**Moving an ASP.NET Application from IIS 6.0 to IIS 7.0**

This topic describes how to move a Web application from Internet Information Services (IIS) 6.0 to IIS 7.0. Web applications in IIS 7.0 can be configured to use either Classic mode or Integrated mode. Classic mode maintains backward compatibility with earlier versions of IIS by using an ISAPI extension to invoke the ASP.NET runtime. This option usually requires few or no modifications to existing applications.

IIS 7.0 Integrated mode is a unified request-processing pipeline that combines the ASP.NET request pipeline with the IIS core request pipeline. The integrated pipeline provides improved performance, provides modularity for configuration and administration, and adds flexibility for extending IIS with managed-code modules. For example, you can define a managed-code module in the App\_Code folder of the Web application and register it to apply to all IIS requests, which includes requests for static files.

Using IIS 7.0 Integrated mode might require minor changes to an application’s Web.config file. Some additional changes might be needed if the application uses any custom modules that implement the [IHttpModule](http://msdn.microsoft.com/en-us/library/system.web.ihttpmodule.aspx) interface.

For an overview of the request processing pipeline in Integrated mode in IIS 7.0, see [ASP.NET Application Life Cycle Overview for IIS 7.0](http://msdn.microsoft.com/en-us/library/bb470252.aspx). When you use IIS 7.0, you can run applications in Classic mode and Integrated mode side-by-side on the same server. Both Classic and Integrated mode support the .NET Framework version 2.0 and later releases. The .NET Framework version 1.1 is supported in Classic mode only. For more information about how to upgrade from earlier versions of IIS to IIS 7.0, see [Upgrading ASP.NET Applications to IIS 7.0: Differences between IIS 7.0 Integrated Mode and Classic mode](http://go.microsoft.com/fwlink/?LinkId=86952).

|  |
| --- |
| **NoteNote** |
| You can use the information in this topic to move Web applications from IIS 5.x to IIS 7.0. However, additional changes might be necessary, which are not discussed here. For more information see [Upgrading ASP.NET 1.1 to IIS7 on Windows Vista & Windows Server 2008](http://go.microsoft.com/fwlink/?LinkID=183152) and [Preparing to Upgrade to IIS 7](http://go.microsoft.com/fwlink/?LinkId=183151). |

This topic contains the following sections:

* [Web.config Files in IIS 7.0](http://msdn.microsoft.com/en-us/library/bb515251.aspx#CONFIG)
* [Moving a Web Application to Classic Mode](http://msdn.microsoft.com/en-us/library/bb515251.aspx#CLASSIC)
* [Moving a Web Application to Integrated Mode](http://msdn.microsoft.com/en-us/library/bb515251.aspx#INTEGRATED)

http://i.msdn.microsoft.com/Global/Images/clear.gif Web.config Files in IIS 7.0

When you move an ASP.NET Web application to IIS 7.0 Integrated mode, you must update the Web.config file. IIS 7.0 includes changes in how you can administer Web.config files and in the types of settings that can be stored in the Web.config files. The new settings are in a new configuration section named **system.webServer**.

In IIS 6.0, the ASP.NET MMC snap-in provides IIS-related administration features for configuring ASP.NET. For more information, see [Walkthrough: Configuring ASP.NET Applications in IIS 6.0 using MMC](http://msdn.microsoft.com/en-us/library/cek8kw6e.aspx).

In IIS 7.0, the administration of ASP.NET applications is integrated more closely with IIS administration, and there is no separate snap-in. Instead, all IIS and ASP.NET configuration is done by using IIS Manager. Because IIS 7.0 configuration information is based on the .NET Framework configuration system, the Web.config file for an application that runs in IIS 7.0 contains both Web server and ASP.NET configuration settings. For example, in a Web.config file for an ASP.NET application running in IIS 7.0, you can specify the default file to return when a browser does not request a specific file. (In IIS 6.0 and earlier versions of IIS, this was a setting that was maintained in the IIS metabase.)

**Editing Web.config Files**

You can change the Web.config file for a Web application that is running in IIS 7.0 in the following ways:

* Editing the Web.config file directly, either by using Visual Studio or Visual Web Developer, or by using a text editor program.
* Using IIS Manager. For more information, see [Internet Information Services (IIS) Manager](http://go.microsoft.com/fwlink/?LinkId=92312).
* Using the ASP.NET Web Site Administration Tool. For more information, see [ASP.NET Web Site Administration Tool](http://msdn.microsoft.com/en-us/library/ms228053.aspx).

|  |
| --- |
| **NoteNote** |
| Changes made in the Web Site Administration Tool do not affect child configuration elements in the **system.webServer** element. |

* Using the IIS 7.0 command-line tool (Appcmd.exe). This utility enables you to specify IIS configuration settings and Web application configuration settings at the command line. For more information, see [IIS 7.0 Command-Line Tool](http://go.microsoft.com/fwlink/?LinkId=92315).

**The system.webServer Section**

The **system.webServer** configuration section in the Web.config file specifies IIS 7.0 settings that are applied to the Web application. The **system.WebServer** section is a child of the [configuration](http://msdn.microsoft.com/en-us/library/ms228147.aspx) element. For more information, see [IIS 7.0: system.webServer Section Group (IIS Settings Schema)](http://go.microsoft.com/fwlink/?LinkId=92335).

Examples of Web server settings that you can set in the **system.WebServer** configuration group include the following:

* The default document that the Web server returns to a client when request does not include a specific resource (**defaultDocument** element).
* The compression settings for responses (**httpCompression** element).
* Custom headers (**customHeaders** element of the **httpProtocol** section).
* Modules (**modules** element).
* Handlers (**handlers** element).

Some settings apply only to IIS 7.0 Integrated mode and do not apply to Classic mode. For example, if the application is running in Classic mode, any managed-code modules and handlers specified in the **system.WebServer** section of the Web.config file are ignored. Instead, the managed-code modules and handlers must be defined as in earlier versions of IIS, by using the [httpModules](http://msdn.microsoft.com/en-us/library/9b9dh535.aspx) and [httpHandlers](http://msdn.microsoft.com/en-us/library/bya7fh0a.aspx) elements of the [system.web](http://msdn.microsoft.com/en-us/library/dayb112d.aspx) section.

For examples of using the **system.webServer** configuration section, see [How to: Configure the <system.webServer> Section for IIS 7.0](http://msdn.microsoft.com/en-us/library/bb763179.aspx).

http://i.msdn.microsoft.com/Global/Images/clear.gif Moving a Web Application to Classic Mode

Typically, moving a Web application from IIS 6.0 to IIS 7.0 Classic mode requires only that you put the application in an application pool that is running in Classic mode. For example, when you install IIS 7.0 with , by default the Web server is configured to operate in Integrated mode. It is also configured to run under the default application pool, which is called **DefaultAppPool**. To run a Web application in Classic mode, use the **Classic.NETAppPool** application or create a new application pool that is configured to run in Classic mode. For information about how to create an application pool, see [Create an Application Pool](http://go.microsoft.com/fwlink/?LinkId=92336).

Any custom modules that implement the [IHttpModule](http://msdn.microsoft.com/en-us/library/system.web.ihttpmodule.aspx) interface in an application that runs in Classic mode are notified only about pipeline requests that are handled by the ASP.NET runtime. For example, they are notified about requests for an .aspx page. The application life cycle for Classic mode is the same as the life cycle for ASP.NET in IIS 6.0. For more information, see [ASP.NET Application Life Cycle Overview for IIS 5.0 and 6.0](http://msdn.microsoft.com/en-us/library/ms178473.aspx).

If an application that runs in Classic mode contains a handler that requires a script map to handle a custom extension in IIS, you must register the handler in both the **httpHandler** group and the **handler** group. You map the custom file-name extension to the ASP.NET ISAPI extension (Aspnet\_isapi.dll) by specifying the **modules** and **scriptProcessor** attributes in the **handler** element. These attributes specify that the module that defines the handler is an ISAPI extension, and they specify the path of that extension. This is how IIS 7.0 in Classic mode provides backward compatibility with earlier versions of IIS. However, if you run the application in Integrated mode, you must remove the **modules** and **scriptProcessor** attributes. For more information, see [How to: Configure an HTTP Handler Extension in IIS](http://msdn.microsoft.com/en-us/library/bb515343.aspx).

When you move a Web application from IIS 6.0 to Classic mode, it is not guaranteed to work in Integrated mode without changes. If you switch an application from Classic mode to Integrated mode (and change any custom modules and handlers), you might have to make additional changes for the application to run correctly in Integrated mode. The next section in this topic explains how to move an application to IIS 7.0 Integrated mode.

http://i.msdn.microsoft.com/Global/Images/clear.gif Moving a Web Application to Integrated Mode

Web applications that do not include custom modules or handlers will usually work without changes in Integrated mode in IIS 7.0. Web applications that rely on custom modules or handlers require the following steps to enable the application to run in Integrated mode:

* Register custom modules and handlers in the **system.webServer** section of the Web.config file by using one of the methods described in the [Migrating a Web Config File to Integrated Mode](http://msdn.microsoft.com/en-us/library/bb515251.aspx#MIGRATING) section later in this topic.
* Define event handlers for [HttpApplication](http://msdn.microsoft.com/en-us/library/system.web.httpapplication.aspx) request pipeline events like [BeginRequest](http://msdn.microsoft.com/en-us/library/system.web.httpapplication.beginrequest.aspx) and [EndRequest](http://msdn.microsoft.com/en-us/library/system.web.httpapplication.endrequest.aspx) only in the [Init](http://msdn.microsoft.com/en-us/library/system.web.ihttpmodule.init.aspx) method of the custom module.
* Make sure that you have addressed any issues discussed in the "Known Differences Between Integrated Mode and Classic Mode" section of [Upgrading ASP.NET Applications to IIS 7.0: Differences between IIS 7.0 Integrated Mode and Classic mode](http://go.microsoft.com/fwlink/?LinkId=86952).

Modules that implement the [IHttpModule](http://msdn.microsoft.com/en-us/library/system.web.ihttpmodule.aspx) interface are referred to as managed-code modules because they are built by using the .NET Framework. Managed-code modules can be registered at the server level or at the application level. Native-code modules are DLLs (non-managed code) that are registered only at the server level. Core ASP.NET features, such as session state and forms authentication, are implemented as managed modules in Integrated mode.

When you move an application from Classic to Integrated mode, you can leave custom modules and handler registrations for Classic mode, or you can remove them. If you do not remove the [httpModules](http://msdn.microsoft.com/en-us/library/9b9dh535.aspx) and [httpHandlers](http://msdn.microsoft.com/en-us/library/bya7fh0a.aspx) registrations that are used in Classic mode, you must set the **validation** element’s **validateIntegratedModeConfiguration** attribute to **false** to avoid errors. The **validation** element is a child element of the **system.webServer** element. For more information, see the section "Disabling the migration message" in [ASP.NET Integration with IIS 7.0](http://go.microsoft.com/fwlink/?LinkId=88850).

**Migrating a Web.config File for Use in Integrated Mode**

If a module or handler is defined at the application level, the module or handler is not automatically invoked. This includes modules or handlers that are defined in an assembly in the Bin folder or as source code in the App\_Code folder, and that are not registered defined in the **system.webServer** section of the Web.config file. In order for the module or handler to participate in the Integrated mode request pipeline, you must register the module or handler by using one of the following methods:

* Edit the Web.config directly and add the **modules** or **handlers** element to the **system.webServer** element. Note the difference from Classic mode in the element name—**modules** compared to [httpModules](http://msdn.microsoft.com/en-us/library/9b9dh535.aspx) and handlers compared to [httpHandlers](http://msdn.microsoft.com/en-us/library/bya7fh0a.aspx).
* Use IIS Manager to configure the module or handler. For more information, see [Configuring Handler Mappings in IIS 7.0](http://go.microsoft.com/fwlink/?LinkId=92338) and [Configuring Modules in IIS 7.0](http://go.microsoft.com/fwlink/?LinkId=92339).
* Use the IIS 7.0 command-line tool (Appcmd.exe). For more information, see [Configure Settings for a Site, Application, Virtual Directory or URL by Using Appcmd.exe](http://go.microsoft.com/fwlink/?LinkId=92340).

**Classes and Properties for Working with Integrated Mode**

When you work with an application in IIS 7.0 Integrated mode and the .NET Framework version 3.0 or later, you can use the following classes and members that are not available in Classic mode:

* The [SubStatusCode](http://msdn.microsoft.com/en-us/library/system.web.httpresponse.substatuscode.aspx) property of the [HttpResponse](http://msdn.microsoft.com/en-us/library/system.web.httpresponse.aspx) object, which lets you set codes that are useful when failed request tracing is configured. For more information, see [Troubleshooting Failed Requests Using Failed Request Tracing in IIS 7.0](http://go.microsoft.com/fwlink/?LinkId=87359).
* The [Headers](http://msdn.microsoft.com/en-us/library/system.web.httpresponse.headers.aspx) property of the [HttpResponse](http://msdn.microsoft.com/en-us/library/system.web.httpresponse.aspx) object, which provides access to response headers.
* The [IsPostNotification](http://msdn.microsoft.com/en-us/library/system.web.httpcontext.ispostnotification.aspx) and [CurrentNotification](http://msdn.microsoft.com/en-us/library/system.web.httpcontext.currentnotification.aspx) properties of the [HttpContext](http://msdn.microsoft.com/en-us/library/system.web.httpcontext.aspx) object, which are used when you provide handlers for [HttpApplication](http://msdn.microsoft.com/en-us/library/system.web.httpapplication.aspx) events.
* The [Headers](http://msdn.microsoft.com/en-us/library/system.web.httprequest.headers.aspx) and [ServerVariables](http://msdn.microsoft.com/en-us/library/system.web.httprequest.servervariables.aspx) properties of the [HttpRequest](http://msdn.microsoft.com/en-us/library/system.web.httprequest.aspx) object, which are write-enabled.