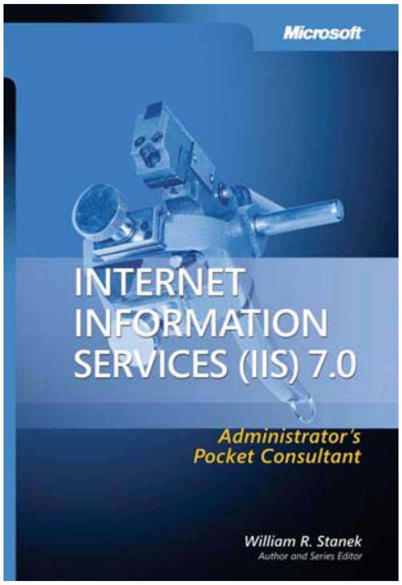

Internet Information Services (IIS) 7.0 Administrator's Pocket Consultant

William R. Stanek

PREVIEW CONTENT This excerpt contains uncorrected manuscript from an upcoming Microsoft Press title, for early preview, and is subject to change prior to release. This excerpt is from *Internet Information Services (IIS) 7.0 Administrator's Pocket Consultant* from Microsoft Press (ISBN 978-0-7356-2364-4, copyright 2007 William Stanek, all rights reserved), and is provided without any express, statutory, or implied warranties

To learn more about this book, visit Microsoft Learning at
http://www.microsoft.com/MSPress/books/10442.aspx

978-0-7356-2364-4

Table of Contents

Chapter 1

IIS 7.0 Administration Overview

Let's start with the bad news right up front: Internet Information Services (IIS) 7.0 isn't what you think it is. Although IIS 7.0 *is* the latest release of Internet Information Services, it *isn't* what it seems. IIS does look a lot like its predecessors, but this is deceiving because under the surface, the architecture is completely different. So much has changed, in fact, that perhaps it might have been better if Microsoft had given IIS 7.0 a new product name. That way you'd know that IIS 7.0 was completely different from its predecessors, allowing you to start with a fresh perspective and a reasonable expectation of having to learn a whole new bag of tricks. Seasoned IIS pros also are going to have to unlearn some old tricks; and that's not only going to be difficult, it might be the single biggest obstacle to mastering IIS 7.0.

IIS 7.0 provides the core services for hosting Web servers, Web applications, and Microsoft Windows SharePoint Services. Throughout this book, I'll refer to administration of IIS, Web applications, and Windows SharePoint Services as *Microsoft Web administration* or simply *Web administration*. As you get started with Microsoft Web administration, you should concentrate on these key areas:

- What's new or changed in IIS 7.0
- How IIS 7.0 configuration schema and global configuration architecture are used
- How IIS 7.0 works with your hardware
- How IIS works with Windows-based operating systems
- Which administration tools are available
- Which administration techniques you can use to manage and maintain IIS

Working with IIS 7.0: What You Need to Know Right Now

Microsoft fully integrated Microsoft ASP.NET and the Microsoft .NET Framework into IIS 7.0. Unlike IIS 6, IIS 7.0 takes ASP.NET and the .NET Framework to the next level by integrating the ASP.NET runtime extensibility model with the core server architecture, allowing developers to fully extend the core server architecture by using ASP.NET and the .NET Framework. This tighter integration makes it possible to use existing ASP.NET features such as .NET Roles, Session Management, Output Caching, and Forms Authentication with all types of content.

IIS 7.0 has generalized the Hypertext Transfer Protocol (HTTP) process activation model that IIS 6 introduced with application pools and made it available for all protocols through an independent service called the Windows Process Activation Service, and developers can use Windows Communications Framework (WCF) protocol adapters to take advantage of the capabilities of this service. You also should know up front that IIS 7.0 includes a metabase compatibility component that allows your existing scripts and applications to continue running but does not use a metabase to store configuration information. Instead of a metabase, IIS 7.0 uses a distributed configuration system with global and application-

---

specific configuration files that are based on a customizable set of Extensible Markup Language (XML) schema files. These XML schema files define the configuration elements and attributes in addition to valid values for those elements and attributes, providing you precise control over exactly how you can configure and use IIS.

Microsoft built the configuration system around the concept of modules. *Modules* are standalone components that provide the core feature set of an IIS server. Microsoft ships more than 40 independent modules with IIS 7.0. Either these modules are IIS 7.0-native modules that use a Win32 DLL or IIS 7.0-managed modules that use a .NET Framework Class Library contained within an assembly. Because all server features are contained within modules, you can modify the available features easily by adding, removing, or replacing a server's modules. Further, by optimizing the installed modules based on the way an IIS server is used, you can enhance security by reducing the attack surface area and improve performance by reducing the resources required to run the core services.

Note Because modules are such an important part of IIS 7.0, you'll find much discussion about them and how they are used in this book. Chapter 2, "Deploying IIS 7.0 in the Enterprise," introduces all the available native and managed modules. Chapter 5, "Managing Global IIS Configuration," details how to install and manage modules. Appendix A, "Comprehensive IIS 7.0 Module and Schema Reference," provides a complete guide to using modules and schemas.

IIS 7.0 is more secure than IIS 6 because of built-in request filtering and rules-based Uniform Resource Locator (URL) authorization support. You can configure request filtering to reject suspicious requests by scanning URLs sent to a server and filtering out unwanted requests. You can configure URL authorization rules to require logon and allow or deny access to specific URLs based on user names, .NET roles, and HTTP request methods. To make it easier to resolve problems with the server and Web applications, IIS 7.0 includes new features for diagnostics, real-time request reviewing, and error reporting. These features allow you to:

- View the current running state of the server.
- Trace failed requests through the core server architecture.
- Obtain detailed error information to pinpoint the source of a problem.

IIS 7.0 has many other new and enhanced features, but few are as important as the new set of administration tools, including new graphical, command-line, and scripting administration tools. The new graphical administration tool uses a browser-like interface and adds features for delegated administration, remote administration over Secure HTTP (HTTPS), and extensibility through custom user interface components. The new command-line administration tool makes it possible to perform most configuration tasks with a single line of command text. With ASP.NET, you can manage IIS configuration through the .NET Framework by using the Microsoft.Web.Administrators Application Programming Interface (API). With scripting, you can manage IIS configuration through the IIS 7.0 Windows Management Instrumentation (WMI) provider.

Because of the many changes, much of what you know about IIS is obsolete or irrelevant. But there's a light at the end of the tunnel—well, it's more like a freight train coming right at you—but it's there. The changes in IIS 7.0 are well worth the time and effort you'll spend learning the new architecture and the new techniques required to manage Web servers. Our dependence on ASP.NET and the .NET Framework will only grow over time, and the more you learn about the heart of the .NET architecture—IIS 7.0—the better prepared you'll be for now and for the future.

With IIS 7.0, key components that were a part of previous IIS releases are no longer available or work in different ways than they did before. Because IIS 7.0 does not use a metabase, applications designed for IIS 6 will not run on IIS 7.0 without special actions being taken. To run IIS 6 applications, you must install the IIS 6 compatibility and metabase feature. To manage IIS 6 applications and features, you must install IIS 6 Manager, IIS 6 scripting tools, and IIS 6 WMI compatibility. Additionally, IIS 7.0 does not include Post Office Protocol Version 3 (POP3) or Simple Mail Transfer Protocol (SMTP) services. With IIS 7.0, you can send e-mail messages from a Web application by using the SMTP E-mail component of ASP.NET.

IIS Manager is the graphical user interface (GUI) for managing both local and remote installations of IIS 7.0. To use IIS Manager to manage an IIS server remotely, Web Management Service (WMSVC) must be installed and started on the IIS server you want to manage remotely. WMSVC is also required when IIS site or application administrators want to manage features over which they've been delegated control.

The Web Management Service provides a hostable Web core that acts as a standalone Web server for remote administration. After you install and start WMSVC on an IIS server, it listens on port 8172 on all unassigned IP addresses for four specific types of requests:

Login Requests

- IIS Manager sends login requests to WMSVC to initiate connections. On the hostable Web core, login requests are handled by Login.axd. The authentication type is either NT LAN Manager (NTLM) or Basic, depending on what you select when you are prompted to provide credentials in the connection dialog box.

Code Download Requests

- If login is successful, WMSVC returns a list of user interface (UI) modules for the connection. Each IIS Manager page corresponds to a specific UI module. If there's a module that IIS Manager doesn't have, it will request to download the module binaries. Code download requests are handled by Download.axd.

Management Service Requests

- After a connection is established, your interactions with IIS Manager cause management service requests. Management service requests direct module services in WMSVC to read or write configuration data, runtime state, and providers on the server. Management service requests are handled by Service.axd.

Ping Requests

- Ping requests are made from within the WMSVC service to the hostable Web core. Ping requests are made by Ping.axd to ensure that the hostable Web core continues to be responsive.

The Web Management Service stores a limited set of editable configuration values in the registry. Each time the service is started, the Web configuration files are regenerated in the following directory: *%SystemRoot%*\ServiceProfiles\LocalService\AppData\Local\Temp\WMSvc. To enhance security, WMSVC requires SSL (HTTPS) for all connections. This ensures that data passed between the remote IIS Manager client and WMSVC is secure. Additionally, WMSVC runs as Local Service with a reduced permission set and a locked down configuration. This ensures that only the minimal set of required modules are loaded when the hostable Web core starts. See Chapter 3, "Core IIS 7.0 Administration," for more information.

Note %*SystemRoot*% refers to the SystemRoot environment variable. The Windows operating system has many environment variables, which are used to refer to user- and system-specific values. Often, I'll refer to environment variables in this book using this syntax: %*VariableName*%.

Introducing IIS 7.0 Configuration Architecture

You can use IIS 7.0 to publish information on intranets, extranets, and the Internet. Because today's Web sites use related features, such as ISAPI filters, ASP, ASP.NET, CGI, and the .NET Framework, IIS bundles these features as part of a comprehensive offering. What you need to know right now about IIS 7.0 is how IIS 7.0 uses the configuration schema and its global configuration system. In Chapter 2, you'll learn about the available setup features and the related configuration modules.

IIS 7.0 Configuration Schema

Unlike IIS 6, in which the main configuration information is stored in metabase files, IIS 7.0 has a unified configuration system for storing server, site, and application settings. You can manage these settings by using an included set of managed code, scripting APIs, and management tools. You can also manage these settings by directly editing the configuration files themselves. Direct editing of configuration files is possible because the files use XML and are written in plain-language text files based on a predefined set of XML schema files.

Note IIS 7.0 always takes the master state for configuration from the configuration files. This is a dramatic change from IIS 6, in which the master state was taken from the in-memory configuration database, which was flushed periodically to disk.

Using the XML schema to specify the configuration settings ensures that the related configuration files are well-structured XML, which is easy to modify and maintain. Because configuration values are stored using easy-to-understand text strings and values, they are easy to work with. By examining the schema itself, you can determine the exact set of acceptable values for any configuration option. IIS shares the same schema with ASP.NET configuration, ensuring that configuration settings for ASP.NET applications are just as easy to manage and maintain.

On an IIS server, schema files are stored in the *%SystemRoot%*\System32\inetsrv\config\schema directory. The three standard schema files are:

IIS_schema.xml

- This file provides the IIS configuration schema.

ASPNET_schema.xml

- This file provides the ASP.NET configuration schema.

FX_schema.xml

- This file provides the .NET Framework configuration schema (providing features beyond what the ASP.NET schema offers).

IIS reads in the schema files automatically during startup of the application pools. The IIS schema file is the master schema file. Within the IIS schema file, you'll find configuration sections for each major feature of IIS, from application pooling to failed request tracing. The ASP.NET schema file builds on and extends the master schema with specific configuration sections for ASP.NET. Within the ASP.NET schema file, you'll find configuration sections for everything from anonymous identification to output cache settings. The FX schema file builds on and extends the ASP.NET schema file. Within the FX schema file, you'll find configuration settings for application settings, connection strings, date-time serialization, and more.

Whereas configuration sections are also grouped together for easier management, section groups do not have schema definitions. If you want to extend the configuration features and options available in IIS, you can do this by extending the XML schema. You extend the schema by following these basic steps:

1. Specify the desired configuration properties and the necessary section container in an XML schema file.
2. Place the schema file in the *%SystemRoot%*\System32\inetsrv\config\schema directory.
3. Reference the new section in IIS 7.0's global configuration file.

The basic syntax for a schema file is as follows:

```
<!–

The text within this section is a comment. It is standard practice to provide
introductory details in the comments at the beginning of the schema file.

-->
<configSchema>
    <sectionSchema name="configSection1">
    </sectionSchema>
    <sectionSchema name="configSection2">
    </sectionSchema>
    <sectionSchema name="configSection3">
    </sectionSchema>
</configSchema>
```

As an administrator or developer, you don't necessarily need to be able to read and interpret XML schemas to succeed. However, because having a basic understanding of schemas is helpful, I'll introduce the essentials. Within schema files, configuration settings are organized into sets of related features called *schema sections*. The schema for a configuration section is defined in a <sectionSchema> XML element. For example, the features related to the HTTP listener in IIS are defined with a schema section named system.applicationHost/listenerAdapters. In the IIS_schema.xml file, this section is defined as follows:

```
<sectionSchema name="system.applicationHost/listenerAdapters">
 <collection addElement="add" >
  <attribute name="name" type="string" required="true" isUniqueKey="true" />
  <attribute name="identity" type="string" />
  <attribute name="protocolManagerDll" type="string" />
  <attribute name="protocolManagerDllInitFunction" type="string" />
 </collection>
</sectionSchema>
```

This schema definition states that the system.applicationHost/listenerAdapters element can contain a collection of add elements with the following attributes:

name
- A unique string that is a required part of the add element.

identity
- An identity string that is an optional part of the add element.

protocolManagerDll
- A string that identifies the protocol manager DLL.

protocolManagerDllInitFunction
- A string that identifies the initialization function for the protocol manager DLL.

An attribute of an element is either optional or required. If the attribute definition states **required="true"** as with the name attribute, the attribute is required and must be provided when you are using the related element. Otherwise, the attribute is considered optional and does not need to be provided when you are using the related element. In addition to being required, attributes can have other enforced conditions, including:

isUniqueKey
- If set to true, the related value must be unique.

encrypted
- If set to true, the related value is expected to be encrypted.

With some attributes, you'll see default values and possibly an enumerated list of the acceptable string values and their related internal values. In the following example, the identityType attribute has a default value of NetworkService and a list of other possible values:

```
<attribute name="identityType" type="enum" defaultValue="NetworkService">
 <enum name="LocalSystem" value="0"/>
 <enum name="LocalService" value="1"/>
 <enum name="NetworkService" value="2"/>
 <enum name="SpecificUser" value="3"/>
</attribute>
```

The friendly-name of a value is provided to make the value easier to work with. The actual value used by IIS is provided in the related value definition. For example, if you set identityType to LocalService, the actual configuration value used internally by IIS is 2.

As a standard rule, you cannot use enumerated values in combination with each other. Because of this, the identityType attribute can have only one possible value. In contrast, attributes can have flags, which can be used together to form combinations of values. In the following example, the logEventOnRecycle attribute uses flags and has a default set of flags that are used in combination with each other:

```
<attribute name="logEventOnRecycle" type="flags" defaultValue="Time, Memory,
PrivateMemory">
 <flag name="Time" value="1"/>
 <flag name="Requests" value="2"/>
 <flag name="Schedule" value="4"/>
 <flag name="Memory" value="8"/>
 <flag name="IsapiUnhealthy" value="16"/>
 <flag name="OnDemand" value="32"/>
 <flag name="ConfigChange" value="64"/>
 <flag name="PrivateMemory" value="128"/>
</attribute>
```

Again, the friendly name is provided to make the value easier to work with. The actual value used by IIS is the sum of the combined flag values. With a setting of "Time, Requests, Schedule," the logEventOnRecycle attribute is set to 7 (1+2+4=7).

Attribute values can also have validation. IIS performs validation of attribute values when parsing the XML and when calling the related API. Table 1-1 provides an overview of the validators you'll see in schemas.

Table 1-1 Summary of Attribute Validation Types in an IIS XML Schema

Validation Type	Validation Parameter	Validation fails if…
validationType= "applicationPool Name"	validationParameter=""	A validated value contains these characters: |<>&\"
validationType=" integerRange"	validationParameter= "<minimum>, <maximum>[,exclude]"	A validated value is outside [inside] range, in integers.
validationType=" nonEmptyString"	validationParameter=""	A validated value has a string value that is not set.
validationType=" siteName"	validationParameter=""	A validated value contains these characters: /\.?
validationType=" timeSpanRange"	validationParameter=" <minimum>,<maximum>, <granularity>[,exclude]"	A validated value is outside [inside] range, in seconds.
validationType=" requireTrimmedS tring"	validationParameter=""	A validated value has white space at start or end of value.

IIS 7.0 Global Configuration System

IIS uses a global configuration system that is difficult to understand at first but gets easier and easier to understand once you've worked with it awhile. Because there's no sense trying to ease into this, I'll dive right in. If you'll hang with me for a few pages, I'll get you through the roughest parts and zero in on exactly what you need to know—I promise.

IIS configuration settings are stored in configuration files that together set the running configuration of IIS and related components. One way to think of a configuration file is as a container for the settings you apply and their related values. You can apply multiple configuration files to a single server and the applications it is running. Generally, you manage configuration files at the .NET Framework root level, the server root level, and the various levels of a server's Web content directories. A server's Web content directories include the root directory of the server itself, the root directories of configured Web sites, and any subdirectories within Web sites. The root levels and the various levels of a server's Web content directories can be described as containers for the settings you apply and their values. If you know a bit about object-oriented programming, you might expect the concepts of parent-child relationship and inheritance to apply—and you'd be right.

Through inheritance, a setting applied at a parent level becomes the default for other levels of the configuration hierarchy. Essentially, this means that a setting applied at a parent level is passed down to a child level by default. For example, if you apply a setting at the server root level, the setting is inherited by all Web sites on the server and by all the content directories within those sites.

The order of inheritance is as follows:

```
.NET Framework root → server root → Web Site root → top-level directory  →
subdirectory
```

This means that the settings for the current .NET Framework root are passed down to IIS, the settings for IIS are passed down to Web sites, and the settings for Web sites are passed down to content directories and subdirectories. As you might expect, you can override inheritance. To do this, you specifically assign a setting for a child level that contradicts a setting for a parent. As long as overriding a setting is allowed (that is, overriding isn't blocked), the child level's setting will be applied appropriately. To learn more about overriding and blocking, see Chapter 5.

When working with the configuration files, keep the following in mind:

- The .NET Framework root IIS applies depends on the current running version of ASP.NET and the .NET Framework. The default configuration files for the .NET Framework root are Machine.config and Web.config, which are stored in the *%SystemRoot%*\microsoft.net\framework\Version\config\machine.config directory. Machine.config sets the global defaults for the .NET Framework settings in addition to some ASP.NET settings. Web.config sets the rest of the global defaults for ASP.NET. See Chapter 8, "Running IIS Applications," and Chapter 9, "Managing Applications, Application Pools, and Worker Processes," for more information about configuring the .NET Framework and ASP.NET.
- The default configuration file for the server root is applicationHost.config, which is stored in the *%SystemRoot%*\System32\inetsrv\config directory. This file controls the global configuration of IIS. See Chapter 5 for more information about configuring IIS servers.
- The default configuration file for a Web site root is Web.config. This file is stored in the root directory of the Web site to which it applies and controls the behavior for the Web site. See Chapters 8 and 9 for more information about configuring IIS applications.

- The default configuration file for a top-level content directory or a content subdirectory is Web.config. This file is stored in the content directory to which it applies and controls the behavior of that level of the content hierarchy and downwards. See Chapter 6 for more information about configuring content directories.

In some cases, you may want a .config file to include some other .config file. This can be done by using the configSource attribute to refer to the .config file containing the settings you want to use. Currently, the referenced .config file must reside in the same directory as the original .config file. Note that this behavior may change to allow .config files in other directories to be used. To see how this works, consider the following example from the applicationHost.config file:

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- applicationHost.config -->
<configuration>
 <system.webServer>
  <httpErrors>

   <error statusCode="401" prefixLanguageFilePath="%SystemDrive%\inetpub\custerr"
path="401.htm" />

   <error statusCode="403" prefixLanguageFilePath="%SystemDrive%\inetpub\custerr"
path="403.htm" />

   <error statusCode="404" prefixLanguageFilePath="%SystemDrive%\inetpub\custerr"
path="404.htm" />

   <error statusCode="405" prefixLanguageFilePath="%SystemDrive%\inetpub\custerr"
path="405.htm" />

   <error statusCode="406" prefixLanguageFilePath="%SystemDrive%\inetpub\custerr"
path="406.htm" />

   <error statusCode="412" prefixLanguageFilePath="%SystemDrive%\inetpub\custerr"
path="412.htm" />

   <error statusCode="500" prefixLanguageFilePath="%SystemDrive%\inetpub\custerr"
path="500.htm" />

   <error statusCode="501" prefixLanguageFilePath="%SystemDrive%\inetpub\custerr"
path="501.htm" />

   <error statusCode="502" prefixLanguageFilePath="%SystemDrive%\inetpub\custerr"
path="502.htm" />

  </httpErrors>

 </system.webServer>
</configuration>
```

In this example, error elements specify how certain types of HTTP error status codes should be handled. If you wanted to customize the error handling for a server, you might want to extend or modify the default values in a separate .config file and then reference the .config file in applicationHost.config. To do this, you would update the applicationHost.config file to point to the additional .config file. An example follows.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- applicationHost.config -->
<configuration>
 <system.webServer>
  <httpErrors configSource=errorMode.config />
</configuration>
```

You would then create the errorMode.config file and store it in the same directory as the applicationHost.config file. The following is an example of the contents of the errorMode.config file:

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- errorMode.config -->

<configuration>
 <system.webServer>
  <httpErrors>

  <error statusCode="401" prefixLanguageFilePath="%SystemDrive%\inetpub\custerr"
path="401.htm" />

  <error statusCode="403" prefixLanguageFilePath="%SystemDrive%\inetpub\custerr"
path="403.htm" />

  <error statusCode="404" prefixLanguageFilePath="%SystemDrive%\inetpub\custerr"
path="404.htm" />

  <error statusCode="405" prefixLanguageFilePath="%SystemDrive%\inetpub\custerr"
path="405.htm" />

  <error statusCode="406" prefixLanguageFilePath="%SystemDrive%\inetpub\custerr"
path="406.htm" />

  <error statusCode="412" prefixLanguageFilePath="%SystemDrive%\inetpub\custerr"
path="412.htm" />

  <error statusCode="500" prefixLanguageFilePath="%SystemDrive%\inetpub\custerr"
path="500.htm" />

  <error statusCode="501" prefixLanguageFilePath="%SystemDrive%\inetpub\custerr"
path="501.htm" />

  <error statusCode="502" prefixLanguageFilePath="%SystemDrive%\inetpub\custerr"
path="502.htm" />

   </httpErrors>

  </system.webServer>

</configuration>
```

When you make these or other types of changes in configuration files, you don't need to worry about restarting IIS or related services. IIS automatically picks up the changes and uses them. In theses examples, you'll note that we're working with the system.webServer section of the configuration file. As per the schema definition files, all settings are defined within specific configuration sections. Although sections cannot be nested, a section can exist within a section group, and that section group can in turn be contained in a parent section group. A section group is simply a container of logically related sections.

In applicationHost.config, section groups and individual sections are defined in the configSections element. The configSections element controls the registration of sections. Every section belongs to one section group. By default, applicationHost.config contains these section groups:

system.applicationHost

- Defines the following sections: applicationPools, configHistory, customMetadata, listenerAdapters, log, sites, and webLimits.

system.webServer

- Defines the following sections: asp, caching, cgi, defaultDocument, directoryBrowse, globalModules, handlers, httpCompression, httpErrors, httpLogging, httpProtocol, httpRedirect, httpTracing, isapiFilters, modules, odbcLogging, serverRuntime, serverSideInclude, staticContent, urlCompression, and validation. Includes the security and tracing subgroups.

system.webServer.security

- A subgroup of system.webServer that defines the following sections: access, applicationDependencies, authorization, ipSecurity, isapiCgiRestriction, and requestFiltering. Includes the authentication subgroup.

system.webServer.security.authentication

- A subgroup of system.webServer.security that defines the following sections: anonymousAuthentication, basicAuthentication, clientCertificateMappingAuthentication, digestAuthentication, iisClientCertificateMappingAuthentication, and windowsAuthentication.

system.webServer.security.tracing

- A subgroup of system.webServer.security that defines the traceFailedRequests and traceProviderDefinitions sections.

In applicationHost.config, section groups and individual sections are defined as follows:

```
<configSections>

 <sectionGroup name="system.applicationHost">

  <section name="applicationPools" allowDefinition="AppHostOnly"
overrideModeDefault="Deny" />

  <section name="configHistory" allowDefinition="AppHostOnly" overrideModeDefault="Deny"
/>

  <section name="customMetadata" allowDefinition="AppHostOnly"
overrideModeDefault="Deny" />

  <section name="listenerAdapters" allowDefinition="AppHostOnly"
overrideModeDefault="Deny" />

  <section name="log" allowDefinition="AppHostOnly" overrideModeDefault="Deny" />

  <section name="sites" allowDefinition="AppHostOnly" overrideModeDefault="Deny" />

  <section name="webLimits" allowDefinition="AppHostOnly" overrideModeDefault="Deny" />

  </sectionGroup>

 <sectionGroup name="system.webServer">

  ...

 </sectionGroup>

</configSections>
```

In machine.config, you'll also find definitions for section groups and individual sections. These are similar to those used in applicationHost.config but are used for configuring the .NET Framework and some ASP.NET settings. When working with either .config file, keep in mind that a section is the basic unit of deployment, locking, searching, and containment for configuration settings. Every section has a name attribute and optional allowDefinition and overrideModeDefault attributes. The name attribute sets the unique section name. The allowDefinition attribute specifies the level at which the section can be set:

Everywhere

- The section can be set in any configuration file including directories mapped to virtual directories that are not application roots, and their subdirectories.

MachineOnly

- The section can be set only in applicationHost.config or machine.config. Because this is the default setting, a section that doesn't have an allowDefinition attribute uses this setting automatically.

MachineToWebRoot

- The section can be set only in the .NET Framework root's machine.config or Web.config file, or in applicationHost.config.

MachineToApplication

- The section can be set only in the .NET Framework root's machine.config or Web.config file, in applicationHost.config, or in Web.config files for application root directories.

AppHostOnly

- The section can be set only in Web.config files for application root directories.

The OverrideModeDefault attribute sets the default lockdown state of a section. Essentially, this means that it controls whether a section is locked down to the level in which it is defined or can be overridden by lower levels of the configuration hierarchy. If this attribute is not set, the default value is Allow. With Allow, lower level configuration files can override the settings of the related section. With Deny, lower level configuration files cannot override the settings of the related section. As discussed in Chapter 5, you'll typically use location tags to lock or unlock sections for specific Web sites or applications.

Because the complete configuration settings of a server and its related sites and applications are stored in the configuration files, you easily can back up or duplicate a server's configuration. Backing up a server's configuration is a simple matter of creating a copy of the configuration files. Similarly, duplicating a server's configuration on another server is a simple matter of copying the source configuration files to the correct locations on another server.

IIS 7.0 and Your Hardware

Before you deploy IIS 7.0, you should carefully plan the server architecture. As part of your planning, you need to look closely at pre-installation requirements and the hardware you will use. IIS 7.0 is no longer the simple solution for hosting Web sites that it once was. It is now provides the core infrastructure for hosting Web servers, Web applications, and Windows SharePoint Services.

Guidelines for choosing hardware for Internet servers are much different from those for choosing other types of servers. A Web hosting provider might host multiple sites on the same computer and might also have service level agreements that determine the level of availability and performance required. On the other hand, a busy e-commerce site might have a dedicated Web server or even multiple load-balanced servers. Given that Internet servers are used in a wide variety of circumstances and might be either shared or dedicated, here are some guidelines for choosing server hardware:

Memory

- The amount of random access memory (RAM) that's required depends on many factors, including the requirements of other services, the size of frequently accessed content files, and the RAM requirements of the Web applications. In most installations, I recommend that you use at least 1 gigabyte (GB) of RAM. High-volume servers should have a minimum of 2 to 4 GB of RAM. More RAM will allow more files to be cached, reducing disk requests. For all IIS installations, the operating system paging file size should at least equal the amount of RAM on the server.

Don't forget that as you add physical memory, virtual paging to disk grows as well. With this in mind, you might want to ensure that the Pagefile.sys file is on the appropriate disk drive, one that has adequate space for the page file to grow, along with providing optimal input/output (I/O) performance.

Moreinfo For detailed information on memory management and performance tuning, see Chapter 12, "Performance Tuning and Monitoring."

CPU

- The CPU processes the instructions received by the computer. The clock speed of the CPU and the size of the data bus determine how quickly information moves among the CPU, RAM, and system buses. Static content, such as HTML and images, place very little burden on the processor, and standard recommended configurations should suffice. Faster clock speeds and multiple processors increase the performance scalability of a Web server, particularly for sites that rely on dynamic content. 32-bit versions of Windows run on Intel x86 or compatible hardware. 64-bit versions of Windows run on the x64 family of processors from AMD and Intel, including AMD64 and Intel Extended Memory 64 Technology (Intel EM64T). IIS provides solid benchmark performance on Intel Xeon, AMD Opteron, and AMD Athlon processors. Any of these CPUs provide good starting points for the typical IIS server. You can achieve significant performance improvements with a large processor cache. Look closely at the L1, L2, and L3 cache options available—a larger cache can yield much better performance overall.

SMP

- IIS supports symmetric multiprocessors (SMPs) and can use additional processors to improve performance. If the system is running only IIS and doesn't handle dynamic content or encryption, a single processor might suffice. You should always use multiple processors if IIS is running alongside other services, such as Microsoft SQL Server or Microsoft Exchange Server.

Disk drives

- The amount of data storage capacity you need depends entirely on the size of content files and the number of sites supported. You need enough disk space to store all your data plus workspace, system files, and virtual memory. I/O throughput is just as important as drive capacity. However, disk I/O is rarely a bottleneck for Web sites on the public Internet—generally, bandwidth limits throughput. High-bandwidth sites should consider hardware-based redundant array of independent disks (RAID) solutions using copper or fiber channel–based small computer system interface (SCSI) drives.

Data protection

- Unless you can tolerate hours of downtime, you should add protection against unexpected drive failures by using RAID. Hardware RAID implementations are always preferred over software RAID implementations. RAID 0 (disk striping without parity) offers optimal read/write performance, but if a drive fails, IIS won't be able to continue operation until the drive is replaced and its contents are restored from backup. Because of this, RAID 0 isn't the recommended choice. RAID 1 (disk mirroring) creates duplicate copies of data on separate physical drives, allowing the server to remain operational when a drive fails, and even while the RAID controller rebuilds a replacement drive in a failed mirror. RAID 5 (disk striping with parity) offers good protection against single-drive failure but has poor write performance. Keep in mind that if you've configured redundant load-balanced servers, you might not need RAID. With load balancing, the additional servers might offer the necessary fault tolerance.

UPS

- Sudden power loss and power spikes can seriously damage hardware. To prevent this, get an uninterruptible power supply (UPS). A properly configured UPS system allows the operating system to automatically shut down the server gracefully in the event of a power outage, and it's also important in maintaining system integrity when the server uses write-back caching controllers that do not have on-board battery backups. Professional hosting providers often offer UPS systems that can maintain power indefinitely during extended power outages.

If you follow these hardware guidelines, you'll be well on your way to success with IIS.

IIS 7.0 Editions and Windows

IIS 7.0 is available for both desktop and server editions of Windows. On Windows Vista, IIS 7.0 offers Web administrators and Web developers a complete platform for building and testing dynamic Web sites and Web applications. IIS 7.0 running on Windows Vista also enables process activation, process management, and the necessary HTTP infrastructure for creating Windows Communications Foundation (WCF)–based applications.

As discussed further in Chapter 2, the way IIS 7.0 works on Windows Vista depends on the edition of Windows Vista you are using. On Windows Vista Starter and Home Basic editions, IIS 7.0 cannot be used to host Web sites, Web applications, or Windows SharePoint Services. On these editions, a limited set of IIS features are available, such as Windows Activation Service components that are used to enable WCF-based applications. Users who install WCF-based applications will not need to install these components. The necessary components are installed automatically by WCF. With these editions, the simultaneous request execution limit for IIS is three, meaning that an application or a group of running applications could make up to three simultaneous requests for Web content through the installed IIS components.

On Windows Vista Home Premium, most of the IIS 7.0 features required for Web site development are available. The available features should allow most casual or hobbyist administrators and developers to build and test dynamic Web sites and Web applications. Many advanced features are missing, however, including advanced authentication components, advanced logging components, and FTP server components. As with Starter and Home Basic editions of Windows Vista, the simultaneous request execution limit for IIS is three for Windows Vista Home Premium, meaning you or running applications could make up to three simultaneous requests for Web content through the installed IIS components.

For Windows Vista Business, Enterprise, and Ultimate Editions, all IIS 7.0 features are available. This means that professional Web administrators and Web developers have everything necessary to design, build, and test Web sites and Web applications. The simultaneous request execution limit is ten for these editions of Windows Vista, meaning you or running applications could make up to ten simultaneous requests for Web content through the installed IIS components.

With server editions of Windows, you can use IIS to host Web servers, Web applications, and Windows SharePoint Services. All features of IIS 7.0 are available on all editions of Microsoft Windows Server 2008. On Windows Server operating systems, IIS 7.0 has no request execution limit. This means that an unlimited number of simultaneous requests can be made to the IIS 7.0 server core.

Web Administration Tools and Techniques

Web administrators will find that there are many ways to manage Web and application servers. The key administration tools and techniques are covered in the following sections.

Managing Resources by Using Key Administration Tools

Many tools are available for managing Web resources. Key tools you'll use are shown in Table 1-2. Most of these tools are available on the Administrative Tools menu. Click Start and choose All Programs, Administrative Tools, and then the tool you want to use. You can use all the tools listed in the table to manage local and remote resources. For example, if you connect to a new computer in IIS Manager, you can manage all its sites and services remotely from your system.

Table 1-2 Quick Reference for Key Web Administration Tools

Administration Tool	Purpose
Active Directory Users and Computers	Manages domain user, group, and computer accounts.
Computer Management	Manages services, storage, and applications. The Services And Applications node provides quick access to Indexing Service catalogs and IIS sites and servers.
Data Sources (ODBC)	Configures and manages Open Database Connectivity (ODBC) data sources and drivers. Data sources link Web front ends with database back ends.
DNS	Public Internet sites must have fully qualified domain names (FQDNs) to resolve properly in browsers. Use the Domain Name System (DNS) administrative snap-in to manage the DNS configuration of your Windows DNS servers .
Event Viewer	Allows you to view and manages events and system logs. If you keep track of system events, you'll know when problems occur.
Internet Information Services (IIS) 6.0 Manager	Manages Web and application server resources that were designed for IIS 6.0. This tool is included for backward compatibility only.
Internet Information Services (IIS) Manager	Manages Web and application server resources that were designed for IIS 7.0.
Web Management Service (WMSVC)	Allows you to use the IIS Manager to manage Web and application server resources on remote servers.
Reliability and Performance Monitor	Tracks system reliability and performance allowing you to pinpoint performance problems.
Services	Views service information, starts and stops system services, and configures service logons and automated recoveries.

When you add services to a server, the tools needed to manage those services are automatically installed. If you want to manage these servers remotely, you might not have these tools installed on your workstation. In that case, you need to install the administration tools on the workstation you're using.

Web Administration Techniques

Web administrators have many options for managing IIS. The key administration tools are:

- IIS Manager (InetMgr.exe)
- IIS Administration objects made available through the IIS 7.0 WMI provider
- IIS command-line administration tool (AppCmd.exe)

IIS Manager provides the standard administration interface for IIS. To start IIS Manager, click Start and choose All Programs, Administrative Tools, and then Internet Information Services (IIS) Manager. When started, IIS Manager displays the start page shown in Figure 1-1 and automatically connects to the local IIS installation, if it's available. On the start page, you have the following options:

Connect to localhost

- Connects you to the IIS installation on the local computer.

Connect to a server

- Allows you to connect to a remote server.

Connect to a site

- Allows you to connect to a specific Web site on a designated Web server.

Connect to an application

- Allows you to connect to a specific Web application on a designated site and server.

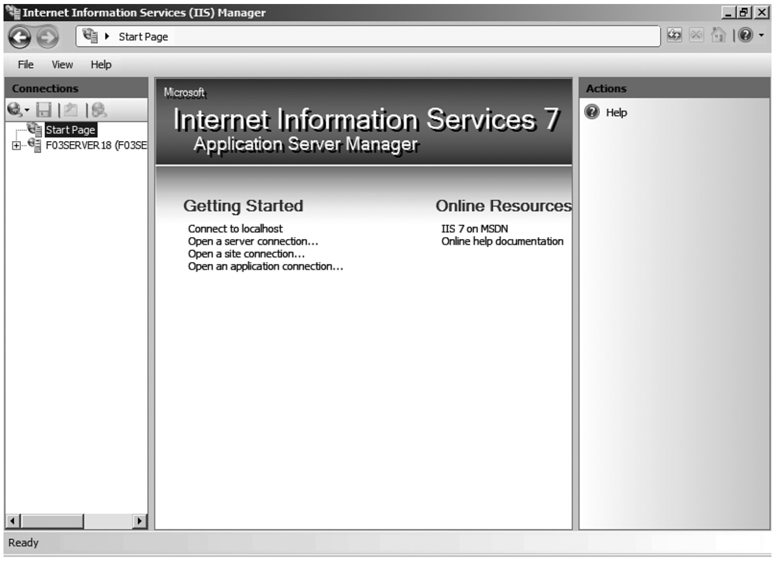
Internet Information Services (IIS) Manager
Start Page
File
View
Help
Connections
Start Page
F03SERVER18 (F03SE
Microsoft
Internet Information Services 7
Application Server Manager
Getting Started
Connect to localhost
Open a server connection...
Open a site connection...
Open an application connection...
Online Resources
IIS 7 on MSDN
Online help documentation
Actions
Help
Ready

Figure 1-1 You can access servers, sites, and applications by using IIS Manager.

As discussed previously, remote access to an IIS server is controlled by the WMSVC. When you install and start WMSVC on an IIS server, it listens on port 8172 on all unassigned IP addresses and allows remote connections from authorized user accounts. You can connect to a remote server by following these steps:

1. In Internet Information Services (IIS) Manager, click Start Page in the console tree and then click Connect To A Server. This starts the Connect To A Server wizard.

2. Type or select the server name in the Server Name box. For a server on the Internet, type the FQDN of the server, such as www.adatum.com. For a server on the local network, type the computer name, such as WEBSVR87. Port 80 is the default port for connections. As necessary, you can provide the port to which you want to connect. For example, if you want to connect to the server on port 8080, you would follow the server name by :8080, such as WEBSVR87:8080.
3. After you type the server name and optionally the port number, click Next. IIS Manager will then try to use your current user credentials to log on to the server. If this fails, you'll need to provide the appropriate credentials on the presented Provide Credentials page before clicking Next to continue. Click Finish to complete the connection.

Tip If IIS Manager displays a connection error stating that the remote server is not accepting connections, you'll need to log on locally or through remote desktop. Once logged on, check to ensure the Management Service is started and configured properly. For more information, see the "Enabling and Configuring Remote Administration" section of Chapter 3.

You can connect to a specific Web site on a designated server by following these steps:

1. In Internet Information Services (IIS) Manager, click Start Page in the console tree and then click Connect To A Site. This starts the Connect To A Site wizard.
2. Type or select the server name in the Server Name box, such as TESTSVR22. In the Site Name box, type or select the name of the Web site to which you want to connect, such as Default Web Site.
3. Click Next. IIS Manager will then try to use your current user credentials to log on to the server. If this fails, you'll need to provide the appropriate credentials on the presented Provide Credentials page before clicking Next to continue. Click Finish to complete the connection.

You can connect to a specific application on a designated site and server by following these steps:

1. In Internet Information Services (IIS) Manager, click Start Page in the console tree and then click Connect To An Application. This starts the Connect To An Application wizard.
2. Type or select the server name in the Server Name box, such as TESTSVR22. In the Site Name box, type or select the name of the Web site to which you want to connect, such as Default Web Site.
3. In the Application Name box, type or select the relative path of the Web application to which you want to connect, such as /MyApplication or /Apps/Myapp.
4. Click Next. IIS Manager will then try to use your current user credentials to log on to the server. If this fails, you'll need to provide the appropriate credentials on the presented Provide Credentials page before clicking Next to continue. Click Finish to complete the connection.

As Figure 1-2 shows, IIS Manager has been completely redesigned for IIS 7.0. Instead of being a snap-in for the Microsoft Management Console, IIS Manager is now a stand-alone application with a browser-like interface. Once you connect to a server, site, or application, IIS Manager automatically connects to these installations upon startup. You can change this behavior by disconnecting from the remote server while in IIS Manager. See Chapter 3 for more information on using IIS Manager.

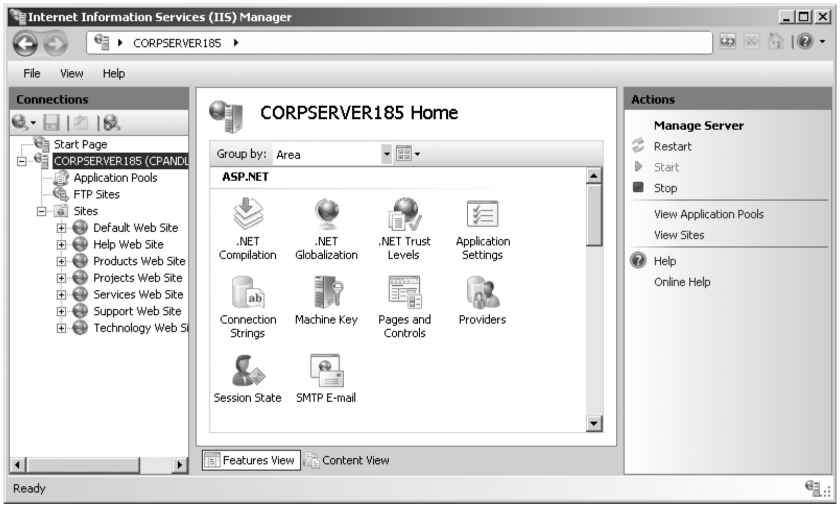

Figure 1-2 IIS Manager has a completely redesigned interface in IIS 7.0.

In IIS 5, you could designate Web site operators who could remotely administer IIS. Web site operators were a special group of users who had elevated privileges on individual Web sites. This feature was removed in IIS 6 because most of the time operator accounts weren't used. IIS 7.0 introduces the concept of delegated administration. With *delegated administration*, a machine administrator can delegate administrative control safely and securely. Delegated administration allows different levels of the configuration hierarchy to be managed by other users, such as site administrators or application developers. In a standard configuration, the default delegation state limits write access to most configuration settings to machine administrators only, and you must explicitly modify the delegation settings to grant write access to others. You'll learn more about IIS security and delegation in Chapter 10, "Managing Web Server Security."

IIS Manager and other graphical tools provide just about everything you need to work with IIS 7.0. Still, there are times when you might want to work from the command line, especially if you want to automate installation or administration tasks. To help you with all your command-line needs, IIS 7.0 includes the IIS command line administration tool (AppCmd.exe). AppCmd.exe is located in the *%SystemRoot%*\system32\inetsrv directory. By default, this directory is not in your command path. Because of this, you'll need either to add this directory to the default path or change to this directory each time you want to use this tool. Add this directory temporarily to your default path by typing the following at an elevated command prompt:

```
path %PATH%;%SystemRoot%\System32\inetsrv
```

Then add this directory permanently to your default path by typing the following at an elevated command prompt:

```
setx PATH %PATH%;%SystemRoot%\System32\inetsrv
```

Note You use Path to temporarily update the command path for the current window. You use SETX PATH to permanently update the command path for future command windows.

Table 1-3 provides a summary of the core set of administration objects for the IIS command line administration tool.

Table 1-3 Administration Objects for the IIS command line administration tool.

Object Type	Description	Related Commands
APP	Allows you to create and manage Web application settings by using related list, set, add, and delete commands	list, set, add, and delete
APPPOOL	Allows you to create and manage application pools by using related list, set, add, delete, start, stop, and recycle commands	list, set, add, delete, start, stop, and recycle
BACKUP	Allows you to create and manage backups of your server configuration by using list, add, delete, and restore commands	list, add, delete, and restore
CONFIG	Allows you to manage general configuration settings by using related list, set, search, lock, unlock, clear, reset, and migrate commands	list, set, search, lock, unlock, clear, reset, and migrate
MODULE	Allows you to manage IIS modules by using related list, set, add, delete, install, and uninstall commands	list, set, add, delete, install, and uninstall
REQUEST	Allows you to list current HTTP requests by using a related list command	list
SITE	Allows you to create and manage virtual sites by using related list, set, add, delete, start, and stop commands	list, set, add, delete, start, and stop
TRACE	Allows you to manage failed request tracing by using related list, configure, and inspect commands	list, configure, and inspect
VDIR	Allows you to create and manage virtual directory settings by using related list, set, add, and delete commands	list, set, add, and delete
WP	Allows you to list running worker processes by using a related list command	list

The basics of working with the IIS command line administration tool are straightforward. Most administration objects support these basic commands:

ADD

- Creates a new object with the properties you specify.

DELETE

- Deletes the object you specify.

LIST

- Displays a list of related objects. Optionally, you can specify a unique object to list, or you can type one or more parameters to match against object properties.

SET

- Sets parameters on the object specified.

Some objects support other commands including:

RECYCLE

- Recycles the object you specify by deleting it and then re-creating it.

START

- Starts the object you specify if it is stopped.

STOP

- Stops the object you specify if it is started or otherwise active.

To type commands, use the following basic syntax:

```
appcmd Command <Object-type>
```

where *Command* is the action to perform, such as list, add, or delete, and Object-type is the object on which you want to perform the action, such as app, site, or vdir. Following this, if you wanted to list the configured sites on a server, you could type the following command at an elevated command prompt:

```
appcmd list site
```

Because the IIS command line administration tool will also accept plural forms of object names, such as apps, sites, or vdirs, you could also use:

```
appcmd list sites
```

In either case, the resulting output is a list of all configured sites on the server with their related properties, such as:

```
SITE "Default Web Site" (id:1,bindings:http/*:80:,state:Started)
```

You'll find a comprehensive discussion of using the IIS Command-line Administration tool in Chapter 4, "Managing IIS 7.0 from the Command Line." In addition, you will see examples of using this tool throughout the book.