ASP.NET 4 Breaking Changes

This document describes changes that have been made for the .NET Framework version 4 release that can potentially affect applications that were created using earlier releases, including the ASP.NET 4 Beta 1 and Beta 2 releases.

Contents

ControlRenderingCompatabilityVersion Setting in the Web.config File 2

ClientIDMode Changes 2

HtmlEncode and UrlEncode Now Encode Single Quotation Marks 3

ASP.NET Page (.aspx) Parser is Stricter 3

Browser Definition Files Updated 3

System.Web.Mobile.dll Removed from Root Web Configuration File 4

ASP.NET Request Validation 4

Default Hashing Algorithm Is Now HMACSHA256 5

Configuration Errors Related to New ASP.NET 4 Root Configuration 5

ASP.NET 4 Child Applications Fail to Start When Under ASP.NET 2.0 or ASP.NET 3.5 Applications 7

ASP.NET 4 Web Sites Fail to Start on Computers Where SharePoint Is Installed 10

The HttpRequest.FilePath Property No Longer Includes PathInfo Values 11

ASP.NET 2.0 Applications Might Generate HttpException Errors that Reference eurl.axd 11

Event Handlers Might Not Be Not Raised in a Default Document in IIS 7 or IIS 7.5 Integrated Mode 12

Changes to the ASP.NET Code Access Security (CAS) Implementation 14

MembershipUser and Other Types in the System.Web.Security Namespace Have Been Moved 15

*Output Caching Changes to Vary * HTTP Header* 16

System.Web.Security Types for Passport are Obsolete 17

The MenuItem.PopOutImageUrl Property Fails to Render an Image in ASP.NET 4 17

Menu.StaticPopOutImageUrl and Menu.DynamicPopOutImageUrl Fail to Render Images When Paths Contain Backslashes 19

Disclaimer 21

ControlRenderingCompatabilityVersion Setting in the Web.config File

ASP.NET controls have been modified in the .NET Framework version 4 in order to let you specify more precisely how they render markup. In previous versions of the .NET Framework, some controls emitted markup that you had no way to disable. By default, ASP.NET 4 this type of markup is no longer generated.

If you use Visual Studio 2010 to upgrade your application from ASP.NET 2.0 or ASP.NET 3.5, the tool automatically adds a setting to the `Web.config` file that preserves legacy rendering. However, if you upgrade an application by changing the application pool in IIS to target the .NET Framework 4, ASP.NET uses the new rendering mode by default. To disable the new rendering mode, add the following setting in the `Web.config` file:

```
<pages controlRenderingCompatibilityVersion="3.5" />
```

The major rendering changes that the new behavior brings are as follows:

- The **Image** and **ImageButton** controls no longer render a `border="0"` attribute.
- The **BaseValidator** class and validation controls that derive from it no longer render red text by default.
- The **HtmlForm** control does not render a **name** attribute.
- The **Table** control no longer renders a `border="0"` attribute.
- Controls that are not designed for user input (for example, the **Label** control) no longer render the `disabled="disabled"` attribute if their **Enabled** property is set to **false** (or if they inherit this setting from a container control).

ClientIDMode Changes

The **ClientIDMode** setting in ASP.NET 4 lets you specify how ASP.NET generates the **id** attribute for HTML elements. In previous versions of ASP.NET, the default behavior was equivalent to the **AutoID** setting of **ClientIDMode**. However, the default setting is now **Predictable**.

If you use Visual Studio 2010 to upgrade your application from ASP.NET 2.0 or ASP.NET 3.5, the tool automatically adds a setting to the `Web.config` file that preserves the behavior of earlier versions of the .NET Framework. However, if you upgrade an application by changing the application pool in IIS to target the .NET Framework 4, ASP.NET uses the new mode by default. To disable the new client ID mode, add the following setting in the `Web.config` file:

```
<pages ClientIDMode="AutoID" / >
```

HtmlEncode and UrlEncode Now Encode Single Quotation Marks

In ASP.NET 4, the **HtmlEncode** and **UrlEncode** methods of the **HttpUtility** and **HttpServerUtility** classes have been updated to encode the single quotation mark character (') as follows:

- The **HtmlEncode** method encodes instances of the single quotation mark as `&#39;` .
- The **UrlEncode** method encodes instances of the single quotation mark as `%27`.

ASP.NET Page (.aspx) Parser is Stricter

The page parser for ASP.NET pages (`.aspx` files) and user controls (`.ascx` files) is stricter in ASP.NET 4 and will reject more instances of invalid markup. For example, the following two snippets would successfully parse in earlier releases of ASP.NET, but will now raise parser errors in ASP.NET 4.

```
<asp:HiddenField runat="server" ID="SomeControl" Value="sampleValue"  ; />
```

Notice the invalid semicolon at the end of the **HiddenField** tag.

```
<asp:LinkButton runat="server" ID="SomeControl" onclick="someControlClicked"
      style="display:inline;  CssClass="searchLink"  />
```

Notice the unclosed **style** attribute that runs into the **CssClass** attribute.

Browser Definition Files Updated

The browser definition files have been updated to include information about new and updated browsers and devices. Older browsers and devices such as Netscape Navigator have been removed, and newer browsers and devices such as Google Chrome and Apple iPhone have been added.

If your application contains custom browser definitions that inherit from one of the browser definitions that have been removed, you will see an error. For example, if the `App_Browsers` folder contains a browser definition that inherits from the IE2 browser definition, you will receive the following configuration error message:

```
The browser or gateway element with ID 'IE2' cannot be found.
```

Note The **HttpBrowserCapabilities** object (which is exposed by the page's **Request.Browser** property) is driven by the browser definitions files. Therefore, the information returned by accessing a property of this object in ASP.NET 4 might be different than the information returned in an earlier version of ASP.NET.

You can revert to the old browser definition files by copying the browser definition files from the following folder:

```
Windows\Microsoft.NET\Framework\v2.0.50727\CONFIG\Browsers
```

Copy the files into the corresponding `\CONFIG\Browsers` folder for ASP.NET 4. After you copy the files, run the Aspnet_regbrowsers.exe command-line tool.

For more information, download the ASP.NET Browser Definition Files release from http://aspnet.codeplex.com/releases/view/41420. This download includes the old browser definition files, the new browser definition files, and instructions for installing the files.

System.Web.Mobile.dll Removed from Root Web Configuration File

In previous versions of ASP.NET, a reference to the System.Web.Mobile.dll assembly was included in the root `Web.config` file in the **assemblies** section under **<system.web><compilation>**. In order to improve performance, the reference to this assembly was removed.

The System.Web.Mobile.dll assembly is included in ASP.NET 4, but it is deprecated. If you want to use types from the System.Web.Mobile.dll assembly, add a reference to this assembly to either the root `Web.config` file or to an application `Web.config` file. For example, if you want to use any of the (deprecated) ASP.NET mobile controls, you must add a reference to the System.Web.Mobile.dll assembly to the `Web.config` file.

ASP.NET Request Validation

The request validation feature in ASP.NET provides a certain level of default protection against cross-site scripting (XSS) attacks. In previous versions of ASP.NET, request validation was enabled by default. However, it applied only to ASP.NET pages (`.aspx` files and their class files) and only when those pages were executing.

In ASP.NET 4, by default, request validation is enabled for all requests, because it is enabled before the **BeginRequest** phase of an HTTP request. As a result, request validation applies to requests for all ASP.NET resources, not just .aspx page requests. This includes requests such as Web service calls and custom HTTP handlers. Request validation is also active when custom HTTP modules are reading the contents of an HTTP request.

As a result, request validation errors might now occur for requests that previously did not trigger errors. To revert to the behavior of the ASP.NET 2.0 request validation feature, add the following setting in the `Web.config` file:

```
<httpRuntime requestValidationMode="2.0" />
```

However, we recommend that you analyze any request validation errors to determine whether existing handlers, modules, or other custom code accesses potentially unsafe HTTP inputs that could be XSS attack vectors.

Default Hashing Algorithm Is Now HMACSHA256

ASP.NET uses both encryption and hashing algorithms to help secure data such as forms authentication cookies and view state. By default, ASP.NET 4 now uses the HMACSHA256 algorithm for hash operations on cookies and view state. Earlier versions of ASP.NET used the older HMACSHA1 algorithm.

Your applications might be affected if you run mixed ASP.NET 2.0/ASP.NET 4 environments where data such as forms authentication cookies must work across.NET Framework versions. To configure an ASP.NET 4 Web application to use the older HMACSHA1 algorithm, add the following setting in the `Web.config` file:

```
<machineKey validation="SHA1" />
```

Configuration Errors Related to New ASP.NET 4 Root Configuration

The root configuration files (the `machine.config` file and the root `Web.config` file) for the .NET Framework 4 (and therefore ASP.NET 4) have been updated to include most of the boilerplate configuration information that in ASP.NET 3.5 was found in the application `Web.config` files. Because of the complexity of the managed IIS 7 and IIS 7.5 configuration systems, running ASP.NET 3.5 applications under ASP.NET 4 and under IIS 7 and IIS 7.5 can result in either ASP.NET or IIS configuration errors.

We recommend that you upgrade ASP.NET 3.5 applications to ASP.NET 4 by using the project upgrade tools in Visual Studio 2010, if practical. Visual Studio 2010 automatically modifies the ASP.NET 3.5 application's `Web.config` file to contain the appropriate settings for ASP.NET 4.

However, it is a supported scenario to run ASP.NET 3.5 applications using the .NET Framework 4 without recompilation. In that case, you might have to manually modify the application's `Web.config` file before you run the application under the .NET Framework 4 and under IIS 7 or IIS 7.5.

The next two sections describe changes that you might need to make for different combinations of software.

Windows Vista SP1 or Windows Server 2008 SP1, where neither hotfix KB958854 nor SP2 are installed. In this configuration, the IIS 7 configuration system incorrectly merges an application's managed configuration by comparing the application-level `Web.config` file to the ASP.NET 2.0 `machine.config` files. Because of this, application-level `Web.config` files from the .NET Framework 3.5 or later must have a **system.web.extensions** configuration section definition (the **<configSections />** element) in order not to cause an IIS 7 validation failure.

However, manually modified application-level `Web.config` file entries that do not precisely match the original boilerplate configuration section definitions that were introduced with Visual Studio 2008 will

cause ASP.NET configuration errors. (The default configuration entries that are generated by Visual Studio 2008 work correctly.) A common problem is that manually modified `Web.config` files leave out the **allowDefinition** and **requirePermission** configuration attributes that are found on various configuration section definitions. This causes a mismatch between the abbreviated configuration section in application-level `Web.config` files and the complete definition in the ASP.NET 4 `machine.config` file. As a result, at run time, the ASP.NET 4 configuration system throws a configuration error.

Windows Vista SP2, Windows Server 2008 SP2, Windows 7, Windows Server 2008 R2, and also Windows Vista SP1 and Windows Server 2008 SP1 where hotfix KB958854 is installed.

In this scenario, the IIS 7 and IIS 7.5 native configuration system returns a configuration error because it performs a text comparison on the **type** attribute that is defined for a managed configuration section handler. Because all `Web.config` files that are generated by Visual Studio 2008 and Visual Studio 2008 SP1 have "3.5" in the type string for the **system.web.extensions** (and related) configuration section handlers, and because the ASP.NET 4 `machine.config` file has "4.0" in the **type** attribute for the same configuration section handlers, applications that are generated in Visual Studio 2008 or Visual Studio 2008 SP1 always fail configuration validation in IIS 7 and IIS 7.5.

Resolving These Issues

The workaround for the first scenario is to update the application-level `Web.config` file by including the boilerplate configuration text from a `Web.config` file that was generated automatically by Visual Studio 2008.

An alternative workaround for the first scenario is to install Service Pack 2 for Vista or Windows Server 2008 on your computer or to install hotfix KB958854 (http://support.microsoft.com/kb/958854) to fix the incorrect configuration-merge behavior of the IIS configuration system. However, after you perform either of these actions, your application will likely encounter a configuration error due to the issue described for the second scenario.

The workaround for the second scenario is to delete or comment out all the **system.web.extensions** configuration section definitions and configuration section group definitions from the application-level `Web.config` file. These definitions are usually at the top of the application-level `Web.config` file and can be identified by the **configSections** element and its children.

For both scenarios, it is recommended that you also manually delete the **system.codedom** section, although this is not required.

ASP.NET 4 Child Applications Fail to Start When Under ASP.NET 2.0 or ASP.NET 3.5 Applications

ASP.NET 4 applications that are configured as children of applications that run earlier versions of ASP.NET might fail to start because of configuration or compilation errors. The following example shows a directory structure for an affected application.

`/parentwebapp` (configured to use ASP.NET 2.0 or ASP.NET 3.5)
    `/childwebapp` (configured to use ASP.NET 4)

The application in the `childwebapp` folder will fail to start on IIS 7 or IIS 7.5 and will report a configuration error. The error text will include a message similar to the following:

```
The requested page cannot be accessed because the related configuration data for
the page is invalid.

The configuration section 'configSections' cannot be read because it is missing a
section declaration.
```

On IIS 6, the application in the `childwebapp` folder will also fail to start, but it will report a different error. For example, the error text might state the following:

```
The value for the 'compilerVersion' attribute in the provider options must be
'v4.0' or later if you are compiling for version 4.0 or later of the .NET
Framework.  To compile this Web application for version 3.5 or earlier of the .NET
Framework, remove the 'targetFramework' attribute from the <compilation> element of
the Web.config file
```

These scenarios occur because the configuration information from the parent application in the `parentwebapp` folder is part of the hierarchy of configuration information that determines the final merged configuration settings that are used by the child web application in the `childwebapp` folder. Depending on whether the ASP.NET 4 Web application is running on IIS 7 (or IIS 7.5) or on IIS 6, either the IIS configuration system or the ASP.NET 4 compilation system will return an error.

The steps that you must follow to resolve this issue and to get the child ASP.NET 4 application to work depend on whether the ASP.NET 4 application runs on IIS 6 or on IIS 7 (or IIS 7.5).

Step 1 (IIS 7 or IIS 7.5 only)

This step is necessary only on operating systems that run IIS 7 or IIS 7.5, which includes Windows Vista, Windows Server 2008, Windows 7, and Windows Server 2008 R2.

Move the **configSections** definition in the `Web.config` file of the parent application (the application that runs ASP.NET 2.0 or ASP.NET 3.5) into the root `Web.config` file for the.NET Framework 2.0. The IIS 7 and IIS 7.5 native configuration system scans the **configSections** element when it merges the hierarchy of configuration files. Moving the **configSections** definition from the parent Web application's

`Web.config` file to the root `Web.config` file effectively hides the element from the configuration merge process that occurs for the child ASP.NET 4 application.

On a 32-bit operating system or for 32-bit application pools, the root `Web.config` file for ASP.NET 2.0 and ASP.NET 3.5 is normally located in the following folder:

```
C:\Windows\Microsoft.NET\Framework\v2.0.50727\CONFIG
```

On a 64-bit operating system or for 64-bit application pools, the root `Web.config` file for ASP.NET 2.0 and ASP.NET 3.5 is normally located in the following folder:

```
C:\Windows\Microsoft.NET\Framework64\v2.0.50727\CONFIG
```

If you run both 32-bit and 64-bit Web applications on a 64-bit computer, you must move the **configSections** element up into root `Web.config` files for both the 32-bit and the 64-bit systems.

When you put the **configSections** element in the root `Web.config` file, paste the section immediately after the **configuration** element. The following example shows what the top portion of the root `Web.config` file should look like when you have finished moving the elements.

Note In the following example, lines have been wrapped for readability.

```
<?xml version="1.0" encoding="utf-8"?>
<!-- The root web configuration file -->
<configuration>
  <configSections>
    <sectionGroup name="system.web.extensions"
        type="System.Web.Configuration.SystemWebExtensionsSectionGroup,
      System.Web.Extensions, Version=3.5.0.0, Culture=neutral,
      PublicKeyToken=31BF3856AD364E35">

      <sectionGroup name="scripting"
        type="System.Web.Configuration.ScriptingSectionGroup,
        System.Web.Extensions, Version=3.5.0.0, Culture=neutral,
        PublicKeyToken=31BF3856AD364E35">

        <section name="scriptResourceHandler"

type="System.Web.Configuration.ScriptingScriptResourceHandlerSection,
          System.Web.Extensions, Version=3.5.0.0, Culture=neutral,
          PublicKeyToken=31BF3856AD364E35" requirePermission="false"
          allowDefinition="MachineToApplication" />

        <sectionGroup name="webServices"
          type="System.Web.Configuration.ScriptingWebServicesSectionGroup,
```

```
            System.Web.Extensions, Version=3.5.0.0, Culture=neutral,
            PublicKeyToken=31BF3856AD364E35">

            <section name="jsonSerialization"
              type="System.Web.Configuration.ScriptingJsonSerializationSection,
              System.Web.Extensions, Version=3.5.0.0, Culture=neutral,
              PublicKeyToken=31BF3856AD364E35" requirePermission="false"
              allowDefinition="Everywhere" />

            <section name="profileService"
              type="System.Web.Configuration.ScriptingProfileServiceSection,
              System.Web.Extensions, Version=3.5.0.0, Culture=neutral,
              PublicKeyToken=31BF3856AD364E35" requirePermission="false"
              allowDefinition="MachineToApplication" />
            <section name="authenticationService"

type="System.Web.Configuration.ScriptingAuthenticationServiceSection,
            System.Web.Extensions, Version=3.5.0.0, Culture=neutral,
            PublicKeyToken=31BF3856AD364E35" requirePermission="false"
              allowDefinition="MachineToApplication" />

            <section name="roleService"
              type="System.Web.Configuration.ScriptingRoleServiceSection,
            System.Web.Extensions, Version=3.5.0.0, Culture=neutral,
            PublicKeyToken=31BF3856AD364E35" requirePermission="false"
              allowDefinition="MachineToApplication" />

          </sectionGroup>
        </sectionGroup>
      </sectionGroup>
    </configSections>
```

Step 2 (all versions of IIS)

This step is required whether the ASP.NET 4 child Web application is running on IIS 6 or on IIS 7 (or IIS 7.5).

In the `Web.config` file of the parent Web application that is running ASP.NET 2 or ASP.NET 3.5, add a **location** tag that explicitly specifies (for both the IIS and ASP.NET configuration systems) that the configuration entries only apply to the parent Web application. The following example shows the syntax of the **location** element to add:

```
<location path="" inheritInChildApplications="false" >
```

```
  <!-- Additional settings -->
</location>
```

The following example shows how the **location** tag is used to wrap all configuration sections starting with the **appSettings** section and ending with **system.webServer** section.

```
<location path="" inheritInChildApplications="false" >
  <appSettings />
  <connectionStrings />
  <system.web>
    <!-- Removed for brevity -->
  </system.web>
  <system.codedom>
    <!-- Removed for brevity -->
  </system.codedom>
  <system.webServer>
    <!-- Removed for brevity -->
</system.webServer>
</location>
```

When you have completed steps 1 and 2, child ASP.NET 4 Web applications will start without errors.

ASP.NET 4 Web Sites Fail to Start on Computers Where SharePoint Is Installed

Web servers that run SharePoint have a `Web.config` file that is deployed at the root of a SharePoint Web site (for example, `c:\inetpub\wwwroot\web.config` for Default Web Site). In this `Web.config` file, SharePoint sets a custom partial-trust level named `WSS_Minimal`.

If you try to run an ASP.NET 4 Web site that is deployed as a child of this type of SharePoint Web site, you will see the following error:

```
Could not find permission set named 'ASP.Net'.
```

This error occurs because the ASP.NET 4 code access security (CAS) infrastructure looks for a permission set named `ASP.Net`. However, the partial trust configuration file that is referenced by `WSS_Minimal` does not contain any permission sets with that name.

Currently there is not a version of SharePoint available that is compatible with ASP.NET. As a result, you should not attempt to run an ASP.NET 4 Web site as a child site underneath SharePoint Web sites.

The HttpRequest.FilePath Property No Longer Includes PathInfo Values

Previous versions of ASP.NET included a **PathInfo** value in the value returned from various file path-related properties, including **HttpRequest.FilePath**, **HttpRequest.AppRelativeCurrentExecutionFilePath**, and **HttpRequest.CurrentExecutionFilePath**. ASP.NET 4 no longer includes the **PathInfo** value in the return values from these properties. Instead, the **PathInfo** information is available in **HttpRequest.PathInfo**. For example, imagine the following URL fragment:

```
/testapp/Action.mvc/SomeAction
```

In earlier versions of ASP.NET, **HttpRequest** properties have the following values:

HttpRequest.FilePath: `/testapp/Action.mvc/SomeAction`

HttpRequest.PathInfo: (empty)

In ASP.NET 4, **HttpRequest** properties instead have the following values:

HttpRequest.FilePath: `/testapp/Action.mvc`

HttpRequest.PathInfo: `SomeAction`

ASP.NET 2.0 Applications Might Generate HttpException Errors that Reference eurl.axd

After ASP.NET 4 has been enabled on IIS 6, ASP.NET 2.0 applications that run on IIS 6 (in either Windows Server 2003 or Windows Server 2003 R2) might generate errors such as the following:

```
System.Web.HttpException: Path '/[yourApplicationRoot]/eurl.axd/[Value]' was
not found.
```

This error occurs because when ASP.NET detects that a Web site is configured to use ASP.NET 4, a native component of ASP.NET 4 passes an extensionless URL to the managed portion of ASP.NET for further processing. However, if virtual directories that are below an ASP.NET 4 Web site are configured to use ASP.NET 2.0, processing the extensionless URL in this way results in a modified URL that contains the string “eurl.axd”. This modified URL is then sent to the ASP.NET 2.0 application. ASP.NET 2.0 cannot recognize the "eurl.axd" format. Therefore, ASP.NET 2.0 tries to find a file named `eurl.axd` and execute it. Because no such file exists, the request fails with an **HttpException** exception.

You can work around this issue using one of the following options.

Option 1

If ASP.NET 4 is not required in order to run the Web site, remap the site to use ASP.NET 2.0 instead.

Option 2

If ASP.NET 4 is required in order to run the Web site, move any child ASP.NET 2.0 virtual directories to a different Web site that is mapped to ASP.NET 2.0.

Option 3

If it is not practical to remap the Web site to ASP.NET 2.0 or to change the location of a virtual directory, explicitly disable extensionless URL processing in ASP.NET 4. Use the following procedure:

1. In the Windows registry, open the following node:

   `HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\ASP.NET\4.0.30319.0`

2. Create a new **DWORD** value named **EnableExtensionlessUrls**.

3. Set **EnableExtensionlessUrls** to 0. This disables extensionless URL behavior.

4. Save the registry value and close the registry editor.

5. Run the **iisreset** command-line tool, which causes IIS to read the new registry value.

   Note Setting **EnableExtensionlessUrls** to 1 enables extensionless URL behavior. This is the default setting if no value is specified.

Event Handlers Might Not Be Not Raised in a Default Document in IIS 7 or IIS 7.5 Integrated Mode

ASP.NET 4 includes modifications that change how the **action** attribute of the HTML **form** element is rendered when an extensionless URL resolves to a default document. An example of an extensionless URL resolving to a default document would be `http://contoso.com/`, resulting in a request to `http://contoso.com/Default.aspx`.

ASP.NET 4 now renders the HTML **form** element's **action** attribute value as an empty string when a request is made to an extensionless URL that has a default document mapped to it. For example, in earlier releases of ASP.NET, a request to `http://contoso.com` would result in a request to `Default.aspx`. In that document, the opening **form** tag would be rendered as in the following example:

```
<form action="Default.aspx" />
```

In ASP.NET 4, a request to `http://contoso.com` also results in a request to `Default.aspx`. However, ASP.NET now renders the HTML opening **form** tag as in the following example:

```
<form action="" />
```

This difference in how the **action** attribute is rendered can cause subtle changes in how a form post is processed by IIS and ASP.NET. When the **action** attribute is an empty string, the IIS **DefaultDocumentModule** object will create a child request to `Default.aspx`. Under most conditions, this child request is transparent to application code, and the `Default.aspx` page runs normally.

However, a potential interaction between managed code and IIS 7 or IIS 7.5 Integrated mode can cause managed .aspx pages to stop working properly during the child request. If the following conditions occur, the child request to a `Default.aspx` document will result in an error or in unexpected behavior:

1. An .aspx page is sent to the browser with the **form** element's **action** attribute set to "".
2. The form is posted back to ASP.NET.
3. A managed HTTP module reads some part of the entity body. For example, a module reads **Request.Form** or **Request.Params**. This causes the entity body of the POST request to be read into managed memory. As a result, the entity body is no longer available to any native code modules that are running in IIS 7 or IIS 7.5 Integrated mode.
4. The IIS **DefaultDocumentModule** object eventually runs and creates a child request to the `Default.aspx` document. However, because the entity body has already been read by a piece of managed code, there is no entity body available to send to the child request.
5. When the HTTP pipeline runs for the child request, the handler for `.aspx` files runs during the handler-execute phase.
6. Because there is no entity body, there are no form variables and no view state, and therefore no information is available for the .aspx page handler to determine what event (if any) is supposed to be raised. As a result, none of the postback event handlers for the affected .aspx page run.

You can work around this behavior in the following ways:

- Identify the HTTP module that is accessing the request's entity body during default document requests and determine whether it can be configured to run only for managed requests. In Integrated mode for both IIS 7 and IIS 7.5, HTTP modules can be marked to run only for managed requests by adding the following attribute to the module's **system.webServer/modules** entry:

  ```
  precondition="managedHandler"
  ```

  This setting disables the module for requests that IIS 7 and IIS 7.5 determine as not being managed requests. For default document requests, the first request is to an extensionless URL. Therefore, IIS does not run any managed modules that are marked with a precondition of `managed Handler` during initial request processing. As a result, managed modules will not accidentally read the entity body and thus the entity body is still available and is passed along to the child request and to the default document.

- If the problematic HTTP modules have to run for all requests (for static files, for extensionless URLs that resolve to the **DefaultDocumentModule** object, for managed requests, etc.), modify the affected .aspx pages by explicitly setting the **Action** property of the page's **System.Web.UI.HtmlControls.HtmlForm** control to a non-empty string. For example, if the default

document is `Default.aspx`, modify the page's code to explicitly set the **HtmlForm** control's **Action** property to "Default.aspx".

Changes to the ASP.NET Code Access Security (CAS) Implementation

ASP.NET 2.0, and by extension the ASP.NET features that were added in 3.5, use the .NET Framework 1.1 and 2.0 code access security (CAS) model. However, the implementation of CAS in ASP.NET 4 has been substantially overhauled. As a result, partial-trust ASP.NET applications that rely on trusted code running in the global assembly cache (GAC) might fail with various security exceptions. Partial-trust applications that rely on extensive modifications to machine CAS policy might also fail with security exceptions.

You can revert partial-trust ASP.NET 4 applications to the behavior of ASP.NET 1.1 and 2.0 using the new **legacyCasModel** attribute in the **trust** configuration element, as shown in the following example:

```
<trust level= "Medium" legacyCasModel="true" />
```

When you revert to the legacy CAS model, the following old CAS behaviors are enabled:

- Machine CAS policy is honored.
- Multiple different permission sets in a single application domain are allowed.
- Explicit permission assertions are not required for assemblies in the GAC that are invoked when only ASP.NET or other .NET Framework code is on the stack.

One scenario cannot be reverted in the .NET Framework 4: non-Web partial-trust applications can no longer call certain APIs in System.Web.dll and System.Web.Extensions.dll. In previous versions of the .NET Framework, it was possible for non-Web partial-trust applications to be explicitly granted **AspNetHostingPermission** permissions. These applications could then use **System.Web.HttpUtility**, types in the **System.Web.ClientServices.*** namespaces, and types related to membership, roles, and profiles. Calling these types from non-Web partial trust applications is no longer supported in the .NET Framework 4.

> **Note** The **HtmlEncode** and **HtmlDecode** functionality of the **System.Web.HttpUtility** class was moved to the new .NET Framework 4 **System.Net.WebUtility** class. If that was the only ASP.NET functionality that was being used, modify the application's code to use the new **WebUtility** class instead.

The following is a high-level summary of the changes to the default CAS implementation in ASP.NET 4:

- ASP.NET application domains are now homogeneous application domains. Only partial-trust and full-trust grant sets are available in an application domain.
- ASP.NET partial-trust grant sets are independent from any enterprise-level, machine-level, or user-level CAS policy.

- ASP.NET assemblies that shipped in 3.5 and 3.5 SP1 have been converted to use the .NET Framework 4 transparency model.
- Use of the ASP.NET **AspNetHostingPermission** attribute has been substantially reduced. Most instances of this attribute have been removed from the public ASP.NET APIs.
- Dynamically compiled assemblies that are created by ASP.NET build providers have been updated to explicitly mark assemblies as transparent.
- All ASP.NET assemblies are now marked in such a way that the APTCA attribute is honored only in Web hosting environments. Partially trusted non-Web hosting environments like ClickOnce will not be able to call into ASP.NET assemblies.

For more information about the new ASP.NET 4 code access security model, see Using Code Access Security in ASP.NET Applications on the MSDN Web site.

MembershipUser and Other Types in the System.Web.Security Namespace Have Been Moved

Some types that are used in ASP.NET membership have been moved from `System.Web.dll` to the new `System.Web.ApplicationServices.dll` assembly. The types were moved in order to resolve architectural layering dependencies between types in the client and in extended .NET Framework SKUs.

Web site projects do not have problems as a result of moving these types, because `System.Web.ApplicationServices.dll` was added to the list of referenced assemblies that is used by default by the ASP.NET compilation system. If you upgrade a Web site project that was created using an earlier version of ASP.NET to ASP.NET 4 by opening it in Visual Studio 2010, the project will compile without errors.

Similarly, if you upgrade a Web application project that was created in an earlier version of ASP.NET to ASP.NET 4 by opening it in Visual Studio 2010, the upgrade process adds a reference to `System.Web.ApplicationServices.dll` to the project. Therefore, upgraded Web application projects will also compile without errors.

Compiled (binary) files that were created using earlier versions of ASP.NET will also run without errors on ASP.NET 4, even though the membership types were moved to a different assembly. Type forwarding information has been added to the ASP.NET 4 version of `System.Web.dll` that automatically routes run-time references for these types to the new location for the types.

However, class libraries that use specific membership types and that have been upgraded from earlier versions of ASP.NET will fail to compile when used in an ASP.NET 4 project. For example, a class library project might fail to compile and report an error such as the following:

```
The type 'System.Web.Security.MembershipUser' is defined in an assembly that is not
referenced. You must add a reference to assembly 'System.Web.ApplicationServices,
Version=4.0.0.0, Culture=neutral, PublicKeyToken=31bf3856ad364e35'.
```

```
The type name 'MembershipUser' could not be found. This type has been forwarded to
assembly 'System.Web.ApplicationServices, Version=4.0.0.0, Culture=neutral,
PublicKeyToken=31bf3856ad364e35'. Consider adding a reference to that assembly.
```

You can work around this problem by adding a reference in your class library project to System.Web.ApplicationServices.dll.

The following list shows the *System.Web.Security* types that were moved from `System.Web.dll` to `System.Web.ApplicationServices.dll`:

- *System.Web.Security.MembershipCreateStatus*
- *System.Web.Security.Membership.CreateUserException*
- *System.Web.Security.MembershipPasswordException*
- *System.Web.Security.MembershipPasswordFormat*
- *System.Web.Security.MembershipProvider*
- *System.Web.Security.MembershipProviderCollection*
- *System.Web.Security.MembershipUser*
- *System.Web.Security.MembershipUserCollection*
- *System.Web.Security.MembershipValidatePasswordEventHandler*
- *System.Web.Security.ValidatePasswordEventArgs*
- *System.Web.Security.RoleProvider*
- *System.Web.Configuration.MembershipPasswordCompatibilityMode*

Output Caching Changes to Vary * HTTP Header

In ASP.NET 1.0, a bug caused cached pages that specified `Location="ServerAndClient"` as an output–cache setting to emit a `Vary:*` HTTP header in the response. This had the effect of telling client browsers to never cache the page locally.

In ASP.NET 1.1, the **System.Web.HttpCachePolicy.SetOmitVaryStar** method was added, which you could call to suppress the `Vary:*` header. This method was chosen because changing the emitted HTTP header was considered a potentially breaking change at the time. However, developers have been confused by the behavior in ASP.NET, and bug reports suggest that developers are unaware of the existing **SetOmitVaryStar** behavior.

In ASP.NET 4, the decision was made to fix the root problem. The `Vary:*` HTTP header is no longer emitted from responses that specify the following directive:

```
<%@OutputCache Location="ServerAndClient" %>
```

As a result, **SetOmitVaryStar** is no longer needed in order to suppress the `Vary:*` header.

In applications that specify `Location="ServerAndClient"` in the **@ OutputCache** directive on a page, you will now see the behavior implied by the name of the **Location** attribute's value – that is, pages will be cacheable in the browser without requiring that you call the **SetOmitVaryStar** method.

If pages in your application must emit `Vary:*`, call the **AppendHeader** method, as in the following example:

```
HttpResponse.AppendHeader("Vary","*");
```

Alternatively, you can change the value of the output caching **Location** attribute to "Server".

System.Web.Security Types for Passport are Obsolete

The Passport support built into ASP.NET 2.0 has been obsolete and unsupported for a few years due to changes in Passport (now LiveID). As a result, the five types related to Passport in **System.Web.Security** are now marked with the **ObsoleteAttribute** attribute.

The MenuItem.PopOutImageUrl Property Fails to Render an Image in ASP.NET 4

In ASP.NET 3.5, the *MenuItem.PopOutImageUrl* property lets you specify the URL for an image that is displayed in a menu item to indicate that the menu item has a dynamic submenu. The following example shows how to specify this property in markup in ASP.NET 3.5.

```
<asp:menu id="NavigationMenu"
    staticdisplaylevels="1"
    staticsubmenuindent="10"
    orientation="Vertical"
    target="_blank"
    runat="server">
  <items>
    <asp:menuitem navigateurl="default2.aspx"
        text="Home"
        PopOutImageUrl="~/Images/Popout.gif"
        tooltip="Home">
      <asp:menuitem navigateurl="default3.aspx"
          text="Movies"
          PopOutImageUrl="~/Images/Popout.gif"
          tooltip="Movies">
      </asp:menuitem>
    </asp:menuitem>
  </items>
</asp:menu>
```

As a result of a design change in ASP.NET 4, no output is rendered for the *PopOutImageUrl* if the property is set for the *MenuItem* class. Instead, you must specify an image URL directly in the *Menu* control using either the *StaticPopOutImageUrl* property or the *DynamicPopOutImageUrl* property. When you work with a static menu, the *Menu.StaticPopOutImageUrl* property specifies the URL for an image that is displayed in order to indicate that the static menu item has a submenu, as shown in the following example:

```
<asp:menu id="NavigationMenu"
    staticdisplaylevels="1"
    staticsubmenuindent="10"
    orientation="Vertical"
    target="_blank"
    StaticPopOutImageTextFormatString="More..."
    StaticPopOutImageUrl="Images/Popout.gif"
    runat="server">
  <items>
    <asp:menuitem navigateurl="default2.aspx"
        text="Home"
        tooltip="Home">
      <asp:menuitem navigateurl="default3.aspx"
          text="Movies"
          tooltip="Movies">
      </asp:menuitem>
    </asp:menuitem>
  </items>
</asp:menu>
```

If you are working with a dynamic menu, you use the *Menu.DynamicPopOutImageUrl* property to specify the URL for an image that indicates that a dynamic menu item has a submenu. The following example is similar to the previous one, but shows how to set the *DynamicPopOutImageUrl* property for a dynamic menu.

```
<asp:menu id="NavigationMenu"
    staticdisplaylevels="1"
    staticsubmenuindent="10"
    orientation="Vertical"
    target="_blank"
    DynamicPopOutImageTextFormatString="More..."
    DynamicPopOutImageUrl="Images/Popout.gif"
    runat="server">
  <items>
    <asp:menuitem navigateurl="default2.aspx"
        text="Home"
        tooltip="Home">
      <asp:menuitem navigateurl="default3.aspx"
          text="Movies"
          tooltip="Movies">
      </asp:menuitem>
```

```
    </asp:menuitem>
  </items>
</asp:menu>
```

If the *Menu.DynamicPopOutImageUrl* property is not set and the *Menu.DynamicEnableDefaultPopOutImage* property is set to *false*, no image is displayed. Similarly, if the *StaticPopOutImageUrl* property is not set and the *StaticEnableDefaultPopOutImage* property is set to *false*, no image is displayed.

When you set the paths for these properties, use a forward slash (/) instead of a backslash (\). For more information, see Menu.StaticPopOutImageUrl and Menu.DynamicPopOutImageUrl Fail to Render Images When Paths Contain Backslashes elsewhere in this document.

Menu.StaticPopOutImageUrl and Menu.DynamicPopOutImageUrl Fail to Render Images When Paths Contain Backslashes

In ASP.NET 4, the images that you specify using the *Menu.StaticPopOutImageUrl* and *Menu.DynamicPopOutImageUrl* properties will not render if the path contains backslashes (\). This is a change from earlier versions of ASP.NET.

The following example of *Menu* control markup shows the *StaticPopOutImageUrl* property set using a path that contains a backslash. In ASP.NET 4, the image specified in the property will not render.

```
<asp:menu id="NavigationMenu"
    staticdisplaylevels="1"
    staticsubmenuindent="10"
    orientation="Vertical
    target="_blank"
    StaticPopOutImageTextFormatString="More..."
    StaticPopOutImageUrl="Images\Popout.gif"
    runat="server">
  <items>
    <asp:menuitem navigateurl="default2.aspx"
        text="Home"
        tooltip="Home">
      <asp:menuitem navigateurl="default3.aspx"
          text="Movies"
          tooltip="Movies">
      </asp:menuitem>
    </asp:menuitem>
  </items>
</asp:menu>
```

To correct this issue, change path values that are specified in the *StaticPopOutImageUrl* and *DynamicPopOutImageUrl* properties to use forward slashes (/). The following example shows this change:

```
<asp:menu id="NavigationMenu"
    staticdisplaylevels="1"
    staticsubmenuindent="10"
    orientation="Vertical
    target="_blank"
    StaticPopOutImageTextFormatString="More..."
    StaticPopOutImageUrl="Images/Popout.gif"
    runat="server">
  <items>
    <asp:menuitem navigateurl="default2.aspx"
        text="Home"
        tooltip="Home">
      <asp:menuitem navigateurl="default3.aspx"
          text="Movies"
          tooltip="Movies">
      </asp:menuitem>
    </asp:menuitem>
  </items>
</asp:menu>
```

Note that applications that were migrated from earlier versions of ASP.NET to ASP.NET 4 might also be affected, because the *MenuItem.PopOutImageUrl* property has been changed. For more information, see The MenuItem.PopOutImageUrl Property Fails to Render an Image in ASP.NET 4 elsewhere in this document.

Disclaimer

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.

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, email address, logo, person, place or event is intended or should be inferred.