Internet Explorer Application Compatibility

Microsoft Corporation

September 2008

Applies to:

   Microsoft Internet Explorer Compatibility Test Tool

   Microsoft Application Compatibility Toolkit

Summary: The Internet Explorer Application Compatibility article describes in detail the application compatibility events that are logged by Windows Internet Explorer 8 and Internet Explorer 7. These events are used by the Microsoft Application Compatibility Toolkit (ACT) to help customers understand their application compatibility situation by identifying which applications are compatible with Internet Explorer 8 and Internet Explorer 7 and which require further testing. This article describes the application compatibility events, when the events are triggered, and workarounds to fix the application compatibility issues.

#### Legal Notice

This is a preliminary document and may be changed substantially prior to final commercial release of the software described herein.

The information contained in this document represents the current view of Microsoft Corporation on the issues discussed as of the date of publication. Because Microsoft must respond to changing market conditions, it should not be interpreted to be a commitment on the part of Microsoft, and Microsoft cannot guarantee the accuracy of any information presented after the date of publication.

This White Paper is for informational purposes only. MICROSOFT MAKES NO WARRANTIES, EXPRESS, IMPLIED OR STATUTORY, AS TO THE INFORMATION IN THIS DOCUMENT.

Complying with all applicable copyright laws is the responsibility of the user. Without limiting the rights under copyright, no part of this document may be reproduced, stored in or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording, or otherwise), or for any purpose, without the express written permission of Microsoft Corporation.

Microsoft may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. Except as expressly provided in any written license agreement from Microsoft, the furnishing of this document does not give you any license to these patents, trademarks, copyrights, or other intellectual property.

Unless otherwise noted, the example companies, organizations, products, domain names, e-mail addresses, logos, people, places and events depicted herein are fictitious, and no association with any real company, organization, product, domain name, e-mail address, logo, person, place or event is intended or should be inferred.

© 2008 Microsoft Corporation. All rights reserved.

Microsoft, MS-DOS, Windows, Windows NT, Windows Server, Windows Vista, Active Directory, ActiveSync, ActiveX, Direct3D, DirectDraw, DirectInput, DirectMusic, DirectPlay, DirectShow, DirectSound, DirectX, Expression, FrontPage, HighMAT, Internet Explorer, JScript, Microsoft Press, MSN, Outlook, PowerPoint, SideShow, Silverlight, Visual Basic, Visual C++, Visual InterDev, Visual J++, Visual Studio, WebTV, Windows Media, Win32, Win32s, and Zune are either registered trademarks or trademarks of Microsoft Corporation in the U.S.A. and/or other countries.

The names of actual companies and products mentioned herein may be the trademarks of their respective owners.

#### Contents

Introduction 1

Events 1

1021 – MIME Handling Restrictions 1

Logged Message 1

What is it? 2

When is this event logged? 2

Example 2

Remediation 3

1022 – Windows Restrictions 4

Logged Message 4

What is it? 4

When is this event logged? 4

Example 4

Remediation 5

1023 – Zone Elevation Restrictions 7

Logged Message 7

What is it? 7

When is this event logged? 7

Example 7

Remediation 8

1024 – Binary Behaviors Restrictions 9

Logged Message 9

What is it? 10

When is this event logged? 10

Example 10

Remediation 11

1025 – Object Caching Protection 12

Logged Message 12

What is it? 12

When is this event logged? 12

Example 13

Remediation 15

1026 – ActiveX Blocking 15

Logged Message 15

What is it? 15

When is this event logged? 17

Example 17

Remediation 17

1027 – Pop-Up Blocking 18

Logged Message 18

What is it? 18

When is this event logged? 19

Example 19

Remediation 19

1028 – Automatic Download Blocking 20

Logged Message 20

What is it? 21

When is this event logged? 21

Example 21

Remediation 21

1030 – Local Machine Zone Lockdown (LMZL) 23

Logged Message 23

What is it? 23

When is this event logged? 23

Example 24

Remediation 25

1031 – Centralized URL Parsing 26

Logged Message 26

What is it? 26

When is this event logged? 26

Example 26

Remediation 27

1032 – Internationalized Domain Names (IDN) Support 27

Logged Message 27

What is it? 27

When is this event logged? 27

Example 27

Remediation 27

1033 – Secure Sockets Layer (SSL) 31

Logged Message 31

What is it? 31

When is this event logged? 32

Example 32

Remediation 32

1034 – Cross-Domain Barrier and Script URL Mitigation 36

Logged Message 36

What is it? 36

When is this event logged? 36

Example 37

Remediation 37

1035 – Anti-Phishing 38

Logged Message 38

What is it? 38

When is this event logged? 39

Example 39

Remediation 40

1036 – Manage Add-ons 40

Logged Message 40

What is it? 41

When is this event logged? 41

Example 41

Remediation 42

1037 – Protected Mode 42

Logged Message 42

What is it? 42

When is this event logged? 44

Remediation 44

1040 – Cascading Style Sheet (CSS Fixes) 45

Logged Message 45

What is it? 45

When is this event logged? 46

Example 46

Remediation 47

1041 – UIPI Extension Blocked 47

Logged Message 47

What is it? 47

When is this event logged? 47

Remediation 47

1042 – UIPI Cross Process Window Message 48

Logged Message 48

What is it? 48

When is this event logged? 48

Remediation 48

1046 – Cross-Site Scripting Filter 48

Logged Message 48

What is it? 49

When is this event logged? 50

Example 50

Remediation 51

1047 – Intranet at Medium Integrity Level 52

Logged Message 52

What is it? 52

When is this event logged? 54

Example 54

Remediation 54

1048 – DEP/NX Crash Recovery 55

Logged Message 55

What is it? 55

When is this event logged? 55

Example 55

Remediation 56

1049 – Standards Mode 56

Logged Message 56

What is it? 57

When is this event logged? 57

Example 58

Remediation 58

1056 – File Name Restriction 59

Logged Message 59

What is it? 59

When is this event logged? 59

Example 59

Remediation 61

1058 – Codepage Sniffing 61

Logged Message 61

What is it? 61

When is this event logged? 61

Example 61

Remediation 62

1059 – Ajax Navigation 62

Logged Message 62

What is it? 63

When is this event logged? 63

Example 63

Remediation 65

1061 – Application Protocol 65

Logged Message 65

What is it? 65

When is this event logged? 65

Example 66

Remediation 67

1062 – Windows Reuse Navigation Restriction 67

Logged Message 67

What is it? 68

When is this event logged? 68

Example 68

Remediation 70

1063 – MIME Restrictions: Authoritative Content Type Handling 70

Logged Message 70

What is it? 70

When is this event logged? 71

Remediation 71

1064 – MIME Sniffing Restrictions – No Image Elevation to HTML 72

Logged Message 72

What is it? 72

When is this event logged? 72

Example 72

Remediation 74

1065 – Web Proxy Error Handling Changes 74

Logged Message 74

What is it? 74

When is this event logged? 75

Example 75

Remediation 76

1073 – Certificate Filtering 76

Logged Message 76

What is it? 76

When is this event logged? 76

Example 76

Remediation 76

# Introduction

This document is designed to be a companion to the **Microsoft Internet Explorer Compatibility Test Tool** (which is part of the **Microsoft Application Compatibility Toolkit)**. The Internet Explorer Compatibility Test Tool logs information about your browsing session in Internet Explorer. As you browse Web pages, Internet Explorer logs events that indicate potential application compatibility issues for Internet Explorer 8. The tool logs the name of each event along with a short description of what each event is. Each description also contains a link to this document. The intention is that users of the Internet Explorer Compatibility Test Tool use this document to find out more about each event and what they can do to remediate the identified compatibility issue.

The remainder of this document contains a subsection for each of the events that can be logged by the Internet Explorer Compatibility Test Tool. After the user has linked to this document, the intention is that the table of contents will be used to link directly to the event that they’re interested in. The section for each event contains the following information:

* **Logged Message** – This is a copy of the event description that you’ll see in the Internet Explorer Compatibility Test Tool.
* **What is it?** – This is an elaboration of the logged message explaining what the event is. Additional references are provided when available.
* **When is this event logged?** – This is a short description of what has to happen in your Web page for this event to be logged in the Internet Explorer Compatibility Test Tool.
* **Example** – Most events include examples that demonstrate how to make the corresponding event create a log entry in the compatibility tool. These examples help make the description of the event more concrete.
* **Remediation** – Guidance on what you can do to eliminate the incompatibility from your Web site.

We’ve tried to provide as complete information as we could in the remediation section. Be aware that sometimes this guidance is short—there isn’t always a workaround. So some guidance is aimed at simply educating you about the issue so that you can design your site appropriately.

In many cases remediation guidance includes steps to disable a particular feature. It’s important to understand that the first and best option is always to redesign your application to eliminate the compatibility issue. Disabling a feature (many of which are security related), may fix a particular compatibility issue, but may also open a vulnerability to your browser. Disabling a feature is mainly useful during troubleshooting to observe behavior in an enabled vs. disabled state. But on an on-going basis, **disabling features should only be used as a last resort**—and even then only as a short term solution.

When security issues come up they will be called out with the label **SECURITY WARNING!** Be sure to pay special attention to these warnings.

# Events

## 1021 – MIME Handling Restrictions

### Logged Message

Windows Internet Explorer 8 and Internet Explorer 7 use Multipurpose Internet Mail Extensions (MIME) type information to decide how to handle files sent by a Web server. For example, when Internet Explorer receives a .jpg file, the user sees the file in an Internet Explorer window. If Internet Explorer receives an executable (.exe) file, it generally asks the user how to handle the file. The MIME Handling Restrictions security feature protects users from accidentally downloading or executing a dangerous file because of misleading MIME or file name extension information.

### What is it?

Internet Explorer uses the following pieces of information to decide how to handle a file.

* File name extension, the corresponding programmatic identifier (PROGID), and the class identifier (CLSID) for the registered handler of that file name extension.
* Content-Type from the HTTP header (MIME type), the corresponding PROGID, and the CLSID for the registered handler of that content or MIME type.
* Content-Disposition from the HTTP header.
* Results of a MIME scan.

During the file download into the cache, Internet Explorer compares the MIME type of the cache file to the extension of the cache file. If there is a mismatch, Internet Explorer reconciles the mismatch by renaming the file in the cache.

Before a file is loaded in its MIME handler or executed by its extension handler, Internet Explorer compares the CLSID of the MIME handler to the CLSID of the extension handler. A file download box comes up when there is a mismatch between mime type and extension handler. This prompt forces the user to confirm whether to load the file in the MIME handler. If the MIME handler rejects the mismatched file, Internet Explorer shows a download error dialog box and does not automatically execute the file in its extension handler.

A related change prevents the execution of a potentially corrupt file in its extension handler. Internet Explorer shows the download error dialog box for any file rejected by its MIME handler with the error code INET\_E\_CANNOT\_LOAD\_DATA and does not execute the file in its extension handler regardless of MIME type or extension.

These changes do not affect cases where the file uses a Content-Disposition: attachment HTTP header.

For these files, the final file name or extension suggested by the server is executed, regardless of any MIME/extension mismatch, if the user accepts the file download prompt.

### When is this event logged?

This event is logged when a possible dangerous or unknown MIME type is detected and the user is prompted what to do with it.

### Example

 Perform the following steps to see this event logged in the compatibility tool:

1. Create a Web page with the following contents. For this example call it 1021.aspx.

<html xmlns="http://www.w3.org/1999/xhtml">

<body>

<div>

<h1>Mime Handling Example</h1>

</div>

</body>

</html>

1. Install the file and the exe file in the root directory of the local web server. On an IIS server this means putting the file in this directory:

\Inetpub\wwwroot

1. Browse to the file:

<http://127.0.0.1/1021.aspx>

The browser downloads the .aspx file which Internet Explorer recognizes to be an HTML file. This causes the MIME Handling restriction event to be fired.

### Remediation

If you receive unwanted file download prompts due to an irreconcilable MIME mismatch, you can register the MIME handler's PROGID and bypass all download prompts, including the prompt on mismatch. Before registering the handler, verify that the MIME handler securely manages any delegated file. For example, confirm that the handler never enables an attacker to gain more user rights than enabled by zone of the originating file. You can do this through threat modeling, a code review for secure failure modes that checks for buffer overruns. If you determine the MIME handler can safely handle the files, you can register it by adding a new value to HKEY\_LOCAL\_MACHINE (or HKEY\_CURRENT\_USER)\SOFTWARE\Microsoft\Windows\CurrentVersion\Internet Settings\Secure Mime Handlers\MIME Handler PROGID.

In addition to changing the client-side settings, you can also change the Web server so it sends the appropriate Content-Type header for any blocked file, when the filename extension does not match the Content-Type header.

You must update the custom MIME handlers that intentionally rely on Internet Explorer to execute files that the custom MIME handler rejects. The most secure scenario is to handle files natively in the MIME handler rather than rejecting them. If you cannot change the MIME handler, there are a few options:

* You can develop a MIME handler and extension handler that are both part of the same CLSID. Internet Explorer accepts the CLSID match and does not prompt to download the file or block the file from execution in the extension handler.
* You can mark the MIME handler to be ignored by Internet Explorer when there is a MIME/extension mismatch. For example, if the MIME handler for a certain media MIME type has a mismatched extension, you can mark the PROGID of the MIME handler to be ignored on the mismatch (when the media file name extension belongs to a different PROGID). To do this, set the following value in the registry with the MIME handler to ignore.

HKEY\_CLASSES\_ROOT\PROG\_ID\_OF\_MIMEHANLDER\_TO\_IGNORE\PreferExecuteOnMismatch=0x00000001

* If neither option is viable, notify users of the incompatibility and explain how to save the mismatched file to the file system and how to launch it manually.

You can also disable this feature through feature control keys.

**Security Warning!** *If you disable this feature, users can unknowingly download malicious content, disguised with an incorrect filename extension. Once downloaded, an incorrect extension handler can run the file, enabling the malicious content to cause damage.*

You manage the restrictions through a security feature control registry key (FEATURE\_MIME\_HANDLING). Internet Explorer (Iexplore.exe) and Windows Explorer (Explorer.exe) run under this feature control by default. The following shows the registry key and enabled processes:

* HKEY\_LOCAL\_MACHINE (or HKEY\_CURRENT\_USER)\SOFTWARE\Microsoft\Internet Explorer\Main\FeatureControl\FEATURE\_MIME\_HANDLING\iexplore.exe= 0x00000001
* HKEY\_LOCAL\_MACHINE (or HKEY\_CURRENT\_USER)\SOFTWARE\Microsoft\Internet Explorer\Main\FeatureControl\FEATURE\_MIME\_HANDLING\explorer.exe= 0x00000001
* HKEY\_LOCAL\_MACHINE (or HKEY\_CURRENT\_USER)\SOFTWARE\Microsoft\Internet Explorer\Main\FeatureControl\FEATURE\_MIME\_HANDLING\process name.exe= 0x00000001

## 1022 – Windows Restrictions

### Logged Message

Windows Internet Explorer 8 and Internet Explorer 7 for Microsoft Windows XP with Service Pack 2 (SP2) and Microsoft Windows Vista place some restrictions on windows to prevent hidden information and user-interface spoofing. Internet Explorer Windows Restrictions are designed to prevent a scripted window from obscuring the Internet Explorer title bar, address bar, and status bar. Window Restrictions affect several Dynamic HTML (DHTML) scripting commands, such as the open (with chrome) and window.createPopup (chromeless) methods.

### What is it?

In Windows Internet Explorer, scripts can open two different types of windows and can resize and reposition existing windows. Malicious coders have used these script-opened windows and the script-driven window positioning to mislead and deceive users. The Window Restrictions security feature in Internet Explorer now restricts the opening and placement of windows by script to prevent malicious coders from misleading users. These restrictions include constraints on new Internet Explorer windows created by the window.open method and HTML pop-up windows created by the window.createPopup method. This also includes positioning and sizing of Internet Explorer windows. By understanding Window Restrictions and how they work, you can write your scripts so that your Web pages function as expected.

### When is this event logged?

This event is logged when an attempt is made to hide user interface elements through script.

### Example

 Perform the following steps to see this event logged in the compatibility tool:

1. Create a Web page with the following contents. For this example call it 1022.html.

<html xmlns="http://www.w3.org/1999/xhtml" >

<head>

<title>Windows Restrictions Example</title>

</head>

<body>

<form id="form1" runat="server">

<div>

<a href="#" onclick="window.open('Popup.html','\_blank','height = 400, width = 400, top = 2000');">Open</a>

</div>

</form>

</body>

</html>

1. Create another Web page with the following contents. For this example call it Popup.html.

<html xmlns="http://www.w3.org/1999/xhtml" >

<body>

<div>

<p>Pop-up Example</p>

</div>

</form>

</body>

</html>

1. Install the file in the root directory of the local web server. On an IIS server this means putting the file in this directory:

\Inetpub\wwwroot

1. Browse to the file:

<http://127.0.0.1/1022.html>

1. Click on the “Open” hyperlink.

The code invoked by the link attempts to create a popup window that is situated off-screen. This is prevented by the Windows Reuse Navigation Restriction. Internet Explorer forces the pop-up window to appear in the viewable area of the desktop and the event is logged to the compatibility tool.

### Remediation

The following are guidelines for working with script-initiated window calls:

For windows opened by using window.open:

* Expect the status bar to be present, and write your code accordingly. The status bar is On by default and is 20–25 pixels in height.
* Adjust your window's size and content so that it fits well visually with the window's overall size. The window does not cover the taskbar, so it might lose 40 pixels if the status bar is On and you do not account for the taskbar. Vertically size the window no more than 30 pixels outside the taskbar.
* Do not open windows off-screen. Internet Explorer moves windows by the smallest offset of x- and y-coordinates, enabling the window to fully display onscreen.
* When designing, you must consider how the display theme, the font size, and the resolution impact the display of the window.

*NOTE: Using the window.open method with Fullscreen=Yes results in a maximized window, not a kiosk mode window.*

For windows opened by using window.createPopup:

* Adjust your window's size and content so that it fits well visually with the window's overall size. With this new feature, the window does not cover its parent window's title bar or status bar, so it might lose 40 pixels if you do not account for the title and status bars. Vertically size the window no larger than the currently visible area of the page.
* Do not open chromeless windows outside of the HTML rendering surface of Internet Explorer. Internet Explorer moves windows by the smallest offset of x- and y-coordinates, allowing the window to fully display inside the client area.

*Important:*

*There is one exception: Up to half of the window can exist outside the left or right edge of the Internet Explorer client area.*

* When designing, you must consider how the display theme, the font size, and the resolution impact the display of the window.

By default, sites in the Local Intranet and Trusted Sites zones enable windows to open and to create pop-up windows. When developing an extranet Web application, you might be able to configure your user's browsers to add the Web site to one of these zones.

You may also disable this feature through feature control keys.

**Security Warning!** *Disabling this feature exposes the browser to possible attacks—including spoofing. The visible security features of Internet Explorer windows, like the status bar, provide information to help users to determine the source and the security level of a Web page. If there are hidden security elements, users might believe they are on a trustworthy page or are interacting with a system process instead of a malicious host. Malicious use of window relocation can present false information to the user, obscure important information, or otherwise spoof important elements of the user interface in an attempt to motivate the user to take unsafe actions or to divulge sensitive information.*

*Disabling the feature should only be used as a temporary measure during troubleshooting—to compare behavior of the application when the feature is enabled or not. It is not recommended that the feature be left disabled on an on-going basis.*

You manage the Window Restrictions with a security feature control registry key (FEATURE\_WINDOW\_RESTRICTIONS). Internet Explorer (Iexplore.exe) and Windows Explorer (Explorer.exe) run under this feature control by default. The following list shows the registry keys and enabled processes:

* HKEY\_LOCAL\_MACHINE (or HKEY\_CURRENT\_USER)\SOFTWARE\Microsoft\Internet Explorer\Main\FeatureControl\FEATURE\_WINDOW\_RESTRICTIONS\iexplore.exe= 0x00000001
* HKEY\_LOCAL\_MACHINE (or HKEY\_CURRENT\_USER)\SOFTWARE\Microsoft\Internet Explorer\Main\FeatureControl\FEATURE\_WINDOW\_RESTRICTIONS\explorer.exe= 0x00000001
* HKEY\_LOCAL\_MACHINE (or HKEY\_CURRENT\_USER)\SOFTWARE\Microsoft\Internet Explorer\Main\FeatureControl\FEATURE\_WINDOW\_RESTRICTIONS\process name.exe=0x00000001

Applications that host the WebBrowser control can also take advantage of the security feature control, by adding their process to the same registry locations. You can do this programmatically, by using the CoInternetSetFeatureEnabled function.

*NOTE: If an application does not run under this security feature control, the WebBrowser control behaves the same as previous versions of Internet Explorer*

## 1023 – Zone Elevation Restrictions

### Logged Message

Zone elevation is an attempt to gain access to a stronger security-enabled zone from a weaker zone. Zone Elevation Restrictions prevent the overall security context of any link on a Web page from being higher than the security context of the root URL. For example, a page in the Internet zone cannot navigate to a page in the Local intranet zone except as the result of a user-initiated action. A script cannot automatically make this sort of navigation without user interaction, such as a mouse click or a keystroke. Zone Elevation Restrictions also disables JavaScript navigation if there is no security context.

### What is it?

Zone Elevation Restrictions prevent the overall security context of any link on a page from being higher than the security context of the root URL. This means, for example, that a page in the Internet zone cannot navigate to a page in the Local intranet zone except as the result of a user-initiated action. A script cannot automatically make this sort of navigation without user interaction, such as a mouse click or a keystroke. Zone Elevation Restrictions also disables JavaScript navigation if there is no security context.

*Note: The security ranking of the zones from least trusted to most trusted is Restricted sites zone, Internet zone, Local intranet zone, Trusted sites zone, and Local Machine zone.*

### When is this event logged?

This event is logged when an attempt is made to gain access to a stronger security-enabled zone from a weaker zone.

### Example

 Perform the following steps to see this event logged in the compatibility tool:

1. Create a Web page with the following contents. For this example call it 1023Target.html. The file can be placed anywhere. For this example, the file is located in C:\.

<html xmlns="http://www.w3.org/1999/xhtml">

<head>

<title>Zone Elevation</title>

</head>

<body>

</body>

</html>

1. Create another Web page with the following contents. For this example call it 1023Source.html.

<html xmlns="http://www.w3.org/1999/xhtml" >

<head>

<title>Zone Elevation</title>

</head>

<body>

<script>

window.location = 'C:\1023Target.html';

</script>

</body>

</html>

1. Install the 1023Source.html file in the root directory of the local web server. On an IIS server this means putting the file in this directory:

\Inetpub\wwwroot

1. Browse to the file:

<http://localhost/1023Source.html>

*A blank page is displayed with an Error (note the yellow exclamation icon in the lower left).*

1. Double-click the **Done** message in the lower left of the browser.

*This displays the detail of the error that occurred. In this case, it’s an* ***Access Denied*** *error.*

The Web page attempts to re-direct to the 1023Target.html file. But since 1023Target.html is located in the Local Machine Zone (and the calling page is located in the Local Intranet Zone) you receive an access denied error and the event is logged to the compatibility tool.

### Remediation

You manage the Zone Elevation Restrictions through a security feature control registry key (FEATURE\_ZONE\_ELEVATION). Internet Explorer (Iexplore.exe) and Windows Explorer (Explorer.exe) run under this feature control by default. The following shows the registry keys and enabled processes:

* HKEY\_LOCAL\_MACHINE (or HKEY\_CURRENT\_USER)\SOFTWARE\Microsoft\Internet Explorer\Main\FeatureControl\FEATURE\_ZONE\_ELEVATION\iexplore.exe= 0x00000001
* HKEY\_LOCAL\_MACHINE (or HKEY\_CURRENT\_USER)\SOFTWARE\Microsoft\Internet Explorer\Main\FeatureControl\FEATURE\_ZONE\_ELEVATION\explorer.exe= 0x00000001
* HKEY\_LOCAL\_MACHINE (or HKEY\_CURRENT\_USER)\SOFTWARE\Microsoft\Internet Explorer\Main\FeatureControl\FEATURE\_ZONE\_ELEVATION\process name.exe=0x00000001

Applications that host the WebBrowser control can also take advantage of the security feature control by adding their process to the same registry locations. You can do this programmatically, using the CoInternetSetFeatureEnabled function.

*Note: If an application does not run under this security feature control, the WebBrowser control behaves the same as previous versions of Internet Explorer.*

Applications can also use the CoInternetIsFeatureZoneElevationEnabled function to determine whether to enable navigation from one URL to another. The function returns an HRESULT based on the value of the URL policy for the URL action flag URLACTION\_FEATURE\_ZONE\_ELEVATION in the zone of the specified URL.

You can modify the Internet Explorer security zone settings if a trusted Web application is unusable.

To modify the security zone settings:

1. In Internet Explorer, click Tools, point to Internet Options, and then click the Security tab.
2. Select the required zone, and then click Custom Level.
3. Scroll down under Miscellaneous and then select the check box for **Web sites in less privileged web content zone can navigate into this zone**.

If you need to open a local file from the Internet or local intranet zone, you can add a Mark of the Web comment in the HTML code. This Internet Explorer feature forces your HTML files into a zone other than the Local Machine zone, based on the URL identified in the comment.

To insert a Mark of the Web comment into your HTML file, add the following comment:

<!-- saved from url= <(0022)http://www.fabrikam.com>-->

Where http://www.fabrikam.com is the URL of the Internet or intranet domain hosting the page, and 0022 is the length of the URL.

You can use the Mark of the Web comment with .mht, .xml, or .htc files; however, if you use a version prior to Internet Explorer 6 for Windows XP with Service Pack 2 (SP2), Mark of the Web is ignored.

If you need to access local resource files like .doc or .xls, you can disable the zone elevation feature for the Local Machine zone.

**Security Warning!** *The Local Machine zone is a prime target for malicious users attempting to gain access to a more secure zone. Disabling this feature reduces security in the Local Machine zone.*

*Zone Elevation restrictions protect personal or other confidential content on your local machine. Disabling Zone Elevation Restrictions can facilitate serious breaches of confidentiality including identity theft.*

The zone elevation feature for the Local Machine zone can be disabled by using the following registry key:

HKEY\_CURRENT\_USER\Software\Microsoft\Windows\CurrentVersion\Internet Settings\Lockdown\_Zones\0\2101 = 0

## 1024 – Binary Behaviors Restrictions

### Logged Message

Windows Internet Explorer 8 and Internet Explorer 7 contain dynamic binary behaviors, which are components attached to HTML elements that encapsulate specific functionality. Internet Explorer security settings do not control binary behaviors, so the components can work on Web pages in the Restricted sites zone. The Binary Behaviors Restrictions security feature disables the binary behavior in the Restricted sites zone by default. In combination with the Local Machine Zone Lockdown security feature, administrative approval is required for binary behaviors to run in the Local machine zone by default.

### What is it?

A behavior is a component that implements two interfaces, but in both cases it relies heavily on COM component technology. You can write behaviors in three ways: as HTML components (HTC), Windows Script Components (WSC), or COM objects. When you use raw COM interfaces, or the WSC infrastructure, the resulting behavior looks unequivocally like a COM object.

*Note: Binary behaviors differ from attached behaviors and element behaviors, which are written in script. For more information, see the* [*Introduction to DHTML Behaviors*](http://msdn.microsoft.com/en-us/library/ms531079(VS.85).aspx) *and* [*About Element Behaviors*](http://msdn.microsoft.com/en-us/library/ms531426.aspx) *topics.*

**Binary Behavior Restriction Setting**

The Binary Behavior Restriction security feature creates a new URL action setting, Binary and Script Behaviors, in each Internet Explorer security zone. The default value for this setting is Enable for all zones except the Restricted sites zone and the Locked-Down Local Machine zone. In the Restricted sites zone, the default value is Disable. In the Locked-Down Local Machine zone, the default value is Administrator-approved.

**Automatic Download Blocking and the Registry**

Applications that host the WebBrowser control and use Internet Explorer functionality in the Restricted sites zone might be affected. For example, e-mail applications that use a binary behavior to render HTML e-mail in the Restricted sites zone might require modification.

### When is this event logged?

This event is logged when a binary behavior is triggered in the Restricted site zone.

### Example

 Perform the following steps to see this event logged in the compatibility tool:

1. Create a file with the following contents. For this example call it MouseOver.htc. The file can be placed anywhere. For this example, the file is located on the desktop.

<PUBLIC:HTC>

<PUBLIC:ATTACH event="onmouseover"

handler="fnOver"/>

<PUBLIC:ATTACH event="onmouseout"

handler="fnOut"/>

<script LANGUAGE="jscript">

function fnOver()

{element.style.color="red";}

function fnOut()

{element.style.color="";}

</script>

</PUBLIC:HTC>

1. Add the following HTML to a Web page and save to the same location as the mouseover.htc file. For this example call it 1024.html.

<html xmlns="http://www.w3.org/1999/xhtml" >

<head>

<title>Binary Behaviors Restrictions</title>

</head>

<body>

<a href="http://foo" style="behavior:url(mouseover.htc)">Click here</a>

</body>

</html>

1. Open a browser and navigate to the Web page. For example:

C:\Users\JohnS\Desktop\1024.html

*A message is displayed in the information bar indicating that the browser is restricting the Web page from running ActiveX controls. This is due to the Mouse Over behavior we created.*

For comparison, you can install the same files in the root directory of the local web server and browse to <http://localhost/1024.html>. This places the page in the Local intranet zone which does not have the binary behavior restriction. In that case, the behavior just runs without a prompt in the information bar.

### Remediation

To use binary behaviors from the Restricted sites zone, an application can implement a custom security manager. For more information about URL security zones, see the [Creating a Customized URL Security Manager](http://msdn.microsoft.com/en-us/library/ms537182(VS.85).aspx) topic on MSDN.

You can also disable this feature through feature control keys.

**Security Warning!** *This feature mitigates attacks from malicious binary behaviors and allows the user to control the use of binary behaviors on a per-zone basis. Disabling the feature should only be used as a temporary measure during troubleshooting—to compare behavior of the application when the feature is enabled or not. It is not recommended that the feature be left disabled on an on-going basis.*

You manage the restrictions through a security feature control registry key (FEATURE\_BEHAVIORS). Internet Explorer (Iexplore.exe) and Windows Explorer (Explorer.exe) are enabled by default. The following shows where to add the registry keys:

* HKEY\_LOCAL\_MACHINE (or HKEY\_CURRENT\_USER)\SOFTWARE\Microsoft\Internet Explorer\Main\FeatureControl\FEATURE\_BEHAVIORS\iexplore.exe= 0x00000001
* HKEY\_LOCAL\_MACHINE (or HKEY\_CURRENT\_USER)\SOFTWARE\Microsoft\Internet Explorer\Main\FeatureControl\FEATURE\_BEHAVIORS\explorer.exe= 0x00000001
* HKEY\_LOCAL\_MACHINE (or HKEY\_CURRENT\_USER)\SOFTWARE\Microsoft\Internet Explorer\Main\FeatureControl\FEATURE\_BEHAVIORS\process name.exe=0x00000001

Applications that host the WebBrowser control can also take advantage of the security feature control by adding their process to the same registry locations. You can do this programmatically by using the CoInternetSetFeatureEnabled function.

*Note: If an application does not run under this security feature control, the WebBrowser control behaves the same as previous versions of Internet Explorer.*

After the security feature control is enabled for a process, the value of the URL action flag URLACTION\_BEHAVIOR\_RUN determines whether binary behaviors are allowed to run. This flag can be set differently for each security zone. The default value for this flag is URLPOLICY\_ALLOW for all zones except the Restricted sites zone. In the Restricted sites zone, the default value is URLPOLICY\_DISALLOW.

The following table shows the new settings for turning on or off the existing binary behaviors functionality.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Setting name** | **Location** | **Previous default** | **Default value** | **Possible values** |
| **\*** | HKCU{LM}\Software\Microsoft\  Internet Explorer\Main\FeatureControl\  FEATURE\_BINARY\_BEHAVIOR\_LOCKDOWN | None | 1 | 0 (off)  1 (on) |
| **2000** | HKCU\Software\Microsoft\Windows\ CurrentVersion\Internet Settings\Zones\3 | None | 3 (disabled for restricted zone)  0 (enable for all other zones) | 3 (disabled)  0 (enabled) |

*Note: The asterisk (\*) is used to indicate that all processes are opted-in for this feature control setting by default. You can also modify the binary behaviors setting through Group Policy as part of the Internet Explorer Security Zones and Content Ratings setting.*

## 1025 – Object Caching Protection

### Logged Message

A reference to an object is no longer accessible when the user browses to a new domain in Windows Internet Explorer 8, Internet Explorer 7 and Internet Explorer 6 for Microsoft Windows XP with Service Pack 2 (SP2). There is a security context on all scriptable objects so that access to all cached objects is blocked. Additionally, Internet Explorer blocks access when browsing within the same domain (fully qualified domain name). A reference to an object is no longer accessible after the context has changed due to navigation.

### What is it?

In earlier versions of Windows, some Web pages were able to access objects cached from another Web site. In Internet Explorer 8, a reference to an object is no longer accessible when the user browses to a new domain. There is a new security context for all scriptable objects so that access to all cached objects is blocked. Additionally, access is blocked when browsing within the same domain (fully qualified domain name). A reference to an object is no longer accessible after the context has changed due to navigation.

### When is this event logged?

This event is logged if script attempts to access a cached object.

### Example

 Perform the following steps to see this event logged in the compatibility tool:

1. Create a Web page with the following contents. For this example call it 1025.html.

<html>

<head>

<title>Object Caching test</title>

<script language="jscript">

var vulObjs={

showModalDialog:{

Cache:function (openWin) { return openWin.showModalDialog },

Exploit:function (openWin,vCache,sInject) {

if (document.implementation) {

function oExploit(iSec) {

return {

};

}

oExploit.length=1;

var oSecurity={

document:{

all:{

tags:function (sTag) {

return sTag=="link" ? oExploit : [];

}

}

}

}

vCache("res://shdoclc.dll/analyze.dlg",oSecurity,"");

}

},

Options:fInject

}

};

function fInject(oVulnItem,openWin) {

oExploitOptions.innerHTML="<input type=\"button\" id=\"oExecute\" value=\"Execute\">";

oExecute.onclick=function () {

try {

openWin.focus();

oVulnItem.Exploit(openWin,oVulnItem.CachedObject,"alert(document.cookie)");

}

catch (oErr) {

alert("Error occurred while injecting ("+oErr.description+").");

}

}

}

function assignLoc(sURL,openWin) {

try {

openWin.location.href=/^(res|file|http|https):\/\//i.test(sURL) ? sURL : "http://"+sURL;

}

catch (oErr) {

openWin.location.href="getLocal.asp?resName="+escape(sURL);

}

}

onload=function () {

var oOption,sMethod,openWin;

openWin=open("blank.html","victim","width=400,height=300");

focus();

setTimeout(

function () {

for (sMethod in vulObjs) {

vulObjs[sMethod].CachedObject=vulObjs[sMethod].Cache(openWin);

}

oStatus.innerText=

"";

oGUI.style.display="block";

assignLoc("http://microsoft.com",openWin);

oExploitOptions.style.display="block";

vulObjs["showModalDialog"].Options(vulObjs["showModalDialog"],openWin);

},

3000

);

}

</script>

</head>

<body>

<div id="oStatus">Caching objects please wait...</div>

<div id="oGUI">

<div id="oExploitOptions"></div>

</div>

</body>

</html>

1. Install the file in the root directory of the local web server. On an IIS server this means putting the file in this directory:

\Inetpub\wwwroot

1. Browse to the file:

<http://127.0.0.1/1025.html>

*A message appears in the information bar indicating that Pop-ups are blocked.*

1. Right click the information bar and select “Always allow pop-ups from This Site”

*This allows the pop-up in the example to be displayed. You can turn off pop-ups later if you like.*

*Notice that a second window appears that redirects to* [*www.microsoft.com*](http://www.microsoft.com)

1. Click the **Execute** button on the 1025.html page.

Internet Explorer displays a dialog that reads:

*Error occurred while injecting (Permission Denied)*

What’s happening is that the main page (1025.html) is opening up a second window that redirects to [www.microsoft.com](http://www.microsoft.com). The main page caches a reference to that second window. When you click the Execute button, you’re invoking code on the main page that attempts to manipulate the second window through its cached reference. Since the security context of the two windows are different, Internet Explorer prevents the attempt, and displays an error. The **Object Caching** event is also logged to the compatibility tool.

### Remediation

Moving your Web site to an intranet site is an alternative that makes object caching acceptable. Changing the Web page so it does not rely on object caching is another option. For more information on updating a Web page that experiences a problem with this security feature, see [Knowledge Base Article 884697](http://support.microsoft.com/kb/884697).

For more articles on object caching protection, including any updates made to this feature, see the following [Knowledge Base Article on object caching.](http://technet.microsoft.com/en-us/library/cc766459.aspx)

## 1026 – ActiveX Blocking

### Logged Message

ActiveX controls are reusable software components based on Microsoft ActiveX technology. ActiveX controls add interactivity and additional functionality, such as animations or pop-up menus to a Web page, application, or software development tool. Windows Internet Explorer 8, Internet Explorer 7, and Internet Explorer 6 for Microsoft Windows XP with Service Pack 2 (SP2) block controls that are unsigned, invalid, or explicitly distrusted by the user. In Internet Explorer 8, users can allow controls to run on more than one Web site, or all Web sites, by responding to the Information bar that drops down when a control is requested for use. These sites can also be edited through the Manage Add-ons interface.

### What is it?

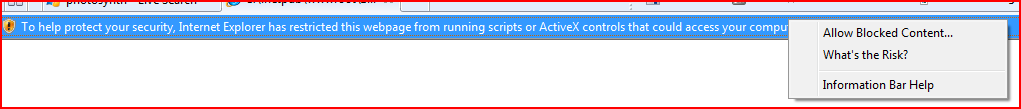
ActiveX controls are very important to web applications because they allow developers to enhance Web pages with additional software application features that won't work in standard HTML Web pages. Web developers use ActiveX controls to add animation, multimedia and other features to their Web sites.

Because ActiveX controls, or any browser extension, add features for Web sites, they also increase the possibility of a security vulnerability. Active-X Opt-in, added in Internet Explorer 7, will reduce the number of ActiveX controls available to Web sites on the Internet and thereby reduce the chances of a security vulnerability. Internet Explorer 8 makes it easy to use common sites with important controls but lets users opt-in to using the advanced features that might be exposed by more obscure ActiveX controls.

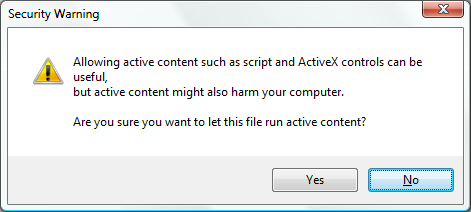
This Internet Explorer feature is called ActiveX Opt-in. By default, ActiveX Opt-in disables the controls on a user's machine. When the user encounters a Web page with a disabled ActiveX control, they will see an Information bar with the following text:

*This Web site wants to run the following add-on "ABC Control" from "XYZ Publisher". If you trust the Web site and the add-on and want to allow it to run, click here …"*

The user can choose to enable the ActiveX control from this Information bar, as shown in the following figure.



After the user selects "Run ActiveX Control" they are presented with the following Authenticode dialog from which they can allow the control to run.



Some ActiveX controls will not be disabled by ActiveX Opt-in.

1. Controls that are commonly used and that were designed with security scrutiny will not be disabled. These controls will appear on a pre-approved list.
2. Controls which were used in Internet Explorer before upgrading to Internet Explorer 8.
3. Controls which the user downloads through Internet Explorer 8 will be automatically enabled during the download and install process.

Controls which are on the pre-approved list will run without the Opt-in prompt.

Users are prompted in the follow circumstances:

* On Internet Explorer 8 the first run of the control requires approval
* Each ActiveX control will require approval on a per site basis
* When the ActiveX control needs to be installed
* ActiveX failed to install / was blocked by a security feature

Per-site activation applies when an ActiveX control is built to only run on a select group of Web sites. If an attempt is made to instantiate such a control on a site other one for which it was built, then Internet Explorer blocks the instantiation.

References:

* <http://blogs.msdn.com/ie/archive/2008/05/07/ie8-security-part-ii-activex-improvements.aspx>
* <http://code.msdn.microsoft.com/ie8whitepapers/Release/ProjectReleases.aspx?ReleaseId=561>

### When is this event logged?

 Logging occurs anytime Internet Explorer disables a control via the ActiveX opt-in feature.

### Example

Perform the following steps to see this event logged in the compatibility tool:

1. Create a Web page with the following contents. For this example call it 1026.html. The file can be placed anywhere. For this example, the file is located on the desktop.

<html xmlns="http://www.w3.org/1999/xhtml" >

<body>

<object classid="clsid:22D6F312-B0F6-11D0-94AB-0080C74C7E95">

</body>

</html>

1. Open a browser and navigate to the Web page. For example:

C:\Users\JohnS\Desktop\1026.html

The page contains an object tag whose behaviour is to instantiate the Media Player ActiveX control. But Internet Explorer prompts the user via the information bar asking if they wish to run the add-on. The event is also logged.

### Remediation

You may configure Internet Explorer to install ActiveX controls automatically, bypassing any prompting in the information bar.

**Security Warning!** *The automatic prompting in Internet Explorer is implemented to help prevent potentially malicious ActiveX controls from being run on the user’s Web page. Disabling the feature should only be used as a temporary measure during troubleshooting—to compare behavior of the application when the feature is enabled or not. It is not recommended that the feature be left disabled on an on-going basis.*

To automatically download ActiveX Controls, perform the following tasks:

1. Select the Tools >Internet Options menu item
2. Select the **Security** tab on the **Internet Options** dialog
3. Select a zone (**Internet**, **Local Intranet**, etc.)
4. Click the **Custom Level** button
5. Enable both of these options:

ActiveX controls and plug-ins >

Download signed ActiveX controls

Download unsigned ActiveX controls

This generates a security warning error but does install the ActiveX control without being prompted.

*Note: This will only work if the control has previously not been installed. If the control is pre-installed, you will still see a per-domain approval information bar. If the control is pre-approved (on the pre-approved list), Internet Explorer will assume that the control is allowed on every site by default. This can still be overridden by the user.*

## 1027 – Pop-Up Blocking

### Logged Message

The Pop-Up Blocker is a feature that blocks pop-up (and pop-under) windows initiated automatically by a Web site. Windows Internet Explorer 8 and Internet Explorer 7 block pop-up windows in the Internet and Restricted sites zones by default. However, Pop-up Blocker allows pop-up windows initiated by a user's actions. Users can configure Internet Explorer 8 and Internet Explorer 7 for Microsoft Windows XP with Service Pack 2 (SP2) and Microsoft Windows Vista to be more or less restrictive. Users can also turn off Pop-up Blocker completely.

### What is it?

Pop-up blocking is a mechanism to prevent new browser windows being opened automatically using script. Generally, the Pop-up Blocker enables a window to open under the following circumstances:

* When initiated by user action, such as clicking a button or hyperlink
* When opened in the Trusted sites and Local intranet zones (considered safe)
* When opened by other applications running on the local computer

The affected script methods are:

* window.open
* window.showHelp
* window.showModalDialog
* window.showModelessDialog
* window.external
* window.NavigateAndFind

*Note: Pop-ups created with window.createPopup are unaffected by the Pop-up Blocker.*

A pop-up block log entry includes the values taken into account to determine whether or not to display the pop-up. This is a combination of multiple INewWindowManager flags (NWMF). For more information, see the [NWMF Enumerated Type](http://msdn.microsoft.com/en-us/library/bb762518.aspx) topic on MSDN.

### When is this event logged?

This event is logged when the pop-up blocker blocks a new pop-up window.

### Example

Perform the following steps to see this event logged in the compatibility tool:

1. Create a Web page with the following contents. For this example call it 1027.html.

<html xmlns="http://www.w3.org/1999/xhtml" >

<head>

<title>Windows Restrictions Example</title>

</head>

<body>

<script type="text/jscript">

window.open("http://www.microsoft.com","\_blank");

</script>

</body>

</html>

1. Install the file in the root directory of the local web server. On an IIS server this means putting the file in this directory:

\Inetpub\wwwroot

1. Ensure the Internet Explorer pop-up blocker is enabled. This setting can be found in Tools > Internet Options > Privacy.
2. Browse to the file:

<http://127.0.0.1/1027.html>

The page will attempt to create a new popup using window.open. Internet Explorer will block the attempt and prompt the user for confirmation that they want to open the pop-up. The event is also logged.

### Remediation

There are two ways to modify the Pop-up Blocker:

* From Internet Explorer, click Tools, click Pop-up Blocker, and then click **Turn On Pop-up Blocker**.
* From the Internet Explorer, Internet Options dialog box, click the Privacy tab, and then check **Turn on Pop-up Blocker**.

From the Internet Options dialog box, you can also click Settings and identify specific Web sites to enable pop-up windows.

Web developers can perform the following actions:

* Modify your Web page to open a new window by user-initiated action, such as a button click or hyperlink.
* Instruct your users to enable pop-ups from specific sites (or use Group Policy to add specific Web sites to the trusted or local intranet zone).
* Check for a NULL return value for any windows you open. This indicates whether the pop-up window opened successfully and enables you to handle either case.

*Note: Some Web sites call the window.open method on an onunload event to get users to stay on the site longer or to serve them with an advertisement. Some Web sites open a new window outside of the visible desktop area or open a window that is larger than the screen. Internet Explorer does not enable any of this to occur. For more information, see* [*Window Restrictions.*](http://msdn.microsoft.com/en-us/library/ms537638(VS.85).aspx)

You can also control the pop-up blocking feature through the registry.

You manage Pop-up Blocking through a separate registry key from the other security features. Internet Explorer (Iexplore.exe) and Windows Explorer (Explorer.exe) run under this feature control by default. The following shows the registry keys and enabled processes:

* HKEY\_LOCAL\_MACHINE (or HKEY\_CURRENT\_USER)\SOFTWARE\Microsoft\Internet Explorer\New Windows\PopupMgr="Yes"
* HKEY\_LOCAL\_MACHINE (or HKEY\_CURRENT\_USER)\SOFTWARE\Microsoft\Internet Explorer\Main\FeatureControl\FEATURE\_WEBOC\_POPUPMANAGEMENT\process name.exe=0x00000001

Applications that host the WebBrowser can also take advantage of the security feature control by adding their process to the same registry locations. You can do this programmatically, using the CoInternetSetFeatureEnabled function.

*Note:*

*If an application does not run under this security feature control, the WebBrowser control behaves the same as previous versions of Internet Explorer.*

An application that hosts the WebBrowser control can also implement the INewWindowManager interface to use or extend the Pop-up Blocker.

#### How Can I Fix Problems Identified by this Security Feature?

You can put trusted Web sites on the allow list so pop-up windows are not blocked, as follows:

* HKEY\_CURRENT\_USER\SOFTWARE\Microsoft\Internet Explorer\New Windows\Allow\URLName=*URLs of trusted sites* ;

For example, setting **URLName** = 127.0.0.1 mean that the pop-up in the example above is not blocked.

## 1028 – Automatic Download Blocking

### Logged Message

Automatic Download Blocking automatically suppresses file download dialog boxes not initiated by the user (such as by clicking the mouse or hitting a key). When a dialog box is blocked, the Information bar appears at the top of the window. Users can download the blocked content by clicking the Information bar.

### What is it?

When a dialog box is automatically blocked, the Information bar appears at the top of the window, stating:

"To help protect your security, Internet Explorer blocked this site from downloading files to your computer. Click here for more options…"

Users then have the option to download the blocked content, by clicking the Information bar.

By moving download prompts to the Information bar, you prevent users from installing unwanted code on their computers. Previously, sites overwhelmed users with file download prompts and, as a result, users accidentally ran unwanted software on their computer. With this change, file download prompts launched automatically are more likely the result of a user's deliberate click and not an accidental action.

### When is this event logged?

This event is logged when an attempt is made to download a file that is not initiated by the user.

### Example

Perform the following steps to see this event logged in the compatibility tool:

1. Create a Web page with the following contents. For this example call it 1028.html.

<html xmlns="http://www.w3.org/1999/xhtml" >

<body>

<script>

window.location="<YOUR EXE>.exe"

</script>

</body>

</html>

1. Replace <YOUR EXE> with the file name of an exe file that you can place on the server.
2. Install the file and the exe file in the root directory of the local web server. On an IIS server this means putting the file in this directory:

\Inetpub\wwwroot

1. Browse to the file:

<http://127.0.0.1/1028.html>

The page will attempt to automatically download the exe file, which will be blocked by Internet Explorer and the user will be prompted.

### Remediation

Applications that host the WebBrowser control can take advantage of the tighter restrictions on file downloads. You manage the restrictions through a security feature control registry key (FEATURE\_RESTRICT\_FILEDOWNLOAD). Internet Explorer (Iexplore.exe) and Windows Explorer (Explorer.exe) run under this feature control by default. The following shows where to add the registry keys:

* HKEY\_LOCAL\_MACHINE (or HKEY\_CURRENT\_USER)\SOFTWARE\Microsoft\Internet Explorer\Main\FeatureControl\FEATURE\_RESTRICT\_FILEDOWNLOAD\iexplore.exe= 0x00000001
* HKEY\_LOCAL\_MACHINE (or HKEY\_CURRENT\_USER)\SOFTWARE\Microsoft\Internet Explorer\Main\FeatureControl\FEATURE\_RESTRICT\_FILEDOWNLOAD\explorer.exe= 0x00000001
* HKEY\_LOCAL\_MACHINE (or HKEY\_CURRENT\_USER)\SOFTWARE\Microsoft\Internet Explorer\Main\FeatureControl\FEATURE\_RESTRICT\_FILEDOWNLOAD\process name.exe=0x00000001

Applications that host the WebBrowser control can also take advantage of the security feature control, by adding their process to the same registry locations. You can do this programmatically, by using the CoInternetSetFeatureEnabled function.

*Note: If an application does not run under this security feature control, the WebBrowser control behaves the same as previous versions of Internet Explorer.*

When a process is running the security feature control for file download restrictions, the URL action flag URLACTION\_AUTOMATIC\_DOWNLOAD\_UI determines whether to use the Information bar for file downloads.

*Note: You might choose to update Web sites that contain an image of the Download dialog box, used to instruct users where to click to accept the control, to reflect the new dialog box for users of the Windows Vista® operating system. Use the userAgent string to determine the correct version of the browser.*

#### How Can I Fix Problems Identified by this Security Feature?

To ensure that your Web site downloads are not blocked, do not automatically launch a file download. Instead, use hyperlinks or buttons that require a user action. If you use a script to navigate to the resource, it must run synchronously within the context of the OnClick event handler for the link.

#### How Can I Work Around this Problem?

As a temporary solution, you can disable the feature that blocks unsolicited downloads, allowing them to come through and not be intercepted by the Information bar.

**Security Warning!** *If you disable this feature, malicious sites can overwhelm users with file download prompts and, as a result, users can accidentally accept downloading and running unwanted or hostile software on their computers. We do not recommend this action and strongly suggest that developers not rely on customers turning this feature off.*

*Disabling the feature should only be used as a temporary measure during troubleshooting—to compare behavior of the application when the feature is enabled or not. It is not recommended that the feature be left disabled on an on-going basis.*

To access the setting that controls automatic download blocking

1. From Internet Explorer, click Tools, and then click Internet Options.

*The Internet Options dialog box appears.*

1. Click the Security tab.

*The Internet Options dialog box shows your security levels and zones.*

1. Select the Internet zone, and then click Custom Level.

*The Security Settings dialog box appears.*

1. Scroll down to Downloads, and then select the Enable option for **Automatic prompting for file downloads**.

Another way to control automatic downloads is to add the Web site to the Intranet zone, where automatic prompting for file downloads is enabled by default. Neither of these options is recommended as a long-term solution.

## 1030 – Local Machine Zone Lockdown (LMZL)

### Logged Message

Local Machine zone lockdown (LMZL) secures the Local Machine zone by tightening restrictions on several URL actions. Any time a restricted URL action is attempted the Information bar appears. Clicking the Information bar removes the lock on the restricted content.

### What is it?

The Local Machine zone is an implicit zone for content that exists on the local computer. The content found on the user's computer (except for content that Internet Explorer caches on the local system) is treated with a high level of trust.

Content that Internet Explorer caches is accessed through the URL of origin and is assigned to the appropriate zone for that URL.

The following table shows the seven URL actions that are more restrictive in the Lockdown zone than in the Local Machine zone.

|  |  |
| --- | --- |
| **URL Action** | **URL Policy** |
| **URLACTION\_ACTIVEX\_OVERRIDE\_OBJECT\_SAFETY** | URLPOLICY\_DISALLOW |
| **URLACTION\_ACTIVEX\_RUN** | URLPOLICY\_DISALLOW |
| **URLACTION\_BEHAVIOR\_RUN** | URLPOLICY\_DISALLOW |
| **URLACTION\_CLIENT\_CERT\_PROMPT** | URLPOLICY\_DISALLOW |
| **URLACTION\_DOWNLOAD\_UNSIGNED\_ACTIVEX** | URLPOLICY\_DISALLOW |
| **URLACTION\_JAVA\_PERMISSIONS** | URLPOLICY\_JAVA\_PROHIBIT |
| **URLACTION\_SCRIPT\_RUN** | URLPOLICY\_DISALLOW |

*Note: For more information on URL actions and pointers to what they mean, see the* [*Introduction to URL Security Zones*](http://msdn.microsoft.com/en-us/library/ms537183.aspx) *topic on MSDN.*

### When is this event logged?

This event is logged any time a restricted URL action is attempted.

### Example

Perform the following steps to see this event logged in the compatibility tool:

1. Create a Web page with the following content. For this example call it 1030.html. The file can be placed anywhere. For this example, the file is located on the desktop.

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN" "http://www.w3.org/TR/REC-html40/strict.dtd">

<html>

<head>

<meta http-equiv="X-UA-Compatible" content="IE=8" >

<title>AJAX MAP</title>

</head>

<script type="text/javascript" src="http://dev.virtualearth.net/mapcontrol/mapcontrol.ashx?v=6"></script>

<script type="text/javascript">

var oMap = null;

var iZoomLevel = 0;

function GetMap()

{

oMap = new VEMap('myMap');

oMap.LoadMap();

oMap.AttachEvent("onendzoom", ZoomHandler);

iZoomLevel = oMap.GetZoomLevel();

window.location.hash = iZoomLevel;

}

function ZoomHandler(e)

{

iZoomLevel = oMap.GetZoomLevel();

window.location.hash = iZoomLevel;

}

function HashChangeHandler()

{

var hash = window.location.hash;

var iNewZoomLevel = hash.substr(1);

if (iNewZoomLevel != iZoomLevel)

{

iZoomLevel = iNewZoomLevel;

oMap.SetZoomLevel(iNewZoomLevel);

}

}

</script>

<body style='overflow: scroll; heigth:100%' onload="GetMap();" onhashchange="HashChangeHandler();">

<div id='myMap' style='position: relative; width: 500px; height: 500px; valign: center'></div>

</body>

</html>

1. Open a browser and navigate to the Web page. For example:

C:\Users\JohnS\Desktop\ 1030.html

This opens the page in the local machine zone. Since the page includes script, the Local Machine Zone Lockdown event is logged to the compatibility tool. The user is also prompted with the following message:

*To help protect your security, Internet Explorer has restricted this Web page from running scripts or ActiveX controls that could access your computer. Click here for options...*

The prompt in the information bar is generated because Internet Explorer is restricting the **URLACTION\_SCRIPT\_RUN** URL action.

### Remediation

If your Web page runs a Microsoft ActiveX control or script, you can add a Mark of the Web comment to the HTML of the page. Mark of the Web is an Internet Explorer feature that forces the HTML file into the security zone of the specified URL. This enables the Web page to run the script or ActiveX control in a less restrictive zone. This only works for Internet Explorer 4.0 and later.

Use the following comment to insert a Mark of the Web comment into a page with an identified domain, replacing http://www.fabrikam.com with the URL of the Internet or intranet domain where the page is hosted.

<!--saved from url= <http://www.fabrikam.com> -->

Use the following comment when you need to insert a generic Mark of the Web.

<!--saved from <url=about:internet>-->

With Internet Explorer 6 and later, you can use the Mark of the Web comment with multipart HTML (.mht) files.

*Note: You host HTML application (.hta) files in a different process; therefore, they are not impacted by the Local Machine zone lockdown.*

The user can also configure the restrictions associated with the zones, through Internet Options on the Tools menu.

**Security Warning!** *This setting prevents content on a user's computer from elevating privilege. Code with elevated privilege can then run any code through an ActiveX control or read information with a script. It’s not recommended that this option be used, or if necessary only as a last resort.*

*Disabling the feature should only be used as a temporary measure during troubleshooting—to compare behavior of the application when the feature is enabled or not. It is not recommended that the feature be left disabled on an on-going basis.*

You may also control this feature through feature control keys.

You manage the Local Machine zone lockdown restrictions through a security feature control registry key (FEATURE\_LOCALMACHINE\_LOCKDOWN). Internet Explorer (Iexplore.exe) and Windows Explorer (Explorer.exe) run under this feature control by default. The following shows the registry keys and the enabled processes:

* HKEY\_LOCAL\_MACHINE (or HKEY\_CURRENT\_USER)\SOFTWARE\Microsoft\Internet Explorer\Main\FeatureControl\FEATURE\_LOCALMACHINE\_LOCKDOWN\iexplore.exe= 0x00000001
* HKEY\_LOCAL\_MACHINE (or HKEY\_CURRENT\_USER)\SOFTWARE\Microsoft\Internet Explorer\Main\FeatureControl\FEATURE\_LOCALMACHINE\_LOCKDOWN\explorer.exe= 0x00000001
* HKEY\_LOCAL\_MACHINE (or HKEY\_CURRENT\_USER)\SOFTWARE\Microsoft\Internet Explorer\Main\FeatureControl\FEATURE\_LOCALMACHINE\_LOCKDOWN\process name.exe=0x00000001

Applications that host the WebBrowser control can also take advantage of the security feature control, by adding their process to the same registry locations. You can do this programmatically, by using the CoInternetSetFeatureEnabled function.

*Note: If an application does not run under this security feature control, the WebBrowser control behaves the same as previous versions of Internet Explorer.*

## 1031 – Centralized URL Parsing

### Logged Message

Centralized URL Parsing (CURL) assists in the prevention of malformed URLs fooling Windows Internet Explorer 8 and Internet Explorer 7 by parsing the URL and verifying that it conforms to the RFC guidelines. When URL construction fails, it is typically because the URL does not conform to RFC guidelines. Internet Explorer 8 and Internet Explorer 7 log an error when URL construction fails, and a warning when a parsed URL results in a valid URL that is different from the URL parsed by Internet Explorer 6. In this specific situation, Internet Explorer 8 and Internet Explorer 7 does not block the Web page; however, the page may behave contrary to expectations.

### What is it?

When a URL is entered into the address bar it is parsed to see if it adheres to RFC guidelines.

### When is this event logged?

This event is logged when URL construction fails.

### Example

Perform the following steps to see this event logged in the compatibility tool:

1. Create a Web page with the following content. For this example call it 1031.html. The file can be placed anywhere. For this example, the file is located on the desktop.

<html xmlns="http://www.w3.org/1999/xhtml" >

<head>

<title> Centralized URL Parsing</title>

</head>

<body>

<p>Link to URL with USERINFO: <a href="http://foo:bar@foobar1/">http://foo:bar@foobar1/</a></p>

</body>

</html>

1. Open a browser and navigate to the Web page. For example:

C:\Users\JohnS\Desktop\1031.html

1. Click on the hyperlink.

This causes the event to log to the compatibility tool. Internet Explorer will also not allow the link to be clicked since the URL does not adhere to RFC guidelines and security best practices.

### Remediation

 There is no workaround. Domain names are required to conform to the RFC guidelines.

## 1032 – Internationalized Domain Names (IDN) Support

### Logged Message

Windows Internet Explorer 8 and Internet Explorer 7 permit navigation to Internationalized Domain Names (IDN). IDN relies upon a standardized mechanism known as "punycode" for encoding Unicode domain names, using only the ASCII characters permitted by the global DNS system. Internet Explorer helps mitigate possible spoofing attacks posed by Unicode look-alike characters by rendering some existing IDN names in the encoded format. This change has implications for URL compatibility in previous versions, since earlier versions of Internet Explorer encoded domain names in different formats (such as ANSI and UTF-8).

### What is it?

An **internationalized domain name** (**IDN**) is [Internet](http://en.wikipedia.org/wiki/Internet) [domain name](http://en.wikipedia.org/wiki/Domain_name) that contains one or more non-[ASCII](http://en.wikipedia.org/wiki/ASCII) characters. IDN uses "punycode" which is a standardized mechanism for encoding Unicode domain names, using only the ASCII characters permitted by the global DNS system.

For further details on IDN’s please refer to [this Wikipedia article](http://en.wikipedia.org/wiki/Internationalized_domain_name).

### When is this event logged?

This event is logged when Internet Explorer encounters an Internationalized Domain Name.

### Example

Perform the following steps to see this event logged in the compatibility tool:

1. Open Internet Explorer and enter the URL:

http://www.café.com?

The URL will be rendered in punycode format and the event will be logged.

### Remediation

End users of Internet Explorer might experience the compatibility impact of IDN support.

**Symptom**: When browsing the Internet, an encoded Web address that starts with xn-- and an Information bar that states:

*This Web address contains letters or symbols that cannot be displayed with the current language settings.*

**Cause**: Internet Explorer shows an encoded Web address to avoid possible spoofing if the domain name contains characters not used by the user's list of preferred content languages.

Internet Explorer also performs the encoding to avoid look-alike attacks, where a user is redirected to a malicious site, for example, www.litwareinc.com versus www.1itwareinc.com where the second URL replaces the letter L with the number 1. With IDN, the character set expands from a few dozen characters to many thousands of characters from all of the world's languages, thereby increasing the risk of spoofing attack.

**Workaround**: The user can add languages to the configured language set, by clicking the Information bar or by clicking Tools, clicking Internet Options, and then clicking the Languages button on the General tab.

*Note: The order of configured languages is important, so the user must ensure that the preferred content language appears first on the list.*

#### IDN Compatibility Impact for Network Administrators

Network Administrators can disable the Information bar by setting the HKLM or HKCU \Software\Policies\Microsoft\Windows\CurrentVersion\Internet Settings\DisableIDNPrompt DWORD registry key to 1.

**Symptom**: When navigating to an intranet Web site where the domain name contains Unicode characters, the browser might fail to find the site and instead show an HTTP/404 error page.

**Cause**: Previous versions of Internet Explorer used a different URL format when representing intranet domain names that contained non-ASCII characters. Some environments might not yet support the new IDN Punycode standard, so some sites might not be reachable using the IDN Punycode address format.

**Workaround**: The network administrator can revert to the way Internet Explorer 6 handled the Unicode domain names, by using the Internet Control Panel.

To change the Unicode domain name handling:

1. On the Tools menu, click Internet Options, and then click the Advanced tab.
2. Scroll down to the International section, and then clear the check box for **Send IDN server names**

-OR-

Open the Registry Editor and change the HKLM or HKCU \Software\Microsoft\Windows\CurrentVersion\Internet Settings\EnablePunycode DWORD value, based on the following:

0: Punycode is never used.

1: Punycode is used when talking directly to origin servers.

2: Punycode is used when talking to a proxy server.

3 (default): Punycode is used when talking to both origin and proxy servers.

#### IDN Compatibility Impact for Web Site Developers

Web site developers might experience the compatibility impact of IDN support in the following ways:

**Symptom**: Internet Explorer shows an encoded Web address that starts with xn-- and an Information bar that states "This Web address contains letters or symbols that cannot be displayed with the current language settings" appears.

**Cause**: Internet Explorer shows an encoded Web address to avoid possible spoofing if the domain name contains characters not used by the user's list of preferred content languages.

Internet Explorer also performs the encoding to avoid look-alike attacks, where a user is redirected to a malicious site, for example, www.litwareinc.com versus www.1itwareinc.com where the second URL replaces the letter L with the number 1. With IDN, the character set expands from a few dozen characters to many thousands of characters from all of the world's languages, thereby increasing the risk of spoofing attack.

#### Internet Explorer and Allowable IDN Addresses

Internet Explorer includes a number of restrictions on allowable IDN addresses in order to protect the user from spoofing attacks including:

* A domain name is displayed in encoded form if any of the following are true:
  + The domain name contains characters outside of the user's chosen languages, except for ASCII-only labels, which are always permitted for compatibility with existing sites.
  + The domain name contains characters that are not part of any language.
  + Any one of the labels contains a mix of scripts that do not appear together within a single language. For instance, Greek characters cannot mix with Cyrillic in a single label.

**Workaround**: Ensure that your domain name does not contain characters from multiple languages within a single label. If you need to use multiple languages, use one label per language.

*Note: A label is a segment of a domain name, delimited by periods (.). For example, www.microsoft.com contains three labels, "www", "microsoft", and "com".*

Ensure that you write the domain name, by using characters from the language that your destination audience is most likely to have configured in their browser. Many Web site owners choose to use IDN domain names as redirects to ASCII-based domain names. ASCII URLs are reachable in all browser versions, and such domain names do not show an Information bar in any locale.

**Symptom**: If you use scripting to retrieve the URL properties for DHTML objects, the URL can be unexpectedly returned as Unicode.

**Cause**: URL properties might be converted into Unicode form when assigned to object model properties. This is particularly relevant if your code attempts to perform comparisons between URL strings and strings elsewhere in the script code.

**Workaround**: Ensure that you write robust URL handling routines to handle the differences in domain name encoding. In particular, ensure that you use Unicode rather than Punycode when comparing JScript strings to URLs. Make sure that you test your scripts with all of the expected combinations of browser languages to ensure that you did not write your URL handling code based on incorrect assumptions.

#### IDN Compatibility Impact for Application Developers

Application developers might experience the compatibility impact of IDN support in the following ways:

**Symptom**: If you develop an application that uses WININET to connect to a Web site where the domain name contains Unicode characters, the networking component might fail to connect to the site.

**Cause**: Previous versions of WININET represented intranet domain names with non-ASCII characters in a different URL format. Some environments might not yet support the new IDN Punycode standard, so some sites might not be reachable, by using the IDN Punycode address format.

**Workaround**: To work around this issue, you must configure IDN using the InternetSetOption function before making an HTTP request. Additionally, you must set the dwIDNSettings registry key based on the following:

0: Punycode is never used.

1: Punycode is used when talking directly to origin servers.

2: Punycode is used when talking to a proxy server.

3 (default): Punycode is used when talking to both origin and proxy servers.

**Symptom**: If you use code to retrieve the URL properties for DHTML objects, the URL can be unexpectedly returned in Punycode or Unicode, depending on the user's settings.

**Cause**: URL properties might be converted into Punycode or Unicode form when assigned to object model properties. This is particularly relevant if your code attempts to perform comparisons between URL strings and strings elsewhere in the code.

**Workaround**: Ensure that you write robust URL handling routines to handle the differences in domain name encoding. In particular, ensure that you use Unicode rather than Punycode when comparing strings to URLs. Make sure that you test your code with all of the expected combinations of browser languages to ensure that you did not write your URL handling code based on incorrect assumptions. When writing native code applications, use the IURI API to parse the URL components.

#### IDN Support Advantages

The following sections discuss how the end user, the network administrator, the Web site developer, and the application developer can take advantage of IDN support.

##### End User:

End users can take advantage of IDN support in Internet Explorer because it now permits navigation to IDN sites written with Unicode characters from all of the world's languages.

##### Network Administrator:

* If your users do not need to navigate to IDN sites, you can force all IDN Web addresses to appear in encoded form by setting the HKLM or HKCU \Software\Policies\Microsoft\Windows\CurrentVersion\Internet Settings\ShowPunycode DWORD registry key to 1.

-OR-

* Manually set the Always show encoded addresses option located in the Advanced tab of the Internet Control Panel on each user's computer. When you select this option, all IDN Web addresses appear in encoded form.

##### Web Site Developer:

Formatting your IDN names in Punycode ensures that users of Internet Explorer prior to Internet Explorer 7 can navigate to your sites. In Internet Explorer 8, your links will appear in Unicode if your user's configured display language permits the characters

##### Application Developer:

In addition to Internet Explorer support for IDN, you can take advantage of other IDN resources available on MSDN.

## 1033 – Secure Sockets Layer (SSL)

### Logged Message

HTTPS uses either the Secure Sockets Layer (SSL) or Transport Layer Security (TLS) protocols to secure Internet traffic and protect your computer from snooping or tampering by others on your network. In order to improve security, Windows Internet Explorer 8 and Internet Explorer 7 automatically block navigation to any HTTPS site with invalid or erroneous security certificates.

### What is it?

New protocol defaults reduce the likelihood of someone taking advantage of configuration or protocol weaknesses to intercept or to modify Web traffic transferred using the HTTPS protocol. New error pages provide a simplified user experience, which also helps to mitigate social engineering and phishing attacks.

As an end user, network administrator, or Web site developer using Internet Explorer, you might experience the compatibility impact of HTTPS Security Improvements in the following ways:

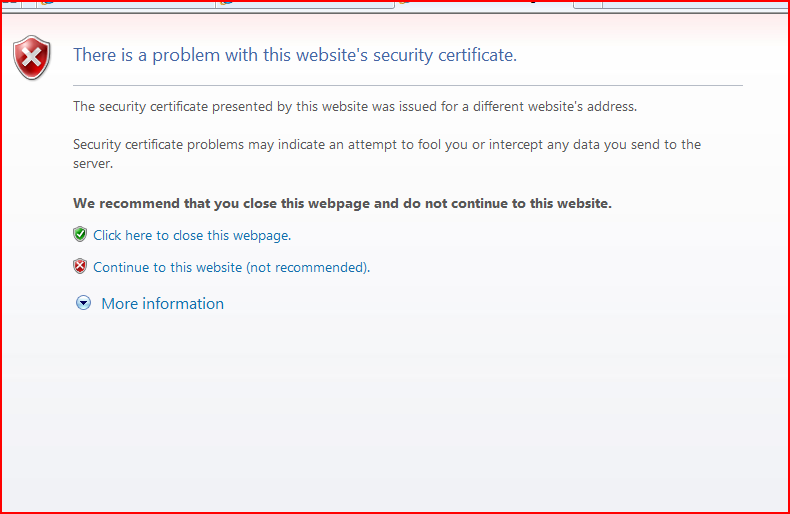
|  |  |
| --- | --- |
| **Symptom** | **Cause** |
| **An error page appears when viewing a site configured to use only the SSL 2.0 protocol.** | Internet Explorer 8 automatically disables the SSL 2.0 protocol. Due to known security issues with the SSL 2.0 protocol, it has been replaced by the SSL 3.0 and TLS 1.0 protocols. |
| **An error page appears when viewing an HTTPS site configured to use weaker ciphers (such as 40-bit and 56-bit encryption) on the Windows Vista® operating system.** | Windows Vista disabled the weaker encryption ciphers, only allowing the stronger ciphers to function properly. |
| **An error page appears when navigating to an HTTPS site with an erroneous security certificate** | Internet Explorer 8 automatically blocks navigation to any HTTPS site with invalid or erroneous security certificates. |
| **An Information bar appears when viewing a page that mixes HTTPS and HTTP content.** | Internet Explorer 8 automatically blocks HTTP content from appearing in HTTPS pages. |
| **An error appears when navigating to an HTTPS site with a revoked security certificate on Windows Vista.** | Windows Vista automatically performs a check for revoked security certificates on HTTPS sites. |

### When is this event logged?

This event is logged when Internet Explorer encounters invalid or erroneous security certificates.

### Example

Attempting to browse to a secure Web site with an invalid certificate prompts Internet Explorer to display the following warning:



### Remediation

The following sections describe possible workarounds for some of the most common Internet Explorer issues, as faced by end users, Network Administrators, and Web site developers.

#### Workarounds for End Users

As an end user of Internet Explorer, you can work around the compatibility impact of HTTPS Security Improvements in the following ways:

|  |  |
| --- | --- |
| **Symptom** | **Workaround** |
| **An error page appears when viewing an HTTPS site configured to use weaker ciphers (such as 40-bit and 56-bit encryption) on Windows Vista.** | There is no workaround for this issue. Please contact the Web site owner and request stronger encryption options. |
| **An error page appears when navigating to an HTTPS site with an erroneous security certificate.** | There are multiple issues when discussing erroneous security certificates and workarounds.   * Expired certificates. There is no workaround for an expired certificate. You must contact the Web site owner and request that they update the certificate. * Non-matching addresses. If the address in the security certificate does not match the Web site's address, you can clear the **Warn about certificate address mismatch** check box, located in the Advanced tab of the Internet Options dialog box and successfully navigate to the Web site.   *Important:*  *It is not recommended to change this setting.*   * Unsigned certificate. If a trusted certification authority did not sign the security certificate, you can manually add the authority.   **Security Warning!**  *Trusting a malicious certification authority puts your computer at risk.*  To manually add an authority   1. Click the Certificate Error button in the Internet Explorer address bar of the Certificate Error page. 2. Click View Details. 3. Select the root certificate in the Certification Path tab, and then click View Certificate. 4. Click Install Certificate in the General tab. |

#### Workarounds for Network Administrators

As a Network Administrator of computers running Internet Explorer 8, you can work around the compatibility impact of HTTPS Security Improvements in the following ways:

|  |  |
| --- | --- |
| **Symptom** | **Workaround** |
| **An error page appears when viewing an HTTPS site configured to use weaker ciphers (such as 40-bit and 56-bit encryption) on Windows Vista.** | You must configure your Web server software to offer stronger encryption options. If the Web server is not in your control, contact the server operator. |
| **An HTTPS error page appears, enabling users to continue on to a Web site that presented the erroneous certificate.** | Enable the Prevent ignoring certificate errors setting from your Group Policy. Enabling this option removes the ability to continue to a Web site from an HTTPS error page.  To enable the setting using Group Policy   1. Start the Group Policy tool (GPEdit.msc). 2. Expand the policy structure to Computer Configuration\Administrative Templates\Windows Components\Internet Explorer\Internet Control Panel. 3. Double-click the Prevent ignoring certificate errors setting. 4. Click Enabled, and then click OK. |

#### Workarounds for Web Site Developers

As a Web site developer for sites viewed with Internet Explorer 8, you can work around the compatibility impact of HTTPS Security Improvements in the following ways:

|  |  |
| --- | --- |
| **Symptom** | **Workaround** |
| **An error page appears when viewing a site configured to use only the SSL 2.0 protocol.** | Enable SSL 3.0 or later in your Web server software. |
| **An error page appears when viewing an HTTPS site configured to use weaker ciphers (such as 40-bit and 56-bit encryption) on Windows Vista.** | Enable strong ciphers (128-bit or higher) in your Web server software. |
| **An error page appears when navigating to an HTTPS site with an erroneous security certificate.** | There are multiple issues when discussing erroneous security certificates and workarounds.   * Expired certificates. Ensure that you are using valid, non-expired security certificates issued by a trusted root certification authority. * Non-matching addresses. Ensure that the address in the certificate matches the certificate for your Web site. This is particularly important for servers that are addressable by multiple hostnames. For example, a certificate issued to email.fabrikam.com is not valid for use on mailbox.fabrikam.com. You must either purchase a certificate that lists both hostnames, or purchase a wildcard (\*) certificate for \*.fabrikam.com. |
| **An Information bar appears when viewing a page that mixes HTTPS and HTTP content.** | Ensure that your HTTPS Web pages do not contain embedded references to resources addressed by the HTTP protocol.  *Note:*  *If you have a Web page that is viewable from either HTTP or HTTPS, make sure you use protocol-relative hyperlinks to address resources.*  *For example, if you have an image on www.fabrikam.com/account.htm that is addressable using either http:// or https:, you must use <img src="//www.fabrikam.com/pic.jpg"> instead of <img src="www.fabrikam.com/pic.jpg">*  *This way, if the user views the site using HTTPS the image is downloaded through HTTPS, but if the user views the Web site using HTTP, the image is downloaded through HTTP.* |

## 1034 – Cross-Domain Barrier and Script URL Mitigation

### Logged Message

Windows Internet Explorer 8 and Internet Explorer 7 block redirected navigation in DOM objects if there is a potential threat of cross-domain exploitation. Internet Explorer 8 and Internet Explorer 7 also block any script URL requiring cross-domain access that could be a potential threat.

### What is it?

Introduced in Windows Internet Explorer 7 and continuing in Internet Explorer 8 there is script URL mitigation for cross-domain exploits, in which a Web page from one domain can either read or manipulate the contents of a Web page located on a different domain. A script URL is a URL that starts with script protocols like JavaScript or VBScript (for example, javascript:doFormPost()). Since script URLs typically run scripts on an Internet Explorer window, they become easy targets for cross-domain exploits.

In earlier versions of Internet Explorer, script URLs were treated like any other URL, allowing navigation to occur. This navigation also allowed the script source to have access and run in the current window. In Internet Explorer 7 and Internet Explorer 8, script URL mitigation removes the script from the script URL, effectively stopping navigation from places that are considered unsafe. Internet Explorer 7 and Internet Explorer 8 then run the script in the domain context of the Web page from where it came instead of actually navigating to the page.

#### How This Issue Appears to the User

If for any reason the script URL navigation occurs from a questionable location or Internet Explorer is unable to retrieve the domain context, the navigation is blocked and the user receives a script error message in the Status bar. If the Web page relies on this blocked navigation, it might become partially unusable, so that when the user clicks a link or button, nothing happens.

#### What are the Possible Causes of This Issue?

The form element’s submit() or onclick handler provides a reference for two common functionalities blocked by Internet Explorer 7 and Internet Explorer 8 due to a JavaScript URL:

* Location.href
* Window.Open

### When is this event logged?

This event is logged when an attempt is made to run a script URL cross domain.

### Example

Perform the following steps to see this event logged in the compatibility tool:

1. Create a Web page with the following contents. For this example call it 1034.html.

<html>

<head><title>Test of Intranet and Restricted Script Links</title></head>

<body>

<iframe src="http://127.0.0.1/1034r.html">

</body>

</html>

1. Create another Web page with the following contents. For this example call it 1034r.html.

<html>

<head><title>Test of Restricted Script Link</title></head>

<body><a href="javascript:alert('I am from the restricted zone')">Restricted Alert Link</a></body>

</html>

1. Install the files in the root directory of the local web server. On an IIS server this means putting the files in this directory:

\Inetpub\wwwroot

1. In the browser go to Tools > Internet Options > Security > Restricted Sites and click the Sites button.
2. Add <http://127.0.0.1/> to the list and close the **Restricted sites** dialog.
3. Close the **Internet Options** dialog.
4. Browse to the file:

<http://localhost/1034.html>

1. Click on the **Restricted Alert Link** hyperlink.

The link attempts to execute a script URL in another domain. Internet Explorer prevents this. The user doesn’t see any indication in the browser window but the corresponding event is logged to the compatibility tool.

### Remediation

To enable the script URL to redirect the navigation to a Web site, you can change the way the script runs by using an event handler, an inline script, or provide a call to the execScript method.

#### Using an Event Handler

You can change the

<a href="javascript:alert('hi')"> Click</a>

script URL to include an event handler, such as

<a onclick="javascript:alert('hi')"> Click</a>.

#### Using an Inline Script

You can run the script URL code, by using an inline script tag. For example:

<script type="text/jscript">

doSomething()

</script>

#### Using the execScript Method

You can call the execScript method to run your script URL code. For example, you can change window.location = javascript:doSomething() to window.execScript("javascript:doSomething()").

#### How Can I Work Around This Issue?

You can work around the script URL mitigation security feature by setting registry named value.

**Security Warning!** *If you disable this security feature, you will be more prone to cross-domain attacks.*

*Disabling the feature should only be used as a temporary measure during troubleshooting—to compare behavior of the application when the feature is enabled or not. It is not recommended that the feature be left disabled on an on-going basis.*

Turn off the security URL mitigation by setting the following named value in the registry:

[HKEY\_CURRENT\_USER\Software\Microsoft\Internet Explorer\Main\FeatureControl\ FEATURE\_SCRIPTURL\_MITIGATION]"iexplore.exe"=dword:00000000

## 1035 – Anti-Phishing

### Logged Message

The Phishing Filter compares the addresses of Web sites that a user attempts to visit to a list of reported and confirmed phishing sites (fraudulent sites used to gather sensitive information from unsuspecting users). The Phishing Filter also analyzes sites for characteristics common to phishing sites and, if the user allows it, sends the addresses of visited sites to Microsoft to be verified against a frequently updated list of confirmed phishing sites. If you find that your Web site is flagged incorrectly as a phishing site, you can report the problem from the Tools menu.

### What is it?

Phishing is one of the fastest growing threats on the Internet, with the number of Internet sites having increased over 400 percent in the last year. Microsoft is committed to protecting Internet users worldwide from becoming victims of phishing scams by promoting valuable consumer education, industry collaboration, legislation, enforcement, and technology innovation.

Phishing (pronounced "fishing") is a way to trick computer users into revealing personal or financial information through an e-mail message or a Web site. A common phishing scam starts with an e-mail message that looks like an official notice from a trusted or reputable source, such as a bank, a credit card company, or an online merchant. In the e-mail message, recipients are directed to a fraudulent Web site where they are asked to provide personal information, such as an account number or password.

#### How the Microsoft Phishing Filter Works

The Phishing Filter works by comparing the addresses of Web sites that a user attempts to visit to the list of reported and confirmed phishing sites, stored on the end user's computer. The Phishing Filter also analyzes sites for characteristics common to phishing sites, and, if the user enables it, sends the addresses of visited sites to Microsoft to be verified against a frequently updated list of confirmed phishing sites.

### When is this event logged?

This event is logged when Internet Explorer detects a suspicious Web site.

### Example

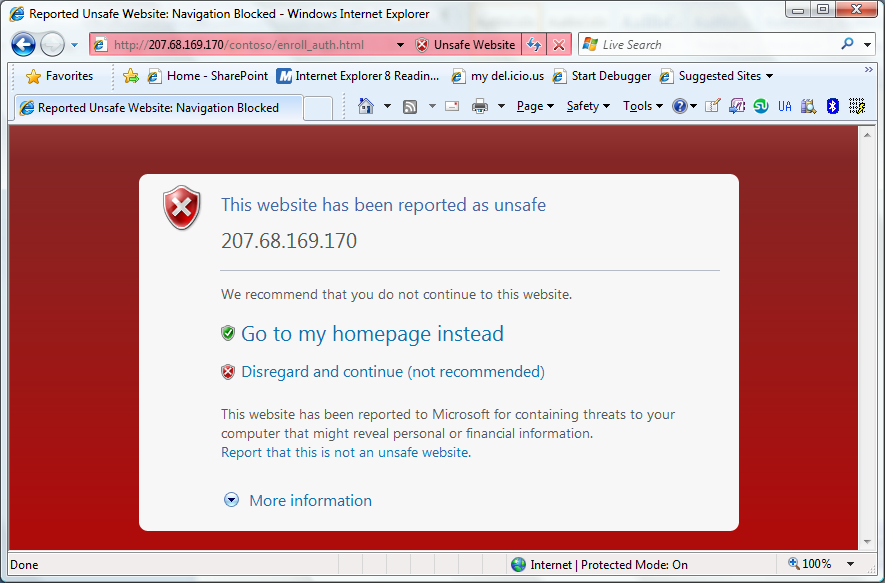
Perform the following steps to see this event logged in the compatibility tool:

1. Enter the following into the address bar:

<http://207.68.169.170/contoso/enroll_auth.html>

*Note: This is a test site for demonstrating the anti-phishing features of Internet Explorer. Do not enter any information into the page.*

*The address bar turns red and a warning message is displayed:*

**

### Remediation

The Phishing Filter works with Internet Explorer 8 to provide feedback on whether a Web site is a known or potential phishing threat.

* **Blocked Web Site** – Reputable sources have confirmed that the Web site is a known phishing Web site.
* **Suspicious Web Site** – Reputable sources have not yet confirmed that the Web site is fraudulent, but the site has characteristics that indicate it is a phishing site. The user must proceed with caution.

#### False URL Warnings and Dispute Submission

The following sections discuss the reasons for which the Phishing Filter might flag a URL, guidelines to prevent false URL warnings, and the process by which you can dispute a false warning.

##### Recommended Guidelines to Prevent False URL Warnings

Microsoft recommends the following best practices to help prevent Phishing Filter flagging.

* If you intend to ask users for personal information, you must use Secure Sockets Layer (SSL) certification.
* Verify that your Web site is secure from outside attacks by maintaining updated firewalls and installing all required security updates. Additionally, verify that your virus detection software is current and your server is virus-free.
* Verify that you are free from cross-site scripting attacks. For more information on this security issue, see the Cross-site scripting, Cross-site scripting questions and answers, and Prevent Cross-site Scripting articles.
* Verify the reliability of your external or third-party hosted content. Also verify that the content is secure and from a known or trusted source.
* Use a reputable domain name and avoid using an IP address. For example, URLs for sites hosted on the Microsoft domain must read "Microsoft.com" instead of "207.46.19.30".
* Verify that you and your users are running the most recent version of Internet Explorer 8.

#### Additional Resources

Refer to the following Web sites for more information about anti-phishing and the Microsoft Phishing Filter:

* [Anti-Phishing Technologies home page](http://go.microsoft.com/fwlink/?LinkId=69167)
* [Microsoft Phishing Filter Frequently Asked Questions (FAQ)](http://go.microsoft.com/fwlink/?LinkId=69168)
* [Anti-Phishing white paper](http://go.microsoft.com/fwlink/?LinkId=69169)

## 1036 – Manage Add-ons

### Logged Message

ActiveX Opt-in disables ActiveX controls on a user's computer in Windows Internet Explorer 8 and Internet Explorer 7. When a user encounters a Web page with a disabled control, the Information bar appears and the user can opt to activate the control.

### What is it?

The Internet Explorer Manage Add-ons dialog box displays information about all of the controls installed on the local computer, including the control name, the publisher, the status, the control type, and the file name. Any time one of these controls attempts to install on the local computer, a security user interface (UI) element called the Information bar appears. The user can click the Information bar to access the Manage Add-ons dialog box and to use the provided information to determine whether to enable the control.

#### Digitally Signing Your Control

A digital signature on your control enables your users to verify the control's publisher and that the control has not been tampered with since being published. Digital signatures help users make better trust decisions when determining if they should install a control, in addition to helping identify the control in the Manage Add-ons dialog box.

*Important: Users are strongly discouraged from installing unsigned controls. Installation is blocked by default on unsigned controls.*

### When is this event logged?

This event is logged when an ActiveX control that is disabled in the managed Add-ons feature is requested by a Web page.

### Example

Perform the following steps to see this event logged in the compatibility tool:

1. Go to a Web site that’s using an ActiveX control. For this example browse to:

<http://tv.msn.com>

1. Select a video and starting playing it.
2. In Internet Explorer go to Tools > Manage Add-ons.

The default filter is **Currently loaded add-ons**. You should see the following add-on in the list:

Shockwave Flash Object

1. Select the **Shockwave Flash Object** and click the **Disable** button. Close the Manage Add-ons dialog.
2. Refresh the Web page.

*The event is logged to the compatibility tool.*

1. Use the Manage Add-ons dialog to re-enable the add-on.
2. Refresh the Web page again.

*The add-on runs correctly.*

### Remediation

If your control is not signed then it is disabled by default. So the best way to avoid this problem is to sign your add-on. If a CAB file or another executable file installs your control, you must sign both the installation file (to enable installation) and the DLL or OCX file that contains the ActiveX control (to ensure your publisher name shows up without the Not verified notice).

The name shown in the Name column comes from the following registry setting:

HKEY\_CLASSES\_ROOT\CLSID\{Control CLSID}

AppName="contoso.exe"

(Default)=(REG\_SZ)"Control Name"

LocalizedStringPolicy=(REG\_SZ) Localized Control Name

If your control has a localized name, you can set LocalizedStringPolicy to point to the resource ID in your control's binary file with the translated name. The Manage Add-ons dialog box verifies this value first and uses it if it exists. If the localized string does not exist, the default non-localized value is used.

#### Related Articles

For more information, please see the following resources on the Microsoft Developer Network (MSDN) Web site.

* [Introduction to Code Signing](http://go.microsoft.com/fwlink/?LinkId=79808)
* [Signing and Checking Code with Authenticode](http://go.microsoft.com/fwlink/?LinkId=79809)

## 1037 – Protected Mode

### Logged Message

Windows Internet Explorer 8 and Internet Explorer 7 running on Windows Vista function at a lower integrity level to protect users from a variety of attacks. Protected Mode Internet Explorer restricts Web sites from writing to the registry and file systems on the local computer. Internet Explorer logs events when write access has been denied and when virtualization to a different location occurs.

### What is it?

Windows Vista Protected Mode reduces the severity of threats to both Internet Explorer and extensions running in Internet Explorer, by eliminating the ability to install malicious code silently through software vulnerabilities. Protected Mode takes advantage of the User Interface Privilege Isolation (UIPI) to block interaction from Internet Explorer with higher integrity applications system resources.

Protected Mode is an important step forward in security for Internet Explorer. It can help protect users from attack, by running an Internet Explorer process with greatly restricted privileges on the Windows Vista operating system. While Protected Mode does not protect against all forms of attack, it significantly reduces the ability of an attack to write, to alter, or to destroy data on the user's machine, or to install malicious code.

While most Internet Explorer security features will be available in Internet Explorer for Windows XP Service Pack 2, Protected Mode is only available on Windows Vista because it is based on security features new to Windows Vista.

* User Account Control (UAC) makes it easy to run without Administrator privileges. When users run programs with limited user privileges, they are safer from attack than when they run with Administrator privileges because Windows can restrict the malicious code from carrying out damaging actions.
* Integrity mechanism restrict write access to securable objects by lower integrity processes, much the same way that user account group membership restricts the rights of users to access sensitive system components.
* UIPI prevents processes from sending selected Windows messages and other USER APIs to processes running with higher integrity.

The Windows Vista security infrastructure enables Protected Mode to provide Internet Explorer with the privileges needed to browse the Web while withholding privileges needed to silently install programs or to modify sensitive system data.

#### Understanding Windows Vista's Integrity Mechanism

Windows Vista uses integrity-level labels for processes and other securable objects, an addition to the access control security mechanism of Windows. The integrity level defines which Internet-facing programs are at higher risk for exploits because they download untrustworthy content from unknown sources. Running these at-risk programs with more restricted permissions or at a lower integrity level than other programs reduces the ability of an exploit to modify the system or harm user data files.

Protected Mode uses the Windows Vista integrity mechanism to run the Internet Explorer process at Low integrity level. The main features of the integrity level mechanism in Windows Vista are as follows:

* Securable objects, like files and registry keys, have security descriptors that define the integrity level, or level of privilege required for write access to the object. This integrity level is defined with a new mandatory access control entry (ACE) in the system access control list (SACL) called a mandatory label. Objects without mandatory labels have an implied default integrity level of Medium.
* Processes have an integrity level defined in the security access token. In Protected Mode, Internet Explorer has a Low integrity level, applications started from the Start menu have a Medium integrity level, and applications that require Administrator permissions run with a High integrity level.
* Low integrity processes cannot gain write access to objects at a higher integrity level, even if the user's SID has write access in the discretionary access control list (DACL). Windows Vista performs the integrity-level checks before user access permission checks.

All files and registry keys on Windows Vista have a default integrity level of Medium. A Low integrity process, like Internet Explorer in Protected Mode, will receive access denied errors when it tries to modify existing files.

Some folders have a Low integrity mandatory label. A Low integrity level process can create and modify files in Low integrity folders. For example, the Temporary Internet Files folder contains a folder called Low, which is a Low integrity folder. Additionally, the Windows Vista integrity mechanism automatically assigns Low integrity mandatory labels to securable objects, files, or other objects created by Low integrity-level processes. By default, child processes started by a Low integrity process will also run with a Low integrity level.

The following table shows supported integrity access levels and the privileges they confer.

|  |  |
| --- | --- |
| **Integrity Access Level (IL)** | **System Privileges** |
| **High** | Administrative. The process can install files to the Program Files folder and write to sensitive registry areas such as HKEY\_LOCAL\_MACHINE. |
| **Medium** | User. The process can create and modify files in the user's Documents folder and write to user-specific areas of the registry, such as HKEY\_CURRENT\_USER. |
| **Low** | Not Trusted. The process can only write to Low integrity locations, such as the Temporary Internet Files\Low folder or the HKEY\_CURRENT\_USER\Software\LowRegistry registry key. |

*Note: For additional Protected Mode information,* [*see Understanding and Working in Protected Mode Internet Explorer*](http://msdn.microsoft.com/en-us/library/bb250462(VS.85).aspx) *on MSDN.*

### When is this event logged?

This event is logged when Internet Explorer or its extensions attempt to write to securable objects in Protected mode (and access is denied).

### Remediation

A user can configure Protected Mode, by using the Internet Explorer Internet Options dialog box.

To configure Protected Mode using the Internet Options dialog box:

1. From the Internet Options dialog box, click the Security tab.
2. Select a Web content zone, and then select or clear the **Enable Protected Mode** check box.
3. Verify that Internet Explorer is running in with Protected Mode on or off, by looking for Protected Mode: On/Off next to the Web content zone displayed in the Internet Explorer status bar.

You can also run Internet Explorer with administrator privileges in elevation mode to disable protected mode. To do this, right-click the Internet Explorer icon or shortcut, then select **Run as Administrator** from the context menu. Click **Allow** in the User Account Control dialog box. You will then see Protected Mode: Off at the status bar.

An administrator will be able to configure Protected Mode, by using Group Policy and the URLACTION\_LOWRIGHTS (0x00002500)URL Action registry key. For more information, see [URL Security Zones Overviews](http://msdn.microsoft.com/en-us/library/ms537187(VS.85).aspx).

#### Debugging Using the Internet Explorer Compatibility Test Tool

You can use the Internet Explorer Compatibility Test Tool to debug Protected Mode issues. When Internet Explorer or its extensions attempt to write to securable objects in Protected Mode, the ACT generates an entry in the log file describing the operation and its results. The following list explains the possible values in the log entries:

* ModuleName: Name of the file that started the process attempting to access securable objects.
* VirtualizationAction: Results of the write operations, including one of the following values:
  + InterceptedWrite: Indicates that the Compatibility Layer intercepted the operation.
  + WriteIgnored: Indicates that Protected Mode ignored the operation because the attempting process is an elevated broker.
  + CreateVirtualCopy: Indicates that the Compatibility Layer made a copy of the object in a virtual location.
  + CreateNew: Indicates that the Compatibility Layer created a new object in a virtual location.
  + ObjectType: Specifies whether the object is a File or Registry setting.
* APIName: Specifies the function attempting the operation, for example CreateFile or RegOpenKey.
* ReqObjectPath: Location of the object the operation attempted to modify. This is blank for objects that do not have paths.
* NewObjectPath: Specifies the object modified by the operation if the write operations succeed.
* APIResult: Specifies the result returned by the API function attempting the write operation.
* LastError: The last error received by an API function.

## 1040 – Cascading Style Sheet (CSS Fixes)

### Logged Message

Windows Internet Explorer 8 and Internet Explorer 7 contain a number of improvements to cascading style sheet (CSS) parsing and rendering over Internet Explorer 6. The goal of the improvements was to increase the consistency of how Internet Explorer interprets cascading style sheets as recommended by the W3C, so that developers can have a reliable set of functionality. Some Web pages may render differently with Internet Explorer 8 and Internet Explorer 7 than with Internet Explorer 6. You may see this when elements are moved to a different area of the page or in overlapping content when viewed in Internet Explorer 8 and Internet Explorer 7. These issues are common to content using particular CSS constructs (often known as filters) to work around bugs that existed under the strict mode in Internet Explorer 6.

### What is it?

Cascading Style Sheets (CSS) provides the ability for developers to separate the layout and styles of a Web page from the content. Internet Explorer 7 went a long way to improve the CSS rendering of Web Pages. Internet Explorer 8 continues this trend as is now the most CSS-compliant release yet. If a Web page is rendering differently in Internet Explorer 8 than in previous versions, it’s important to understand how CSS works so that you can properly diagnose the problem.

Some of the common causes of rendering issues are:

* User Agent Strings and Browser Detection
* XML Prolog and the Strict Switch Method
* Box Model Changes
* CSS Filters

CSS is a broad topic. Aside from how it’s logged, it’s beyond the scope of this document to explain the technology comprehensively. For more detail on CSS please visit the following resources:

* <http://msdn.microsoft.com/en-us/library/cc351024(VS.85).aspx>
* [http://technet.microsoft.com/en-us/library/cc766445.aspx](http://technet.microsoft.com/en-us/library/cc766445.aspx%20)
* <http://www.msdn.com/iecompat>
* <http://www.w3.org/Style/CSS/>
* <http://www.w3.org/TR/CSS21/>

### When is this event logged?

This event is logged when one of the 4 categories of CSS issues mentioned above are encountered.

### Example

Perform the following steps to see this event logged in the compatibility tool:

1. Create a Web page with the following contents. For this example call it 1040.html. The file can be placed anywhere. But for this example, the file is located on the desktop

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">

<head>

<title>CSS Test</title>

<style type="text/css">

.myclass {

min-height: 300px;

\_height: 300px;

}

</style>

</head>

<body>

<div>

<p class="myclass">Test text</p>

</div>

</body>

</html>

1. Open a browser and navigate to the Web page. For example:

C:\Users\JohnS\Desktop\1040.html

The event is fired due to the \_height property in the .myclass style. The underscore filter was used to show properties exclusive to Internet Explorer. Internet Explorer 7 and later versions treat these constructs as a custom property, which means that it is still in the object model and can be queried through a script, but it does not natively apply its value

### Remediation

For help on working around these issues please refer to the section “My Web Site is Broken; What Can I Do?” in the following document:

<http://technet.microsoft.com/en-us/library/cc766445.aspx>

## 1041 – UIPI Extension Blocked

### Logged Message

User Interface Privilege Isolation (UIPI) Extension blocking prevents loading some Internet Explorer extensions that are known to be incompatible with Windows Vista, or have potential security issues. For example this blocking prevents the Microsoft PowerPoint handler from starting without explicit user interaction.

### What is it?

User Interface Privilege Isolation (UIPI) blocks lower-integrity processes from accessing higher-integrity processes. For example, a lower-integrity process cannot send window messages or hook or attach to higher priority processes. UIPI in this scenario blocks extensions that are incompatible with Windows Vista as well as extensions that could pose a security risk.

Extensions that could be blocked are:

* ActiveX Control
* Toolbar
* Browser Helper Object
* Browser Extension
* Mime Handler
* Pluggable Protocol

A list of known extensions are listed in the registry:

HKEY\_LOCAL\_MACHINE\SOFTWARE\Microsoft\Internet Explorer\Extension Compatibility.

### When is this event logged?

This event is logged when an Internet Explorer Extension is encountered that is not Vista compatible or has potential security implications.

### Remediation

There is no workaround. You should be aware of this issue and design your site accordingly.

## 1042 – UIPI Cross Process Window Message

### Logged Message

User Interface Privilege Isolation (UIPI) prevents application processes running with lower privileges from using Windows messages to send information to a higher privilege process. For example, if you are running as a Limited User, Windows Internet Explorer 8 and Internet Explorer 7 prevent Web sites from sending messages to the Microsoft Management Console (MMC) or an Administrative Control Panel (CPL). Without this prevention, application processes can inject hostile information without requiring user interaction.

### What is it?

User Interface Privilege Isolation (UIPI) blocks lower-integrity from accessing higher-integrity processes. For example, a lower-integrity process cannot send window messages or hook or attach to higher priority processes. This helps protect against "shatter attacks." A shatter attack is when one process tries to elevate privileges by injecting code into another process using windows messages.

A shatter attack is a programming technique employed in malicious software that can be used to bypass security restrictions between processes in a session.

### When is this event logged?

This event is logged when an application process running with lower privileges attempts to use Windows messages to send information to a higher privilege process.

### Remediation

There is no workaround . The solution is to modify your extension so that it operates within the protected mode integrity checks. For guidance on writing extensions that are compatible with Internet Explorer protected mode in Vista, see the article [Understanding and Working in Protected Mode Internet Explorer](http://msdn.microsoft.com/en-us/library/bb250462.aspx) on MSDN.

## 1046 – Cross-Site Scripting Filter

### Logged Message

 The Cross-Site Scripting Filter event is logged when Internet Explorer 8 detects and mitigates a cross-site scripting (XSS) attack. Cross-site scripting attacks occur when one Web site, generally malicious, injects (adds) JavaScript to otherwise legitimate requests to another Web site. The original request is generally innocent, such as a link to another page or a Common Gateway Interface (CGI) script providing a common service (such as a guestbook). The injected script generally attempts to access privileged information or services that the second Web site does not intend to allow. The response or the request generally reflects results back to the malicious Web site. The XSS Filter, a feature new to Internet Explorer 8, detects JavaScript in URL and HTTP POST requests. If JavaScript is detected, the XSS Filter searches evidence of reflection, information that would be returned to the attacking Web site if the attacking request were submitted unchanged. If reflection is detected, the XSS Filter sanitizes the original request so that the additional JavaScript cannot be executed. The XSS Filter then logs that action as a Cross-Site Script Filter event.

### What is it?

Internet Explorer 8 contains a new feature to detect reflected cross-site scripting (XSS) vulnerabilities. XSS vulnerabilities enable an attacker to control the relationship between a user and a Web site or web application that they trust. Cross-site scripting can enable attacks such as:

* Cookie theft, including the theft of sessions cookies that can lead to account hijacking
* Monitoring keystrokes input to the victim Web site / application
* Performing actions on the victim Web site on behalf of the victim user. For example, an XSS attack on Windows Live Mail might enable an attacker to read and forward e-mail messages, set new calendar appointments, etc.

The XSS Filter operates as an Internet Explorer 8 component with visibility into all requests / responses flowing through the browser. When the filter discovers likely XSS in a cross-site request, it identifies and neuters the attack if it is replayed in the server’s response.

With the new XSS Filter, Internet Explorer 8 users encountering a Type-1 XSS attack will see a notification like the following:



The page has been modified and the XSS attack is blocked. Users are not presented with a question about what they would like to do in this case (a question most users would be unable to answer). Internet Explorer simply blocks the malicious script from executing.

In this case the XSS Filter has identified a cross-site scripting attack in the URL. It has neutered this attack as the identified script was replayed back into the response page. In this way the filter is effective without modifying an initial request to the server or blocking an entire response.

### When is this event logged?

This event is logged when Internet Explorer detects a cross-site scripting attack and the user is notified as explained above.

### Example

Perform the following steps to see this event logged in the compatibility tool:

For this example you will need Microsoft Visual Web Developer Express (VWDE). This is available free from <http://www.microsoft.com/express/vwd/>

1. Create a new ASP.Net c# Web site called XSS in VWDE.

1. Open the file default.aspx within VWDE and replace the content of with the following:

<%@ Page Language="C#" AutoEventWireup="true" ValidateRequest="false" CodeFile="Default.aspx.cs" Inherits="\_Default" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">

<head runat="server">

<title>XSS Example</title>

</head>

<body>

<form id="form1" runat="server">

<div>

<a href=\" <asp:Literal ID="sid" runat="server" />#"></a>

</div>

</form>

</body>

</html>

1. Now open Default.aspx.cs and replace its contents with the following:

using System;

public partial class \_Default : System.Web.UI.Page

{

protected void Page\_Load(object sender, EventArgs e)

{

this.sid.Text = Request["SID"];

}

}

1. Press F5 to compile and run the site.

*Take note of the port number the site is running on (the number after the “localhost:” portion of the URL)*

1. Create a Web page with the following contents. For this example call it 1046.html.

<html xmlns="http://www.w3.org/1999/xhtml">

<head>

<title>XSS Launch Page</title>

</head>

<body>

<div>

<a href="http://127.0.0.1:<SITE PORT NUMBER>/XSS/Default.aspx?SID=%22%3E%3C%73%63%72%69%70%74%20%66%6F%72%3D%77%69%6E%64%6F%77%20%65%76%65%6E%74%3D%6F%6E%6C%6F%61%64%3E%3C%2F%73%63%72%69%70%74%3E%3C%2F%61%3E%3C%61%20%68%72%65%66%3D%22">Click</a>

</div>

</body>

</html>

1. Edit the above file to replace <SITE PORT NUMBER> with the port number you noted earlier.

*Note: The ASCII chars in the hyperlink href above are the encoded version of the following text:*

*"><script for=window event=onload></script></a><a href="*

*This is encoded JScript and is what causes the XSS warning event.*

1. Install the 1046.html file in the root directory of the local web server. On an IIS server this means putting the file in this directory:

\Inetpub\wwwroot

1. Browse to the Web page:

<http://localhost/1046.html>

1. Click the hyperlink on the page.

Internet Explorer recognizes a possible cross-site scripting attack. It logs the event and displays an appropriate message to the user.

### Remediation

This feature can be disabled through an HTTP header.

**Security Warning!** *The reason for disabling this feature is to support sites that depend on the reflection behavior that Internet Explorer is looking for. However, this should only be used as a temporary measure while the site is redesigned.*

*Disabling the feature should only be used as a temporary measure during troubleshooting—to compare behavior of the application when the feature is enabled or not. It is not recommended that the feature be left disabled on an on-going basis.*

You can disable this feature by setting the HTTP response header:

X-XSS-Protection: 0

The end user can also control the XSS Filter (which is ON by default in the Restricted, Internet and Trusted Zones) using the Internet Control panel.

## 1047 – Intranet at Medium Integrity Level

### Logged Message

Windows Internet Explorer 8 helps protect users from attack by running an Internet Explorer process with greatly restricted privileges on Windows Vista. In Internet Explorer 8, browsing intranet Web sites occurs at the medium integrity level. At this level, processes have user level system privileges and can write to user-specific areas of the registry. In Internet Explorer 7, browsing intranet Web sites operates with untrusted system privileges and writes only to specific low-integrity locations.

### What is it?

While most Internet Explorer 8 security features will be available in Internet Explorer 8 for Windows XP Service Pack 2, Protected Mode is only available on Windows Vista because it is based on security features new to Windows Vista.

* [User Account Control](http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnlong/html/AccProtVista.asp) (UAC) makes it easy to run without Administrator privileges. When users run programs with limited user privileges, they are safer from attack than when they run with Administrator privileges because Windows can restrict the malicious code from carrying out damaging actions.
* Integrity mechanism restrict write access to [securable objects](http://msdn.microsoft.com/library/default.asp?url=/library/en-us/secauthz/security/securable_objects.asp) by lower integrity processes, much the same way that user account group membership restricts the rights of users to access sensitive system components.
* [User Interface Privilege Isolation](http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnlong/html/AccProtVista.asp) (UIPI) prevents processes from sending selected window messages and other USER APIs to processes running with higher integrity.

The Windows Vista security infrastructure allows Protected Mode to provide Internet Explorer with the privileges needed to browse the Web while withholding privileges needed to silently install programs or modify sensitive system data.

#### Understanding Windows Vista's Integrity Mechanism

Windows Vista includes an addition to the access control security mechanism of Windows that labels processes and other securable objects with an integrity level. Internet-facing programs are at higher risk for exploits than other programs because they download untrustworthy content from unknown sources. Running these programs with fewer permissions, or at a lower integrity level, than other programs reduces the ability of an exploit to modify the system or harm user data files.

Protected Mode uses the Windows Vista integrity mechanism to run the Internet Explorer process at low integrity. The main features of the integrity level mechanism in Windows Vista are as follows:

* [Securable objects](http://msdn.microsoft.com/library/default.asp?url=/library/en-us/secauthz/security/securable_objects.asp), like files and registry keys, have security descriptors that define the integrity level, or level of privilege required for write access to the object. This integrity level is defined with a new mandatory access control entry (ACE) in the System access control list (SACL). The new mandatory ACE is called a mandatory label. Objects without mandatory labels have an implied default integrity level of Medium.
* Processes have an integrity level defined in the security access token. In Protected Mode, Internet Explorer has a low integrity level. Applications run from the Start menu have a medium integrity level. Applications that require administrator permissions run with a high integrity level.
* Low integrity processes cannot gain write access to objects at a higher integrity levels, even if the user's SID is granted write access in the discretionary access control list (DACL). Integrity level checks are performed before user access permission checks.

All files and registry keys on Windows Vista have a default integrity level of Medium. A Low integrity process, like Internet Explorer in Protected Mode, will receive access denied errors when it tries to modify existing files.

Some folders have a low integrity mandatory label. A low integrity process, such as Internet Explorer in Protected Mode, can create and modify files in low integrity folders. For example, the temporary Internet files folder contains a folder called Low, which is a low integrity folder. The Windows Vista integrity mechanism automatically assigns low integrity mandatory labels to securable objects created by low integrity processes. As a result, all files and other objects created by Internet Explorer in Protected Mode or any other low integrity process are automatically assigned low integrity mandatory labels. By default, child processes started by a low integrity process will also run with a low integrity level. Protected mode allows processes to be created with higher integrity. For details, see [Starting Processes from Protected Mode](http://msdn.microsoft.com/en-us/library/bb250462.aspx#wpm_elebp) section.

The following table shows supported integrity access levels and the privileges they confer.

|  |  |
| --- | --- |
| **Integrity Access Level (IL)** | **System Privileges** |
| **High** | Administrative (Process can install files to the Program Files folder and write to sensitive registry areas like HKEY\_LOCAL\_MACHINE.) |
| **Medium** | User (Process can create and modify files in the user's Documents folder and write to user-specific areas of the registry, such as HKEY\_CURRENT\_USER.) |
| **Low** | Untrusted (Process can only write to low integrity locations, such as the Temporary Internet Files\Low folder or the HKEY\_CURRENT\_USER\Software\LowRegistry key) |

#### Changes in Internet Explorer 8

 In Internet Explorer 7 browsing a site in the Intranet or Internet zone occurs with protected mode on. This means that browsing the low integrity level is used.

This behavior changes in Internet Explorer 8. Internet pages continue to run with protected mode on—meaning a low integrity level. However, Intranet pages run with protected mode off—and a medium integrity level. This facilitates a level of security that you may want for internal sites—such as accessing local files belonging to the user. However, when the user clicks a link to an internet site from an intranet site, the page is opened in a new window running under a different, low integrity security context. This ensures that internet sites have little chance of causing harm, even if an attacker was able to take control of the browser.

### When is this event logged?

 This is logged when the user clicks a link on an intranet page that leads to an internet page.

### Example

Perform the following steps to see this event logged in the compatibility tool:

1. Create a Web page with the following contents. For this example call it 1047.html.

<html xmlns="http://www.w3.org/1999/xhtml" >

<body>

<div>

<a href="http://www.microsoft.com">Microsoft.com</a>

</div>

</body>

</html>

1. Install the file in the root directory of the local web server. On an IIS server this means putting the file in this directory:

\Inetpub\wwwroot

1. Browse to the file:

<http://localhost/1047.html>

*The file is opened with protected mode off—medium integrity level.*

1. Click on the “Microsoft.com” link.

Internet Explorer opens the Microsoft home page in a new window, with protected mode on—at low integrity level. The event is logged to the compatibility tool.

### Remediation

This feature can be disabled by modifying the registry.

**Security Warning!** *Disabling the feature should only be used as a temporary measure during troubleshooting—to compare behavior of the application when the feature is enabled or not. It is not recommended that the feature be left disabled on an on-going basis.*

You can disable this feature by setting the following named values in the registry:

* HKCU\Software\Microsoft\Internet Explorer\Main \PPT (dword)
* HKCU\Software\Microsoft\Internet Explorer\Main\TabProcGrowth (dword)

PPT (Process Per Tab) allows you to turn Low Rights Internet Explorer (LoRIE)(/Protected Mode off. Set PPT=0 to turn LoRIE off but note, by default, you will run with tab processes at medium integrity. If you additionally set TabProcGrowth=0, you will run Internet Explorer at medium-MIC (mandatory integrity control) in a single process (both frames and tabs in the same process.

## 1048 – DEP/NX Crash Recovery

### Logged Message

 Data Execution Prevention/No Execute (DEP/NX) option in Windows Internet Explorer 8 prevents code from running in non-executable memory. When a violation occurs, the browser stops responding instead of running malicious code. When Internet Explorer 8 has recovered from a crash caused by DEP/NX, this event is logged. Typically, DEP/NX failures occur due to attempts to exploit the browser or its add-ons. But it is possible that a browser add-on is not compatible with DEP/NX, and failures occur even without malicious content. If an incompatible add-on is found, contact the add-on developer for an updated version.

### What is it?

Internet Explorer 7 on Windows Vista introduced an off-by-default Internet Control Panel option to “Enable memory protection to help mitigate online attacks.”  This option is also referred to as Data Execution Prevention ([DEP](http://msdn2.microsoft.com/en-us/library/aa366553.aspx)) or No-Execute (NX). This option is enabled by default for Internet Explorer 8 on Windows Server 2008 and Windows Vista SP1 and later.

DEP/NX helps to foil attacks by preventing code from running in memory that is marked non-executable. DEP/NX, combined with other technologies like Address Space Layout Randomization ([ASLR](http://blogs.msdn.com/michael_howard/archive/2006/05/26/address-space-layout-randomization-in-windows-vista.aspx)), make it harder for attackers to exploit certain types of memory-related vulnerabilities like buffer overruns. Best of all, the protection applies to both Internet Explorer and the add-ons it loads. No additional user interaction is required to provide this protection, and no new prompts are introduced.

If a process running a particular Web page crashes, Internet Explorer automatically captures information about the crash and then recovers the process (the crash recovery portion of this is part of the new Loosely-Coupled Internet Explorer architecture). For example, you might have a page running an ActiveX control that performs an illegal memory access (whether accidental or malicious).

When a crash occurs, Internet Explorer automatically logs:

* An exception code
* Exception address
* Exception flags

See the Platform SDK for more information about the possible exception codes and exception flags:

<http://msdn.microsoft.com/en-us/library/aa363082.aspx>

### When is this event logged?

 This event is logged when a process running a page crashes.

### Example

An example is a page that contains an ActiveX control that performs a divide-by-zero operation. In this case, the process will be recovered but DEP/NX will also log details about the crash. It’s not possible to provide a complete example in this document. But here are the basic steps you would need to complete to create a sample that logs this event in the compatibility tool.

1. Create an ActiveX control that exposes a callable function called DepTest and performs a divide-by-zero operation.
2. Create a Web page with the following contents. For this example call it 1048.html:

<html xmlns="http://www.w3.org/1999/xhtml" >

<head runat="server">

<title>DEP/NX Test</title>

<script type="text/jscript">

function CreateCrash()

{

objCC.DepTest();

}

</script>

</head>

<body>

<object id="objCC" codeBase="<YourActiveX.Cab>"

classid="<YOUR CLSID>" width="32" height="32"></object>

<input id="Button1" onclick="CreateCrash()"

value="DEP/NX Crash Recovery" type=button />

</body>

</html>

1. Edit the name and CLSID of your ActiveX control in the <object> tag to match those of the ActiveX control that you wrote.
2. Press the “DEP/NX Crash Recovery” button.

*This invokes the JScript CreateCrash function which calls into your ActiveX control method DepTest. This method performs a divide-by-zero operation that causes the browser tab to crash. The event is logged when the crash occurs.*

### Remediation

No workaround is needed for this feature. The information captured in the log can be used to isolate the cause of the crash.

## 1049 – Standards Mode

### Logged Message

To adhere to the latest Web standards, Windows Internet Explorer 8 includes an update to the Standards Mode engine. The Standards Mode engine is enabled when a Standards Mode DOCTYPE declaration is found in the HTML source of a Web page. In Standards Mode, many components that affect the layout and rendering of Web pages have been changed, including HTML and CSS interpretation, document object model APIs, JScript performance and security, and accessibility APIs.

### What is it?

Each major release of Internet Explorer adds features designed to make the browser easier to use, to increase security, and to more closely support industry standards. As Internet Explorer gains features, there is a risk that older Web sites may not display correctly.

To minimize this risk, Internet Explorer 8 supports a number of document compatibility modes that enable different features and can affect the way content is displayed.

* IE5 mode renders content as if it were displayed by Internet Explorer 7's quirks mode, which is very similar to the way content was displayed in Internet Explorer 5.
* IE7 mode renders content as if it were displayed by Internet Explorer 7's standards mode, whether or not the page contains a <!DOCTYPE> directive.
* Emulate IE7 mode tells Internet Explorer to behave as IE7 does without any additional hints as to what the compatibility mode should be. In this case the browser uses the <!DOCTYPE> directive of the Web page to determine how to render content. Standards mode directives are displayed in Internet Explorer 7 standards mode and quirks mode directives are displayed in IE5 mode. Unlike IE7 mode, Emulate IE7 mode respects the <!DOCTYPE> directive.
* IE8 mode provides the highest support available for industry standards, including the [W3C Cascading Style Sheets Level 2.1 Specification](http://www.w3.org/TR/CSS21/) World Wide Web link and the [W3C Selectors API](http://www.w3.org/tr/selectors-api) World Wide Web link, and limited support for the [W3C Cascading Style Sheets Level 3 Specification (Working Draft)](http://www.w3.org/TR/css3-roadmap/) World Wide Web link.
* Emulate IE8 mode tells Internet Explorer to use the <!DOCTYPE> directive of the Web page to determine how to render content. Standards mode directives are displayed in IE8 standards mode and quirks mode directives are displayed in quirks mode.
* Edge mode tells Internet Explorer to display content in the highest mode available. With Internet Explorer 8, this is equivalent to IE8 mode. If a (hypothetical) future release of Internet Explorer supported a higher compatibility mode, pages set to edge mode would appear in the highest mode supported by that version. Those same pages would still appear in IE8 mode when viewed with Internet Explorer 8.

Internet Explorer 8 renders Internet sites in IE8 standards mode by default (Intranet sites are rendered in IE7 emulation mode by default). The fact that Internet pages default to standards mode, may cause problems for pages that don’t explicitly opt-in to that mode. A page explicitly opts into standards mode using one of the following techniques:

* Including a metatag in the page that specifies IE=8 or IE=EmulateIE8
* Configuring the web server to send a compatibility header with the page that specifies IE=8 or IE=EmulateIE8

In cases where an Internet page hasn’t explicitly opted in to IE8 standards mode, it’s prudent to warn the user that their page may have rendering problems. That’s what this event does.

### When is this event logged?

This event is logged when a page is rendered in standards mode where the page did NOT explicitly opt in to standards mode (i.e., there’s not a metatag that specifies an IE8 emulation mode).

If a page explicitly opts-in to standards mode (e.g., it contains a metatag that indicates IE8 standards mode), then the event is NOT logged.

### Example

Perform the following steps to see this event logged in the compatibility tool:

1. Create a Web page with the following contents. For this example call it 1049.html.

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" >

<body>

<div>

<h1>Standards Mode Example</h1>

</div>

</body>

</html>

1. Install the file in the root directory of the local web server. On an IIS server this means putting the file in this directory:

\Inetpub\wwwroot

1. Browse to the file:

<http://127.0.0.1/1049.html>

The page will render in the Internet Zone and in Internet Explorer 8 mode. Since there is no explicit Internet Explorer 8 meta tag or HTTP header this event will be logged.

### Remediation

Put a compatibility metatag in the page or add a HTTP Header at site level to explicitly indicate that IE8 standards mode should be used to render the page.

For example, in an ASP.NET application the following web.config file could be used to set the compatibility header for the corresponding application:

<system.webServer>

<httpProtocol>

<customHeaders>

<clear />

<add name="X-UA-Compatible" value="IE=IE8" />

</customHeaders>

</httpProtocol>

</system.webServer>

Or you may add the meta tag directory to the page:

<meta http-equiv="X-UA-Compatible" content="IE=IE8" />

*Note: The value of IE can be IE8 or EmulateIE8. Both are sufficient to opt-in to IE8 standards mode.*

## 1056 – File Name Restriction

### Logged Message

Windows Internet Explorer 8 form submission has been changed so that a FILE UPLOAD control (input type="’file’") only submits the file path to the server. Previously, the full path was sent to the server. Also, programmatic access to the .value property of the FILE UPLOAD control also removes the path information from the file name.

### What is it?

The file upload control contains a property that holds the path of the file being uploaded. This is retrievable via a property on the control and is included in any HTTP POST involving a form that includes the file upload control.

In Internet Explorer 8 an enhancement was made so that on Internet and Restricted sites, the path is not available (instead access to the path returns only the filename portion).

You can find full documentation on the INPUT type=file html control here:

<http://msdn.microsoft.com/en-us/library/ms535263(VS.85).aspx>

### When is this event logged?

 This event is logged when the file path is stripped away and replaced with only the file name.

### Example

For this example you will need Microsoft Visual Web Developer Express (VWDE) that is available free from <http://www.microsoft.com/express/vwd/>

1. Create a new ASP.Net C# Web site in VWDE. Call it FileNameTest.

1. Open the file default.aspx within VWDE and replace the content with the following:

<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Default.aspx.cs" Inherits="\_Default" %>

<html xmlns="http://www.w3.org/1999/xhtml" >

<head id="Head1" runat="server">

<title>FileUploadDemo</title>

<script type="text/jscript">

var enable = false;

function CheckFileName()

{

var file = document.getElementById("file");

alert(file.value);

}

</script>

</head>

<body>

<form id="form1" runat="server">

<div>

<input type="file" id="file" name="file" />

<input type="submit" value="submit" />

<a href="#" onclick="CheckFileName()">Check file name through

JScript</a>

</div>

</form>

</body>

</html>

1. Now open Default.aspx.cs and replace its contents with the following:

using System;

public partial class \_Default : System.Web.UI.Page

{

protected void Page\_Load(object sender, EventArgs e)

{

if (IsPostBack)

{

Response.Write(Request["file"]);

}

}

}

1. Press F5 to compile and run the site.
2. Once the default page is running in the browser change the server name in the URL from “localhost” to “127.0.0.1”.

*This forces the Web page to run in the Internet Zone.*

1. Click the browse button and open a file (it doesn’t matter which one).

*The full path of the file is displayed in the text box.*

1. Click the “Check file name through JScript” hyperlink

*An alert box appears that displays the* ***value*** *property of the file input control. Note that only the file name is displayed. The path has been stripped out.*

1. Click the **Browse** button and open another file.
2. Click the **submit** button .

This POSTs the form to the server. The server writes the value of the **file** property at the top of the page. Once again, only the file name is accessible. The path has been stripped out.

Note that both client side and server side actions will trigger the File Name Restriction event to be logged.

### Remediation

There is no workaround for this feature and it may not be turned off. Providing access to the full path of the file (for Internet and Restricted sites) is a security measure. Stripping out the full path in these instances prevents relatively uncontrolled sites from accessing information that can potentially be exploited.

## 1058 – Codepage Sniffing

### Logged Message

Windows Internet Explorer 8 prevents certain codepages from participating in its Codepage Sniffing heuristic. Any pages that rely on this heuristic to be recognized as UTF7 will no longer be detected.

### What is it?

 UTF-7 is a character encoding used to represent Unicode-encoded text using a stream of ASCII characters. This encoding was proposed for use in internet email messages. The simple mail transport protocol (SMTP) standard for transmitting mail messages does not allow byte values above the ASCII range. So UTF-7 must include some provision to encode these higher value characters. For example, the following text:

<script>MyMethod()</script>

Is encoded in UTF-7 as:

+ADw-script+AD4-MyMethod()+ADw-/script+AD4-

Note that the left angle bracket is encoded as +ADw- and the right angle bracket is encoded as +AD4-.

If Internet Explorer renders a page where the character set is not explicitly specified, then it uses a set of heuristics to “sniff” the page and determine the encoding. If the characters belonging to the UTF-7 encoding are found early enough in the Web page, Internet Explorer may guess that the encoding is UTF-7. What might have been a harmless string in one character set could be interpreted as a potentially malicious script if the encoding is assumed to be UTF-7.

Internet Explorer 8 includes a feature to look for UTF-7 encoded strings. If it finds them, it escapes the text so that any embedded script cannot execute.

### When is this event logged?

This event is logged when Internet Explorer detects a page that is encoded using UTF7.

### Example

Perform the following steps to see this event logged in the compatibility tool:

1. Create a Web page with the following contents. For this example call it 1058.html. The file can be placed anywhere. But for this example, the file is located on the desktop.

<html>

<body>if you see a pop up, this test fails. otherwise, it's a pass...

+ADw-script+AD4-alert(document.location)+ADw-/script+AD4-

</body>

</html>

1. Open Internet Explorer.
2. Ensure Internet Explorer is set to auto-select the page encoding. You can do this by selecting the Page > Encoding > Auto-Select menu item.
3. Open a browser and navigate to the Web page. For example:

C:\Users\JohnS\Desktop\1058.html

The page contains script encoded in UTF-7. Internet Explorer detects this and forces the page’s encoding to Windows-1252. The result is that the page is rendered as plain text (as opposed to the script being executed). When Internet Explorer 8 detects the UTF7 encoded script the event is logged.

If you run the same script on Internet Explorer 7 the UF7 encoded script will not be detected and an alert box will appear.

### Remediation

The best way to avoid this issue is to always specify the encoding of your Web page. You can do so with a meta tag as in the following example:

<meta http-equiv="Content-type" content="text/html; charset=utf-8">

You can also set a HTTP header:

Content-Type: text/html; charset=UTF-8

Of course, when you do specify a character set, make sure it’s not UTF-7—because of the susceptibility this character set has to cross-site scripting (XSS) attacks.

This feature can also be disabled by modifying the registry.

**Security Warning!** *Disabling the feature should only be used as a temporary measure during troubleshooting—to compare behavior of the application when the feature is enabled or not. It is not recommended that the feature be left disabled on an on-going basis.*

This feature can also be directly disabled in Internet Explorer by configuring the following registry keys:

HKEY\_CURRENT\_USER\Software\Microsoft\Internet Explorer\Main\FeatureControl\FEATURE\_DISABLE\_UTF7\_SNIFFIN\iexplore.exe = 0x00000000

## 1059 – Ajax Navigation

### Logged Message

Windows Internet Explorer 8 has a new HTML5.0 AJAX navigation feature that allows sites to maintain and track changes in AJAX states by treating them as a navigation. This means sites setting the location.hash on the window to mark a state change (like zooming in a map) now create an entry in the travelog/back button in Internet Explorer, allowing the user to go back to the previous state. This can be problematic for sites that were using the location.hash to send data between cross-domain components, like IFrames in the page.

### What is it?

A page can update the hash of the URL by changing the window.location.hash property. Doing so adds an entry to the travel log of the current page. The effect is that when you click the back button you’ll navigate to the last value of window.location.hash rather than the last page you visited. Consider a simple example:

Assume you have a mapping site that uses Ajax controls. The user navigates to the site which adds an entry to the travel log. The user then performs the following functions:

1. Enters a street address and clicks a button to display the corresponding map
2. Places a push pin at a location on the map
3. Zooms in

The developer has written the site so that each of these actions results in updating the window.location.hash property. Each time this happens, Internet Explorer 8 updates the travel log. The end result is that when the user clicks the back button 3 times, this is what happens (in order):

1. The user is returned to [B] – the push pin exists but the map is no longer zoomed in.
2. The user is returned to [A] – the map of the entered address is displayed but without the push pin
3. The user is returned to the initial page for the mapping site

Each time you set the window.location.hash property two important things happen:

* The previous URL, which may contain a previous hash fragment is added to the travel log (so clicking the back button returns to that URL).
* A hashChange event fires

This behavior works well for new Ajax enabled Web sites. The potential problem is that some Web sites manually manipulate the hash of the URL for their own purposes. So introducing code that explicitly sets window.location.hash (and modifies the URL) might break those Web pages.

### When is this event logged?

This event is logged when a Web page sets window.location.hash. This applies only if you’re in IE8 standards mode.

Example  
Perform the following steps to see this event logged in the compatibility tool:

1. Create a Web page with the following contents. For this example call it 1059.html .

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN" "http://www.w3.org/TR/REC-html40/strict.dtd">

<html>

<head>

<meta http-equiv="X-UA-Compatible" content="IE=8" >

<title>AJAX MAP</title>

</head>

<script type="text/javascript" src="http://dev.virtualearth.net/mapcontrol/mapcontrol.ashx?v=6"></script>

<script type="text/javascript">

var oMap = null;

var iZoomLevel = 0;

function GetMap()

{

oMap = new VEMap('myMap');

oMap.LoadMap();

oMap.AttachEvent("onendzoom", ZoomHandler);

iZoomLevel = oMap.GetZoomLevel();

window.location.hash = iZoomLevel;

}

function ZoomHandler(e)

{

iZoomLevel = oMap.GetZoomLevel();

window.location.hash = iZoomLevel;

}

function HashChangeHandler()

{

var hash = window.location.hash;

var iNewZoomLevel = hash.substr(1);

if (iNewZoomLevel != iZoomLevel)

{

iZoomLevel = iNewZoomLevel;

oMap.SetZoomLevel(iNewZoomLevel);

}

}

</script>

<body style='overflow: scroll; heigth:100%' onload="GetMap();" onhashchange="HashChangeHandler();">

<div id='myMap' style='position: relative; width: 500px; height: 500px; valign: center'></div>

</body>

</html>

1. Install the file in the root directory of the local web server. On an IIS server this means putting the file in this directory:

\Inetpub\wwwroot

1. Browse to the file:

<http://localhost/1059.html>

1. Click in the zoom-control (the magnifying glass in the upper left).

*The URL changes after the #.*

1. Click the back button in the browser

You’re returned to the previous zoom setting. And notice that the hash fragment in the URL is restored to the value before you zoomed into the map.

Each time the window.location.hash is set, this event is logged.

### Remediation

The behavior described for this event only occurs if the page is being rendered using IE8 standards mode. So putting the page into Internet Explorer 7 strict mode is a workaround. In this case, setting window.location.hash will not affect the travel log.

## 1061 – Application Protocol

### Logged Message

 Windows Internet Explorer 8 displays a dialog before starting an application that is registered to handle an application protocol. The user must confirm that they intended to start the application. The Application Protocol Handler Dialog security feature protects users from accidentally executing an application with dangerous content.

### What is it?

 A protocol in this context is referenced by the prefix of the URL. For example

* <http://microsoft.com> - refers to the http protocol

1. javascript:alert('hello world') - refers to the javascript protocol

 Developers can register their own application to a URL protocol. See:

<http://msdn.microsoft.com/en-us/library/aa767914.aspx>

 Registering your own application would allow you to have a new URL such as:

thing:SomethingThatThingUnderstands

In Internet Explorer 7 when you "launched" an application registered to a URL protocol, Internet Explorer would launch the application without further interaction from the user. In Internet Explorer 8, this behavior is being modified so that the user is first asked to confirm that the application should be launched. This request is made every time the application is launched unless the user indicates otherwise (each confirmation also asks the user if they want to be asked for confirmation in the future).

### When is this event logged?

This event is logged when an application registered to a URL protocol is launched.

Binary protocol registered handlers such as http:, https:, file:, mk:, ms-help:, and ftp: will not be logged. A binary protocol is a protocol which is intended or expected to be read by a machine rather than a human being. Binary protocols have the advantage of terseness, which translates into speed of transmission and interpretation.

### Example

 Perform the following steps to see this event logged in the compatibility tool:

1. Create a Web page with the following contents. For this example call it 1061.html . The file can be placed anywhere. For this example, the file is located on the desktop.

<html>

<head>

<title>Test of onenote: application protocol</title>

</head>

<body>

<p>Open the <a href="[onenote://C:\Program Files\Microsoft Office\Templates\1033\ONENOTE\12\Stationery](onenote://C:\Program%20Files\Microsoft%20Office\Templates\1033\ONENOTE\12\Stationery)\BLANK.ONE

"> Notebook.</a></p>

</body>

</html>

1. Open a browser and navigate to the Web page. For example:

C:\Users\JohnS\Desktop\1061.html

1. Click the Notebook link.

Note: this assumes you have OneNote installed.

A dialog is displayed asking you to confirm that you want to launch "Microsoft Office OneNote". The same dialog asks you if you want to be asked this question in the future (uncheck the checkbox if you don't).

If you don’t have OneNote on your system you can create your own protocol handler. The following example shows how to register an application, alert.exe in this case, to handle an alert protocol.

1. Use the registry editor to create the following entries (keys are shown in bold and named values are shown in plain text)

**HKEY\_CLASSES\_ROOT**

***alert***  
(Default) = "URL:Alert Protocol"  
URL Protocol **=** ""

**DefaultIcon**  
(Default) = "alert.exe"

**shell**

**open**

**command**  
(Default) = "C:\Program Files\Alert\alert.exe" "%1"

1. The following sample code contains a simple C# console application demonstrating a protocol handler for the alert protocol. Compile this code into an executable called Alert.

using System;

using System.Collections.Generic;

using System.Text;

namespace Alert1

{

class Program

{

static string ProcessInput(string s)

{

// TODO Verify and validate the input

// string as appropriate for your application.

return s;

}

static void Main(string[] args)

{

Console.WriteLine("Alert.exe invoked with the following parameters.\r\n");

Console.WriteLine("Raw command-line: \n\t" + Environment.CommandLine);

Console.WriteLine("\n\nArguments:\n");

foreach (string s in args)

{

Console.WriteLine("\t" + ProcessInput(s));

}

Console.WriteLine("\nPress any key to continue...");

Console.ReadKey();

}

}

}

1. Place the Alert.exe into the following directory:

C:\Program Files\Alert

1. Open a browser and enter the following into the address bar:

alert:Hello

A prompt will now be displayed asking if you want the Alert program to be run on your computer. Pressing “Allow” will cause this event to be logged. The event is also logged to the compatibility tool.

### Remediation

If you don't want to be prompted then uncheck the 'Always ask before opening this type of address' checkbox on the first prompt. You won't get the confirmation dialog again. However, the event will be logged each time you launch a registered application—regardless of whether you’re prompted or not.

## 1062 – Windows Reuse Navigation Restriction

### Logged Message

To help prevent spoofing attacks, Windows Internet Explorer 8 will prevent a top-level frame owned by one Web site from being navigated by another Web site from a different security context. In cases where this restriction takes effect, a new window or tab is opened rather than navigating the existing frame. This change conforms to the HTML5 draft specification for window navigation.

### What is it?

An HTML link can specify the name of a window in which the linked page should be displayed. This allows a link appearing in one frame to change the contents of another. As an example, consider a link in a ‘navigation’ frame that changes the page displayed in a ‘contents’ frame.

There’s a new feature in Internet explorer 8 that enforces some limits on the target frame in which a window can be opened.

The idea for the change is based on a security concern. You don’t want contoso.com to have a link that allows it to change the contents of a frame owned by woodgrovebank.com (potentially spoofing a woodgrovebank.com page).

If Internet Explorer detects a link on a page (running in one security context) that is attempting to replace the contents of another frame (in a different security context) then the new page is forced open in a new window (rather than being allowed to change the contents of the existing frame).

The security context of a source and target frame are considered different if the host name and zone is different between the two.

### When is this event logged?

When Internet Explorer forces a page to open in a new window (due to a different security context between the source and target frame), then this event is logged.

Example  
This example requires you to host a set of html files on two different web servers so that each can have a separate security context (host name and zone). For this example we’ll have two sites running on two different servers: site 1 and site 2.

Let’s start with Web Site 1:

1. Create a Web page with the following contents. For this example call it Site1.html.

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" >

<head>

<title>Windows Reuse Demo</title>

</head>

<body>

<div>

<h1>Site with IFrame</h1>

<iframe src="Site1Page1.html" id="site1" name="site1"></iframe>

<iframe src="http://<IP OF SITE 2 HOST>/site2.html"></iframe>

</div>

</body>

</html>

*Note: Be sure to replace <IP OF SITE 2 HOST> with the IP of the second web server. You will need to use the IP address to ensure site2 is accessed in the Internet Zone.*

1. Create another Web page with the following contents. For this example call it Site1Page1.html.

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" >

<head>

<title>Windows Reuse Demo</title>

</head>

<body>

<h2>Site 1 frame page</h2>

</body>

</html>

1. Install the files in the root directory of the Site 1 web server. On an IIS server this means putting the file in this directory:

\Inetpub\wwwroot

Now let’s move onto Web Site 2:

1. Create another Web page with the following contents. For this example call it Site2.html

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" >

<head runat="server">

<title>Windows Reuse Demo</title>

</head>

<body>

<div>

<h1>Site 2</h1>

<a href="Site2Page.html" target="\_parent">Target site 1 frame to load Site2Page.html</a>

</div>

</body>

</html>

1. Create another Web page with the following contents. For this example call it Site2Page.html

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" >

<head>

<title>Windows Reuse Demo</title>

</head>

<body>

<div>

<h2>Site 2 frame page</h2>

</div>

</body>

</html>

1. Install the files in the root directory of the Site 2 web server. On an IIS server this means putting the file in this directory:

\Inetpub\wwwroot

1. Browse to the file on the Site 1 web server where the Site1.html page is located:

[http://127.0.0.1/ Site1.html](http://127.0.0.1/%20Site1.html)

1. Click on the “Target site 1 frame to load Site2Page.html” hyperlink.

Due to the Site2 files being hosted on a different domain and in a different zone (site 1 is running in the Intranet Zone and Site 2 is running in the Internet Zone) Site 2’s attempts to load the Site2Page.html into Site1’s frame results in a Windows Reuse Navigation Restriction event and the page is loaded into the full browser window replacing Site1. The event is also logged to the compatibility tool.

### Remediation

You should be aware of how this feature works and design your site to take the behavior into account.

However, you can modify the registry to disable this feature.

**Security Warning!** *It is not recommended that you leave this feature disabled on an on-going basis since doing so circumvents a security feature that exposes your web application to attack.*

*Disabling the feature should only be used as a temporary measure during troubleshooting—to compare behavior of the application when the feature is enabled or not. It is not recommended that the feature be left disabled on an on-going basis.*

You may disable the feature by setting the following feature key:

* HKCU\Software\Microsoft\Internet Explorer\MAIN\FeatureControl\FEATURE\_Isolate\_Named\_Windows\iexplore.exe = 0x00000000

Note: For further details on how to use feature control keys please reference:

<http://msdn.microsoft.com/en-us/library/ms537184(VS.85).aspx>

## 1063 – MIME Restrictions: Authoritative Content Type Handling

### Logged Message

Windows Internet Explorer 8 uses Multipurpose Internet Mail Extensions (MIME) information to determine how to handle files sent by a Web server. The MIME Handling Restrictions feature reports an unsafe content handler when the reported MIME file type does not match the observed MIME file type, and the content handler for the observed MIME file type is unsafe.

### What is it?

For any page that’s sent to the browser, Internet Explorer verifies that the content type that’s reported by the server matches the content type that it derives by parsing the page (sometimes called “sniffing”). For example, a server may send a page to a browser that contains a MIME type that the server claims is a text file but that Internet Explorer “sniffs” to actually be an executable file.

In Internet Explorer 7, the browser treats the content type of a page as the type that it sniffs, ignoring whatever content type is specified by the server (i.e., it’ll treat an executable as an executable even though the server has marked it with a “text file” content type).

Internet Explorer 8 contains a feature called authoritative content type handling that changes this behavior.

In Internet Explorer 8, if a page is sent to the browser that has the “authoritative content type” header equal to true, then the above logic is modified. In this case, Internet Explorer will treat the content as the server specified but it will still sniff the content. If the content type that’s specified by the server is different from what’s sniffed and the sniffed type is considered “dangerous”, then a warning is displayed to the user. So in the above example, the executable that the server claims is a text file will be treated as a text file—but the browser will display a warning indicating that the type is potentially unsafe.

The server may insist that the browser treat its content as specified (e.g., insisting that an executable is actually text). However, as a precaution, the browser continues to perform its security checks and warns the user if it sniffs a potentially dangerous MIME type.

In many cases the server may not be playing a purposely malicious role. Consider an Internet photo sharing site as an example. Such a server may just assume that the files its customers are uploading are pictures. But some customers may be uploading other files types (e.g., scripts or executables). The server could interrogate uploaded files to look for unexpected content. But many site implementers will probably not choose to expend the processing resources required to do so. The MIME restrictions feature in Internet Explorer helps protect the user regardless.

Here is an example of the types of extensions that will trigger the user prompt:

* ade
* adp
* app
* asp
* bas
* bat
* exe
* fxp
* gadget
* mshxml
* msi

### When is this event logged?

 This event is logged when:

* the server’s specified content type is different from what is sniffed and
* the sniffed type is dangerous and
* the “X-Content-Type-Options” header = nosniff.

### Remediation

The best strategy to avoid this issue is to make the specified type of your content match the sniffed type. Just be aware that this may not always be possible. Think of the photo sharing example mentioned earlier. In that example, there’s nothing to prevent users from uploading executables that are given a .jpg file extension (thereby “fooling” the server).

This feature can be disabled.

**Security Warning!** *Disabling the feature should only be used as a temporary measure during troubleshooting—to compare behavior of the application when the feature is enabled or not. It is not recommended that the feature be left disabled on an on-going basis.*

If you want to disable this feature then simply don’t set the X-Content-Type-Options header. If you don’t, this logic is never executed and the event is never logged.

In general, disabling this feature should be avoided but may be useful as a short term solution.

## 1064 – MIME Sniffing Restrictions – No Image Elevation to HTML

### Logged Message

Windows Internet Explorer 8 uses Multipurpose Internet Mail Extensions (MIME) information to determine how to handle files sent by a Web server. For example, when Internet Explorer receives a .jpg file, the user sees the file in an Internet Explorer window. The MIME Handling Restrictions feature helps prevent script injection attacks against Web servers by ensuring that any content delivered with an IMAGE MIME is not treated as HTML or XML.

### What is it?

The encoding of images (such as JPEG files) include provisions to embed comment blocks within the image. For example, you might put text in the comment block of a JPEG file to note when the picture was taken, copyright information, etc.

If you put HTML code in the comment block of an image, some browsers may interpret it as executable markup and process it. This is an opportunity for hackers to include malicious code in the comment blocks of images that the browser unintentionally executes.

Internet Explorer 8 contains a feature to prevent this type of attack from occurring. If an HTTP server sends content with a MIME type of image/\* then Internet Explorer interrogates the content to verify its type. If Internet Explorer determines that the content is not an image, but (potentially malicious) XML or HTML then it prevents the markup from executing.

### When is this event logged?

This event fires when:

* An HTTP server sends content with a MIME type of image/\* and
* Internet Explorer does not recognize the content as a known image type (.gif, .png, etc.) and
* Internet Explorer does detect that the content is actually XML or HTML.

### Example

For this example you will need Microsoft Visual Web Developer Express (VWDE). This is available free from <http://www.microsoft.com/express/vwd/>

1. Create a new ASP.Net c# Web site in VWDE. Call the application MIMETest
2. Open the file default.aspx within VWDE and replace the content of with the following:

<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Default.aspx.cs" Inherits="\_Default" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">

<head runat="server">

<title>MIME Sniffing Restrictions</title>

</head>

<body>

<form id="form1" runat="server">

<div>

<a href="Handler.ashx">Send html as Gif Mime type.</a>

</div>

</form>

</body>

</html>

1. Add a new item to the web project. When prompted, select the Generic Handler Template. Name the file Handler.ashx.

*A Handler.ashx file is added to the project.*

1. Now open Handler.ashx and replace its contents with the following:

<%@ WebHandler Language="C#" Class="Handler" %>

using System;

using System.Web;

public class Handler : IHttpHandler {

public void ProcessRequest (HttpContext context) {

context.Response.ContentType = "image/gif";

context.Response.Write("<html><body><script>alert('test');</script><br /></body></html>");

}

public bool IsReusable {

get {

return false;

}

}

}

1. Make Default.aspx the start page of the application
2. Press F5 to compile and run the site.
3. Click the hyperlink on the Web page.

*The HTTP response sends html to the browser with a “image/gif” MIME type. Since Internet Explorer interrogates the page and discovers it’s actually HTML (and it’s potentially dangerous), the event is logged to the compatibility tool.*

Note that no message is displayed to the user in the browser when this event occurs. Instead the event is silently logged to the compatibility tool.

### Remediation

There’s no workaround for this event. There’s not considered to be a legitimate reason to embed content in an image that is executable html/xml.

## 1065 – Web Proxy Error Handling Changes

### Logged Message

Windows Internet Explorer 8 blocks content returned by a proxy from a failed CONNECT command, or displays the content in a context based on the hostname of the proxy rather than in the context of the origin server.

### What is it?

The motivation for this feature is to help address a security issue that exists in the HTTPS protocol. Explaining this is more easily done with an example.

Let’s assume that a browser requests a page from a secure site (<https://www.woodgrovebank.co.uk>). Let’s also assume that there’s a proxy between the browser and that site. The proxy’s job is to mediate the establishment of a secure tunnel between the browser and the target server (once established, the proxy doesn’t have access to the encrypted traffic flowing between the two). If the proxy fails, and the secure connection is not successfully made all the way to the target server, then the proxy returns an error to the client. It’s common in such cases for the proxy to return a Web page to the browser with information about the error.

Internet Explorer 7 would simply render the page returned from the proxy—and execute that page in the context of the target server. Internet Explorer 8 has a feature that ensures that the secure connection is made all the way to the target server. If it isn’t, then no page is displayed.

To facilitate this new feature, Internet Explorer 8 includes a modification to WinInet—if the connection isn’t established all the way to the target server, then the WinInet connection returns nothing back to the caller (the caller in this case, is the browser). When the browser detects that the connection failed at the WinInet level, it downgrades the protocol to http and changes the host name in the address bar to be the name of the proxy. So the user starts by browsing to <https://www.woodgrovebank.co.uk>. But after the connection fails the user sees <http://AddressOfProxyServer> in the address bar. Additionally, no content is rendered to the user—even an error page returned from the proxy.

In this way, any potentially malicious page returned from the proxy is not rendered for the user. And since a page from the proxy is never processed, it would not gain access to cookies belonging to the target domain that the user was trying to connect to.

### When is this event logged?

When Internet Explorer fails to make a secure connection to a target server, this event is logged. Internet Explorer detects the event by the failure to create an SSL tunnel to the origin server through the proxy.

### Example

To recreate this scenario you can use Fiddler. Fiddler is a Web Debugging Proxy which logs all HTTP(S) traffic between your computer and the Internet. You can also manipulate the traffic sent between the browser and web server. You can download fiddler [here](http://www.fiddler2.com/fiddler2/version.asp).

Once you’ve installed Fiddler, perform the following tasks:

1. Start Fiddler. It’s accessible as a menu item in the Tools menu called Fiddler2.

*From now on the tool will be referred to as Fiddler. Fiddler2 is the same thing.*

1. In the Fiddler window, select the Rules > Customize Rules menu item.

*This opens Notepad with the CustomRules.js file. You can edit this file.*

1. Find this function in the text file:

static OnBeforeRequest(oSession: Session)

1. Place these lines before any other code in the method:

if (oSession.HTTPMethodIs("CONNECT")){

oSession["ui-color"]="orange";

oSession["ui-bold"]="true";

oSession.oRequest.FailSession(404, "Blocked", "Fiddler blocked CONNECT request.");

}

1. Save and close the file.
2. Attempt to browse to a secure Web site using HTTPS protocol.

*The attempt fails and the event is logged.*

Basically we’ve inserted code into Fiddler instructing it to always return failure from any connection attempt. This simulates a proxy that fails to make a connection.

Be sure to remove this code before you use Fiddler again or any connection attempt will fail.

### Remediation

There is no workaround to this feature. You should be aware of it and design your site accordingly.

## 1073 – Certificate Filtering

### Logged Message

Windows Internet Explorer 8 and Internet Explorer 7 use Certificate Filtering to select the appropriate certificate for client authentication. In Internet Explorer 8, this feature has been improved to remove certificates that are likely to be rejected by the server. For instance, explicitly untrusted certificate chains or certificates not associated with a private key will not show up in the list.

### What is it?

Internet Explorer 8 includes improved certificate selection logic than was found in the previous versions of the browser. When a client certificate is called for, Internet Explorer is smart enough to present the user with only those *usable* certificates that apply to the immediate situation. In many cases, the list of certificates can be automatically reduced to a single one, eliminating the need for the user to make a choice at all.

### When is this event logged?

This event is logged when certificate-based authentication occurs when you attempt to access a secured resource.

### Example

As an example, in client authentication using smartcards, when the user plugs in a smartcard, he/she is presented with a list of certificates to choose from. This may confuse the user as he/she cannot make out the differences between certificates and is often not knowledgeable enough to pick the right one.

Internet Explorer 8 implements certificate selection logic which filters out certificates inappropriate to the situation. In this example, Internet Explorer automatically picks the certificate appropriate for smart card authentication. From the user’s perspective, the authentication “just works”.

### Remediation

Set the following named value in the registry to use the old certificate selection logic.

* HKCU\Software\Microsoft\Internet Explorer\MAIN\FeatureControl\ FEATURE\_CLIENTAUTHCERTFILTER\iexplore.exe =0x00000002

Generally, there’s no reason to go back to the old certificate selection behavior. Unless this is causing a specific issue in your application, leave the default behavior as it is.