1

IIS 7 and ASP.NET Integrated Architecture

Internet Information Services 7.0 (IIS 7) is the latest version of Microsoft Web server. IIS 7 has gone through significant architectural changes since the last version. The most notable change for ASP.NET developers is the deep integration of the IIS 7 and ASP.NET framework. This provides both ASP.NET developers and IIS 7 administrators with an integrated programming environment that allows them to implement features and functionalities that were not possible before. The main goal of this chapter is twofold. First, it covers the IIS 7 package updates and their constituent feature modules, discusses five different IIS 7 setup options, and shows you how to use each option to custom-build your own Web server from these package updates. Second, it provides you with an overview of the IIS 7 and ASP.NET integrated architecture and its constituent systems, setting the stage for the next chapters where you'll dive into the details of this integrated architecture and programming framework.

Modular Architecture of IIS 7

The main priority of the Microsoft IIS team for IIS 6.0 was to improve its security, performance, and reliability. For that reason, modularity and extensibility didn't make it to the list of top priorities for IIS 6.0. That said, IIS 6.0 introduced a very important notion: selectively disabling IIS 7 features such as ISAPI extensions and CGI components. One of the main problems with the earlier versions of IIS was that all features of IIS had to be installed and enabled. There was no way to disable features that your application scenario did not need.

IIS 6.0 enables only static file serving by default on a clean install of the Web server. In other words, dynamic features such as ISAPI extensions and CGI components are disabled by default unless the administrator explicitly enables them. Such customization of the Web server allows you to decrease the attack surface of your Web server by giving attackers fewer opportunities for attacks.

Disabling unwanted features was the first step toward the customizability of IIS. However, this step didn't go far enough because IIS 6.0 still installs everything, which introduces the following problems:

- Disabled features consume server resources such as memory, and therefore increase the Web server footprint.
- Administrators still need to install service packs that address bugs in the disabled features, even though they're never used.
- Administrators still need to install software updates for the disabled features.

In other words, administrators have to maintain the service features that are never used. All these problems stem from the fact that the architecture of IIS 6.0 is relatively monolithic. The main installation problem with a monolithic architecture is that it's based on an all-or-nothing paradigm where you have no choice but to install the whole system.

IIS 7.0 is modular to the bone! Its architecture consists of more than 40 feature modules from which you can choose. This allows you to install only the needed feature modules to build a highly customized, thin Web server. This provides the following important benefits:

- Decreases the footprint of your Web server.
- Administrators need to install only those service packs that address bugs in the installed feature modules.
- Administrators need to install software updates only for the installed feature modules.

So, administrators have to maintain and service only installed feature modules.

Next, I provide an overview of the IIS 7 feature modules or components. These feature components are grouped into what are known as *functional areas*, where each functional area maps to a specific IIS package update. That is, each package update contains one or more feature modules or components. As you'll see later, you'll use these package updates to custom-build your Web server.

The top-level IIS update is known as IIS-WebServerRole, and contains the updates shown in Figure 1-1. As the name suggests, the IIS-WebServerRole update enables Windows Server 2008 and Windows Vista to adopt a Web server role, which enables them to exchange information over the Internet, an intranet, or an extranet.

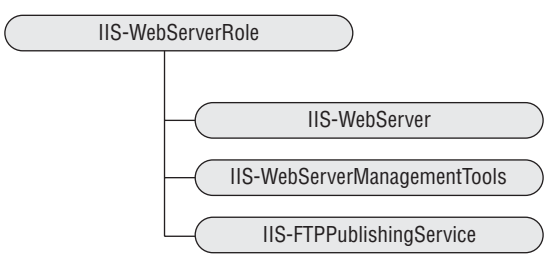

Figure 1-1

IIS-WebServer

The IIS-WebServer update contains five updates as shown in Figure 1-2. As you can see, this update contains the feature modules that make up the core functionality of a Web server.

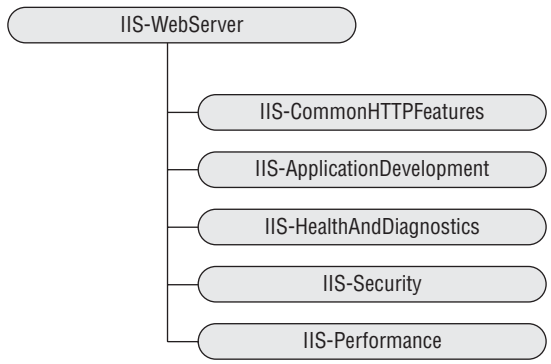

Figure 1-2

IIS-CommonHttpFeatures

The IIS-CommonHttpFeatures update contains the feature modules or components described in the following table:

Feature Module	Description
IIS-StaticContent	Use this module to enable your Web server to service requests for static content. Web site resources with file extensions such as `.html`, `.htm`, `.jpg`, and the like that can be serviced without server-side processing are known as static content.
IIS-DefaultDocument	This module allows you to specify a Web resource that will be used as the default resource when the request URL does not contain the name of the requested resource.
IIS-DirectoryBrowsing	Use this module to enable your Web server to display the contents of a specified directory to end users when they directly access the directory and no default document exists in the directory.
IIS-HttpErrors	Use this module to enable your Web server to support sending custom error messages to end users.
IIS-HttpRedirect	Use this module to enable your Web server to support request redirects.

IIS-ApplicationDevelopment

The IIS-ApplicationDevelopment update contains the feature modules that support different application types as described in the following table:

Feature Module	Description
IIS-ASPNET	Use this module to enable your Web server to host ASP.NET applications.
IIS-NetFxExtensibility	Use this module to enable your Web server to host managed modules.
IIS-ASP	Use this module to enable your Web server to host ASP applications.
IIS-CGI	Use this module to enable your Web server to support CGI executables.
IIS-ISAPIExtensions	Use this module to enable your Web server to use ISAPI extension modules to process requests.
IIS-ISAPIFilter	Use this module to enable your Web server to use ISAPI filter to customize the server behavior.
IIS-ServerSideIncludes	Use this module to enable your Web server to support `.stm`, `.shtm`, and `.shtml` include files.

IIS-HealthAndDiagnostics

The IIS-HealthAndDiagnostics package update contains the feature modules described in the following table:

Feature Module	Description
IIS-HttpLogging	Use this module to enable your Web server to log Web site activities.
IIS-LoggingLibraries	Use this module to install logging tools and scripts on your Web server.
IIS-RequestMonitor	Use this module to enable your Web server to monitor the health of the Web server and its sites and applications.
IIS-HttpTracing	Use this module to enable your Web server to support tracing for ASP.NET applications and failed requests.
IIS-CustomLogging	Use this module to enable your Web server to support custom logging for the Web server and its sites and applications.
IIS-ODBCLogging	Use this module to enable your Web server to support logging to an ODBC-compliant database.

IIS-Security

The IIS-Security package update contains the feature modules described in the following table:

Security Feature Module	Description
IIS-BasicAuthentication	Use this module to enable your Web server to support the HTTP 1.1 Basic Authentication scheme. This module authenticates user credentials against Windows accounts.
IIS-WindowsAuthentication	Use this module to enable your Web server to authenticate requests using NTLM or Kerberos.
IIS-DigestAuthentication	Use this module to enable your Web server to support the Digest authentication scheme. The main difference between Digest and Basic is that Digest sends password hashes over the network as opposed to the passwords themselves.
IIS-ClientCertificateMappingAuthentication	Use this module to enable your Web server to authenticate client certificates with Active Directory accounts.
IIS-IISCertificateMappingAuthentication	Use this module to enable your Web server to map client certificates 1-to-1 or many-to-1 to a Windows security identity.
IIS-URLAuthorization	Use this module to enable your Web server to perform URL authorization.
IIS-RequestFiltering	Use this module to enable your Web server to deny access based on specified configured rules.
IIS-IPSecurity	Use this module to enable your Web server to deny access based on domain name or IP address.

IIS-Performance

The following table describes the performance feature modules:

Performance Feature Module	Description
IIS-HttpCompressionStatic	Use this module to enable your Web server to compress static content before sending it to the client to improve the performance.
IIS-HttpCompressionDynamic	Use this module to enable your Web server to compress dynamic content before sending it to the client to improve the performance.

IIS-WebServerManagementTools

Figure 1-3 presents the Web server management feature modules.

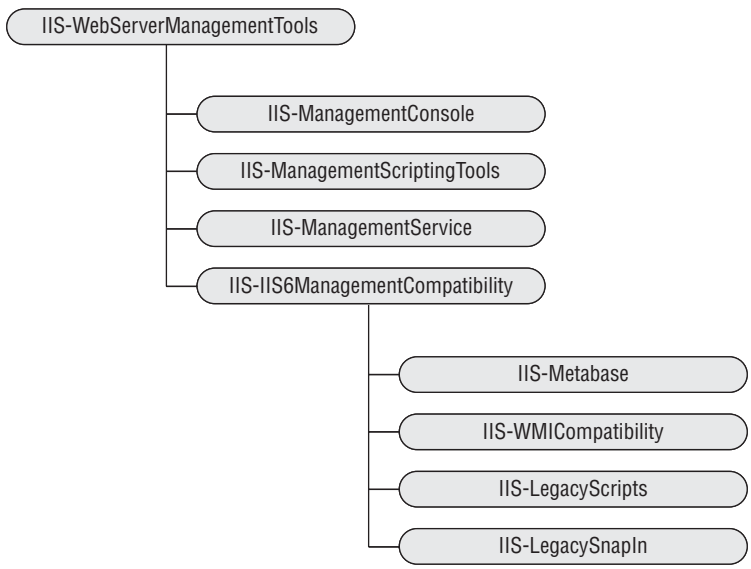

Figure 1-3

The following table describes the feature modules contained in the IIS-WebServerManagementTools update:

Feature Module	Description
IIS-ManagementConsole	This module installs the Web Server Management Console, which allows administration of local and remote IIS Web servers.
IIS-ManagementScriptingTools	Use this module to enable your Web server to support local Web server management via IIS configuration scripts.
IIS-ManagementService	Use this module to enable your Web server to be managed remotely via Web Server Management Console.

The following table presents the feature modules in the IIS-IIS6ManagementCompatibility update:

Feature Module	Description
IIS-Metabase	Use this module to enable your Web server to support metabase calls to the new IIS 7 configuration store.
IIS-WMICompatibility	Use this module to install the IIS 6.0 WMI scripting interfaces to enable your Web server to support these interfaces.
IIS-LegacyScripts	Use this module to install the IIS 6.0 configuration scripts to enable your Web server to support these scripts.
IIS-LegacySnapIn	Use this module to install the IIS 6.0 Management Console to enable administration of remote IIS 6.0 servers from this computer.

IIS-FTPPublishingService

The feature modules contained in the IIS-FTPPublishingService package update are discussed in the following table.

At the time of this writing, Microsoft announced that it would be releasing a significantly enhanced IIS 7 FTP server for Longhorn and (as a separate download) for Vista.

Feature Module	Description
IIS-FTPServer	Use this module to install the FTP service.
IIS-FTPManagement	Use this module to install the FTP Management Console.

WAS-WindowsActivationService

Figure 1-4 presents the feature modules in the WAS-WindowsActivationService package update. These modules provide the base infrastructure for process activation and management.

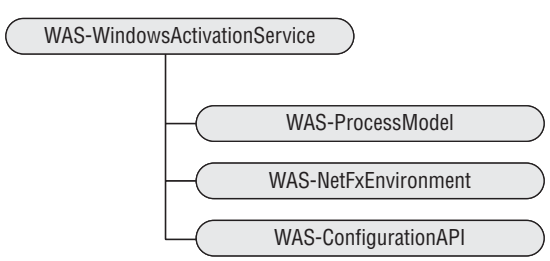

Figure 1-4

Extensible Architecture of IIS 7

IIS 6.0 allows you to extend the functionality of the Web server by implementing and plugging in your own custom ISAPI filter and extension modules. Unfortunately, ISAPI suffers from fundamental problems such as:

- Because ISAPI is not a convenient or friendly API, writing an ISAPI filter or extension module is not an easy task to accomplish. It can take a lot of time and tends to be error-prone.
- ISAPI is not a managed API, which means that ASP.NET developers cannot benefit from the rich features of the .NET Framework when they're writing ISAPI filter and extension modules.

IIS 7.0 has replaced ISAPI with a new set of convenient object-oriented APIs that make writing new feature modules a piece of cake. These APIs come in two different flavors: managed and native. The native API is a convenient C++ API that you can use to develop and plug native modules into the core Web server. The managed API, on the other hand, allows you to take full advantage of the .NET Framework and its rich environment. This allows both ASP.NET developers and IIS 7 administrators to use convenient ASP.NET APIs to extend the core Web server.

IIS 7 and ASP.NET Integrated Request Processing Pipeline

Take a look at the request processing model of IIS 6.0 for processing requests for ASP.NET content as shown in Figure 1-5. Notice that this figure contains two different request processing pipelines: IIS 6.0 and ASP.NET. Each request processing pipeline is a pipeline of components that are invoked one after another to perform their specific request processing tasks. For example, both pipelines contain an authentication component, which is called to authenticate the request.

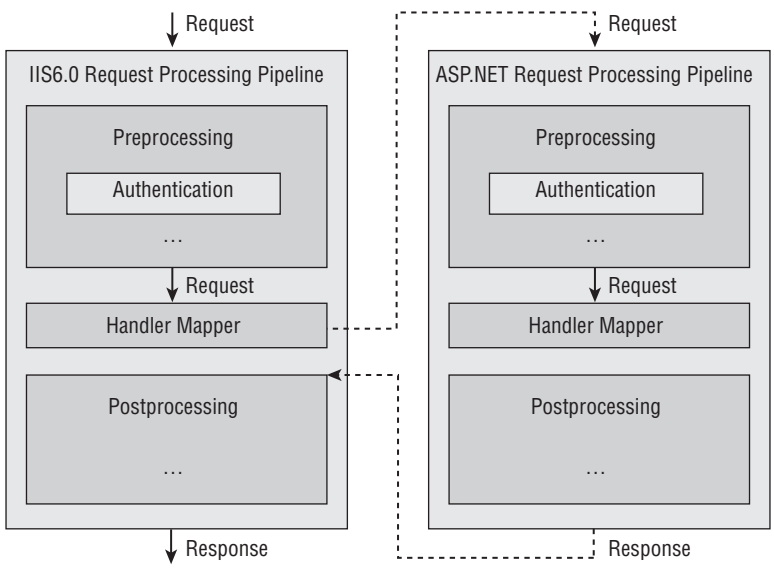

Figure 1-5

As Figure 1-5 shows, the incoming request first goes through the IIS 6.0 pipeline. At some point along this pipeline, IIS 6.0 uses its metabase to map the request to a particular handler. The requests for ASP.NET resources such as ASP.NET pages are mapped to the `aspnet_isapi.dll` handler. This handler then loads the CLR and the target ASP.NET application, if they haven't already been loaded. This is where the ASP.NET request processing pipeline kicks in.

At the beginning of the request, ASP.NET allows the components in its request processing pipeline to register one or more event handlers for one or more ASP.NET application-level events. ASP.NET then fires these events one after another and calls these event handlers to allow each component to perform its specific request processing task. At some point along the pipeline, ASP.NET uses the configuration file to map the request to a particular handler. The main responsibility of the handler is to process the request and generate the appropriate markup text, which will then be sent back to the requesting browser.

Having two separate pipelines, that is, IIS and ASP.NET pipelines working on the same request, introduces the following problems:

- There's a fair amount of duplication. For example, both pipelines contain an authentication component, which means that the same request gets authenticated twice.
- Because the ASP.NET pipeline begins after the IIS pipeline maps the request to the `aspnet_isapi` extension module, the ASP.NET pipeline has no impact on the IIS pipeline steps prior to handler mapping.
- Because the rest of the IIS pipeline steps don't occur until the ASP.NET pipeline finishes, the ASP.NET pipeline has no impact on the IIS pipeline steps either.
- Because the ASP.NET pipeline kicks in only when the IIS pipeline maps the request to the `aspnet_isapi` extension module, and because this mapping is done only for requests to ASP.NET content, the ASP.NET pipeline components cannot be applied to requests to non-ASP.NET content such as `.jpg`, `.js`, `asp`, CGI, and the like. For example, you cannot use the ASP.NET authentication and authorization modules to protect the non-ASP.NET contents of your application.

IIS 7 has changed all that by removing the `aspnet_isapi` extension module and combining the ASP.NET and IIS pipelines into a single integrated request processing pipeline as shown in Figure 1-6.

This resolves all the previously mentioned problems as follows:

- The integrated pipeline does not contain any duplicate components. For example, the request is authenticated once.
- The ASP.NET modules are now first-class citizens in the integrated pipeline. They can come before, replace, or come after any native IIS 7 modules. This allows ASP.NET to intervene at any stage of the request processing pipeline.
- Because the integrated pipeline treats managed modules like native modules, you can apply your ASP.NET managed modules to non-ASP.NET content. For example, you can use the ASP.NET authentication and authorization modules to protect the non-ASP.NET contents of your application, such as `asp` pages.

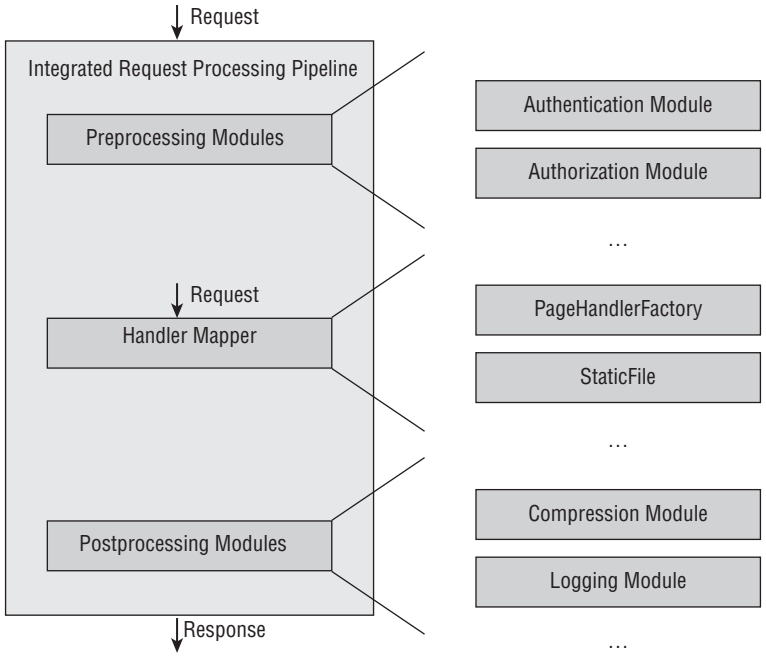

Figure 1-6

IIS 7 and ASP.NET Integrated Configuration Systems

In IIS 6.0, two separate configuration systems govern the IIS and ASP.NET pipelines. These configuration systems store their configuration settings in two different storage media, with two different schemas. IIS configuration settings are stored in the IIS 6.0 metabase, whereas ASP.NET configuration settings are stored in ASP.NET configuration files. Such separation of configuration systems makes the task of administering the Web server and its sites and applications much more complex and troublesome. For one thing, there's no way to delegate site- and application-specific IIS configuration settings to site and application administrators without compromising the integrity and security of the Web server, because all IIS configuration settings are centralized. This also takes away from the ASP.NET developers the opportunity to tailor the IIS configuration settings toward their own applications. Having two separate configuration systems for IIS and ASP.NET configuration settings also means that you have to learn two separate APIs to programmatically access and edit these configuration settings.

IIS 7 has changed all that. Having a single integrated pipeline made it possible for the IIS 7 team to introduce a single integrated configuration system for both IIS and ASP.NET settings. Because this integrated configuration system is an extension of the ASP.NET configuration system, the existing ASP.NET configuration files can easily merge into the new integrated configuration system with a little or no changes.

This integrated configuration system provides a lot of benefits to system administrators and developers alike. For one thing, both IIS and ASP.NET configuration settings are stored in storage media with the same schema. This is great news for ASP.NET developers because the new integrated schema is an extension of the ASP.NET configuration schema. Another obvious benefit of the integrated configuration system is that you can use the same set of APIs to programmatically access and set both IIS 7 and ASP.NET configuration settings.

One of the great new features of the IIS 7 and ASP.NET integrated configuration system is its declarative extensibility through a new integrated declarative schema extension markup language. Thanks to this integrated markup language, you can extend this integrated configuration system to add support for new configuration sections without writing a single line of imperative code such as C# or VB. This is a departure from the imperative extensibility model of the ASP.NET configuration system, which requires developers to write a fair amount of imperative code to extend the system.

IIS 7 and ASP.NET Integrated Administration

Having two separate configuration systems for ASP.NET and IIS in IIS 6.0 also means having two separate administration tools, GUIs, and APIs to administer and to manage them. Having a single integrated configuration system made it possible for the IIS 7 team to introduce brand new administration or management tools, GUIs, and APIs that make the task of server, site, and application administration a whole lot easier. This allows you to use the same integrated management tools, GUIs, and APIs to configure ASP.NET and IIS.

Two very important components of the IIS 7 and ASP.NET integrated administration are the integrated graphical management system and the integrated imperative management system. This book covers both of these systems and their extensibility models in detail. You will learn how to extend these two systems to add graphical and imperative management support for your own custom configuration sections.

Building a Customized Web Server

IIS 7 setup is completely modular, allowing you to custom-build your Web server from a list of more than 40 available feature modules. This ensures that your Web server contains only the feature modules you need, thereby decreasing the attack surface and footprint of your server. In this section, I walk you through the steps that you need to take to build your very own custom Web server on Windows Vista (including Windows Vista Home Premium, Windows Vista Professional, and Windows Vista Ultimate editions) and Windows Server 2008 operating systems.

In general, there are five different IIS 7 setup options:

- Windows Features dialog (Windows Vista only)
- Server Manager tool (Windows Server 2008 only)
- `pkgmgr.exe` command-line tool (both Windows Vista and Windows Server 2008)

- Unattended (both Windows Vista and Windows Server 2008)
- Upgrade (both Windows Vista and Windows Server 2008)

Before drilling down into the details of these five setup options, you need to understand the dependencies between the installable updates.

Update Dependencies

When you're installing an update, you must also install the updates that it depends on. In general, there are two types of dependencies: interdependencies and parent-dependencies. The following table presents the update interdependencies:

Update	Depends On
IIS-WebServer	WAS-ProcessModel
IIS-ASP	IIS-ISAPIExtensions IIS-RequestFiltering
IIS-ASPNET	IIS-DefaultDocument IIS-NetFxExtensibility WAS-NetFxEnvironment IIS-ISAPIExtensions IIS-ISAPIFilter IIS-RequestFiltering
IIS-NetFxExtensibility	WAS-NetFxEnvironment IIS-RequestFiltering
IIS-ManagementService	IIS-WebServer IIS-ManagementConsole WAS-NetFxEnvironment WAS-ConfigurationAPI
IIS-ManagementConsole	WAS-ConfigurationAPI
IIS-ManagementScriptingTools	WAS-ConfigurationAPI
IIS-LegacyScripts	IIS-Metabase IIS-WMICompatibility

Every update also depends on its parent update. For example, to install IIS-WebServer, you must also install its parent update, IIS-WebServerRole.

Windows Features Dialog

Follow these steps to use the Windows Features dialog to set up and custom-build your Web server on Windows Vista:

1. Launch the Control Panel.
2. Click the "Programs" option if Control Panel is displayed in its default view, or the "Programs and Features" option if Control Panel is displayed in Classic View.
3. Click "Turn on or off Windows features" to launch the Windows Features dialog shown in Figure 1-7. If you haven't logged in as the built-in Administrator account, Vista will launch the User Account Control dialog. The content of this dialog depends on the privileges of your account. If your account has administrator privileges, the dialog will just ask you for confirmation. If your account does not have administrator privileges, the dialog will present you with the list of accounts with administrator privileges asking you to choose one and enter the required password. Keep in mind that you'll get this dialog even if you have logged in as an account that has administrator privileges. This is one of the new security features.

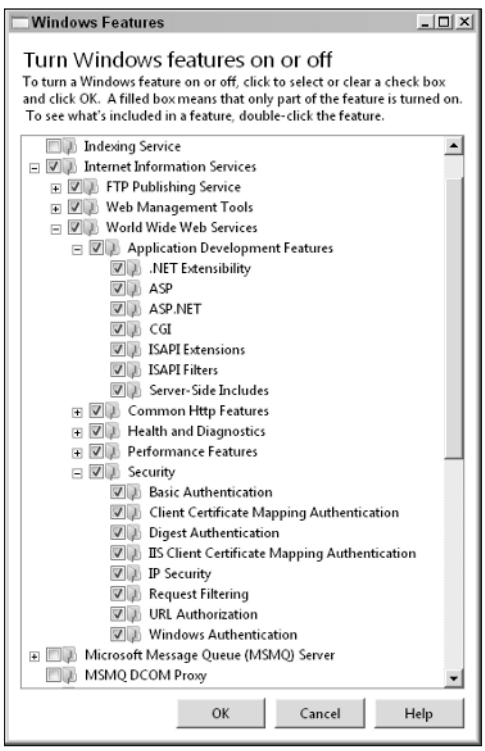

Figure 1-7

4. Expand the Internet Information Services option to see the tree of update nodes discussed in the previous sections. You can install or uninstall each update by simply toggling it on or off and finally clicking the OK button. Notice that when you select an update, its parent update and the update that it depends on are automatically selected.

As you can see, building your own custom Web server with the Windows Features dialog is a piece of cake. You don't have to worry about the update dependencies; it's all taken care of behind the scenes. As you'll see in the following section, you don't have this luxury if you use the other two IIS 7 installation options.

Server Manager

In this section, I show you how to use the Server Manager tool to build your customized Web server on the Windows Server 2008 operating system. Before doing so, you need to familiarize yourself with three basic Windows Server 2008 terms known as *roles*, *role services*, and *features*.

Every server provides its clients with a set of services. These services are grouped into what are known as roles. Installing a server role means installing one or more role services that belong to the role. In other words, when you're installing a server role, you don't have to install all its associated role services.

Here is an example: There is a server role known as Web Server, which enables a server to exchange information over the Internet, an intranet, or an extranet. Another example of a server role is UDDI Services. This role enables a server to provide its clients with Universal Description, Discovery, and Integration (UDDI) services to exchange information about Web services over the Internet, an intranet, or an extranet.

A feature is a piece of software that does not belong to any particular role, but it provides services to one or more server roles and their associated role services. An example of a feature is the Windows Process Activation Service. This service enables the server in the Web Server role to process requests made through all kinds of communication protocols, such as TCP or HTTP.

A role, role service, or feature may depend on other roles, role services, and features. For example, the UDDI Services depend on the Web Server role for the actual exchange of information over the Internet, intranet, or extranet. When you attempt to install a role, role service, or feature that depends on other roles, role services, and features, the Server Manager prompts you to approve the installation of the roles, role services, and features on which the role, role service, or feature being installed depends.

Now back to the business at hand, which is building a customized Web server. Take one of the following steps to launch the Server Manager:

- Select Start ➪ All Programs ➪ Administrative Tools ➪ Server Manager from Administrative Tools to launch the Server Manager tool shown in Figure 1-8.
- First launch the Control Panel, double-click the Administrative Tools icon in the Control Panel, and then double-click the Server Manager to launch the Server Manager tool shown in Figure 1-8.

As Figure 1-8 shows, the left pane contains a node named Server Manager, which in turn contains a child node named Roles. As just discussed, a server can be in one or more roles. As you can see from Figure 1-8, in a clean install of Windows Server 2008 the server is originally in no roles. The role that you're interested in is the Web Server role. Recall that this is the role that allows the server to share information on the Internet, an intranet, or an extranet. The first order of business is to launch a wizard named Add Roles to add this role to your server.

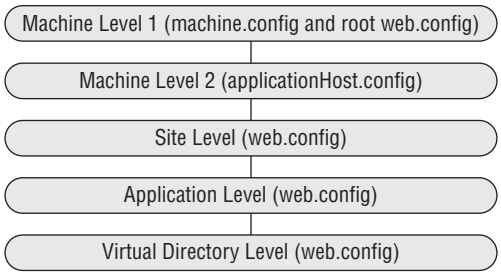

Figure 1-8

To launch the Add Roles Wizard, do one of the following:

- Click the Add Roles link button in Roles Summary panel.
- Right-click the Roles node in the Server Manager panel and select Add Roles.
- Click the Action menu and select Add Roles.

The first page of the Add Roles Wizard provides you with some preliminary instruction. Read the instructions and make sure your account meets the specified requirements as shown in Figure 1-9.

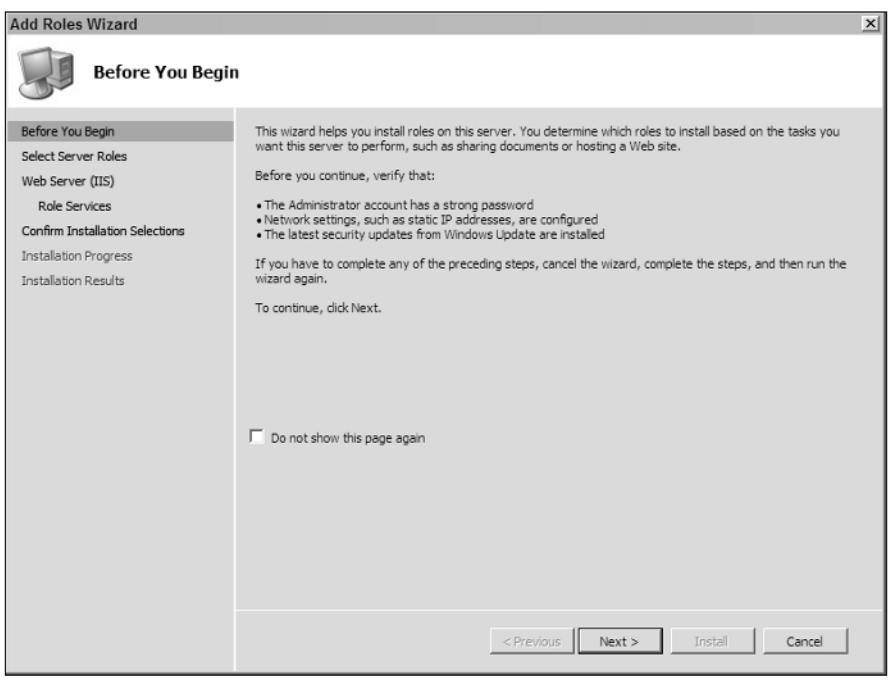

Figure 1-9

Click the Next button to go to the page shown in Figure 1-10.

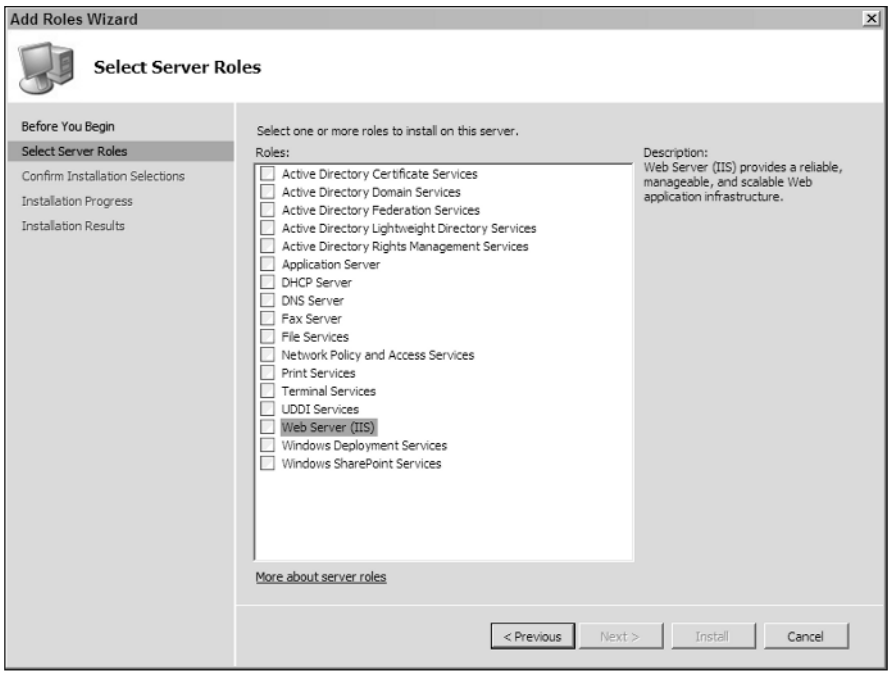

Figure 1-10

Check the Web Server (IIS) item shown in Figure 1-10. It should show you the popup shown in Figure 1-11 informing you that you need to install the Windows Process Activation Service. Click the Add Required Features button on this popup to install the Windows Process Activation Service.

Now click Next to go the next page, which provides some preliminary information. Click Next again to go to the page shown in Figure 1-12.

Notice that some package updates are already selected. These updates form the default installation of the Web server. Note that when you turn on an update that depends on other updates, the Server Manager tool pops up a message showing the updates on which the selected update depends and informing you that you need to install the dependent updates as well. For example, when you check the ASP.NET option, the Server Manager pops up the message shown in Figure 1-13.

After you're done with toggling on the desired updates, click the Next button in Figure 1-13 to move on to the confirmation page shown in Figure 1-14, which lists all the selected updates and their dependent updates. At this point these updates have not been installed yet.

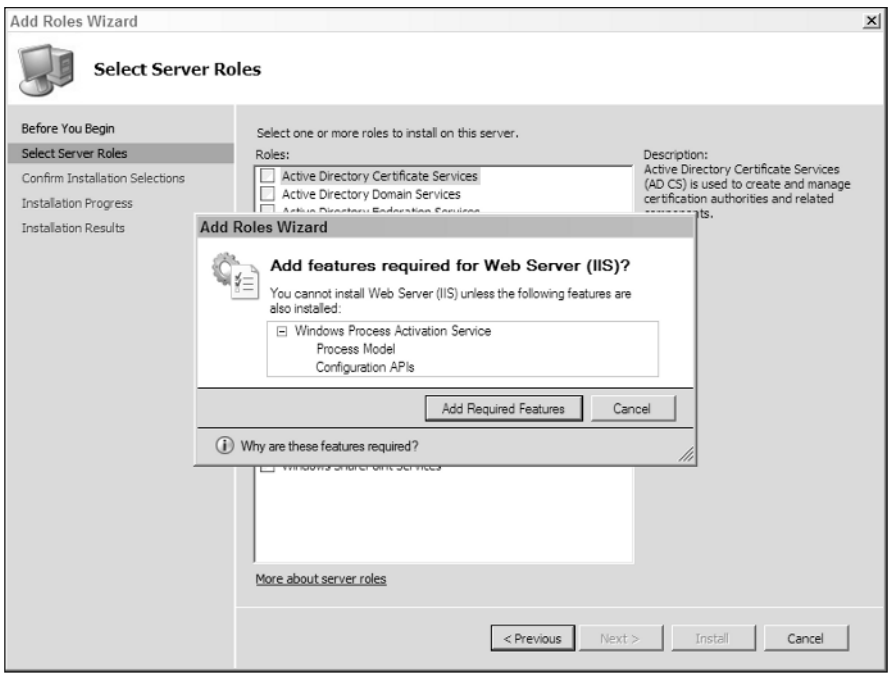

Figure 1-11

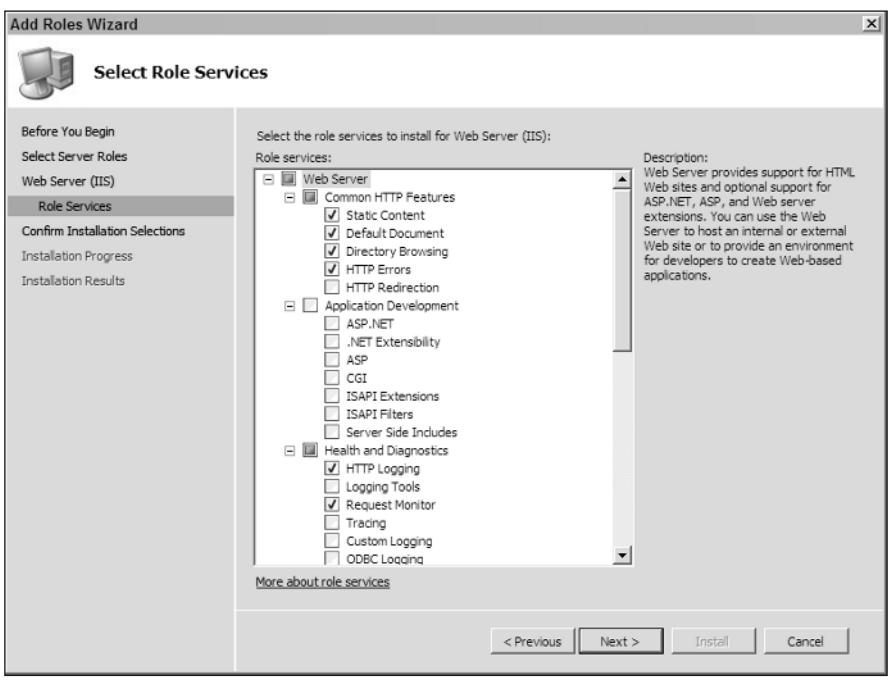

Figure 1-12

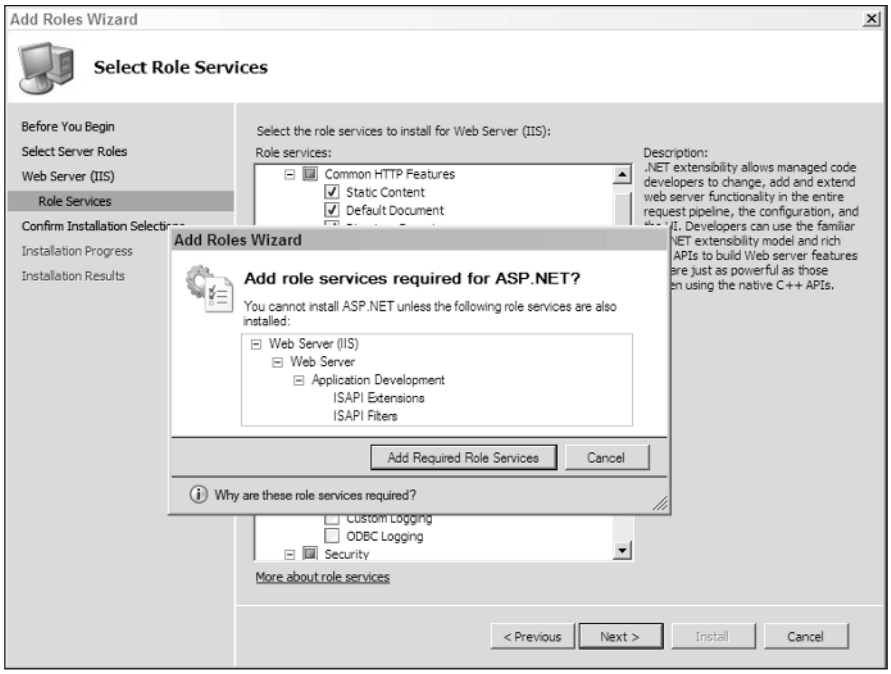

Figure 1-13

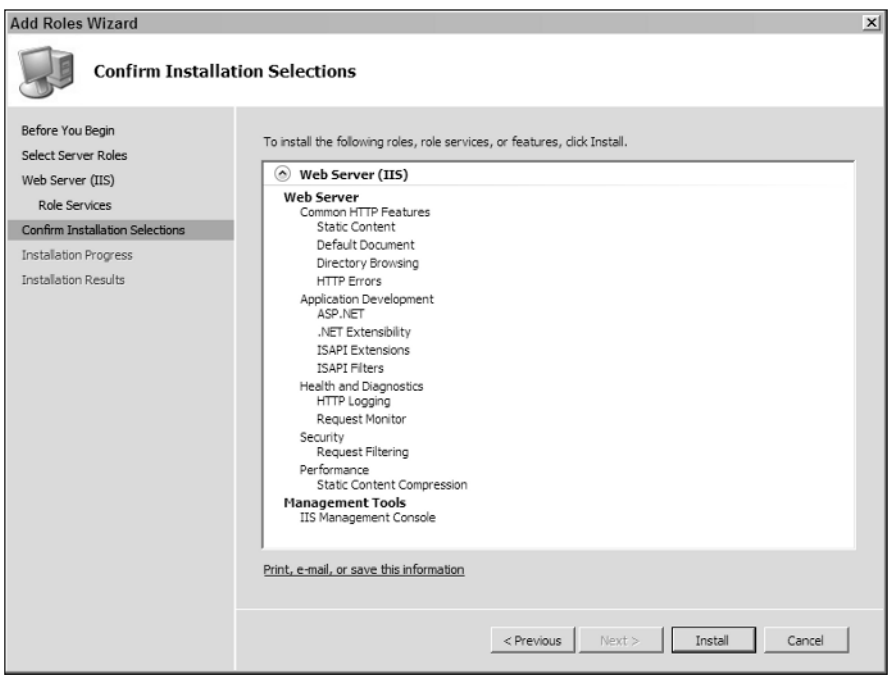

Figure 1-14

Click the Install button in Figure 1-14 to have Server Manager install the specified updates. This will take you to the progress page where you have to wait for a while for the updates to be installed. When the installation completes, the Add Roles Wizard automatically takes you to the page shown in Figure 1-15.

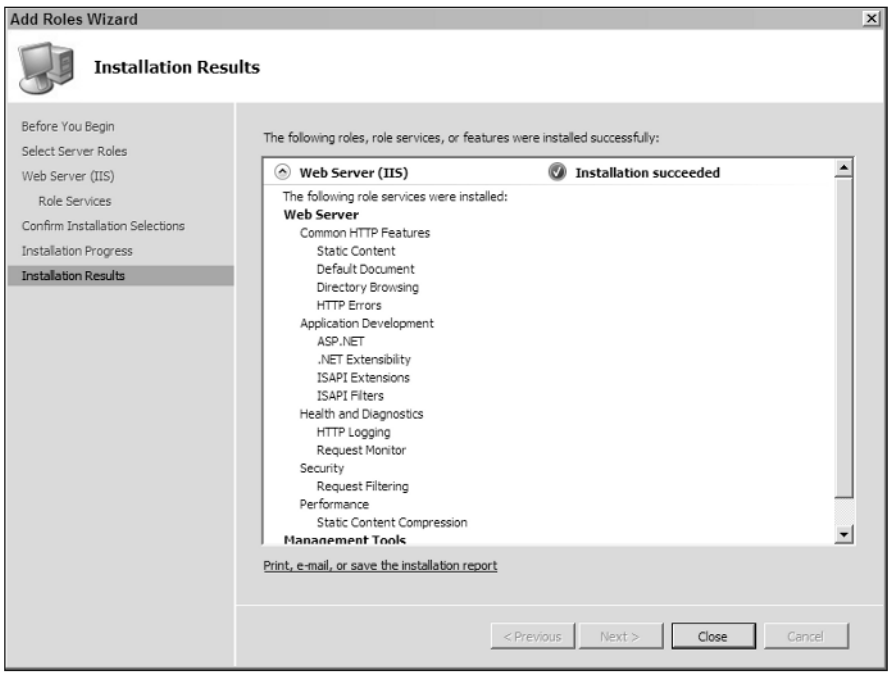

Figure 1-15

If you click the Close button in Figure 1-15, you'll be back to the Server Manager shown in Figure 1-16. Note that the Roles nodes on the left panel and the middle panel now contain a role named Web Server (IIS).

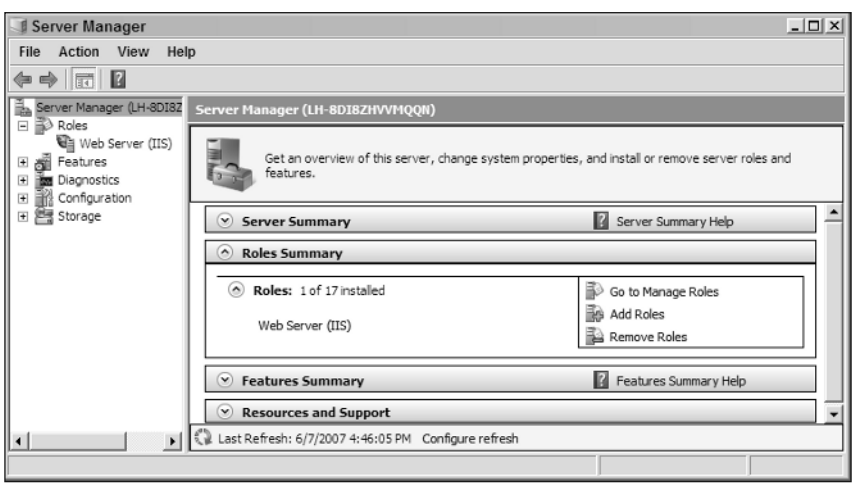

Figure 1-16

Command-Line Setup Option

Windows Vista and Windows Server 2008 come with a new command-line tool named `pkgmgr.exe` that you can use to custom install IIS 7. The following table describes the available options on this command-line tool:

Option	Description
`/iu:update1;update2…`	Run the tool with this option to install the specified updates. Notice that the update list contains a semicolon-separated list of update names discussed in the previous sections.
`/uu:update1;update2…`	Run the tool with this option to uninstall the specified updates. Notice that the update list contains a semicolon-separated list of update names discussed in the previous sections.
`/n:unattend.xml`	Run the tool with this option to install or uninstall the updates specified in the specified `unattend.xml` file. I cover this file in the following section.

When you use the `pkgmgr.exe` command-line tool to install specified updates, you must also explicitly specify and install the updates that your specified updates depend on. For example, if you decide to install the IIS-CommonHttpFeatures update, you must also install its parent update, that is, IIS-WebServer. To install the IIS-WebServer update you must also install its parent update, IIS-WebServerRole, and the update that it depends on, WAS-ProcessModel (see the update dependencies table). To install the WAS-ProcessModel update you must also install its parent update, WAS-WindowsActivationService update:

```
start /w /iu:IIS-WebServerRole;WAS-WindowsActivationService;WAS-ProcessModel;
IIS-WebServer;IIS-CommonHttpFeatures
```

Notice that if you don't specify the `start /w` option, the command-line tool will return immediately and process everything in the background, which means that you won't be able to see when the setup is completed.

Unattended Setup Option

As mentioned earlier, the `pkgmgr.exe` command-line tool comes with the `/n:unattend.xml` option. `unattend.xml` is the XML file that contains the updates to be installed or uninstalled. This XML file provides you with two benefits. First, you don't have to directly enter the names of the updates on the command line. Second, you can store this file somewhere for reuse in other Web server machines. This XML file must have the same schema as the XML file shown in Listing 1-1. This listing installs the IIS-CommandHttpFeatures update and the updates that it depends on as discussed in the previous section.

Listing 1-1: The unattend.xml File

```
<?xml version="1.0" ?>
<unattend xmlns="urn:schemas-microsoft-com:unattend"
xmlns:wcm="http://schemas.microsoft.com/WMIConfig/2002/State">
  <servicing>
    <!--Install a selectable update in a package that is in the
     Windows Foundation namespace-->
    <package action="configure">
      <assemblyIdentity name="Microsoft-Windows-Foundation-Package"
       version="6.0.5308.6" language="neutral" processorArchitecture="x86"
       publicKeyToken="31bf3856ad364e35" versionScope="nonSxS" />

      <selection name="IIS-WebServerRole" state="true"/>
      <selection name="WAS-WindowsActivationService" state="true"/>
      <selection name="WAS-ProcessModel" state="true"/>
      <selection name="IIS-WebServer" state="true"/>
      <selection name="IIS-CommonHttpFeatures" state="true"/>
    </package>
  </servicing>
</unattend>
```

Notice that the `<servicing>` element contains one or more `<selection>` child elements, and each child element specifies a particular update. The `<selection>` child element features two attributes named `name` and `state`. The `name` attribute contains the update name to be installed or uninstalled. Set the `state` attribute to `true` to install or `false` to uninstall the specified update.

Upgrade

If you're upgrading from Windows XP to Windows Vista, or from Windows Server 2003 to Windows Server 2008, and if your old operating system has IIS installed, the Windows Vista or Windows Server 2008 setup automatically scans through the capabilities of the installed IIS and ensures that the new install of IIS 7 supports those features and capabilities. Unfortunately, due to the monolithic architecture of IIS 5.1 and IIS 6.0, this installation ends up installing almost all of the feature modules of IIS 7. I highly recommend that after the upgrade you use one of the previously discussed installation options to uninstall the updates that you do not need to decrease the attack surface and footprint of your Web server.

Summary

This chapter first covered the IIS 7 package updates and their constituent feature modules, and showed you how to custom-build your own Web server from the desired package updates to decrease the footprint of your Web server. The chapter then provided in-depth coverage of five different IIS7 setup options. The chapter also gave an overview of the main systems that make up the IIS7 and ASP.NET integrated infrastructure. As discussed, one of these systems is the IIS7 and ASP.NET integrated configuration system, which will be discussed thoroughly in the next chapter.