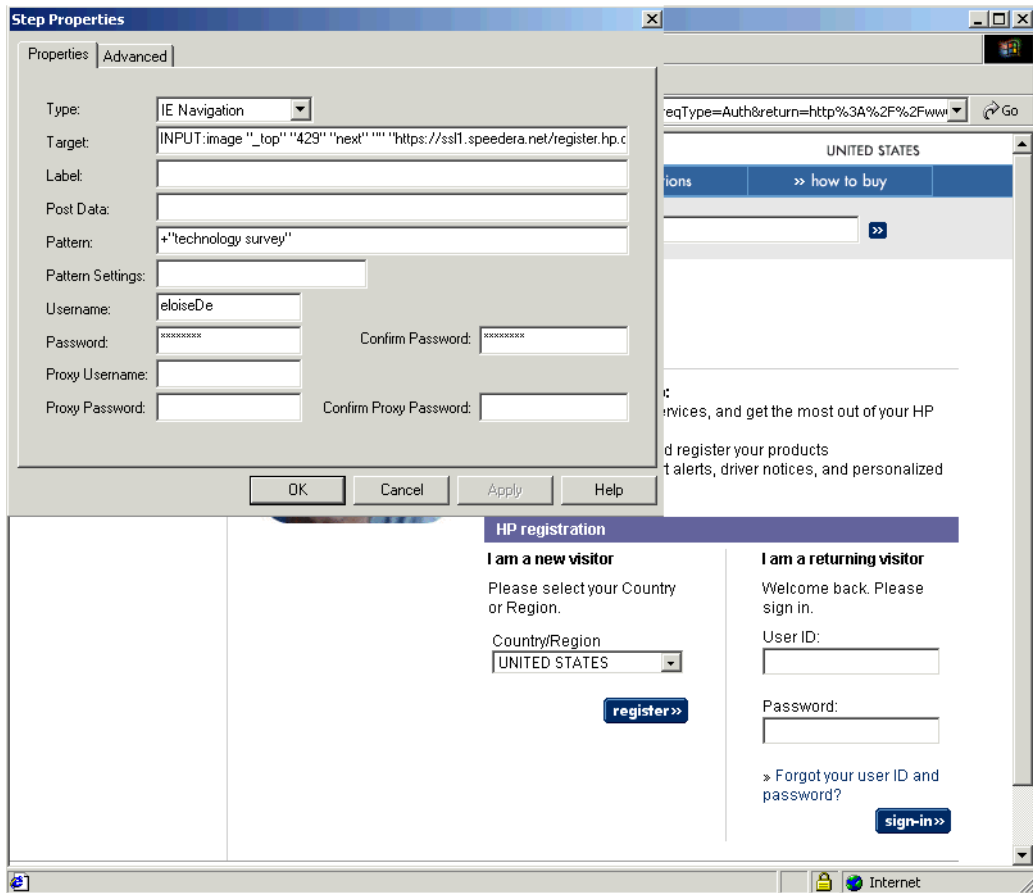
## Advanced Usage

See the following Advanced Topics for information on how to use script statements to handle more complex web functions. See [Transaction Script Reference on page 73](#) for details on the script statements available.

- [Internet Explorer Security Dialog Boxes](#)
- [Dynamic URL Substitution](#)
- [External Scripts for Form Data Input \(IE Mode\)](#)
- [External Script for Availability Check \(IE Mode\)](#)
- [Dynamic Frame Names](#)
- [Changing the Log On Account for the Probe](#)
- [Cookie Settings](#)
- [Loading and Saving Session Cookies](#)
- [Authentication](#)
- [Browser Plug-Ins](#)
- [Complex Page Redirections](#)
- [Duplicate Input Variables](#)
- [You can also use some registry settings to influence probe results. See Registry Settings that Influence Probe Results \(IE mode\) on page 62.](#)
- [HTML Dialog and Pull-down Menu Internals](#)
- [Position Independent Navigation Points](#)

## Internet Explorer Security Dialog Boxes

While recording a transaction, popup dialog boxes, such as security checks and certificates, may appear. These can be handled by manually adding the input information such as Username and Password to the transaction step after the recording. Use the Step Properties dialog box for the step to enter a Username and Password. See the example below.





Dialog boxes such as certificate selection can be handled by adding the `dlgCmd` statement to the transaction step after recording. Use the Advanced Step Properties dialog box after recording to add the script statement to the step. This causes the probe to execute a user action such as pressing the specified button. (See `dlgCmd` in the [Table on page 79](#).)

The following is an example `dlgCmd` statement.

```
dlgCmd=Button "Client Authentication" "OK"
```

The first value indicates the type (Button), followed by the Dialog title (Client Authentication) and button text (OK). This statement causes the probe to execute the user action of pressing the **OK** button in the Client Authentication dialog box.

More complex dialog boxes can be handled by the `dlgCmd=Sequence` `<sequence>` command. See the examples below.

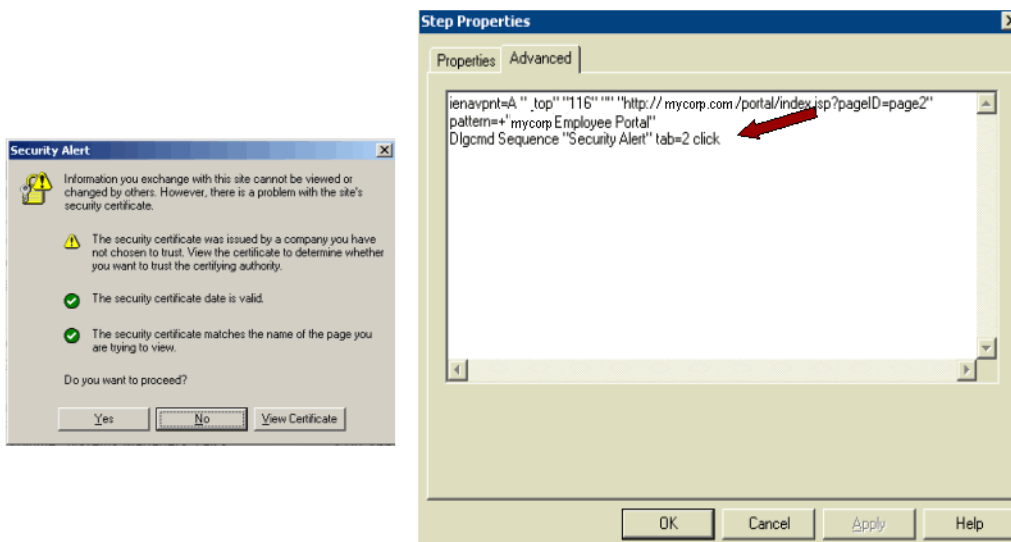
Add the command with either of the following syntaxes:

```
dlgCmd=button <Dialog caption> <button label>
```

```
dlgCmd=Sequence <Dialog caption> <seq-1>..<seq-n>
```

In these examples, `seq` can be `tab=x`, `up=x`, `down=x` or `click`.

In this example, `tab=2 click` indicates selecting the **No** button in the dialog box.



## Dynamic URL Substitution

When recording a transaction where steps consist of the type STEP/IESTEP, you can modify the transaction step to accommodate dynamic URLs. A dynamic URL is a URL where parts of the path or query string change every time you access the page.

Examples:

```
http://www.dyn.hp.com/12345/index.html
```

```
http://www.dyn.hp.com/test?sessionid=123455&arg=login
```

In the first example, the path contains the component 12345, which changes every time you access the main page. In the second example, the `sessionid` value 123455 in the query string changes every time. Some web applications encode this type of session identifier into the URL or POST data that changes with each visit to the web site.

### Dynamic URL Substitution Example

In the following example, `http://shopping.hp.com` creates two unique IDs in the query string of the URL:

```
http://www.shopping.hp.com/cgi-bin/hpdirect/shopping/scripts/  
computer_store/  
computer_landing_view.jsp?BV_SessionID=@@@0722665204.107040506  
3@@@&BV_EngineID=deadckdgjfgjgcfngcfkmdf11dfgg.0&landing=deskt  
ops&selectMenu=desktops
```

The two IDs, `BV_SessionID` and `BV_EngineID`, time out after a while. The transaction you recorded will not play back correctly because the IDs are no longer valid.

If a step is IESTEP (in IE mode) or STEP (in URL mode) and contains those session IDs, modify the transaction step to use the following commands to define a substitute valid. The substitution rule consists of the following:

<code>dynCmdPattern0</code>	Specifies the start of the string that should be used in the substitution.
<code>dynCmdDelim0</code>	Specifies the end of the string.
<code>dynCmdSubst0</code>	Specifies which part of the URL or POST should be replaced by the string.

These commands form a rule. The identifier of the rule is the last part of each command (0 in the table above). This allows multiple rules within a step. A rule uses information from the previous web page and modifies the URL or POST data on the step where the rule is defined.

The following is an example of the two-step recorded transaction:

```
[HTTP_TRANS]
IESTEP=http://www.shopping.hp.com/
IESTEP=http://www.shopping.hp.com/cgi-bin/hpdirect/shopping/
scripts/computer_store/
computer_landing_view.jsp?BV_SessionID=@@@0722665204.107040506
3@@@&BV_EngineID=deadckdgjfgjgcfngcfkmdfl1ldfgg.0&landing=deskt
ops&selectMenu=desktops
```

The web page in the first step (<http://www.shopping.hp.com/>) actually contains the current values for `BV_SessionID` and `BV_EngineID`. If you look at the web page <http://www.shopping.hp.com> (go to **View** → **Source**), you can see that the values are different from the ones you recorded in the URL of the second step. See an example below of the web page in the first step:

```
...
<SCRIPT language=JavaScript type=text/javascript>
<!--
var menuLink1 = "http://www.shopping.hp.com/cgi-bin/hpdirect/
shopping/scripts/computer_store/
computer_landing_view.jsp?BV_SessionID=@@@0573045601.107041251
8@@@&BV_EngineID=ccdeadckdgjfgjgcfngcfkmdfl1ldfgg.0&landing=des
ktops&selectMenu=desktops";var menuLink2 = ...
```

Note the difference in the recorded URL of the second step and the current URL that is contained in the web page [www.shopping.hp.com](http://www.shopping.hp.com). To handle the dynamic URL, a substitution rule is needed to define the `BV_SessionID` and `BV_EngineID` values that should be used in the second step.

To replace `BV_SessionID` and `BV_EngineID` in the second step, add the following two substitution rules.

```
IESTEP=http://www.shopping.hp.com/cgi-bin/hpdirect/shopping/
scripts/computer_store/
computer_landing_view.jsp?BV_SessionID=@@@1224306641.107041396
3@@@&BV_EngineID=xxdeadckdgjfgjgcfngcfkmdfl1ldfgg.0&landing=des
ktops&selectMenu=desktops
DYNCMDPATTERN0=BV_SessionID=
DYNCMDDELIM0=&
DYNCMDSUBST0=QUERY BV_SessionID
```

```
DYNCMPATTERN1=BV_EngineID=  
DYNCMDELIM1=&  
DYNCMDSUBST1=QUERY BV_EngineID
```

During playback, the following URL is replaced.

```
http://www.shopping.hp.com/cgi-bin/hpdirect/shopping/scripts/  
computer_store/  
computer_landing_view.jsp?BV_SessionID=@@@1224306641.107041396  
3@@@&BV_EngineID=xxdeadckdgjfgjgcfngcfkmdf1ldfgg.0&landing=des  
ktops&selectMenu=desktops
```

It is replaced with the following.

```
http://www.shopping.hp.com/cgi-bin/hpdirect/shopping/scripts/  
computer_store/computer_landing_view.jsp?BV_SessionID  
=@@@0573045601.1070412518@@@&BV_EngineID=ccdeadckdgjfgjgcfngc  
fkmdf1ldfgg.0&landing=desktops&selectMenu=desktops
```

The complete transaction looks like the following.

```
[HTTP_TRANS]  
IESTEP=http://www.shopping.hp.com/  
IESTEP=http://www.shopping.hp.com/cgi-bin/hpdirect/shopping/  
scripts/computer_store/  
computer_landing_view.jsp?BV_SessionID=@@@1224306641.107041396  
3@@@&BV_EngineID=xxdeadckdgjfgjgcfngcfkmdf1ldfgg.0&landing=des  
ktops&selectMenu=desktops  
DYNCMPATTERN0=BV_SessionID=  
DYNCMDELIM0=&  
DYNCMDSUBST0=QUERY BV_SessionID  
DYNCMPATTERN1=BV_EngineID=  
DYNCMDELIM1=&  
DYNCMDSUBST1=QUERY BV_EngineID
```

## Specifying a Script that Returns the Value

A script can be specified that returns the value for a particular PATH, POST, or QUERY. The HTTP\_TRANS probe supports getting values from a script in addition to getting values from the previous web page. This allows replacing parts of a URL, POST, or QUERY string with values generated by a script.

For example, if a date/time component is part of a POST and needs to be adjusted to a time in the future, a script can be developed that does the calculation and returns the value to the probe. This is similar to the script execution for FORMDATA in the IE mode mode.

Component:

```

dynCmdScript1=getid.sh
dynCmdSubst1=QUERY sessionid

```

The output of the script `getid.sh` replaces the value of the `sessionid` parameter. Without an absolute path, the scripts are executed from the `\probes` directory.

### Substitution Rule Elements and Location Index

You can add these rules in the **Advanced** tab of the Step Properties dialog box. To get a new line in this dialog box, press **CTRL-ENTER**.

Rules are grouped together by the index at the end of the rule element. For example, `dynCmdPattern0`, `dynCmdDelim0` and `dynCmdSubst0` all belong together.



To test a pattern, single-step through the transaction until you hit the page from which the next page will result in the dynamic URL. Go to **Tools** → **Test Pattern**, which brings up a dialog box with the current document. You can now specify and test a pattern.

The substitution rule element specifies the type and the variable/position that will be replaced. Possible values include the following:

```

QUERY <variable in query string>
PATH <location index>
POST <variable in POST string>
HOST <host portion of the URL>

```

#### Examples:

```

dynCmdSubst0=QUERY pageGenTime
dynCmdPattern1=AUTHMETHOD' value='
dynCmdDelim1='>dynCmdSubst1=POST AUTHMETHOD
dynCmdPattern2=pageGenTime' value='
dynCmdDelim2='>dynCmdSubst2=PATH 1

```

The location index can be negative to specify the position of the component from the right.

Example:

```
http://www.dyn.hp.com/first/second/third
```

**Table 2 Location Index**

0	substitutes the whole path (/first/second/third)
-1	third (or last component)
-2	would be second
-3	would be first
1	would be first (or first component)
2	would be second
3	would be third

### Advanced Substitution

If a standard dynamic substitution rule is not specific enough to work, another format option is available. For example, if you have the following URL:

```
“http://www.dyn.hp.com/dynamic/test&arg=login;sesssionid=123455=”;
```

To make the probe aware of the changing session ID in the last path component of the URL (test&arg=login;sesssionid=123455=“);), you must use a rule with a format option in it:

```
DynCmdPattern0=test&arg=login
DynCmdDelim0=“
DynCmdSubst0=PATH -1=test&arg=login%s
```

This says to search for the pattern “test&arg=login” in the last position and substitute the string that directly follows, which in this case is sessionid. The DynCmdSubst0 parameter substitutes the last path component of the path specified in the IESStep and replaces it with the string that follows the = sign (test&arg=login%s). Before the string is actually replaced, the %s is substituted with the content of the string that was specified by the pattern and delimiter. For example, if the pattern and delimiter found “;session=xyz=”, the new URL would look like the following:

```
http://www.dyn.hp.com/dynamic/test&arg=login;sesssionid=xyz=”;
```

Similarly, the search string can be used with POST. If the post looks something like the following, the rule will contain a search string.

```
post=Login=Guest&Password=test;sid=123455;IgnorePassword=false&
```

The search string consists of the following.

```
DynCmdPattern0=Password=test
```

```
DynCmdDelim0=;
```

```
DynCmdSubst0=POST Password=test%s
```

## External Scripts for Form Data Input (IE Mode)

In IE mode, it is possible to set input variables through a script. For example, consider a train schedule web application that requires the starting date of a train schedule. The application might not support dates that are in the past or too far in the future. You can create a script that calculates a valid date. This script can be specified in a form element.

**Example:**

Originally, the recorder would have recorded the following transaction step:

```
IESTEP=http://www.travel.hp.com/schedule?get
FORMDATA=date1=05-23-2001
FORMDATA=date2=06-01-2001
```

To specify scripts for input elements `date1` and `date2`, add `@` in front of the input element name, and replace the value with a script:

```
IESTEP=http://www.travel.hp.com/schedule?get
FORMDATA=@date1=c:\temp\t.bat
FORMDATA=@date2=cscript //nologo t.vbs
```

If scripts are specified as relative, it is recommended that these scripts are distributed by OVIS to the probe location (from `<install dir>\newconfig` to `<data dir>\bin\instrumentation\probe\scripts`). The WebRecorder appends the `<data dir>\bin\instrumentation\probe\scripts` directory to the PATH environment variable so that the scripts can be located by the operating system. In addition, the OVIS\_SCRIPTS environment variable also points to the `<data dir>\bin\instrumentation\probe\scripts` directory and can be used as follows:

```
cmd /c perl "%OVIS_SCRIPTS%\chk.pl" OpenView
```

This is necessary if a script engine doesn't use the PATH variable to locate the program script.

Alternatively, the scripts can be placed in the <install dir>\bin directory. However, file name clashes with other OpenView products may occur.

The following is an example Visual Basic script that returns tomorrow's date:

```
Set WshShell = WScript.CreateObject("WScript.Shell")
dteCurrent = Now()+1
dteDay = Day(dteCurrent)
dteMonth = Month(dteCurrent)
dteYear = Year(dteCurrent)
MyDate = dteDay &"-" & dteMonth & "-" & dteYear
Wscript.echo MyDate
```

## External Script for Availability Check (IE Mode)

The IE mode probe supports executing an external script to do an advanced availability check. Usually, the availability of a step is determined by the specified pattern, the timeout, or an HTTP status code other than 200.

A script can be used for more complex availability checks on the web page. The probe passes a filename containing the HTML of the web page (including all frames) to the script and reads a return value indicating availability ("1") or unavailability ("0").



If a pattern and a script are specified, the pattern is ignored and only the script is used for availability checking.

If scripts are specified as relative, it is recommended that these scripts are distributed by OVIS to the probe location (from <install dir>\newconfig to <data dir>\bin\instrumentation\probe\scripts). The WebRecorder appends the <data dir>\bin\instrumentation\probe\scripts directory to the PATH environment variable so that the scripts can be located by the operating system. In addition, the OVIS\_SCRIPTS environment variable also points to the <data dir>\bin\instrumentation\probe\scripts directory and can be used as follows:

```
cmd /c perl "%OVIS_SCRIPTS%\chk.pl" OpenView
```

This is necessary if a script engine doesn't use the PATH variable to locate the program script.



Alternatively, the scripts can be placed in the `<install dir>\bin` directory. However, file name clashes with other OpenView products may occur.

Example:

The `checkAvailability` script statement can be added to a recorded step in the **Advanced Step Properties** dialog box:

```
checkAvailability=perl match.pl OpenView
```

The `match.pl` script is the following Perl script:

```
-----
open FH,$ARGV[1];
undef $/;
if (defined($block = <FH>)) {
    if ($block =~ /$ARGV[0]/) {
        print "1";
    }
    else {
        print "0";
    }
}
else {
    print "0";
}
close FH;
unlink($ARGV[1]);
-----
```

The probe appends the file name containing the HTML of the web page to the command line specified by `perl match.pl OpenView`. In the `match.pl` script, the argument 0 is the string "OpenView", and the file name with the HTML text is argument 1. The script reads the HTML text. If it finds the pattern "OpenView" in the input file, the script prints 1; otherwise, it prints 0.

Lastly, it closes the file and removes the file containing the HTML.



It is important to remove the file because the probe will not remove the file.

## Dynamic Frame Names

Some frame windows are created by JavaScript, which may lead to a randomly generated frame name. This may cause problems during playback if form data was recorded in this frame. An error like the following is displayed:

```
ERROR: Unable to find frame "1234567890".
```

The following example shows a transaction step with dynamic frame names:

```
IENAVPNT=A "1234567890" "252" "button"  
"javascript:addToCartUsingToggle();"   
FORMDATA=1234567890.items.item=Submit=True
```

In the IENAVPNT and FORMDATA statement, the items form is located on the 1234567890 frame. Because the form name may change the next time the transaction is played back, the probe will no longer find the form data location.

To work around this issue, you can either remove the frame name (leave it unspecified), which uses the top-most frame, or specify the index of the frame.

**Example A:**

```
IENAVPNT=A "" "252" "button"  
"javascript:addToCartUsingToggle();"   
FORMDATA=.items.item=Submit=True
```

**Example B:**

```
IENAVPNT=A "[2]" "252" "button"  
"javascript:addToCartUsingToggle();"   
FORMDATA=[2].items.item=Submit=True
```

In Example A, the frame name is empty in IENAVPNT and FORMDATA. This instructs the probe to use the top-most frame. In Example B, the correct frame is frame 2 in the frame collections object. To determine the frame number, use the following menu item in the Main Web Transaction Recorder window: **View > Source (DOM)**. This lists all the HTML element indexes and frame names.

## Changing the Log On Account for the Probe

By default, the IE mode probe runs under the SYSTEM account. This account is usually different from the account used to record the transaction. Cookies and client certificates are all associated with an account. As a result, different behavior can be observed when the transaction is being run. To run the transaction under a different account, run the **Services** control panel applet and change the startup parameters for the HP Internet Services account. Enter the Username and Password of the account under which the recording was performed. Note that you can also enter a user in the Run As User field on the initial dialog when entering the Web Transaction Recorder.

## Cookie Settings

Some web sites use cookies to save various settings (for example, language settings) and direct users to a different page than the one they came in from originally. To get consistent transaction results, it is recommended to set the **Scrub Cookies (IE)** flag in the Configure dialog box. This instructs the scheduler to delete all cookies before a transaction probe is run.



This is a global flag and cannot be selected for an individual transaction.

## Loading and Saving Session Cookies

You can also save and load session cookies from a file by specifying the `SAVESESSION` or `LOADSESSION` statements in the Advanced Step Properties dialog box. See [Transaction Script Reference on page 73](#) and look for the `SAVESESSION` statement in the table.

## Authentication

Because the recorder handles the authentication dialog box for web pages that are protected by Username and Password, the usual IE authentication dialog box does not show up. Instead, the step fails. When it fails, double-click the step and enter Username and Password information in the Step Properties dialog box. Replay the transaction. After the transaction replays, continue the recording by pressing the **Record** button.

## Browser Plug-Ins

Navigation in browser plug-ins (for example, Macromedia FLASH) must be added manually as IE Steps.

To do this, first turn tracing to level 1 in the Web Transaction Recorder. This shows the URL and POST string to which the browser is navigating. For example, the log may show `Loading: url='http://ww.hp.com/', post='`.

After the page completely loads, create a new step (right-click in the transaction step list box and choose **Add** from the pop-up menu). Copy the URL and POST string (if applicable) shown in the log to the newly created step.

## Complex Page Redirections

Some web applications perform redirections to other pages during login verification or account lookup. In these instances, Web Transaction Recorder may prematurely end the transaction step during multiple redirections.

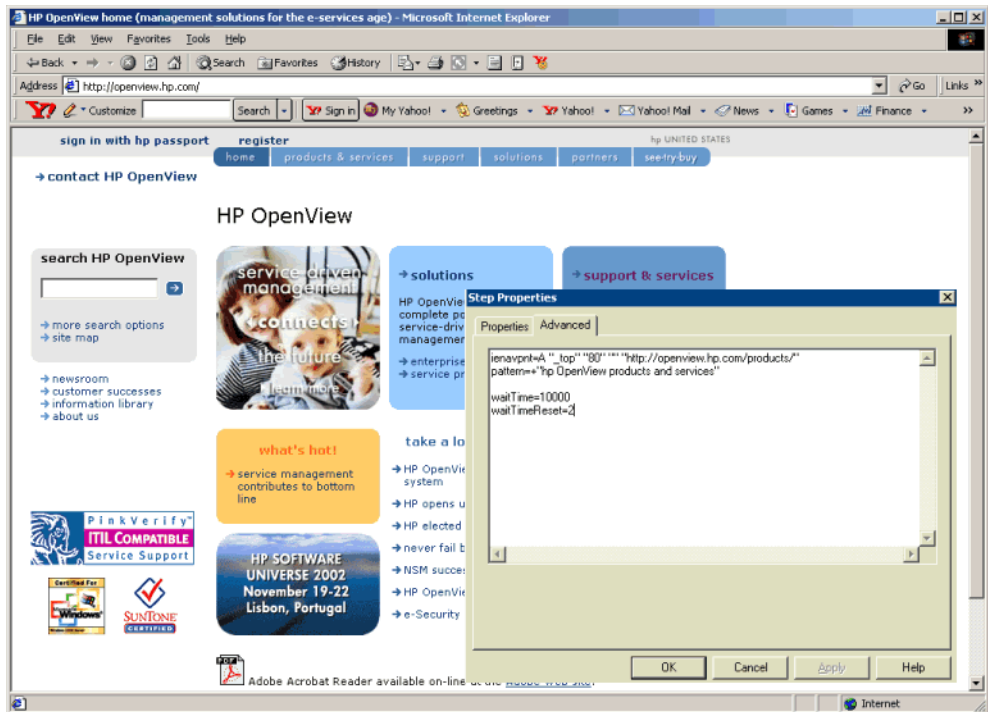
As a default Web Transaction Recorder uses a 1.5-second timeout to determine whether a page was completely loaded. If this timeout is not enough to allow for the page redirections, you can increase it by manually adding a `waitTime` value after the recording. Use the Advanced Step Properties dialog box to add `waitTime` and `waitTimeReset`. The `waitTimeReset` value resets the timeout after some number of redirections.

### Example:

To set the wait time for redirection detection to 10 seconds, right-click the specific step and choose **Properties** from the pop-up menu. Click the **Advanced** tab to see the advanced properties of the step. Add `"waitTime=10000"` at the end of the dialog box. (Press **CTRL-ENTER** to add a new line.) The `waitTime` property requires the value to be in milliseconds.

Alternatively, it is possible to instruct the probe to determine the end of a transaction step by specifying the exact number of documents that a step downloads from the server. For each document that is downloaded, the trace shows the `DocCompletes` counter; for example, `DocCompletes: 31`. The probe can be instructed to wait for an exact number of these `DocCompletes` to determine that a transaction step is completed. Simply enter `"docCompletes=<number of document completes from trace>"` in the Advanced Properties dialog box of a step.

In this example, if the web page had two redirections, the wait time would be set to 10 seconds after the first redirection, and 1.5 seconds (the global default) after the second.



## Duplicate Input Variables

If a FORM has no name, multiple `formData` statements with the same input element may appear and cause incorrect navigation. This can be seen in the Advanced Properties dialog box.

Example:

```
formData=_top..index=music
formData=_top..group=e-card-recommendations=True
formData=_top..index=foo
formData=_top..group=bar=True
```

In the example, two unnamed forms use the same name for input elements, which may cause navigation problems. The workaround is to delete the `formData` statements causing problems (usually the last ones since the probe tries to set the input elements from top to bottom).

## Registry Settings that Influence Probe Results (IE mode)

Certain flags that are accessible in IE under **Internet Options** → **Advanced and Internet Options** → **Settings** influence how IE downloads content.

Because these flags are saved per user, the IE mode probe requires registry changes to enable these flags if the scheduler runs under the SYSTEM account (default).

IE stores these flags under two registry entries:

```
HKEY_USERS\DEFAULT\Software\Microsoft\Windows\CurrentVersion\Internet Settings
```

```
HKEY_USERS\DEFAULT\Software\Microsoft\Internet Explorer
```

Names of more flags can be found on the Microsoft support page or by observing the current users keys under

```
HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Internet Settings
```

```
HKEY_CURRENT_USER\Software\Microsoft\Internet Explorer
```

The following registry keys influence how IE downloads content:

- **Internet Options** → **Settings: Check for newer versions of stored pages**

Add the `SyncMode5` DWORD key under

```
HKEY_USERS\DEFAULT\Software\Microsoft\Windows\CurrentVersion\Internet Settings
```

The data values for the `SyncMode5` value include:

- 0: Never
- 2: Every time you start Internet Explorer
- 3: Every visit to the page
- 4: Automatically

- **Internet Options** → **Advanced: Use HTTP 1.1 and Use HTTP 1.1 through proxy connections**

Add the `EnableHttp1_1` `DWORD` and `ProxyHttp1.1` `DWORD` key under

```
HKEY_USERS\.DEFAULT\Software\Microsoft\Windows\CurrentVersion\Internet Settings
```

The data values for the keys include:

1: Enable

- **Internet Options → Advanced: Disable script debugging**

Add the "Disable Script Debugger" string value under

```
HKEY_USERS\.DEFAULT\Software\Microsoft\Internet Explorer\Main
```

The data values for the keys include:

yes: script debugger is disabled

no: script debugger is enabled

- **Internet Options → Advanced: Empty Temporary Internet Files folder when browser is closed**

Add the `Persistent` `DWORD` key under

```
HKEY_USERS\.DEFAULT\Software\Microsoft\Windows\CurrentVersion\Internet Settings\Cache
```

The values for the `Persistent` key include:

1: do not empty Temporary Internet Files folder

0: empty Temporary Internet Files folder

## HTML Dialog and Pull-down Menu Internals

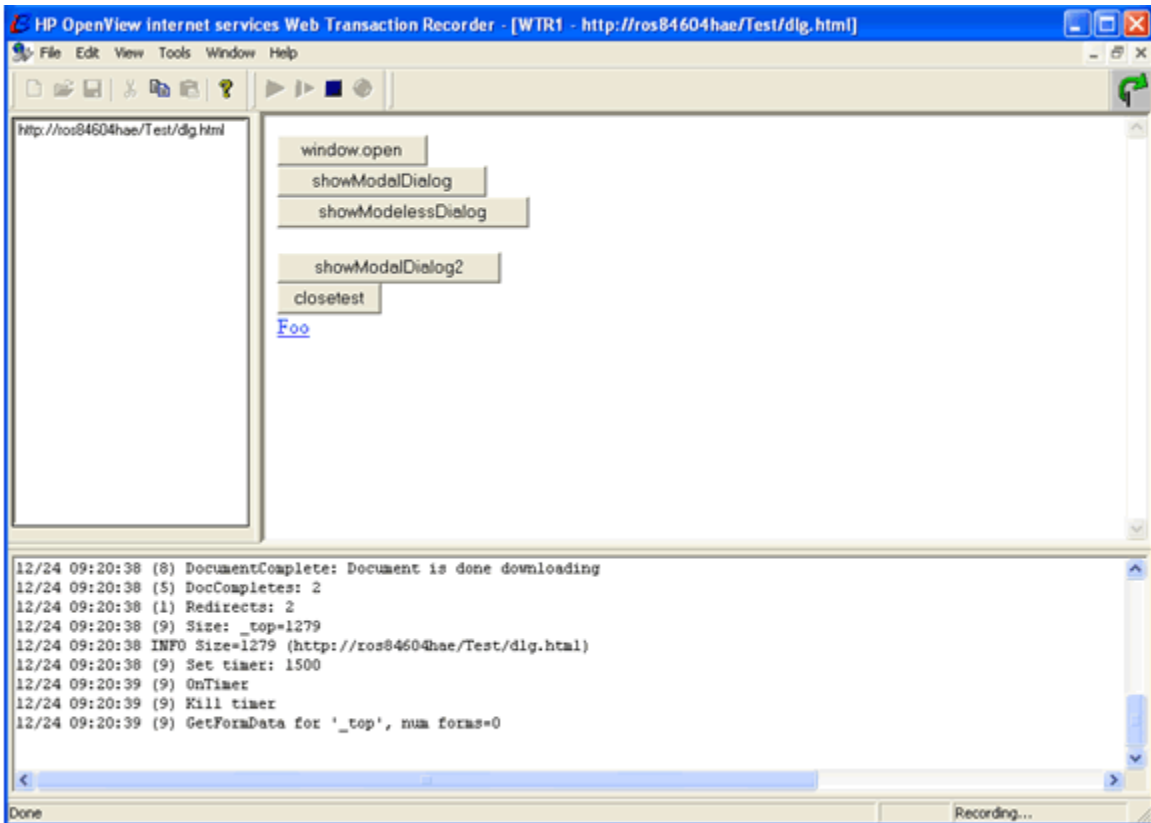
### HTML Dialogs

HTML dialogs are modal or modeless dialogs that are opened through the following JavaScript functions:

```
window.showModelessDialog("http://www.hp.com");
window.showModalDialog("http://www.hp.com");
```

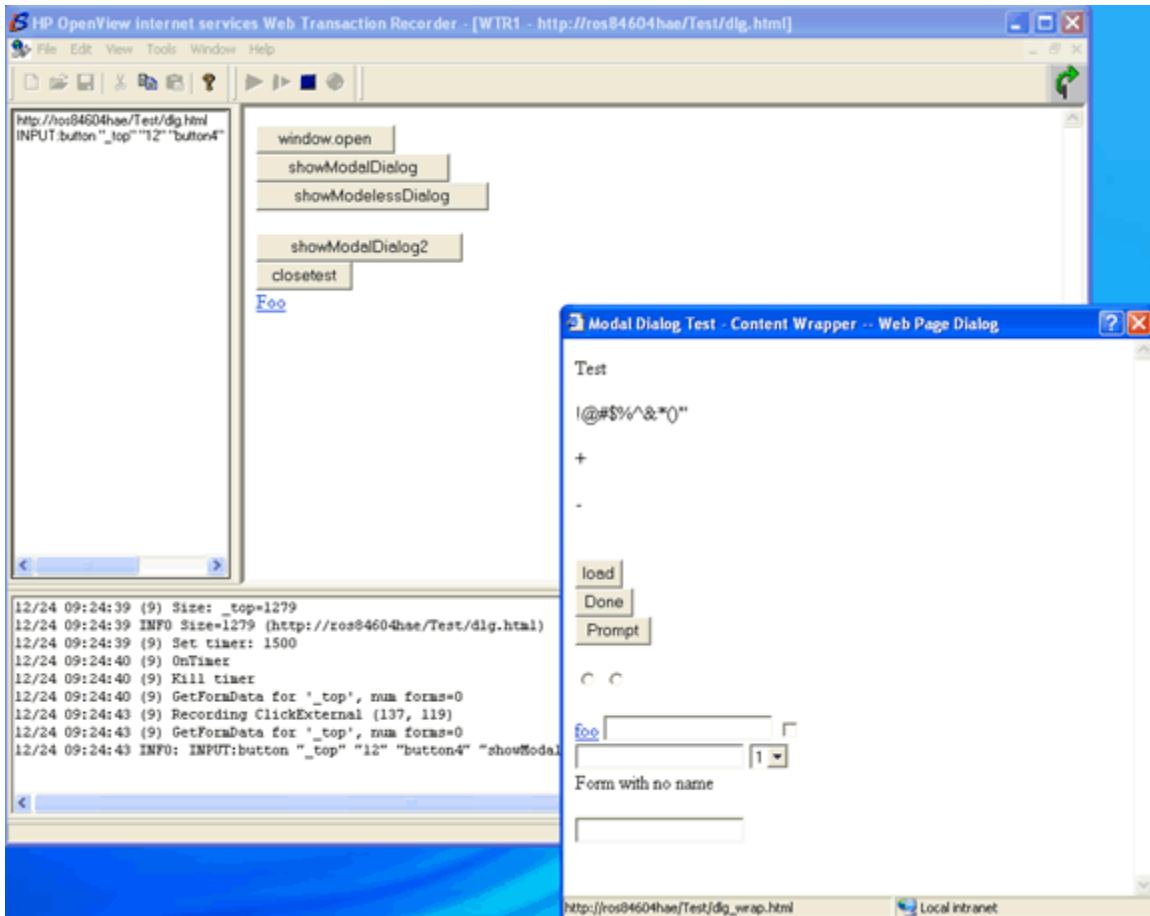
The two functions allow the application to create a modeless or modal dialog box that displays the specified HTML document. Windows opened through these functions are not part of the IE Object Model and WebRecorder will show these dialogs as new windows outside of the MDI (Multi-Document Interface) view.

The following screen shots show a transaction that opens a modal dialog. After navigating to the start URL, WebRecorder shows a single window inside the MDI view with a test page that allows opening of dialogs.



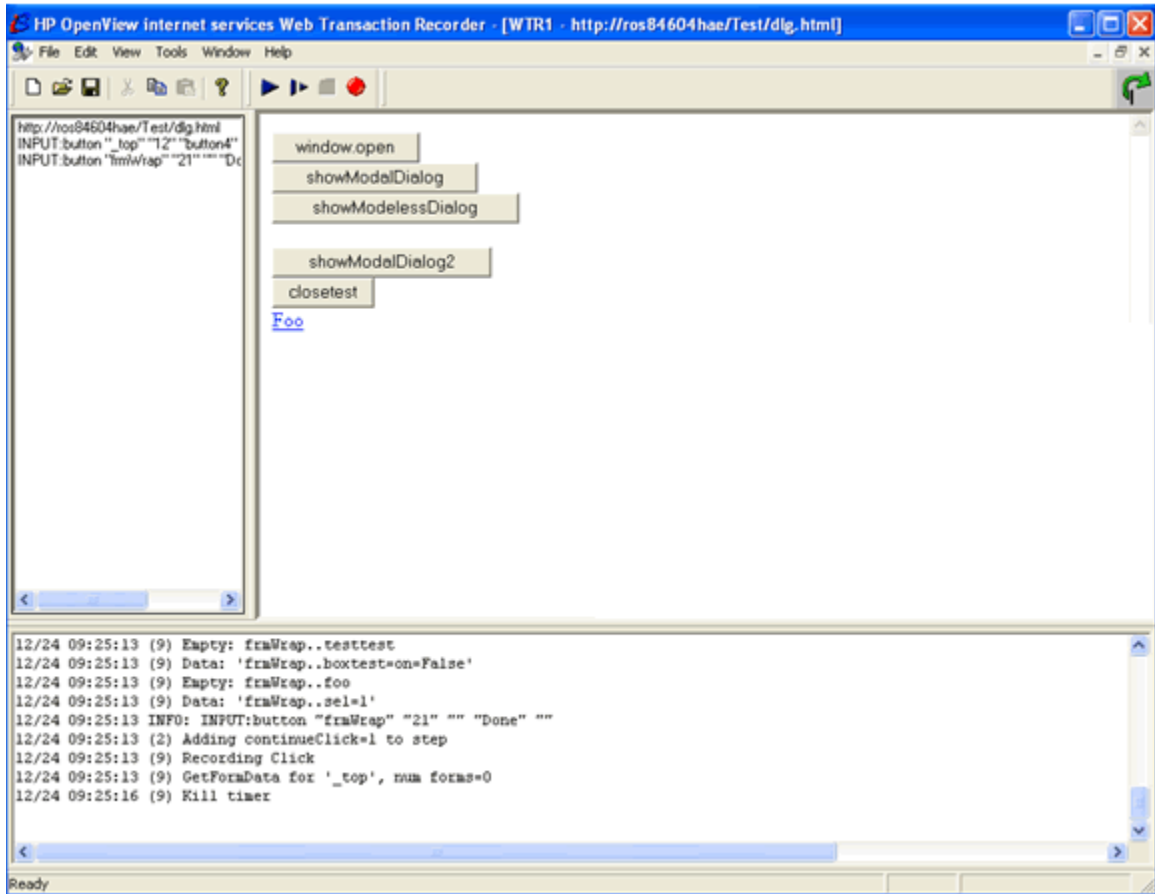
After pressing the "showModalDialog2" button, the modal HTML dialog window "Modal Dialog Test ..." is opened outside of WebRecorder. The code for opening the dialog is "window.showModalDialog("dlg\_wrap.html");".





After pressing the "Done" button in the HTML dialog window, the dialog is closed (`window.close()` in the button's onclick handler).

WebRecorder detects the close of the window and records a step for this action:



Here's the complete script:

```
[HTTP_TRANS]
```

```
FLAGS=cache=0,silent=0,noNewWindow=0,waitTime=1500,incWaitTime=0  
,runJava=1,aboutBlankInit=1,IgnoreExitExceptions=0,captureWindow  
=0
```

*Step 1 (Initial URL):*

```
IESTEP=http://ros84604hae/Test/dlg.html
```

```
PATTERN="+\"Html Dialog Test Page\"
```

**Step 2 (Click on "showModalDialog2"):**

```
IENAVPNT=INPUT:button "_top" "12" "button4" "showModalDialog2"
""
```

```
PATTERN="+Modal Dialog Test - Content Wrapper"
```

**Step 3 (Click on "Done"):**

```
IENAVPNT=INPUT:button "frmWrap" "21" "" "Done" ""
```

```
FORMDATA=frmWrap.form2.CARS=on=False
```

```
FORMDATA=frmWrap.form2.CARS=on=False
```

```
FORMDATA=frmWrap..boxtest=on=False
```

```
FORMDATA=frmWrap..sel=1
```

```
CONTINUECLICK=1
```

```
HTMLDIALOG=Modal Dialog Test - Content Wrapper -- Web Page Dialog
```

When clicking on the "Done" button in the dialog, WebRecorder adds two additional parameters for the step:

- **HTMLDIALOG=<caption of the HTML dialog window>**
- **CONTINUECLICK=1** (instruction to not wait for any page navigation)

The **HTMLDIALOG** parameter tells WebRecorder during playback that it needs to find an HTML dialog window with the specified caption (window title). This is necessary since these dialog windows are not part of the IE Object Model and therefore WebRecorder doesn't get notified by the IE API that such a window has been created (only standard `window.open()` calls generate the "NewWindow" notification to the application that hosts IE). In order to find the window during playback, WebRecorder iterates through all windows that belong to the current process and tries to find a window that matches the caption specified by the **HTMLDIALOG** parameter). Once the window has been found, WebRecorder can access the Object Model of the window and perform the instructions in the step (i.e. setting form values and simulating the click on the button).

The **CONTINUECLICK=1** parameter is necessary because the closing of the HTML dialog doesn't generate a notification either: WebRecorder waits for a "BeforeNavigate" event to detect that the current step has ended and a new step is about to start. Since the HTML dialog doesn't generate such an event, WebRecorder adds this parameter which tells it to not wait for an event and simply continue on with the next step.

### Special handling for pull-down menus that are dynamically loaded

Pull-down menus that dynamically load information from the server depending on the previous selection require some special handling. Consider the following example:

Selection 1:

Selection 2:

Selection 3:

In the above example, a selection of pull-down menu "Selection 1" sets the items of "Selection 2":

Selection 1:

Selection 2:

Selection 3:

After selecting an item in "Selection 2", the items for pull-down menu "Selection 3" are set:

Selection 1:

Selection 2:

Selection 3:

After selecting an item in "Selection 3", the form is send to the server via the submit button.

The standard recording will record one step

```
INPUT:image "_top" "296" "Submit"
```

after all pull-down menus have been selected and "Submit" has been clicked.

The playback will **fail** since the web application requires that each pull-down menu is selected individually. WebRecorder, however, tries to set all pull-down menus before the "Submit" button is pressed (formdata parameter):

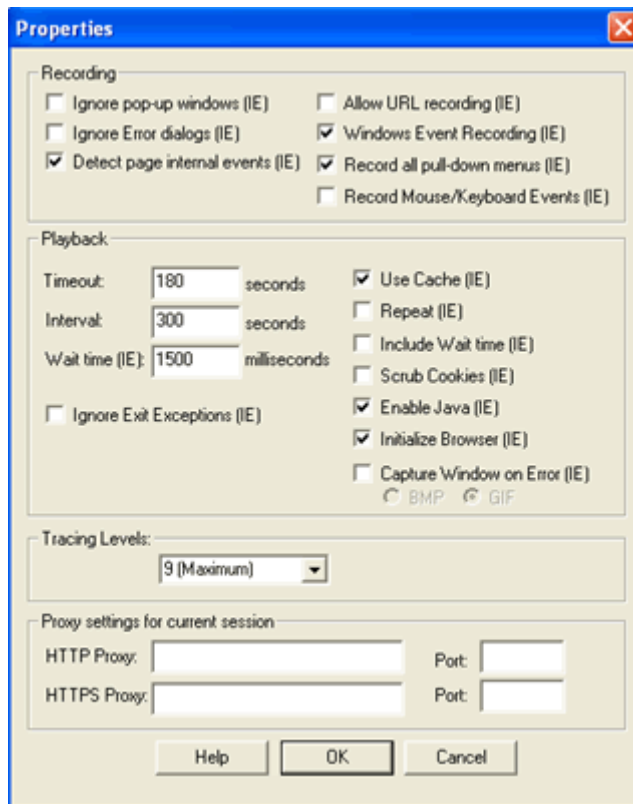
```
ienavnpt=INPUT:image "_top" "296" "submit" "Submit" "http://
server/app/submit.gif"
```

```
formData=_top.AForm.Selection1=1 - Value B
```

```
formData=_top.AForm.Selection2=2 - Value A
```

```
formData=_top.AForm.Selection3=3 - Value C
```

In order to simulate the individual setting of the pull-down menus, the WebRecorder needs to set the "Record all pull-down menu" option under File->Configure:



After selecting the option and filling out the page, the transaction will record additional steps for each pull-down menu:

```
ienavpnt=SELECT "_top" "125" "Selection1" "1"  
formData=_top.AForm.Selection1=1 - Value B
```

```
ienavpnt=SELECT "_top" "178" "Selection2" "37"  
formData=_top.AForm.Selection1=1 - Value B  
formData=_top.AForm.Selection2=2 - Value A
```

```
ienavpnt=SELECT "_top" "235" "Selection3" "2"
```

```
formData=_top.AForm.Selection1=1 - Value B
```

```
formData=_top.AForm.Selection2=2 - Value A
```

```
formData=_top.AForm.Selection3=3 - Value C
```

Note that the above three "SELECT" navigation points have been inserted before the final submit step.

However, if the transaction is played back, the web application will still return an error when the first SELECT navigation point is executed.

This is due to the default behavior of the SELECT navigation point where it tries to execute the action of the HTML form in which it is contained in. Since the form action of the web page is submitting the data to the server when the "Submit" step is pressed, the web application will not have all the information of the pull-down menus.

The default behavior of the SELECT navigation point can be altered by adding "nosubmit" at the end of the SELECT text. The "nosubmit" flag instructs WebRecorder to simply set the pull-down menu and not execute the form action.

In addition to the "nosubmit" change, the navigation point also needs a "continueClick=1" instruction and all the formData statements except for the one that corresponds to the pull-down menu need to be removed.

After altering the three SELECT navigation points, they will look like the following:

```
ienavpnt=SELECT "_top" "125" "Selection1" "1" "noSubmit"
```

```
formData=_top.AForm.Selection1=1 - Value B
```

```
continueClick=1
```

```
ienavpnt=SELECT "_top" "178" "Selection2" "37" "noSubmit"
```

```
formData=_top.AForm.Selection2=2 - Value A
```

```
continueClick=1
```

```
ienavpnt=SELECT "_top" "235" "Selection3" "2"
```

```
formData=_top.AForm.Selection3=3 - Value C
```

```
continueClick=1
```

Now the transaction will be able to correctly playback the pull-down menus.

## Position Independent Navigation Points

If a navigation point (ienavpnt) can't be uniquely identified by its attributes like index and name, it may be possible to use the procedure below to locate the correct navigation point.

Consider the following web page:

```
<a href=x1.html>X1</a><br>
<a href=x2.html>X2</a><br>
<a href=x3.html>X3</a><br>
```

Clicking on X2 in Web Transaction Recorder records the following navigation point:

```
ienavpnt=A "_top" "6" "" "http127.0.0.1/Test/moving/x2.html"
```

If the web page changes constantly (when lists of information are returned), the playback might not play back the correct link. For example, if the above web page changes to:

```
<a href=x1.html>X2</a><br>
<a href=x2.html>X1</a><br>
<a href=x3.html>X3</a><br>
```

the playback might fail.

The navigation point can be replaced by to search for the markup and perform a click on the element that contains the markup. With the above example, the navigation point can be changed to the following:

```
ienavpnt=!Find A "_top" "X2"
```

This will instruct the playback to look for the markup X2 in all links ("A" tag) in frame "\_top". If a case insensitive match of the markup specified in the navigation point is found, the WebRecorder performs the click on the element that contains the markup.

In addition to !Find, there is !FindEx which executes a script that has to return the markup to standard out (stdout):

```
ienavpnt=!FindEx "A" "top" "cmd /c c:\temp\findex.bat"
```



## Transaction Script Reference

The following tables give you details on the different script statements you can use in handling complex web pages. The script statements are entered manually in the Advance Step Properties dialog box after recording a transaction.

### IE Mode

**Table 3 IE Mode Script Statements**

Statement	Arguments	Comment
iestep	URL	<p>Navigates to the specified URL.</p> <p><b>Example:</b></p> <pre>iestep=http://www.hp.com/go/openview</pre> <p>The default is to load the specified URL in iestep into the main frame (_top). Depending on the web application, it may be necessary to specify the frame in which the URL should be loaded. To specify a frame, use the frame statement.</p> <p><b>Example:</b></p> <pre>iestep=http://www.hp.com/go/ frame=salesFrame</pre>
frame	frame name	<p>Specifies the frame into which the URL specified by iestep should be loaded. Use this statement to direct the content to the specified frame.</p>

**Table 3 IE Mode Script Statements (cont'd)**

Statement	Arguments	Comment
ienavpnt	IE Navigation Point	<p>Locates the navigation point in the current URL and simulates a user clicking on the found HTML element.</p> <p><b>Formats:</b></p> <pre>A "frame" "index" "name" "url" AREA "frame" "index" "url" INPUT:type "frame" "index" "name" "value" "url" (type can be BUTTON, IMAGE or SUBMIT)</pre> <p>The first part of the statement specifies the HTML tag, followed by the frame where the element is located. The index specifies the location of the element. (An automatic search is done if the element is not found at the index.) Name specifies the value of the name attribute. urlL specifies the value of the href tag, which is part of the element.</p> <p>The “frame” can be a name (“_top”), a name with index for duplicate frame names (“_top[2]”), or an index that specifies the position in the frames collection (“[2]”).</p> <p><b>Example:</b></p> <pre>ienavpnt=A "_top" "151" "" "http:// www.hp.com/"</pre>

**Table 3 IE Mode Script Statements (cont'd)**

Statement	Arguments	Comment
ienavpnt	IE Navigation Point	<p>Locates the navigation point in the current URL and simulates a user clicking on the found HTML element.</p> <p><b>Formats:</b></p> <pre>A "frame" "index" "name" "url" AREA "frame" "index" "url" INPUT:type "frame" "index" "name" "value" "url" (type can be BUTTON, IMAGE or SUBMIT)</pre> <p>The first part of the statement specifies the HTML tag, followed by the frame where the element is located. The index specifies the location of the element. (An automatic search is done if the element is not found at the index.) Name specifies the value of the name attribute. urlL specifies the value of the href tag, which is part of the element.</p> <p>The “frame” can be a name (“_top”), a name with index for duplicate frame names (“_top[2]”), or an index that specifies the position in the frames collection (“[2]”).</p> <p><b>Example:</b></p> <pre>ienavpnt=A "_top" "151" "" "http:// www.hp.com/"</pre>

**Table 3 IE Mode Script Statements (cont'd)**

Statement	Arguments	Comment
formdata	input element	<p>Specifies data entered into the web page during recording. Scripts and absolute positions in a SELECT element (selection box) can be specified by the @ character.</p> <p><b>Format:</b></p> <pre>frame.form.element=&lt;value&gt;</pre> <p>The “frame” can be a name (“_top”), a name with index for duplicate frame names (“_top[2]”), or an index that specifies the position in the frames collection (“[2]”).</p> <p><b>Example:</b></p> <pre>formdata=_top.form1.index=blended formdata=_top..group=credit=True formdata=.form2.search=foo formdata=@date1=c:\temp\t.bat arg1 arg2 formdata=@date2=cscript //nologo t.vbs formdata=form.sel=@1</pre>
post	HTTP Post string	<p>Specify the data used in a HTTP POST operation (iestep in IE mode only)</p> <p><b>Example:</b></p> <pre>post=product_oid=427&amp;arg=search</pre>

**Table 3 IE Mode Script Statements (cont'd)**

<b>Statement</b>	<b>Arguments</b>	<b>Comment</b>
pattern	Pattern string	<p>Specify the pattern. If the pattern is not found, the step is considered to be unavailable.</p> <p>Certain HTML characters are encoded. For example, &amp; is &amp;amp; and " is &amp;quot;. When setting up a pattern, look at the HTML source to see whether the string displayed in the browser contains any encoded characters. If encoded characters are part of the string, adjust the pattern accordingly.</p> <p><b>Example:</b></p> <pre>pattern="+HP OpenView"</pre>
patternconfig	Pattern options	<p>The default operator for word concatenation is AND. This can be changed to OR. A word compare is case sensitive by default. This can be changed to case insensitive by entering ICASE.</p>
username	Username	<p>Username to be used in HTTP authentication.</p> <p><b>Example:</b></p> <pre>username=ovros</pre>
password	Password	<p>Password to be used in HTTP authentication. Note the password is encrypted when you enter them in the Advanced Step Properties dialog.</p> <p><b>Example:</b></p> <pre>password=artros</pre>
proxyUsername	Username	<p>Username to be used in HTTP proxy authentication.</p> <p><b>Example:</b></p> <pre>proxyUsername=secureros</pre>

**Table 3 IE Mode Script Statements (cont'd)**

<b>Statement</b>	<b>Arguments</b>	<b>Comment</b>
proxyPassword	Password	Password to be used in HTTP proxy authentication. Note the password is encrypted when you enter them in the Advanced Step Properties dialog.  Example: proxyPassword=artros
label	Label	Symbolic name of the transaction step.  Example: label=Check-out

**Table 3 IE Mode Script Statements (cont'd)**

Statement	Arguments	Comment
dlgCmd	Type Dialog Title Button Title or Type Dialog Title Sequence	<p>The command can be used to navigate in an IE dialog box and press a button. It has two formats:</p> <ul style="list-style-type: none"> <li>Presses the button indicated by button title in the dialog box indicated by dialog title.</li> </ul> <p><b>Example:</b></p> <pre>dlgCmd=Button "Client Authentication" "OK"</pre> <ul style="list-style-type: none"> <li>Executes a sequence of TAB, UP/DOWN arrow keys, and mouse-click for the dialog box specified by dialog title. Format:</li> </ul> <pre>dlgCmd=Sequence "&lt;Dialog Title&gt;" &lt;seq1&gt; &lt;seq2&gt; ...</pre> <p>where seq1, seq2, ... can be one of the following:</p> <pre>tab=&lt;number of times to simulate TAB (positive number) or Shift-TAB (negative number&gt;</pre> <pre>up=&lt;number of times to simulate UP key&gt;</pre> <pre>down=&lt;number of times to simulate DOWN key&gt;</pre> <pre>clicktext="Text for text field that is currently in focus"</pre> <pre>close</pre> <p><b>Example: The following sequence simulates pressing the DOWN key twice, focusing on the next dialog element and clicking it:</b></p> <pre>dlgCmd=Sequence "Client Authentication" down=2 tab=1 click</pre>

**Table 3 IE Mode Script Statements (cont'd)**

Statement	Arguments	Comment
dynCmdPattern<idx> dynCmdDelim<idx> dynCmdSubst<idx>	Substitution Rule (Pattern, Delimiter, Substitution)	<p>Specifies substitution rule for dynamic URLs. Statements are grouped by &lt;idx&gt;. For example, dynCmdPattern3, dynCmdDelim3, dynCmdSubst3 statements belong to the same rule.</p> <p>Pattern can be specified as a regular expression. Delimiter is a string that delimits the text that follows right after the pattern and before the delimiter. This text is used to replace the variable or path component as specified in the Substitution. Arguments for the substitution:</p> <p>QUERY &lt;variable in query string&gt;            PATH &lt;location index&gt;            POST &lt;variable in POST string&gt;            HOST &lt;host portion of the URL&gt;</p> <p>Location index can be negative to reference component from the end; for example, -1 is the last path component.</p> <p><b>Example:</b></p> <pre> dynCmdPattern0=pageGenTime' value=' dynCmdDelim0='&gt; dynCmdSubst0=QUERY pageGenTime dynCmdPattern1=AUTHMETHOD' value=' dynCmdDelim1='&gt;dynCmdSubst1=POST AUTHMETHOD dynCmdPattern2=pageGenTime' value=' dynCmdDelim2='&gt;dynCmdSubst2=PATH 1           </pre> <p>If a standard dynamic substitution rule is not specific enough to work, another format option is available.</p>



**Table 3 IE Mode Script Statements (cont'd)**

Statement	Arguments	Comment
		<p>Example:</p> <pre>DynCmdPattern0=test&amp;arg=login DynCmdDelim0=" DynCmdSubst0=PATH -1=test&amp;arg=login%s</pre> <p>This says to search for the pattern "test&amp;arg=login" in the last position and substitute the string that directly follows, in this case, sessionid. The DynCmdSubst0 parameter substitutes the last path component of the path specified in the IESStep and replaces it with the string that follows the = sign (test&amp;arg=login%s). Before the string is actually replaced, the %s is substituted with the content of the string that was specified by the pattern and delimiter.</p> <p>Similarly, the search string can be used with POST. Example:</p> <pre>DynCmdPattern0&gt;Password=test DynCmdDelim0=; DynCmdSubst0=POST Password=test%s</pre>

**Table 3 IE Mode Script Statements (cont'd)**

Statement	Arguments	Comment
dynCmdScript<idx> dynCmdSubst<idx>	Substitution Rule (Script, Substitution)	<p>Specifies substitution rule for dynamic URLs. Statements are grouped by &lt;idx&gt;. For example, dynCmdScript1, dynCmdSubst1 statements belong to the same rule.</p> <p>Script specifies the script that returns the value for the substitution part. Arguments for the substitution:</p> <p>QUERY &lt;variable in query string&gt;</p> <p>PATH &lt;location index&gt;</p> <p>POST &lt;variable in POST string&gt;</p> <p>Location index can be negative to reference a component from the end; for example, -1 is the last path component.</p> <p><b>Example:</b></p> <p>DynCmdScript1=getid.shDynCmdSubst1=POST            sessionId</p>
waitTime	Milliseconds	<p>Wait time between steps in milliseconds. Default is 1500 milliseconds. Increase this value if the playback terminates prematurely during redirections.</p> <p><b>Example:</b></p> <p>waitTime=10000</p>

**Table 3 IE Mode Script Statements (cont'd)**

Statement	Arguments	Comment
waitTimeReset	Number	<p>Resets <code>waitTime</code> to the global value after the specified number of redirections that occurred within the transaction step.</p> <p><b>Example:</b></p> <pre>waitTime=5000 waitTimeReset=2</pre> <p>In the example, the <code>waitTime</code> will be 5000 milliseconds after the first redirection and 1500 milliseconds after the second one.</p> <p>To see the number of redirections, set the trace level to 1 during recording.</p>
continueClick	Number	<p>Instructs the playback to not wait for any events from the browser. For example, a JavaScript program might update a page without loading a URL from the server. In such a case, the <code>continueClick=1</code> statement tells the playback to continue with the next transaction step.</p> <p><b>Example:</b></p> <pre>continueClick=1</pre>
noWindowClose	Number	<p>Instructs the recorder not to close any windows. Some web sites that use multiple windows might close one of the windows when the user logs out. In case the transaction step does not stop on such an event and the progress indicator continues spinning, adding the <code>noWindowClose</code> statement might solve this.</p> <p><b>Example:</b></p> <pre>noWindowClose=1</pre>

**Table 3 IE Mode Script Statements (cont'd)**

<b>Statement</b>	<b>Arguments</b>	<b>Comment</b>
forceWindowClose	Number	<p>Bypasses the closing of a browser window through IE. The Web Transaction Recorder and the probe handle the closing of the window. Setting this flag to 1 may help for sites that open and close multiple windows, such as temporary progress indication windows.</p> <p><b>Example:</b></p> <pre>forceWindowClose=1</pre>
completeCheck	Number	<p>Instructs the playback to change the algorithm for detecting the end of a transaction step. Add this flag to a transaction step if the playback does not detect the end of the transaction step.</p> <p><b>Example:</b></p> <pre>completeCheck=1</pre> <p>If the last event sequence in the trace is BeforeNavigate2, DownloadComplete, use completeCheck=2.</p>
noAbout	Number	<p>Instructs the playback to not load a blank page (about:blank) when a new window is opened. Loading a blank page is required if the playback uses authentication.</p> <p><b>Example:</b></p> <pre>noAbout=1</pre>
checkAvailability	Script	<p>Name of a script that provides the availability check on the HTML text of the web page.</p> <p><b>Example:</b></p> <pre>checkAvailability=check.bat</pre>

**Table 3 IE Mode Script Statements (cont'd)**

Statement	Arguments	Comment
docCompletes	Number	<p>Instructs the playback to wait for the specified number of documents before determining the end of a transaction step. The exact number can be found in the trace (DocCompletes: &lt;num&gt;).</p> <p><b>Example:</b></p> <pre>docCompletes=31</pre> <p>The above example waits for 31 DocCompletes before determining the end of a transaction step.</p>
window	Window Number	<p>Specify the window number to which the step should be applied. Use this when the web application opens multiple windows so you can guide the probe to the correct window during playback.</p> <p>The number of the first window opened starts with 1, the second with 2, and so on. Use "window=0" to always specify the top most window.</p> <p><b>Example:</b></p> <p>After the first page is loaded, the web application opens another window without closing the first window. (Note: You can see the number of windows by clicking on the maximize button of the window).</p> <p>If the next step should do an action in the second window, specify "window=2" in this next step.</p>
enableIEDialog	Number	<p>Instructs the playback to not automatically suppress pop-up dialog boxes generated by a JavaScript <code>alert()</code> function. For example, a JavaScript function might generate a warning message in a pop-up dialog boxes, the <code>enableIEDialog=1</code> statement tells the playback to display the dialog box during that step.</p> <p><b>Example:</b></p> <pre>enableIEDialog=1</pre>

**Table 3 IE Mode Script Statements (cont'd)**

Statement	Arguments	Comment
WaitForDlgCaption	Caption for dialog to wait for before continuing on to the next step	<p>With some complex web pages, there are times when the Web Transaction Recorder cannot determine when to move on to the next step. In these cases, it is usually necessary to include long wait times for a step to be sure that everything has loaded properly before moving on. Although this method works and does not affect availability, the response time for that step will not be accurate. If a dialog appears at the time when the step should end, that can be used to signal Web Recorder to move on. To determine if dialogs are appearing (many dialogs are invisible), turn on tracing and watch the trace as you are recording. If the step that needs a long wait shows a dialog appearing, such as:</p> <p>Handle Dialogs class=ThunderRT6FormDC and wintext=Filters</p> <p>Then you can add this command to your step:</p> <p>WaitForDlgCaption="Filters"</p> <p>When Web Recorder detects that this dialog has appeared, it will stop its internal timer and move on to the next step which will give a more accurate measure of the true response time for the step. It is best to use the last unique dialog that appears as the caption to avoid moving to the next step too soon. If you find that you are still moving too soon, then you can put a 'delay' command in the NEXT step to give the page time to finish loading before executing the step.</p>

**Table 3 IE Mode Script Statements (cont'd)**

<b>Statement</b>	<b>Arguments</b>	<b>Comment</b>
Delay	number of milliseconds to delay execution of the step	Sometimes it is necessary to delay execution of a page for a brief amount of time. For example, if the previous step contained a WaitForDlgCaption command that triggered a new step when a dialog appeared, but the page still wasn't quite ready for the next step (but very close to ready) you can use this command to delay step execution for the specified number of milliseconds. This time is not counted for the response time of the step or the transaction.
maximize	Number	Specifies whether the browser window should be maximized during playback. The default is set to maximize each new browser window. If a web application is sensitive to page resizes, use this flag to instruct the playback to not maximize the browser window.  Example: maximize=0  The above example does not maximize the browser window for the current step.
formInputMode	Number	If this parameter is set to 2 (formInputMode=2), then the Web Recorder will use window events to fill out form input fields. Keyboard related events associated with form input elements are then fired.  The default is not to use window events.
clickMode	Number	If this parameter is set to 2 (clickMode=2), then the Web Recorder will use window events to simulate a click on an HTML element such as a button, image, etc.  The default is not to use window events.  Note: Only elements that are visible within the window can receive window events. If the element is in a section of the page that is not visible, clickMode=2 will have no effect.

**Table 3 IE Mode Script Statements (cont'd)**

Statement	Arguments	Comment
noContinue	number	<p>If an MCLICK or KEY statement generates interaction with the web server, it may move too quickly to the next step, before the web server has updated the page. To keep the step from ending too soon, you can add the following statement to the MCLICK step:</p> <pre>noContinue=1 script</pre> <p>The noContinue statement takes precedence over the continueClick statement.</p>
savesession loadsession	Cookie Filename	<p>The savesession statement will cause ALL the cookies in the IE cookie cache to be written to the specified file. If the file exists, it will be overwritten. This action happens at the end of the step.</p> <pre>savesession=cookie C:\COOKIES.TXT</pre> <p>The loadsession statement will load all the cookies specified in the named file into the IE cookie cache before the step is executed. If the IE cookie cache already contains a cookie with the same name and URL, it will be overwritten.</p> <pre>loadsession=cookie C:\COOKIES.TXT</pre> <p>The cookie file format is interchangeable so for example, you can save cookies with the HTTP probe and load them using the HTTP_TRANS probe.</p>



## URL/Navigation Point Mode

**Table 4 URL/Navigation Point Mode Script Statements**

Statement	Arguments	Comment
step	URL	Navigates to the specified URL. Example: step=http://www.hp.com/go/openview
dstep	Navigation Point	Locates the navigation point in the current document loaded by the probe, resolves it to an URL, and loads this URL. The identifier between = and : specifies the type of the navigation point: A: Anchor FORM: Form (HTTP operation POST) FORM_GET: Form (HTTP operation GET) Examples: dstep=A:Desktops and Monitors dstep= FORM:BV_SessionID=550&BV_EngineID=fmg.0 &product_oid=427
pos	Index	Specifies the index of a navigation point. If multiple identical navigation points are found, the index specifies the one that should be resolved.
post	HTTP Post string	Specifies the data used in a HTTP POST operation. Example: post=product_oid=427&arg=search

**Table 4 URL/Navigation Point Mode Script Statements (cont'd)**

Statement	Arguments	Comment
pattern	Pattern string	<p>Specifies the pattern. If the pattern is not found, the step is considered to be unavailable.</p> <p>Certain HTML characters are encoded. For example, &amp; is &amp;amp; and " is &amp;quot;. When setting up a pattern, look at the HTML source to see whether the string that is displayed in the browser contains any encoded characters. If encoded characters are part of the string, adjust the pattern accordingly.</p> <p><b>Example</b></p> <pre>pattern="+HP OpenView"</pre>
patternconfig	Pattern options	<p>The default operator for word concatenation is AND. This can be changed to OR. A word compare is case sensitive by default. This can be changed to case insensitive by entering ICASE.</p>
username	Username	<p>Specifies the Username to be used in HTTP authentication.</p> <p><b>Example:</b></p> <pre>username=ovros</pre>
password	Password	<p>Specifies the Password to be used in HTTP authentication.</p> <p><b>Example:</b></p> <pre>password=artros</pre>
proxyUsername	Username	<p>Specifies the Username to be used in HTTP proxy authentication.</p> <p><b>Example:</b></p> <pre>proxyUsername=secureros</pre>

**Table 4 URL/Navigation Point Mode Script Statements (cont'd)**

<b>Statement</b>	<b>Arguments</b>	<b>Comment</b>
proxyPassword	Password	Specifies the Password to be used in HTTP proxy authentication. Example: proxyPassword=artros
label	Label	Symbolic name of the transaction step. Example: label=Check-out

**Table 4 URL/Navigation Point Mode Script Statements (cont'd)**

Statement	Arguments	Comment
dynCmdPattern<idx> dynCmdDelim<idx> dynCmdSubst<idx>	Substitution Rule (Pattern, Delimiter, Substitution)	<p>Specifies substitution rule for dynamic URLs. Statements are grouped by &lt;idx&gt;, e.g. dynCmdPattern3, dynCmdDelim3, dynCmdSubst3 statements belong to the same rule. Pattern can be specified as a regular expression. Delimiter is a string that delimits the text that follows right after the pattern and before the delimiter. This text is used to replace the variable or path component as specified in the Substitution. Arguments for the substitution:</p> <p>QUERY &lt;variable in query string&gt;            PATH &lt;location index&gt;            POST &lt;variable in POST string&gt;            HOST &lt;host portion of the URL&gt;</p> <p>Location index can be negative to reference component from the end (e.g. -1 is last path component).</p> <p><b>Example:</b></p> <pre> dynCmdPattern0=pageGenTime' value=' dynCmdDelim0='&gt; dynCmdSubst0=QUERY pageGenTime dynCmdPattern1=AUTHMETHOD' value=' dynCmdDelim1='&gt;dynCmdSubst1=POST AUTHMETHOD dynCmdPattern2=pageGenTime' value=' dynCmdDelim2='&gt;dynCmdSubst2=PATH 1           </pre> <p>If a standard dynamic substitution rule is not specific enough to work, another format option is available.</p>

**Table 4 URL/Navigation Point Mode Script Statements (cont'd)**

Statement	Arguments	Comment
		<p>Example:</p> <pre>DynCmdPattern0=test&amp;arg=login DynCmdDelim0=" DynCmdSubst0=PATH -1=test&amp;arg=login%s</pre> <p>This says to search for the pattern "test&amp;arg=login" in the last position and substitute the string that directly follows, in this case, sessionid. The DynCmdSubst0 parameter substitutes the last path component of the path specified in the IESStep and replaces it with the string that follows the = sign (test&amp;arg=login%s). Before the string is actually replaced, the %s is substituted with the content of the string that was specified by the pattern and delimiter.</p> <p>Similarly, the search string can be used with POST. Example:</p> <pre>DynCmdPattern0&gt;Password=test DynCmdDelim0=; DynCmdSubst0=POST Password=test%s</pre>

**Table 4 URL/Navigation Point Mode Script Statements (cont'd)**

Statement	Arguments	Comment
dynCmdScript<idx> dynCmdSubst<idx>	Substitution Rule (Script, Substitution)	<p>Specifies substitution rule for dynamic URLs. Statements are grouped by &lt;idx&gt;, for example, dynCmdScript1, dynCmdSubst1 statements belong to the same rule.</p> <p>Script specifies the script that returns the value for the substitution part. Arguments for the substitution:</p> <p>QUERY &lt;variable in query string&gt;</p> <p>PATH &lt;location index&gt;</p> <p>POST &lt;variable in POST string&gt;</p> <p>Location index can be negative to reference a component from the end (for example -1 is the last path component).</p> <p><b>Example:</b></p> <p>DynCmdScript1=getid.shDynCmdSubst1=POST            sessionid</p>

# Web Transaction Recorder Tips

This section contains tips that can help you if you encounter problems in recording your transaction.

## Picking Pattern Matching Values

Turn tracing to 1 in the Web Transaction Recorder (this tracing level only lasts while you are in the Recorder) and move to the step that you need pattern matching for. At the bottom of the window you will see what it picked up for the title. Put this in double quotes with a plus sign in front of it in the Pattern field in the Step Properties dialog box.

## Only Alpha-numeric Characters in Pattern Matching

Certain HTML characters are encoded. For example, & is &amp; and " is &quot;. When setting up a pattern, look at the HTML source to see whether the string displayed in the browser contains any encoded characters. If encoded characters are part of the string, adjust the pattern accordingly.

## Scrub Cookies

Existing cookies on a system can impact recording, playback, and running a probe.

To remove cookies each time prior to the HTTP\_TRANS probe running, go to **File** → **Configure** in the Web Transaction Recorder main window. Use the Transaction Properties dialog box to set Scrub Cookies at the global transaction level. This flag is only for the OVIS scheduler running the probe. It has no effect when you play back the recording in the Web Transaction Recorder.

To remove cookies before recording your transaction, go to **Tools** → **Cache Viewer** in the Web Transaction Recorder main window. Click the **Delete All** button and **OK**. This is an optional setting because you sometimes might want to record without deleting cookies.

To remove cookies before doing a playback in the Web Transaction Recorder, go to **Tools** → **Cache Viewer** and delete cookies before playing back a recording.

## Button or Drop Down Does Not Work in IE Mode

When you find a button or drop-down that does not work properly in playback, you can go into the Step Properties dialog box for that step and set the type to IE URL. Then go into **View** → **View Source** for the page and find out what the URL is for that button or drop-down and put that in the Target field in the Step Properties dialog box. If there is a Post part to the URL, cut that part out of the Target field and put it in the Post Data field. Fill in the pattern matching.

## Wait Time

You can define a `waitTime` value on individual steps in the transaction. If you have a step with lots of redirection, you can put in a `waitTime` script statement so that there is enough time for the page to come up before the next step tries to use pattern matching to check for the correct page. Add the `waitTime` statement in the Advanced Step Properties dialog box (see `waitTime` in the [Table on page 82](#)).

## ForceWindowClose

If a site uses a pop-up status window that displays on top of the web page, you need to use the `ForceWindowClose` script statement. Without this, the playback will not remove the blank white top window, and the match will fail since it only looks at the top window. See `ForceWindowClose` in [Table on page 84](#)).

## Dynamic Host Name

If you have a host name (in the URL portion of a navigation step) that changes because of load balancing every time the transaction is played back, the playback will fail with an `Unable to find closest match and Unable to resolve step error`.

```
A "_top" "35" "" "https://server1/servlet/a-b"
```

To remedy this, edit the step to specify a `*` for the host name part of the URL. This instructs the probe to ignore the host name in the check.

```
A "_top" "35" "" https://*/servlet/a-b
```



# Troubleshooting

Troubleshooting the HTTP\_TRANS probe may involve problems that occur while you are recording and problems that occur during playback of the recording.

This section covers recording and playback issues, as well as providing an example trace file for HTTP\_TRANS that is annotated with explanations of the trace output.

# Recording and Playback Issues

## Recording Issues

Recording Issues may include the following:

- [Transaction Step Not Recorded \(Menus, Plug-Ins\) in IE Mode](#)
- [Pop-Up Windows in IE Mode](#)
- [Form Values are Not Recorded](#)



Note that if you are using the Web Transaction Recorder and Microsoft Script Debugger is installed, it is recommended that you turn off script debugging in Internet Explorer. For example, in IE select **Tools > Internet Options > Advanced:** and check **Disable Script Debugging**. Having script debugging enabled interferes with Web Transaction Recorder playback and recording in cases where the page contains a script with an error.

### Transaction Step Not Recorded (Menus, Plug-Ins) in IE Mode

Some elements within a web application, such as complex JavaScript or plug-in (for example, Macromedia FLASH) based menus, internally handle mouse events that are not forwarded to the recorder. As a result, a transaction step is not recorded in the left pane, even though a new page is loaded in the browser window.

To remedy this, select **Allow URL recording** in the **Configure** dialog box (see [Transaction Step Properties Basic on page 44](#)). This instructs the recorder to capture and add URLs to the transaction step list.

You will be prompted by a dialog box for each URL that is loaded by the browser. Usually, only the first URL needs to be recorded. The rest can be ignored until the page has been completely loaded.

If the URL (or POST data) contains dynamic components, apply a substitution rule (see [Dynamic URL Substitution on page 50](#)).

If you still are not able to record the step, you can manually add a URL. To do this, enable tracing to look for messages that start with `Loading:.` Paste the URL and POST information you find in the trace list into the Web Transaction Recorder as a step.

It may be necessary to specify the frame in which the URL should be loaded. Use the `frame` statement with the `iestep` command. (See `iestep` and `frame` statements in the [Table on page 73](#) for more information.) The name of the frame is shown at the end of the `BeforeNavigate2` message.

Example:

```
BeforeNavigate2: 'http://www.hp.com/' ('mainFrame')
```

## Pop-Up Windows in IE Mode

If the main window of a web application opens multiple windows, the main window might be hidden behind the pop-up window. To continue recording in the main window, select the **Cascade** button in the browser view window and select the main window.

Alternatively, you can configure the Web Transaction Recorder to ignore pop-up windows containing advertisements by setting the Ignore pop-up windows or Ignore Error dialogs check boxes in the Set Global Properties dialog box (see [Specifying Global Transaction Properties on page 37](#)).

Also make sure to check the pattern that was automatically generated for the transaction step. It may be the one for the pop-up and not the one for the main web page.

## Form Values are Not Recorded

Under certain circumstances, data that has been entered during the recording is not recorded. This can happen when the form sets up an `OnClick` handler that empties these form elements. As a result, the recorder is not able to collect the entered information. The only workaround possible is to manually construct the `formdata` statements.

**Syntax:** `formdata=<frame name>.<form>.<variable>=value`

**Example:** `formdata=_top.form1.index=blended`

## Playback Issues:

- Transaction Times Out
- What the HTTP\_TRANS Probe Means to Your Resource Usage
- Dynamic Frame Names (ERROR: Unable to find frame x)
- Initialization Error Dialog Boxes
- Response Time and/or System Load Increases
- Java Applets Causing Abort or Incorrect Behavior
- Playback Does Not Detect End of a Transaction Step
- Empty Message Box during Playback/Stepping
- New Window is Not Loaded Correctly
- Content of a Selection Box Changes with Every Playback
- IE Mode Registry Settings
- Dynamic ID Fields
- Scripts Against Web Sites with Active-X Controls/Java Applets
- Limitations on Windows 2000 Playback

### Transaction Times Out

Possible reasons for the probe timing out (IE mode):

- Proxy not set for the Probe Location under which the transaction runs.  
Add the proxy setting in the Probe Location dialog box of the Configuration Manager for the transaction and press **Save**.
- The default SYSTEM account under which the probe runs might not have all the same IE settings as the user account that recorded the transaction.  
Use the Control Panel to change the **HP Internet Services** service to run under the account that recorded the transaction (see [Changing the Log On Account for the Probe on page 58](#)). Or manually set the IE settings for the SYSTEM user in the registry (see [Registry Settings that Influence Probe Results \(IE mode\) on page 62](#)). Or enter a user in the Run As User field on the initial dialog when entering the Web Transaction Recorder.
- Some IE dialog boxes are not handled and sit waiting for user interaction.

Modify the script to handle IE dialog boxes (see [Internet Explorer Security Dialog Boxes](#) on page 48).

- A transaction step in IE mode does not load a new URL and only modifies the page itself.

Add `continueClick=1` command to transaction step causing the time-out (see `continueClick` in the [Table](#) on page 83).

- Probe timing out can also be impacted by the Request Timeout value in the Probe Location dialog box and the setting for Probe Retries.

## What the HTTP\_TRANS Probe Means to Your Resource Usage

The HTTP\_TRANS IE mode probe (recorded in IE mode and run as `httptrans2.exe`) runs the IE engine, so that if you are running several HTTP\_TRANS probes the resource usage can be high. For example, running five HTTP\_TRANS IE mode probes concurrently means running five IE engines. This is like opening five IE browser windows at the same time, with all of them opening and traversing web pages at the same time. Because of this overhead, it is recommended that you set concurrency to a smaller number (for example, between 1-10) for the HTTP\_TRANS probe. Also, consider running the HTTP\_TRANS probes on a dedicated remote probe server.

To determine what concurrency setting will best reflect response time of the real user, you can run a series of tests. Configure your HTTP\_TRANS probes and assign them to a network connection with `concurrency=1` (use the Probe Location dialog in the Configuration Manager). Allow the probe to run on the probe system and collect Response Time measurements. Run the probes again and set concurrency to a higher number and see if the Response Time values are similar. If the values are close, you can run the probes with the higher concurrency and still have response times reflect the real user experience. If the Response Time values are higher than expected, the overhead of multiple probes running is too high. This impacts the response times and, thus, is not truly representative of the user experience. In the latter case, you need to set concurrency to a lower number. However if you are only interested in availability, the higher response time might be acceptable.

Other options you can look at to reduce the resource usage include the following:

- You can run the probe with concurrency of 1. If you have lots of these probes you can spread them over multiple remote probe systems.

- You might be able to use a different probe type (other than HTTP\_TRANS in IE mode) to monitor the transaction that would be lower overhead. If the transaction you are measuring is a simple one with only a single step, you might be able to use the HTTP probe. If the transaction did not use Java or plug-ins, you might be able to use the URL mode probe. This probe would run without incurring the overhead of bringing up the IE browser.
- If you are only measuring availability, you might be able to have a higher concurrency. You could also use the HTTP\_TRANS URL mode probe to measure only availability because you are not as interested in closely simulating the user experience. If you need to measure response times, set concurrency for the network connection lower. You may need to set it to 1; otherwise, the Response Time will not reflect what a typical user experiences, since they would typically only have one IE browser window open.

## Dynamic Frame Names (ERROR: Unable to find frame x)

Some frame windows are created by JavaScript, which may lead to a randomly generated frame name. This may cause problems during playback if form data was recorded in this frame. An error like the following is displayed:

```
ERROR: Unable to find frame "1234567890".
```

Example transaction step with dynamic frame names:

```
IENAVPNT=A "1234567890" "252" "button"  
"javascript:addToCartUsingToggle();"   
  
FORMDATA=1234567890.items.item=Submit=True
```

In the IENAVPNT and FORMDATA statement, the items form is located on the 1234567890 frame. Because the form name may change the next time the transaction is played back, the probe will no longer find the form data location.

To remedy this, either remove the frame name (leave it unspecified), which uses the top-most frame, or specify the index of the frame.

Example A:

```
IENAVPNT=A "" "252" "button"  
"javascript:addToCartUsingToggle();"   
  
FORMDATA=.items.item=Submit=True
```

**Example B:**

```
IENAVPNT=A "[2]" "252" "button"
"javascript:addToCartUsingToggle();"
FORMDATA=[2].items.item=Submit=True
```

In Example A, the frame name is empty in IENAVPNT and FORMDATA. This instructs the probe to use the top-most frame. In Example B, the correct frame is frame 2 in the frame collections object. To determine the frame number, select the **View > Source (DOM)** menu item in the Web Transaction Recorder Main window, which lists all the HTML element indexes and frame names.

## Initialization Error Dialog Boxes

If you encounter pop-up boxes indicating that processes (particularly the probe processes or scheduler) are terminating abnormally due to initialization errors, reconfigure the HP Internet Services System account setting by selecting **This account:** and entering the account (user name) and password that recorded the transaction. This applies to both the management server and remote probe systems (every probe location where the transaction is executed). This configuration information is accessed on Windows 2000 via the **Computer Management → Services and Applications → Services → Properties → Log On** selection.

Note that you can also enter a user in the Run As User field on the initial dialog when entering the Web Transaction Recorder.

## Response Time and/or System Load Increases

The IE Mode in the Web Transaction Recorder programmatically runs the IE engine to do its probing.

By default, probes use the DEFAULT network connection with concurrency set to 32. When using IE mode probing, you should configure a network connection with reduced concurrency between one and 10. Because this is a global setting for the network connection, you must choose a concurrency that will not overload the system but will run probes with a high enough concurrency to complete in the probe interval you specified. You must also allow some slack time for exceptional situations where probes are running longer than usual and take their full timeout allocation. This concurrency can be configured via the Configuration Manager in the Edit Probe Location/Edit Connection dialog box in the **Number of concurrent requests** field.

## Java Applets Causing Abort or Incorrect Behavior

In case the playback of a page containing a Java applet causes problems, disable the downloading of Java applets. In the Web Transaction Recorder, uncheck the **Enable Java (IE)** check box in the **File** → **Configure** → **Properties** dialog box.

## Playback Does Not Detect End of a Transaction Step

In very rare cases, the playback does not detect the end of a transaction step and the transaction times out. In such a case, add the `completeCheck=1` flag to the transaction step that is causing the problem.

This flag is necessary when the browser does not retrieve any document from the server. For example, the web application opens a menu or similar action implemented in JavaScript. Since the trace only shows `DownloadBegin/DownloadComplete` sequences, it would not detect the end of the transaction step.

If the last three trace messages show a `BeforeNavigate2, DownloadBegin, DownloadComplete` sequence without a following `DocumentComplete` message, set the `completeCheck` flag to 2 (`completeCheck=2`). This changes how the Web Transaction Recorder detects the end of a transaction step. When the flag is set to 2, the Web Transaction Recorder uses the above event sequence for the end of transaction detection.

If a web application performs a lot of redirects and additional page loads in the background (for example, through a JavaScript), it is possible to instruct the probe to determine the end of a transaction step by specifying the exact number of documents that a step downloads from the server. For each document that is downloaded the trace shows the `docCompletes` counter (for example, `docCompletes=31`). The probe can be instructed to wait for an exact number of these `docCompletes` to determine that a transaction step is completed. Simply enter `docCompletes=<number of document completes from trace>` in the **Advanced Properties** dialog box of a step. See `docCompletes` in the [Table on page 85](#).



## Empty Message Box during Playback/Stepping

If an empty message box is shown during playback/stepping and the transaction had more than one window open, add the `forceWindowClose=1` flag to the step (see `forceWindowClose` in the [Table on page 84](#)) that is closing the window.

## New Window is Not Loaded Correctly

If a transaction step loads a new window and the content is not displayed correctly (for example, blank page), try setting the `noAbout=1` flag for this transaction step (see `noAbout` in the [Table on page 84](#)). This instructs the recorder to not initialize the new window with a blank page (`about:blank`). The initialization is required when the transaction uses authentication.

Alternatively, you can disable this initialization for the whole transaction by unchecking the **Initialize Browser** flag in the Properties dialog box (see [Specifying Global Transaction Properties on page 37](#)).

## Content of a Selection Box Changes with Every Playback

If a transaction step references an element in a selection box that changes every time the transaction is played back, the recorder will not find the original element and will abort. For example, in a bank transaction, an account pull-down menu may contain the account number and the current balance. Since the balance constantly changes, the recorder will not find the correct entry. To remedy this, locate the corresponding `formData` statement and specify the absolute position of the element in the selection box. For example, `formdata=form.sel=@1` selects the element at position 1 (positions usually start at 0).

## IE Mode Registry Settings

You can set some registry keys that can be useful in troubleshooting the IE mode probe. To do this, add DWORD keys to the following registry key:

```
HKEY_LOCAL_MACHINE\SOFTWARE\Hewlett-Packard\Internet
Services\CurrentVersion
```



On remote probe systems, this key does not exist. Instead, you will see the key `HKEY_LOCAL_MACHINE\SOFTWARE\Hewlett-Packard\Internet Services Remote Probes\CurrentVersion`.

Do not add the DWORD keys to this key.

On remote probe systems, create an `Internet Services\CurrentVersion` key underneath `HKEY_LOCAL_MACHINE\SOFTWARE\Hewlett-Packard\`.

(First create the `Internet Services` key and then the `CurrentVersion` key under `Internet Services`.)

Add any of the following DWORD keys and set their value to 1 to enable. Rebooting is not necessary after adding these keys.

- `ContinuousCapture`

If `Capture Window` is enabled, the probe will do continuous capture. Each file name contains the time and date when the image or trace was created. This option is useful if the behavior of a problem is to be observed over a longer period of time.



Make sure that there is plenty of disk space.

- `CaptureTraceOnAvail`

Write the current trace information that is appended to `trace.log` to a separate file (saved in the following location `<data dir>\data\tmp\probe\capture\`) even if availability is 1. This is helpful to troubleshoot response time-related issues.

- `ShowProgress`

Instructs the probe and recorder to add the information that is usually displayed in the lower left status bar of IE. This information may show which file(s) are currently downloaded.

## Dynamic ID Fields

If the ID fields for HTML elements like table data (TD) or division (DIV) are dynamically generated by the web server, the Web Transaction Recorder will not be able to resolve the step during playback. To remedy this, locate any inner strings associated with the element and add them to the step target.

This problem may be seen on Ariba servers. When an Ariba server is rebooted, field IDs are changed, and the Web Transaction Recorder may no longer be able to play back a previously recorded transaction. One workaround is to try setting the Ariba parameter NamePrefix=True. On some versions of Ariba, this may stop IDs from being dynamically created on reboot. If this does not work, using the steps in the example below should solve the problem.

### Example

If the original HTML is as follows:

```
<td id-"47A">example</td>
```

The navigation step generated would be:

```
TD "_top" "25" "47A"
```

If the ID of "47A" is changed by the web server, this step will fail during playback. Fix the problem by manually adding the inner text "example" to the step. The step would be as follows:

```
TD "_top" "25" "47A" "example"
```

## Scripts Against Web Sites with Active-X Controls/Java Applets

If you record a script against an Active-X control, be aware that the playback may not work if the scheduler is running under the "LocalSystem" account. The LocalSystem account does not have permission to run Active-X controls. You must change the logon account of the scheduler to one that has appropriate permissions. Also, IE has different security settings that are based on the zone that contains the site. Therefore, you must make sure that you know which zone contains the site and whether the security settings are correct. Or enter a user in the Run As User field on the initial dialog when entering the Web Transaction Recorder.

## Limitations on Windows 2000 Playback

Running the HTTP\_TRANS IE mode probe (probehttptrans2) with the -print option will not produce results on Windows 2000 because that platform does not support the ability to write to a console from a Windows GUI application. This means that the **Test on Management Server** function in the Configuration Manager will not return results for the IE mode HTTP\_TRANS probe on Windows 2000 (it does work on XP or higher). It also means that the Probe Re-Execution TIP for this probe on Windows 2000 will give the message “No results returned for command”.

## HTTP\_TRANS Probe Configuration for TIPs

If you see the error “Directory/file cannot be found” in the Dashboard TIPs Viewer when running a TIP for the HTTP\_TRANS probe, you need to be sure the **Capture Window on Error** option is checked in the Web Transaction Recorder **File > Configure > Properties** dialog box. The probe does not produce an error log or error screen image that the TIP is trying to retrieve, until this option is checked.

## Annotated Trace File for HTTP\_TRANS

The following is an annotated trace file for the HTTP\_TRANS probe.

```
Setting proxies: `http=web-proxy.corp.hp.com:8088
https=web-proxy.corp.hp.com:8088'
Proxies Configured: `https=web-proxy.corp.hp.com:8088
http=web-proxy.corp.hp.com:8088'
Using proxies: `web-proxy.corp.hp.com:8088', `web-proxy.corp.hp.com:8088`
^^^ Proxies passed in from command line, configured in IE and actual proxies
```

```
Load `about:blank`. This is a workaround for an IE bug where NTLM
authentication will fail if this page is not loaded. This can be turned off
(Initialize Browser (IE) or noAbout=1).
```

```
BeforeNavigate2: `about:blank' (``)
^^^ BeforeNavigate2 event. First parameter is URL, second parameter is frame
name (``)
BeforeNavigate2: KillTimer
^^^ Watch dog is reset since a URL is about to be loaded
DownloadBegin
DownloadComplete
Clearing select cache
DownloadBegin
NavigateComplete2
DownloadComplete
IE Version: `4.0 (compatible; MSIE 6.0; Windows NT 5.0; H010818;
Hewlett-Packard IE5.5-SP2; .NET CLR 1.0.3705; .NET CLR 1.1.4322)`
^^^ Dump of the IE version: IE 6.0 on Win2K with security patches
DocumentComplete: b=0:rs=4 about:blank
^^^ Document "about:blank" has been loaded (DocumentComplete event)
DocumentComplete: Document is done downloading
^^^ This message indicates that all frames have been downloaded (about:blank
has only 1 frame)
DocCompletes: 1
^^^ Number of document completes for current step
Redirects: 1
^^^ Number of redirects for current step
ElapsedTimer Proc 180
```

Start of first step iestep=http://www.hp.com/

```
BeforeNavigate2: `http://www.hp.com/' (``)
BeforeNavigate2: KillTimer
DownloadBegin
DownloadComplete
```

## Troubleshooting

```
DownloadBegin
NavigateComplete2
DownloadComplete
DocumentComplete: b=0:rs=4 http://www.hp.com/
DocumentComplete: Document is done downloading
DocCompletes: 2
Redirects: 2
SelectFrame: ` _top `
^^^ Select the _top frame and move the MDI window of document to the front
FrameFinder: ` _top `
^^^ Trace from low-level function that locates the document for a given frame name
SelectFrame: found
Size:
_top=36843,s.gif=43,hpweb_1-2_arrw_sbmt.gif=79,b1_smb_text_oct_09.gif=2582,b1_smb_photo_oct_09.jpg=16772,b2_b3_smb_valuepromo_oct_16.gif=4371,&s=1280x960&c=32&j=1.3&v=Y&k=Y&bw=752&bh=362&ct=1an&hp=N&[AQE]=43,hpweb_utilities.js=7478,hpweb_country_pulldown.js=10992,setcookie.js=1082,metrics.js=16044
^^^ Show objects with their sizes that are used to determine the document size
INFO Size=96329 (http://www.hp.com/)
Set timer: 1500
^^^ Set the watch dog timer to 1500 ms
OnTimer
^^^ The watch dog timer expired (1500ms later), determine availability and take measurements
Kill timer
^^^ Low-level trace from function that disables the watch dog
```

Next section

```
Title: Welcome to HP
^^^ <TITLE> tag of the current document
Matched: +"Welcome to HP"
^^^ Shows the value of the "pattern" parameter and whether it matched
IPAUtil:: Cookies retrieved='cc=us; lang=en-us; s_sq; s_cc=true'
^^^ OVTA integration specific message. Shows current cookies
10/21 09:52:33 (1) host='www.hp.com', target='Step 00: http://www.hp.com',
avail=1, setup=0, sresp=10, trans=1414, totRespTime=1424, bytes=96329,
tput=66.06, reqs=1 , stepUrl=http://www.hp.com/
^^^ All measurements for step 0 that are written to the queue file
IPAUtil:: corr=, nodeId=97513558-e711-473e-841d-9fe4c9326903
^^^ OVTA integration specific message. Shows corr and nodeId cookies
```

Start of second step: 'INPUT:image "\_top" "475" "submit" "" "http://welcome.hp-ww.com/img/hpweb\_1-2\_arrw\_sbmt.gif"'

```
SelectFrame: ` _top `
FrameFinder: ` _top `
```

```

SelectFrame: found
^^^ Find the frame specified in the HTML element instruction
Clearing select cache
^^^ Internal message trace message
Find: element='_top.searchForm.qt', value='Internet Services'
^^^ The step has some form data. Find the INPUT element 'qt' in the form
"searchForm" in the "_top" frame
Set form data: form='_searchForm', element='qt', value='Internet Services'
^^^ Found, now set the data value ("Internet Services")
Playing back: `INPUT:image " top" "475" "submit" "" "http://
welcome.hp-ww.com/img/hpweb_1-2_arrw_sbmt.gif"`
^^^ Now prepare to click on the "submit" image
10/21 09:52:33 (9) FindFrame: find:'_top' current:'_top'
^^^ Trace message from frame finder
^^^ No error message here indicates that HTML element was found and clicked on
10/21 09:52:33 (8) BeforeNavigate2: `http://www.hp.com/search/
?cc=us&lang=en&qt=Internet+Services&submit.x=0&submit.y=0' (`)
^^^ BeforeNavigate2 event with the URL that is executed by the submit button
-> Click was successful

```

### Next Section

```

BeforeNavigate2: KillTimer
DownloadBegin
Security dialog (problem=12168)
^^^ Indicates a security "issue" (12168 ==
ERROR HTTP_REDIRECT_NEEDS_CONFIRMATION)
NavigateComplete2
DownloadComplete
DocumentComplete: b=0:rs=4 http://search.hp.com/gwuseng/
query.html?col=hpcom+ccen+ccenfor&la=en&qt=Internet+Services&.
DocumentComplete: Document is done downloading
DocCompletes: 1
Redirects: 1
Size:
_top=37569,s.gif=43,hpweb_1-2_topnav_home.gif=274,hpweb_1-2_topnav_prdsrv.gif
=483,.
INFO Size=75653 (http://search.hp.com/gwuseng/
query.html?col=hpcom+ccen+ccenfor&la=en&qt=Internet+Services.
Set timer: 1500
OnTimer
Kill timer
Title: Search HP US - Search results for 'Internet Services'
Matched: +"Search HP US - Search results for 'Internet Services'"
host='www.hp.com', target='Step 01: ', avail=1, setup=30, sresp=0,
trans=2256, totRespTime=2286, bytes=75653, tput=32.32, reqs=1 ,
stepUrl=http://www.hp.com/search/
?cc=us&lang=en&qt=Internet+Services&submit.x=0&submit.y=0
^^^ All measurements for step 1 that are written to the queue file

```

## Troubleshooting

Kill timer

INFO: Stepping/Playback for '/' done

**^^^ No more steps.**

host='Trans: www.hp.com', target='Trans: :', avail=1, setup=30, sresp=10,  
trans=3670, respTime=3710, bytes=171982, tput=45.27, reqs=2

**^^^ Transaction summary that are written to the queue file**

ElapsedTimer stopped



## A

authentication, **14**  
authentication dialogs, **59**  
availability check with external script, **56**

## B

browser plug-in navigation manually added,  
**59**  
button or drop down doesn't work, **96**

## C

cache, **15**  
cascading style sheet css, **11**  
certificate, **14**  
    root, **14**  
certification authority, **14**  
checkAvailability, **84**  
clickMode, **87**  
codebase parameter, **13**  
completeCheck, **84**  
concurrency settings, **20**  
concurrency set to 1 for IE mode, **20**  
continueClick, **83**

cookie  
    defined, **16**  
    persistent, **16**  
    session, **16**  
    settings, **59**  
cookies  
    save session, **88**

## D

delay, **87**  
dlgCmd, **79**  
docCompletes, **85**  
docCompletes example use in redirections,  
**60**  
DOM document object model, **25**  
dstep, **89**  
duplicate input variables, **61**  
dynamically loaded menus, **68**  
dynamic hostname  
    tips for dealing with, **96**  
dynamic IDs in HTML elements, **107**  
dynamic URLs, **50**  
dynCmdPattern, **80, 92**

dynCmdPattern, dynCmdDelim,  
    dynCmdSubst  
    examples, **51**  
dynCmdScript, **82, 94**

## E

enableIEDialog, **85**

## F

forceWindowClose, **84**  
    tips, **96**  
formdata, **76**  
formInputMode, **87**  
forms with no names, **61**  
frame, **13, 73**  
frame names  
    dynamic, **58**

## G

GUI  
    example, **34**

## H

handlers, **12**  
heavyweight recording mode, **19**  
how to record a transaction, **35**  
HTML  
    defined, **10**  
HTMLDIALOG example, **67**  
HTML dialogs, **63**  
HTTP, **13**  
HTTP\_TRANS probe, **32**  
HTTPS, **14**

## I

IE mode, **19**  
    playback, **27**  
    recording, **23**  
    tips for determining when to use, **21**  
ienavpnt, **74, 75**  
IE settings, **62**  
iestep, **73**  
input variables  
    set through an external script, **55**

## J

JavaScript, **11**  
    and plug-in examples, **12**

## L

label, **78, 91**  
launch Web Transaction Recorder, **32**  
load balancing, **96**  
loadsession, **88**  
log on account  
    changing to run under different, **58**

## M

maximize browser window, **87**  
MCLICK, **88**  
menus dynamically loaded, **68**  
modal dialogs, **63**  
mode  
    heavyweight recording, **19**  
    IE, **19**  
    navigation point, **19**  
    URL, **19**

**N**

navigation point mode, **19**  
noAbout, **84**  
noContinue, **88**  
noWindowClose, **83**

**P**

page redirections, **60**  
parameter  
    codebase, **13**  
    param, **13**  
param parameter, **13**  
password, **77, 90**  
pattern, **77, 90**  
patternconfig, **77, 90**  
pattern matching tips, **95**  
persistent cookie, **16**  
playback options  
    setting, **40**  
pop-up window  
    tips for dealing with, **96**  
pos, **89**  
post, **76, 89**  
probe  
    HTTP\_TRANS, **32**  
    location, **36**  
properties dialog, **37**  
proxyPassword, **78, 91**  
proxy server, **15**  
proxy settings, **43**  
proxyUsername, **77, 90**  
pull-down menus, **39**  
pull-down menus dynamically loaded, **68**

**R**

recording  
    example, **24**  
recording options  
    setting, **39**  
redirection, **15**  
    page, **60**  
    waitTime example use, **60**  
registry settings that influence probe  
    results, **62**  
root certificate, **14**  
run as another user, **32**

**S**

savesession, **88**  
scr attribute, **13**  
script options, **46**  
script statements, **73**  
security dialogs  
    tips for handling, **49**  
service level objectives, **36**  
session cookie, **16**  
settings  
    concurrency, **20**  
SSL, **14**  
step, **89**  
step alarming, **36**  
step properties  
    basic, **44**  
step properties dialog  
    advanced, **46**  
substitution rules for dynamic URLs, **50**

## T

- tag, **13**
- tag, **12**
- test on management server, **36, 108**
- TIPs error on HTTP\_TRANS probe, **108**
- trace file example, **109**
- tracing levels
  - setting, **42**
- transaction
  - how to record, **35**
- transaction properties, **37**
- transaction script reference, **73**

## U

- URL
  - defined, **15**
  - dynamic, **50**
  - steps to display, **16**
  - substitution rules for, **50**
- URL mode, **19**
  - playback, **27**
  - recording, **23**
- user actions executed with DlgCmd, **49**
- user impersonation, **32**
- username, **77, 90**
- username and password added for authentication, **59**

## W

- waitForDlgCaption, **86**
- waitTime, **82**
  - tips, **96**
- waitTime example use in redirections, **60**
- waitTimeReset, **83**

- web recorder
  - how it works, **23**
- Web Transaction Recorder
  - GUI example, **34**
  - launch, **32**
  - steps to using, **34**
- window, **85**
- windows events used to fill out forms, **87**
- windows events used to simulate clicks, **87**