C H A P T E R 5

Upgrading an IIS Server to IIS 6.0

You can move Web sites and applications to Internet Information Services (IIS) 6.0 in two ways. *Upgrading* is installing the Microsoft® Windows® Server 2003 operating system and IIS 6.0 on an existing server running the Windows NT® Server 4.0 operating system and IIS 4.0 or the Windows® 2000 Server operating system and IIS 5.0. *Migrating*, on the other hand, is installing Windows Server 2003 and IIS 6.0 on a new server and then moving, or reinstalling, existing Web sites and applications to that server. Both processes involve minimal outage of service to users that access the Web sites and applications, while retaining the majority of the original configuration settings and fully preserving the content of the Web sites and applications.

In This Chapter

Related Information

- For information about migrating IIS Web sites, see “Migrating IIS Web Sites to IIS 6.0” in this book.
- For information about migrating Apache Web sites, see “Migrating Apache Web Sites to IIS 6.0” in this book.

Overview of Upgrading an IIS Server to IIS 6.0

The advantages of upgrading the server that hosts your existing IIS Web sites and applications to IIS 6.0 include the following:

- **Reduces the time required to deploy IIS 6.0.** Upgrading requires minimal user interaction.
- **Reduces possible configuration errors.** Because the majority of the current operating system, IIS, and Web site configuration settings are retained during upgrade, fewer configuration errors result.

The disadvantages of upgrading the server that hosts your existing IIS Web sites and applications to IIS 6.0 include the following:

- **Retains previous versions of software.** The upgrade process upgrades only software components identified by Microsoft® Windows® Server 2003, Standard Edition; Windows® Server 2003, Enterprise Edition; and Windows® Server 2003, Web Edition. Any other installed software components, such as applications or application dynamic-link libraries (DLLs), remain unchanged.

  During the IT lifecycle of the server, you can install and subsequently remove many of these additional software components. The software removal process often leaves DLLs, other files, and folders on the system. Over time, these unused components can accumulate, consume available disk space, and potentially cause instability with existing applications.
- **Retains the previous registry configuration.** The upgrade process makes the appropriate modifications to the registry entries identified by Windows Server 2003. Registry entries are created by an application as it is installed, but are not removed when the application is removed. Cumulatively, these registry entries consume disk storage and can make troubleshooting registry-related problems more difficult because the unused registry entries can have similar naming conventions to active registry entries.

After upgrading a server to Windows Server 2003 and IIS 6.0, the server runs in *IIS 5.0 isolation mode*, which allows IIS 6.0 to emulate the memory and request processing model in IIS 5.0. However, to take advantage of the security and performance benefits of IIS 6.0, you need to configure the server to run in *worker process isolation mode*, which is the new IIS 6.0 memory and request processing model.

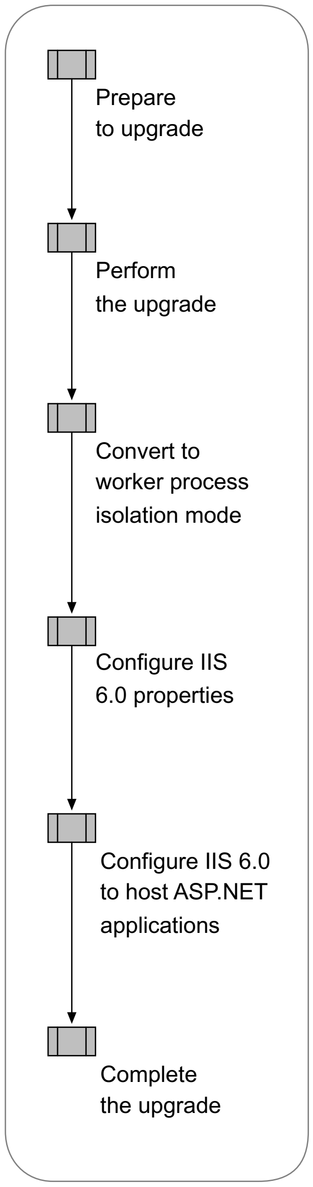
When upgrading existing IIS Web sites is not the best solution for your existing server, consider migrating your IIS Web sites to a newly installed server running IIS 6.0. For information about migrating your existing IIS Web sites to IIS 6.0, see "Migrating IIS Web Sites to IIS 6.0" in this book.

Process for Upgrading an IIS Server to IIS 6.0

The process for upgrading an existing IIS server hosting IIS Web sites and applications includes preparing for the upgrade, as well as performing the upgrade. Upon completing the upgrade process outlined in this chapter, you can then configure IIS 6.0 to run your applications in worker process isolation mode. Lastly, if the newly upgraded IIS 6.0 server hosts existing ASP.NET applications, you must configure IIS to use the proper version of the Microsoft .NET Framework for these applications.

Figure 5.1 illustrates the process for upgrading an existing IIS server hosting Web sites and applications to IIS 6.0.

Figure 5.1 Upgrading an Existing IIS Server to IIS 6.0


Prepare to upgrade
Perform the upgrade
Convert to worker process isolation mode
Configure IIS 6.0 properties
Configure IIS 6.0 to host ASP.NET applications
Complete the upgrade

In addition to upgrading the existing operating system and IIS, the upgrade process automatically upgrades any components that are a part Windows Server 2003, such as FrontPage® Server Extensions from Microsoft. As a part of the upgrade process, you need to evaluate the compatibility of the software that is installed on your existing server, including device drivers, other Windows components, and software, before performing the upgrade of the server.

Tip
To upgrade a Web farm, use the process described in this chapter to upgrade each server in the Web farm.

Depending on your familiarity with IIS and its upgrade process, you might require less information to complete the upgrade process. To facilitate the fastest possible upgrade, the following quick-start guide is provided. You can use this guide to help identify the steps of the upgrade process that you need additional information to complete, and then you can skip the information with which you are already familiar. In addition, all of the procedures that are required to complete the upgrade process are documented in "IIS Deployment Procedures" in this book.

Prepare to Upgrade

1. Determine compatibility with Windows Server 2003.
2. Identify and compensate for changes to IIS 6.0:
   - Ensure that the World Wide Web Publishing Service (WWW service) is enabled after upgrade.
   - Compensate for changes to IIS components.
3. Determine application compatibility with worker process isolation mode by completing the following steps:
   - Evaluate the benefits of worker process isolation mode.
   - Evaluate application changes required for worker process isolation mode.
   - Evaluate management and provisioning script changes required for worker process isolation mode.
   - Verify application compatibility with worker process isolation mode in a lab.
4. Determine application compatibility with the .NET Framework.

Perform the Upgrade

1. Back up the Web server before upgrade.
2. Verify that clients are not accessing Web sites.
3. Prevent the WWW service from being disabled.
4. Upgrade the Web server to IIS 6.0.
5. Verify that the operating system upgrade was successful.
6. Back up the IIS 6.0 metabase after upgrade.

Convert to Worker Process Isolation Mode

1. Document the current application isolation settings.
2. Configure IIS to run in worker process isolation mode.
3. Configure application isolation settings in IIS 6.0.

Configure IIS 6.0 Properties

1. Enable the WWW service.
2. Configure Web service extensions.
3. Configure Multipurpose Internet Mail Extensions (MIME) types.
4. Modify references to IIS 6.0 metabase properties that have changed or are no longer supported.
5. Upgrade FrontPage extended Web sites.
6. Determine whether to run the IIS Lockdown Tool and UrlScan.
7. Make the following security-related configuration changes:
    - Enable essential IIS components and services.
    - Remove any unnecessary IIS virtual directories.
    - Configure the anonymous user identity.

Configure IIS to Host ASP.NET Applications

1. Configure IIS to use the correct version of the .NET Framework.
2. Configure the .NET Framework.
3. Review how ASP.NET applications run in each application isolation mode.
4. Migrate Machine.config attributes to IIS 6.0 metabase property settings by completing the following steps:
    - Migrate recycling-related attributes.
    - Migrate performance-related attributes.
    - Migrate health-related attributes.
    - Migrate identity-related attributes.

Complete the Upgrade

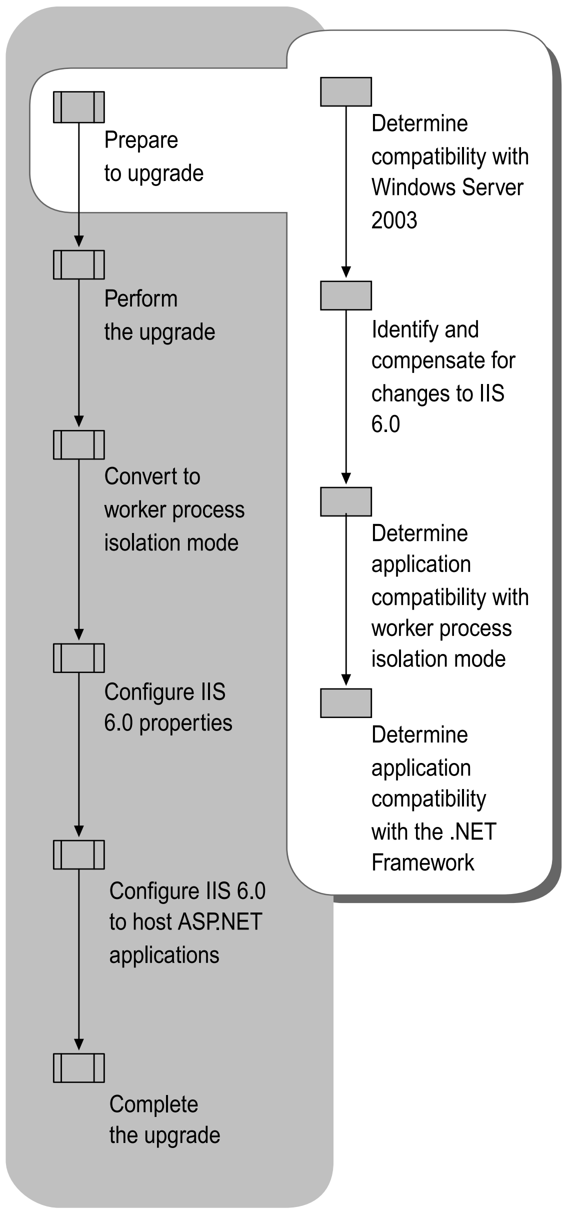
1. Verify that the Web sites and applications run properly.
2. Back up the Web server.
3. Enable client access after upgrade is complete.

Preparing to Upgrade

Before upgrading your existing server running IIS, ensure that Windows Server 2003 supports all of the hardware devices that are on the server. You must also ensure that your Web sites and applications are compatible with IIS 6.0, Windows Server 2003, and worker process isolation mode. Lastly, you must determine whether any of your existing ASP.NET applications are compatible with version 1.1 of the .NET Framework that is installed with Windows Server 2003.

Figure 5.2 illustrates the process for preparing to upgrade an IIS server to IIS 6.0.

Figure 5.2 Preparing to Upgrade


Prepare to upgrade
Perform the upgrade
Convert to worker process isolation mode
Configure IIS 6.0 properties
Configure IIS 6.0 to host ASP.NET applications
Complete the upgrade
Determine compatibility with Windows Server 2003
Identify and compensate for changes to IIS 6.0
Determine application compatibility with worker process isolation mode
Determine application compatibility with the .NET Framework

Determining Compatibility with Windows Server 2003

At a minimum, your existing system hardware and software must be compatible with Windows Server 2003 before you upgrade your existing server to IIS 6.0. The upgrade process identifies any software or hardware devices that are incompatible with Windows Server 2003 and allows you to stop the upgrade process after notifying you of any incompatible software or device drivers.

Determining the Compatibility of Existing Windows Servers with Upgrade

Your current Windows server operating system can limit which version of Windows Server 2003 that you can upgrade to. Table 5.1 lists the existing Windows server operating systems and the versions of Windows Server 2003 that are supported for upgrade.

Table 5.1 Windows Server Upgrades Supported by Windows Server 2003

Existing Windows Server	Standard	Enterprise	Web
Windows NT Server 4.0	●	●	
Windows NT Server 4.0, Enterprise Edition		●	
Windows 2000 Server	●	●	
Windows 2000 Advanced Server		●	
Windows Server 2003, Standard Edition		●	
Windows Server 2003, Web Edition			●

If you want to move from an existing Windows server operating system to a version of Windows Server 2003 that is not supported by upgrade, migrate the existing Web server to a new Web server. For more information about migrating Web sites to a new Web server running IIS 6.0, see “Migrating IIS Web Sites to IIS 6.0” in this book.

Determining the Compatibility of Existing Hardware

In most cases, your existing hardware will be compatible with Windows Server 2003. The most common hardware incompatibility is a device driver that is no longer supported. When devices are no longer supported, remove the existing devices and install an equivalent device that is supported by Windows Server 2003. It is also important that you have the latest BIOS version that is available from your computer manufacturer.

For example, the Web server might have an ISA network adapter that is no longer supported in Windows Server 2003. In this situation, you can obtain and install a new network adapter that is compatible with Windows Server 2003.

For more information about the hardware devices that are supported on Windows Server 2003, see the *Hardware Compatibility List* on the product disc or see the Hardware Driver Quality link on the Web Resources page at http://www.microsoft.com/windows/reskits/webresources.

Determining the Compatibility of Existing Software

Before upgrading, you need to consider the compatibility of your existing applications, or other software that is installed on your server, with Windows Server 2003. This includes software and tools from manufacturers other than Microsoft, as well as Microsoft server products that do not ship with the Windows operating system. Make sure that you have the latest versions of all pre-existing software or service packs that are compatible with Windows Server 2003.

The most common software incompatibilities are caused when your application depends on software from another manufacturer that does not support Windows Server 2003. Or, you might have applications that were designed to run on the Microsoft Windows NT® Server 4.0 or Windows 2000 Server operating systems, which reference application programming interfaces (APIs) that have been changed or removed in Windows Server 2003.

To help you determine the compatibility of your existing software with Windows Server 2003, use the Windows Application Compatibility Toolkit before upgrading your server. To download the latest version of the Windows Application Compatibility Toolkit, see the Windows Application Compatibility link on the Web Resources page at http://www.microsoft.com/windows/reskits/webresources.

For the latest information about application compatibility with Windows Server 2003, see the Windows Server 2003 link on the Web Resources page at http://www.microsoft.com/windows/reskits/webresources.

Identifying and Compensating for Changes to IIS 6.0

Although most of the changes to IIS that are made during the upgrade process have little affect, some of the changes can affect the Web sites and applications, as well as the administration of the Web server. Before upgrading your existing IIS server, review the changes that will be made to IIS during upgrade and determine whether these changes can affect your Web sites and applications. Otherwise, after the upgrade, applications might not function as they were originally designed to. In addition, you need to identify the changes made to IIS during upgrade so that you can properly administer and configure the Web server upon completion of the upgrade.

Identify the changes incurred during the upgrade by completing the following steps:

1. Select a method to ensure that the WWW service is enabled after upgrading a Web server running Windows 2000 Server.
2. Identify changes made to IIS components during upgrade.

Ensuring That the WWW Service is Enabled After Upgrade

For security reasons, the World Wide Web Publishing Service (WWW service) is disabled when you upgrade from Windows 2000 Server or Windows 2000 Advanced Server to Windows Server 2003 or Windows Server 2003, Enterprise Edition, respectively.. The WWW service is disabled on upgrade because IIS 5.0 was installed by default during the Windows 2000 Server installation, and, in many cases, the WWW service was not used. To decrease the attack surface of the server, IIS 6.0 is disabled by default when you upgrade from Windows 2000 Server to Windows Server 2003 unless you do one of the following:

- For manual or unattended upgrades, run the IIS Lockdown Tool on your Windows 2000 Server before starting the upgrade process.
- For manual or unattended upgrades, add the registry entry **RetainW3SVCStatus** to the registry.
- For unattended upgrades, add the entry "DisableWebServiceOnUpgrade = False" in the unattended install script.
- Continue with the upgrade and enable the WWW service after upgrade. For more information about enabling the WWW service, see "Preventing the WWW Service from Being Disabled" later in this chapter.

Note

Because IIS 4.0 is not installed by default on computers running Windows NT Server 4.0, the Windows Server 2003 upgrade does not disable the WWW service when upgrading from IIS 4.0 to IIS 6.0.

Compensating for Changes to IIS Components

Because of increased security measures, which remove any unnecessary Web sites or configuration settings from the Web server, the upgrade process can remove or change the existing IIS configuration. The upgrade process ensures that you take the necessary steps to accommodate these changes so that your applications run as they did before the upgrade.

Table 5.2 lists the changes in the behavior, administration, and support of IIS 6.0

Table 5.2 Changes to IIS Components After Upgrade

Before Upgrade	After Upgrade
The IISAdmin virtual directory exists.	**The IISAdmin virtual directory is removed and a default Web page is placed in the directory.** [1]
Web service extensions are mapped to 404.dll.[2]	**All extensions mapped to 404.dll are disabled. All other extensions are enabled.**

(continued)

Table 5.2 Changes to IIS Components After Upgrade *(continued)*

Before Upgrade	After Upgrade
Web Distributed Authoring and Versioning (WebDAV) is disabled by using NTFS file system permissions. 3	**WebDAV permissions are changed to allow proper operation, and WebDAV is added to the Web service extensions list as *prohibited*.**
Indexing Service is disabled by using NTFS permissions. 3	**Indexing Service remains disabled with the same NTFS permissions.**
Web services hosted on IIS 5.0 are enabled.	**Web services supported by the WWW service are disabled.** 4
FrontPage 2000 Server Extensions included lightweight server support.	**FrontPage 2002 Server Extensions does not include lightweight server support.**
FrontPage 2000 Server Extensions support.	**FrontPage 2000 Server Extensions is not supported. Upgrade all Web sites to FrontPage 2002 Server Extensions.**

1 Directory is removed if the IIS Lockdown Tool is run before upgrade.

2 The IIS Lockdown Tool maps all Web service extensions, used by IIS, to 404.dll to disable the extensions. Additional Web service extensions installed beyond the default IIS Web service extensions are not mapped to 404.dll.

3 The IIS Lockdown Tool disables WebDAV and the Indexing Service by changing the permissions.

4 If the IIS Lockdown Tool runs before upgrade or the appropriate registry modifications are made, the Web services remain enabled after upgrade. For more information about preventing the Web services from being disabled, see "Preventing the WWW service from Being Disabled" later in this chapter.

As a part of the architectural changes to IIS 6.0, many of the features that were previously implemented as Internet Server API (ISAPI) filters are now integrated into IIS directly. Table 5.3 lists the ISAPI filters that have been removed and have the functionality of the ISAPI filter implemented in IIS 6.0

Table 5.3 ISAPI Filters Removed During Upgrade

ISAPI Filter Removed	ISAPI Filter File Name	Implemention in IIS 6.0
Compression	**Sspifilt.dll**	**Integrated into IIS 6.0 directly**
SSL encryption	**Compfilt.dll**	**Integrated into IIS 6.0 directly**
Kerberos authentication	**Md5filt.dll**	**Integrated into IIS 6.0 directly**

Determining Application Compatibility with Worker Process Isolation Mode

IIS 6.0 can run in one of two distinct modes of operation, which are called application isolation modes. *Application isolation* is the separation of applications by process boundaries that prevent the applications from affecting one another, and it is configured differently for each of the two IIS application isolation modes: IIS 5.0 isolation mode and worker process isolation mode.

Worker process isolation mode uses the redesigned architecture for IIS 6.0. This application isolation mode runs all application code in an isolated environment. However, unlike earlier versions of IIS, IIS 6.0 provides isolation without a performance penalty because fewer processor instructions are run when switching from one application pool to another. Worker process isolation mode is compatible with most existing Web sites and applications. Whenever possible, run IIS 6.0 in worker process isolation mode benefit from the enhanced performance and security in IIS 6.0.

IIS 5.0 isolation mode provides compatibility for applications that depend upon the process behavior and memory model of IIS 5.0. Run IIS in this mode only when a Web site or application cannot run in worker process isolation mode, and run it only until the compatibility issues are resolved.

Important

IIS 6.0 cannot run both application isolation modes simultaneously on the same server. Therefore, on a single server running IIS 6.0, you cannot run some Web applications in worker process isolation mode and others in IIS 5.0 isolation mode. If you have applications that require separate modes, you must run them on separate servers.

After you perform an upgrade of the operating system, IIS is configured to run in IIS 5.0 isolation mode by default. Before configuring IIS 6.0 to run in worker process isolation mode, you should evaluate whether your Web sites and applications are compatible with worker process isolation mode. Most of the compatibility issues with IIS 6.0 occur when configuring IIS 6.0 to run in worker process isolation mode.

One of the most common reasons for incompatibility with worker process isolation mode is that applications do not recognize custom ISAPI extensions or DLLs that depend on the memory and request processing models used by earlier versions of IIS. Determine application compatibility in your lab before upgrading your existing IIS server, and if you determine that your applications are not compatible with worker process isolation mode, you can run the applications in IIS 5.0 isolation mode.

Note
Identifying a complete list of potential incompatibilities that applications can experience with worker process isolation mode is beyond the scope of this book. Even after following the guidelines in this chapter, you need to verify in a lab whether your Web sites and applications are compatible with worker process isolation mode.

Determine the compatibility of an application with worker process isolation mode by completing the following steps:

1. Evaluate the benefits of moving to worker process isolation mode.
2. Evaluate the application changes that are required so that the applications can run in worker process isolation mode.
3. Evaluate the management and provisioning script changes that are required to set up programs and provisioning scripts in worker process isolation mode.
4. Verify the compatibility of the application with worker process isolation mode in a lab.

Evaluating the Benefits of Worker Process Isolation Mode

Worker process isolation mode provides higher levels of security and availability for Web sites and applications than IIS 5.0 isolation mode. Therefore, it is recommended that you configure IIS 6.0 to run in worker process isolation mode.

Worker process isolation mode provides the following improvements to IIS 6.0.

Security Enhancements

IIS 6.0 includes a variety of security features and technologies that help ensure the integrity of your Web site content, and of the data that is transmitted through your sites. The following security enhancement is only available when IIS 6.0 is running in worker process isolation mode.

Default process identity for Web sites and applications is set to NetworkService

In IIS 5.0 isolation mode, the default process identity is LocalSystem, which enables access to, and the ability to alter, nearly all of the resources on the Web server. The potential of attacks is reduced in worker process isolation mode because Web sites and applications run under the NetworkService identity. The NetworkService identity is granted less privileges, which helps prevent an attack from compromising the Web server, which is possible with the LocalSystem identity.

Performance and Scaling Enhancements

Future growth in the utilization of your Web sites and applications requires increased performance and scalability of Web servers. By increasing the speed at which HTTP requests can be processed and by allowing more applications and sites to run on one Web server, the number of Web servers that you need to host a site is reduced. The following are a few of the performance improvements included in worker process isolation mode.

Support for processor affinity for worker processes in an application pool

You can configure all of the worker processes in an application pool to have affinity with specific processors in a multiprocessor or server. Processor affinity allows the worker processes to take advantage of more frequent processor caching (Level 1 or Level 2).

Elimination of inactive worker processes and reclamation of unused resources

You can configure application pools to have worker processes request a shutdown if they are idle for a certain amount of time. This can free unused resources for other active worker processes. New worker processes are then started only when they are needed.

Distributing client connections across multiple worker processes

You can configure an application pool to have more than one worker process servicing client connections, also known as a *Web garden*. Because there are multiple worker processes, the incoming client connections are distributed across the worker processes and throughput is not constrained by a single worker process.

Ability to Isolate Web sites and applications from each other

You can isolate applications without incurring a performance penalty.

Availability Enhancements

Because worker process boundaries isolate the applications in an application pool from the applications in other application pools, if an application fails, it does not affect the availability of other applications running on the server. Deploying applications in application pools is a primary advantage of running IIS 6.0 in worker process isolation mode.

Reduced number of Web server restarts required when administering Web sites and applications

Many of the common operation tasks do not force the restart of the server or the Web service restart. These tasks, such as upgrading site content or components, debugging Web applications, or dealing with faulty Web applications, can be performed without affecting service to other sites or applications on the server.

A fault-tolerant request processing model for Web sites and applications

In IIS 5.0 isolation mode, each Web site or application has only one worker process. However, in worker process isolation mode, you can create a *Web garden* by configuring a number of worker processes to share the processing. The benefit of a Web garden is that if one worker process stops responding, other worker processes are available to accept and process requests.

Isolation of failed worker processes from healthy worker processes

In worker process isolation mode, IIS can determine that a worker process is failing and start a new worker process to replace the failing worker process. Because a new worker process is created before the old worker process terminates, users requesting the Web site or application experience no interruption of service. After IIS creates the new worker process, the failed worker process can be separated, or *orphaned*, from the application pool. The advantage of orphaning a worker process rather than terminating it is that debugging can be performed on the orphaned worker process.

Health monitoring of Web sites and applications

In worker process isolation mode, you can configure an application pool to monitor not only the health of the entire application pool, but also individual worker processes servicing the application pool. Monitoring the health of a worker process allows IIS to detect that a worker process is unable to serve requests and to take corrective action, such as recycling the failed worker process.

In addition, worker process isolation supports other responses when a failed worker process or application pool is detected. For example, IIS can attach a debugger to an orphaned worker process or notify an administrator that an application pool has failed due to rapid-fail protection.

Prevention of Web sites or applications that fail quickly from consuming system resources

In some cases, availability can be affected by Web sites and applications that fail very quickly, are automatically restarted, and then fail quickly again. The endless cycle of failure and restarting can consume system resources, causing other Web sites and applications to experience denial of services because of system resource shortages.

Worker process isolation mode includes *rapid-fail protection* that stops an application pool when too many of the worker processes assigned to an application pool are found to be unhealthy within a specified period of time.

Automatic restart of poorly performing Web sites and applications

Some Web sites and applications have memory leaks, are poorly coded, or have other unidentified problems. In IIS 5.0 isolation mode, these applications can force you to restart the entire Web server. The recycling feature in worker process isolation mode can periodically restart the worker processes in an application pool to manage faulty applications. Worker processes can be scheduled to restart based on several options, such as elapsed time or the number of requests served.

Evaluating Application Changes Required for Worker Process Isolation Mode

In most cases, the existing Web sites and applications can be hosted in worker process isolation mode without modification. However, the following are known application issues that can create incompatibilities with worker process isolation mode and require you to run IIS 6.0 in IIS 5.0 isolation mode.

- **Requires Inetinfo.exe.** When the application must run in the same process with Inetinfo.exe, IIS must be configured to run in IIS 5.0 isolation mode because Inetinfo.exe does not process Web requests in worker process isolation mode. In worker process isolation mode, W3wp.exe processes Web requests.
- **Requires Dllhost.exe.** When the application depends on Dllhost.exe to process Web requests for the application, IIS must be configured to run in IIS 5.0 isolation mode because Dllhost.exe is not available in worker process isolation mode.
- **Requires read raw data filters.** If the application uses an ISAPI raw data filter, IIS must be configured to run in IIS 5.0 isolation mode.
- **Requires version 1.0 of the .NET Framework.** When the application requires version 1.0 of the .NET Framework, IIS must be configured to run in IIS 5.0 isolation mode because version 1.0 of the .NET Framework is only supported in IIS 5.0 isolation mode.
- **Requires ISAPI filters or extensions that are incompatible with worker process isolation mode.** When the application requires ISAPI filters or extensions that are incompatible with worker process isolation mode, IIS must be configured to run in IIS 5.0 isolation mode. ISAPI filters or extensions might be incompatible if the filter or extension has one of the following characteristics:
    - Runs in multiple instances and expects to be recycled by using the recycling provided in IIS 5.0.
    - Expects to have exclusive lock on a resource, such as a log file.

If any of the Web sites and applications running on the existing Web server has one or more of these characteristics, then do one of the following:

- Configure IIS to run in IIS 5.0 isolation mode to ensure Web site and application compatibility. If the Web sites and applications need to be hosted in IIS 5.0 isolation mode, then the upgrade process is complete.
- Modify the existing applications to remove the dependencies.

Evaluating Management and Provisioning Script Changes Required for Worker Process Isolation Mode

When management or provisioning scripts exist for your Web sites and applications, you might need to modify them so that they properly set up the Web sites and applications for the application isolation mode that is running on the Web server — IIS 5.0 isolation mode or worker process isolation mode. If you do not modify your management and provisioning scripts as required, you will be unable to use them to install and configure your Web sites and applications on IIS 6.0.

For example, you might have a provisioning script that uses Microsoft Active Directory® Service Interfaces (ADSI) to create a Web site and configure the site for High isolation in IIS 5.0 isolation mode, which does not exist in worker process isolation mode. After upgrade when the Web server is running in worker process isolation mode, you need to modify the script to create application pools instead.

Common modifications that might be necessary to your setup programs and provisioning scripts include:

- When the script installs an ISAPI extension or filter, you might need to modify the script to add an entry for the ISAPI extension or filter to the Web service extensions restriction list and set the status of the entry to **Allowed**.

  For more information about modifying your setup programs or provisioning scripts to install and enable an ISAPI extension or filter, see "Configuring Web Service Extensions" later in this chapter.

- When the script installs an application that contains dynamic content, you might need to modify the script to set the status of the appropriate Web service extensions to **Allowed**, so that IIS allows the dynamic content to run.

  For more information about modifying your setup programs or provisioning scripts to enable dynamic content, see "Configuring Web Service Extensions" later in this chapter.

- When the script installs a Web site or application that runs in High isolation in IIS 5.0, you might need to modify the script to create an application pool and configures the application pool with settings that are comparable to the original IIS 5.0 isolation settings.

  For more information about modifying your setup programs or provisioning scripts to create and configure application pools, see "Configuring Application Isolation Settings in Worker Process Isolation Mode" later in this chapter.

- When the script references metabase properties, you might need to modify the script if it references metabase properties that have changed or are no longer supported in IIS 6.0.

  For more information about modifying your setup programs or provisioning scripts to reference the proper metabase properties in IIS 6.0, see "Modifying References to IIS 6.0 Metabase Properties" later in this chapter.

Verifying Application Compatibility with Worker Process Isolation Mode in a Lab

After modifying your Web sites, applications, setup programs, and provisioning scripts to be compatible with worker process isolation mode, you need to test your modifications in a lab. Be sure to test for compatibility before performing the upgrade process on a production Web server.

Verify the compatibility of your Web sites, applications, setup programs, and provisioning scripts with worker process isolation mode in a lab by completing the following steps:

1. Make an image backup of the Web server you are going to upgrade.
2. Restore the backup to a Web server in your lab, referred to hereafter as a *test Web server*. Ensure that the test Web server is not connected to your production network to prevent any problems encountered during the upgrade from affecting your production network.
3. Perform an upgrade on the test Web server.
4. Configure the Web server to run in worker processor isolation mode.
5. Make the necessary modifications to the Web sites, applications, setup programs, and provisioning scripts so that they are compatible with worker process isolation mode.
6. Verify the Web sites, applications, setup programs, and provisioning scripts run correctly on the test Web server.

For more information about setting up a test lab, see “Designing a Test Environment” in *Planning, Testing, and Piloting Deployment Projects* of the *Microsoft® Windows® Server 2003 Deployment Kit*.

Determining Application Compatibility with the .NET Framework

For a successful upgrade to Windows Server 2003 and IIS 6.0, you need to determine whether your ASP.NET applications are dependent on specific versions of the .NET framework. Windows Server 2003 includes version 1.1 of the .NET Framework, although most of the .NET applications that were developed before the release of Windows Server 2003 were designed to run on version 1.0 of the .NET framework.

Note

Version 1.0 of the .NET Framework is only supported in IIS 5.0 isolation mode. Therefore, you can only run version 1.0 and version 1.1 of the .NET Framework on the same server when IIS 6.0 is configured to run in IIS 5.0 isolation mode. Version 1.1 of the .NET Framework is supported in IIS 5.0 isolation mode or worker process isolation mode.

Typically, ASP.NET applications running on version 1.0 of the .NET Framework are compatible with version 1.1 of the .NET Framework. However, there might be some incompatibilities, the majority of which are security related and can be corrected by configuring the .NET Framework to be less restrictive.

For example, in the .NET Framework version 1.0, the .NET Framework only examines the **SQLPermission.AllowBlankPassword** attribute if the user includes the password keyword in their connection string. If an administrator or user sets “SQLPermision.AllowBlankPassword=False”, it is possible to specify a connection string like “server=(localhost);uid=sam” and succeed. In the .NET Framework version 1.1, this connection string fails.

You can use both version 1.0 and version 1.1 of the .NET Framework on the same server running IIS 6.0, also known as *side-by-side configuration*. Side-by-side configuration allows you to run a mixture of applications that require version 1.0 or version 1.1 of the .NET Framework. During your lab testing, determine the version of the .NET Framework that is required by your applications.

For a current list of possible compatibility issues when upgrading from version 1.0 to version 1.1 of the .NET Framework, see the Compatibility Considerations and Version Changes link on the Web Resources page at http://www.microsoft.com/windows/reskits/webresources. For more information about using multiple versions of the .NET Framework in side-by-side configuration, see “Configuring IIS 6.0 to Use the Correct Version of the .NET Framework” later in this chapter.

Important

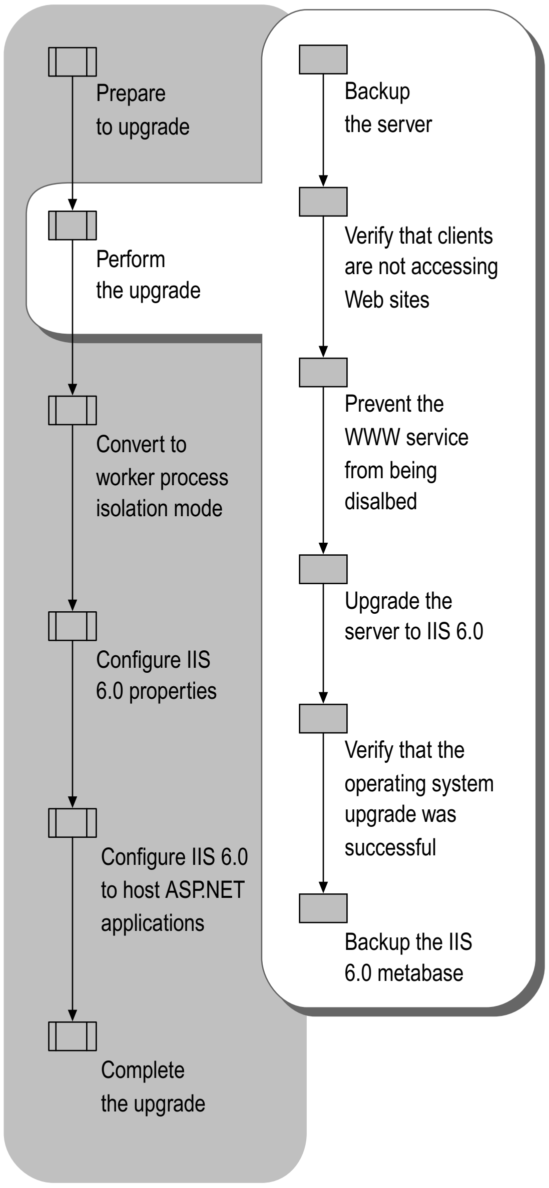
Before upgrading the script maps for your ASP.NET applications, point to version 1.0 of the .NET Framework. Upon completion of the upgrade, version 1.1 of the .NET Framework is installed and the script maps are modified to use version 1.1. When your applications require version 1.0, you need to modify the script maps to use version 1.0, rather than version 1.1. For more information about how to modify the script maps to use version 1.0 of the .NET Framework, see “Configuring IIS 6.0 to Use the Correct Version of the .NET Framework” later in this chapter.

Performing the Upgrade

Before upgrading your existing server to Windows Server 2003 and IIS 6.0, you must back up the server, verify that clients are not accessing Web sites on the server, and optionally prevent the WWW service from being disabled. Then upgrade the Web server to Windows Server 2003 and IIS 6.0. Finally, verify that the upgrade to Windows Server 2003 completed successfully.

Figure 5.3 illustrates the process for performing to upgrade an IIS server to IIS 6.0.

Figure 5.3 Performing the Upgrade


Prepare to upgrade
Perform the upgrade
Convert to worker process isolation mode
Configure IIS 6.0 properties
Configure IIS 6.0 to host ASP.NET applications
Complete the upgrade
Backup the server
Verify that clients are not accessing Web sites
Prevent the WWW service from being disalbed
Upgrade the server to IIS 6.0
Verify that the operating system upgrade was successful
Backup the IIS 6.0 metabase

Backing Up the Server

Before you change any of the configuration settings on the existing Web server, perform a complete image backup. The purpose of this image backup is to provide a point-in-time snapshot of the Web server. If unforeseen problems occur during the upgrade, you can use this backup to restore the Web server to a known configuration.

Important

Do not continue with the upgrade process unless you have a successful backup of the entire Web server or you have another Web server that has the same Web sites and applications. Otherwise, you can lose Web sites, applications, or data stored on the Web server.

For more information about how to back up the Web server, see "Back Up and Restore the Web Server to a File or Tape" in "IIS Deployment Procedures" in this book.

Verifying That Clients Are Not Accessing Web Sites

Before you upgrade the existing server, ensure that no active client sessions are running. Upgrading the server without doing this can result in abnormally terminated client processes and a loss of information.

Verify that clients are no longer accessing Web or File Transfer Protocol (FTP) sites by completing the following steps:

1. Prevent new clients from accessing the sites by pausing the sites.

   For more information about how to pause Web or FTP sites, see "Pause Web or FTP Sites" in "IIS Deployment Procedures" in this book.

2. Enable monitoring of active Web and FTP connections.

   For more information about how to monitor the active Web and FTP connections, see "Monitor Active Web and FTP Connections" in "IIS Deployment Procedures" in this book.

3. When the number of active Web and FTP counters is zero, disable the network adapter that clients use to access the Web server.

   For more information about how to disable the network adapter used by clients, see "Disable Network Adapters" in "IIS Deployment Procedures" in this book.

Preventing the WWW Service from Being Disabled

Earlier in the deployment process, you selected the method for enabling the WWW service after upgrading from a server running Windows 2000 Server and IIS 5.0. If you decided to prevent the WWW service from being disabled after completion of the upgrade, you must do one of the following:

- Modify the registry or unattended setup script.
- Run the IIS Lockdown Tool.

Note

If you are upgrading a Web server running Windows NT Server 4.0 and IIS 4.0, you do not need to run the IIS Lockdown Tool before upgrade.

Modifying the Registry or Unattended Setup Script

Prevent the WWW service from being disabled during upgrade by doing one of the following:

- Create the registry entry **do_not_disable** in the subkey HKLM\SYSTEM\CurrentControlSet\Services\W3SVC\RetainW3SVCStatus as data type REG_DWORD with a value of 0x1. For more information about how to configure the registry, see “Configure the Registry” in “IIS Deployment Procedures” in this book.

Caution

Do not edit the registry unless you have no alternative. The registry editor bypasses standard safeguards, allowing settings that can damage your system, or even require you to reinstall Windows. If you must edit the registry, back it up first and see the Registry Reference on the *Microsoft Windows Server 2003 Deployment Kit* companion CD or on the Web at http://www.microsoft.com/reskit.

- Include the entry “DisableWebServiceOnUpgrade = False” in the [InternetServer] section in an unattended installation script.

Running the IIS Lockdown Tool

The IIS Lockdown Tool is designed to help secure earlier versions of IIS by doing the following:

- Preventing the WWW service from being disabled after upgrade on Web servers that are currently running Windows 2000 Server and IIS 5.0. Disabling the WWW service prevents any Web sites or applications from functioning.
- Helping to secure the existing Web server by disabling or removing unnecessary features that are present in IIS 4.0 and IIS 5.0 installations. These features would otherwise remain on the Web server after upgrading, leaving it vulnerable to attacks.

The IIS Lockdown Tool works by turning off unnecessary features, thereby reducing the attack surface that is available to malicious users. To provide multiple layers of protection for an in-depth defense against potential attackers, the IIS Lockdown Tool includes UrlScan and customized security templates based on supported server roles.

Note

The WWW service is enabled after the upgrade process is complete on servers running IIS 4.0 because IIS is installed by default in Windows 2000 Server.

Custom Server Configurations with the IIS Lockdown Tool

The IIS Lockdown Tool secures the existing IIS server by performing one or more of the following user-specified transactions:

- Enabling or disabling IIS services such as the WWW service, the FTP service, or the Simple Mail Transfer Protocol (SMTP) service
- Removing services that are disabled
- Disabling active Active Server Pages (ASP) applications on the server
- Disabling optional components, including:
  - Index Server Web interface
  - Server-side includes (SSI)
  - Internet Data Connector (IDC)
  - Internet printing
  - HTR scripting
  - WebDAV
- Disabling anonymous user access to the server by denying:
  - Execute permissions on the operating system executables and DLLs
  - Write permissions on all Web site content directories

- Removing unnecessary virtual directories, including:
  - IIS Samples
  - Scripts
  - Microsoft Data Access Components (MDAC)
  - IIS Admin
- Installing UrlScan

Tip
The IIS Lockdown Tool helps secure IIS. However, to maintain IIS security, install all available security patches and hotfixes to help protect against known security vulnerabilities.

Server Roles in the IIS Lockdown Tool

Depending upon the applications that you are hosting and the software that you are using on the existing IIS server, select the server role that most closely corresponds to the server you are upgrading. The IIS Lockdown Tool uses the specified server role to determine the appropriate actions to configure the existing IIS server.

Regardless of the server role selected, UrlScan is not required for the purposes of upgrade. UrlScan can be installed by using the IIS Lockdown Tool or separately. For more information about determining whether you need to run UrlScan after upgrade, see "Determining Whether to Run the IIS Lockdown Tool and UrlScan" later in this chapter.

The server roles that are included in the IIS Lockdown Tool include the following:

- Small Business Server for Windows NT Server 4.0
- Small Business Server 2000
- Exchange Server 5.5
- Exchange Server 2000
- Microsoft SharePoint™ Portal Server
- FrontPage Server Extensions
- SharePoint Team Services
- BizTalk® Server 2000
- Commerce Server 2000
- Proxy Server
- Static Web server
- Dynamic Web server (ASP-enabled)

- Server that does not require IIS
- Other (a server that does not match any of the roles in this list)

Each of the server roles in the IIS Lockdown Tool secures the Web server by performing a different combination of the security configuration changes listed earlier. For example, if you select the Exchange Server 2000 (OWA, PF Management, IM, SMTP, NNTP) server role, then FTP is disabled, and SMTP and NNTP are enabled. However, if you select the SharePoint Portal Server, server role then FTP, SMTP, and NNTP are disabled. To determine the configuration performed by each server role, review the contents of the IisLockd.ini file in the same folder that contains the IIS Lockdown Tool.

After running the IIS Lockdown Tool, thoroughly test the server before upgrading to ensure that your Web sites and applications function as they did before. If you find that the configuration settings adversely affect your applications, run the IIS Lockdown Tool again to undo the changes that were made. If you are uncertain whether the IIS Lockdown Tool has been run on the server, you can run the tool again without adversely affecting the system.

Tip
When the IIS Lockdown Tool locks down a server, it creates a log file named Oblt-log.log and saves it in the folder that contains IISLockd.exe. This file contains information about every action the IIS Lockdown Tool implemented on the system.

Administrators can run the IIS Lockdown Tool unattended, allowing consistent configuration across many servers through unattended scripts. For more information about running the IIS Lockdown Tool unattended, see RunLockdUnattended.doc, which is located in the folder that contains the files for the IIS Lockdown Tool.

To download the latest version of the IIS Lockdown Tool, see the IIS Lockdown Tool link on the Web Resources page at http://www.microsoft.com/windows/reskits/webresources.

Upgrading the Server to IIS 6.0

The upgrade process from IIS 4.0 or IIS 5.0 to IIS 6.0 completes with minimal interaction because the majority of the IIS 4.0 or IIS 5.0 settings are retained. After the upgrade is complete, the Web sites and applications typically function as they did before the upgrade.

The upgrade process runs a number of compatibility tests before actually performing the upgrade. The compatibility tests relate directly to IIS 6.0 and determine the following:

- Whether system volume for the server is formatted with the NTFS file system
- Whether the IIS Lockdown Tool has been run on the server
- Whether the existing server is currently a node in a Microsoft server cluster

A dialog in the Windows Server 2003 upgrade notifies you if any potential compatibility issues exist. After reviewing the potential compatibility issues, you can abort the upgrade process and resolve the compatibility issues, or you can continue the process. None of the compatibility issues prevent the upgrade process from completing.

Verifying That the Operating System Upgrade Was Successful

In most cases, the upgrade process completes without any difficulties. However, before continuing with the upgrade process, verify that the operating system was upgraded successful by completing the following steps:

1. Open *systemroot*\Setuperr.log in Notepad and search for "IIS" to determine if any IIS-related errors occurred.

   During the upgrade process, the Windows Server 2003 upgrade creates an error log file (Setuperr.log) for the entire Windows Server 2003 operating system that records any errors encountered during upgrade. In addition to resolving any IIS related-problems encountered during upgrade, resolve any other upgrade problems listed in Setuperr.log before continuing with the upgrade process.

2. If you find any IIS-related errors, open *systemroot*\Iis6.log in Notepad and search the log file for "fail" to determine the source of the errors.

   During the upgrade of the IIS components, the Windows Server 2003 upgrade creates an IIS-specific log file, Iis6.log in the systemroot folder that records any IIS-specific errors encountered during upgrade. Iis6.log entries with the word "fail" reflect problems encountered during the upgrade process. Review Iis6.log and resolve the problems encountered during upgrade before continuing further in the upgrade process.

   Resolve any operating system upgrade-related problems before continuing further in the upgrade process. Subsequent steps in the upgrade process, such as moving the applications to worker process isolation mode, are dependent upon these issues being resolved.

Backing Up the IIS 6.0 Metabase

Metabase backups created with IIS 4.0 or IIS 5.0 cannot be restored on IIS 6.0. As a result, you cannot use any existing metabase backups of the Web server after the upgrade. After you have verified that IIS 6.0 hosts Web sites and applications as it did before the upgrade, back up the metabase before continuing with the upgrade process.

The remaining steps in the upgrade process focus on hosting the Web sites and applications in worker process isolation mode. Before changing the IIS configuration to worker process isolation mode, verify that you have a backup of the current IIS configuration by backing up the metabase.

In the event that an unforeseen problem occurs while you are configuring IIS 6.0 to run the applications in worker process isolation mode, you can restore the applications to a known operational state. This will provide a known starting place from which to retry the configuration of the server.

For information about how to back up the IIS 6.0 metabase, see "Back Up and Restore the IIS Metabase" in "IIS Deployment Procedures" in this book.

Note

Upon completion of the upgrade process, a backup of the IIS 6.0 metabase is automatically created. However, changes to the Web service extensions list are not reflected in that backup.

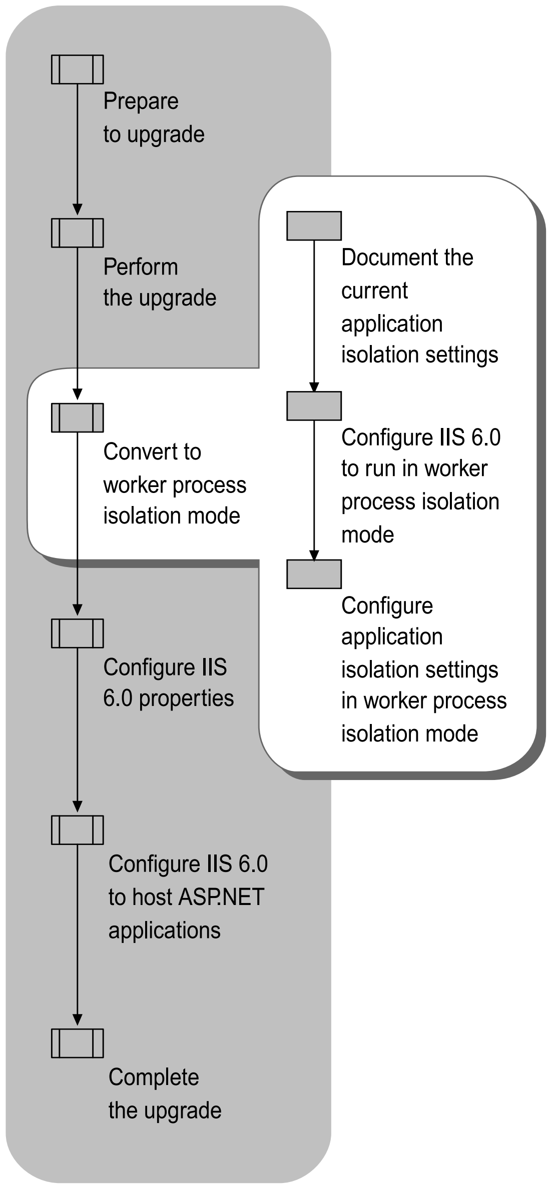
Converting to Worker Process Isolation Mode

When the upgrade process is complete, IIS 6.0 is configured to run in IIS 5.0 isolation mode. Although IIS 5.0 isolation mode provides compatibility with existing applications, it is unable to provide all the security, availability, and performance improvements in worker process isolation mode. To take full advantage of these improvements, you must configure IIS 6.0 to run in worker process isolation mode.

After upgrade, most Web sites and applications function in worker process isolation mode without modification. IIS can run in only one application isolation mode at a time — either IIS 5.0 isolation mode or worker process isolation mode. As a result, configuring IIS to run in worker process isolation mode affects all of the Web sites and applications hosted by IIS. Therefore, before configuring IIS to run in worker process isolation mode, you must determine whether your applications are compatible with this isolation mode. If the existing Web sites and applications are not compatible with worker process isolation mode, you can continue to "Configuring IIS for ASP.NET Applications" later in this chapter.

Figure 5.4 illustrates the process for converting the Web server to worker process isolation mode.

Figure 5.4 Converting to Worker Process Isolation Mode


Prepare to upgrade
Perform the upgrade
Convert to worker process isolation mode
Configure IIS 6.0 properties
Configure IIS 6.0 to host ASP.NET applications
Complete the upgrade
Document the current application isolation settings
Configure IIS 6.0 to run in worker process isolation mode
Configure application isolation settings in worker process isolation mode

For more information about worker process isolation mode and IIS 5.0 isolation mode, see "Determining Application Compatibility with Worker Process Isolation Mode" earlier in this chapter.

Documenting the Current Application Isolation Settings

Before you configure IIS 6.0 to run in worker process isolation mode, document the existing application isolation settings of the Web sites and applications that are hosted by IIS. Later in the upgrade process, you will use this baseline configuration for configuring the application isolation mode for your Web sites and applications.

For each Web site and application currently running on the server, document the following:

Application isolation settings

Earlier versions of IIS can host Web sites and applications in pooled or isolated process configurations. For information about how to view the current application isolation mode, see "View Application Isolation Configuration" in "IIS Deployment Procedures" in this book.

If you are running IIS 4.0 on Windows NT Server 4.0, your applications are isolated in one of the following ways:

- In-process (running in-process with Inetinfo.exe)
- Isolated (running under Microsoft Transaction Server [MTS])

If you are running IIS 5.0 on Windows 2000, your applications are isolated in one of the following ways:

- In-process (running in-process with Inetinfo.exe)
- Pooled (running in a pooled COM+ package)
- Isolated (running in an isolated COM+ package)

Process identity that is used by the Web site or application

Each Web site or application configured in high isolation, or pooled isolation, uses an *identity*. An identity is a user account that provides a security context for worker process servicing the Web site or application. The identity can be used to secure content, by using NTFS permissions, or data, such as data stored in Microsoft SQL Server™. For more information about how to view the identity for each Web site or application, see "View Web Site and Application Process Identities" in "IIS Deployment Procedures" in this book.

Note

All Web sites and applications without identities run under the security context of LocalSystem.

Configuring IIS 6.0 to Run in Worker Process Isolation Mode

As previously mentioned, configuring the IIS application isolation mode is a Web server-wide configuration setting that affects all Web sites and applications running on the Web server. The server is currently configured in IIS 5.0 isolation mode.

If you determined that one or more of your Web sites or applications are incompatible with worker process isolation mode, leave the Web server in IIS 5.0 isolation mode. Otherwise, configure IIS 6.0 to run in worker process isolation mode either in IIS Manager, or by setting the IIS metabase property **IIs5IsolationModeEnabled** to a value of **False**.

Note

If you configure IIS 6.0 to run in IIS 5.0 isolation mode and then decide to change the configuration back to worker process isolation mode, the original worker process isolation mode settings are retained. Similarly, if you configure IIS 6.0 to run in IIS 5.0 isolation mode, change to worker process isolation mode, and then change back to IIS 5.0 isolation mode, the IIS 5.0 isolation mode settings are retained.

For more information about how to configure IIS to run in worker process isolation mode or in IIS 5.0 isolation mode, see "Configure Application Isolation Modes" in "IIS Deployment Procedures" in this book. For more information about determining compatibility with worker process isolation mode, see "Evaluating Application Changes Required for Worker Process Isolation Mode" earlier in this chapter.

Configuring Application Isolation Settings in Worker Process Isolation Mode

Immediately after configuring IIS to run in worker process isolation mode, you need to configure the application isolation settings to closely approximate their configuration in IIS 5.0 isolation mode by assigning them to *application pools*. An application pool is a grouping of one or more Web sites or applications served by one or more worker processes. You might need to apply additional configurations so that the applications retain their original isolation settings.

After converting to worker process isolation mode, all applications run in the pre-existing application pool named "DefaultAppPool." If all of the applications run in the same process in the previous version of IIS, then they all are assigned to the default application pool.

However, if any one of the applications in the same application pool fails, the other applications can be adversely affected. For this reason it is recommended that you isolate your applications into separate application pools whenever possible.

Configure Web sites and applications to run in their own application pool by completing the following steps:

For each Web site or application configured in High isolation in IIS 5.0

1. Create a new application pool to be used by the Web site or application.

   For information about how to create application pools, see "Isolate Applications in Worker Process Isolation Mode" in "IIS Deployment Procedures" in this book.

2. If the Web site or application previously ran under an identity that is still required by the Web site or application, configure the application pool to use that same identity.

   For information about how to configure the identity for an application pool, see "Configure Application Pool Identity" in "IIS Deployment Procedures" in this book.

3. Assign the Web site or application to the new application pool.

   For information about how to assign the Web site to the new application pool, see "Isolate Applications in Worker Process Isolation Mode" in "IIS Deployment Procedures" in this book.

For each Web site or application configured in Low or Medium isolation in IIS 5.0

In earlier versions of IIS, applications ran in-process as DLLs in Inetinfo.exe (Low isolation) and the default process identity (account the application runs as) was LocalSystem. With worker process isolation mode in IIS 6.0, applications never run in Inetinfo.exe. However, any applications that are not explicitly assigned to an application pool are assigned to the default application pool, which runs under the NetworkService process identity by default. Because LocalSystem has an elevated security context, run Web sites and applications under the security context of the NetworkService account.

For each Web site or application that ran in Low or Medium isolation in IIS 5.0, do one of the following:

- When the Web site or application is able to function under the identity of the NetworkService account in the default application pool, continue to host the Web sites or applications in the default application pool, named "DefaultAppPool."
- When the Web site or application is unable to function under the identity of the NetworkService account in the default application pool, perform the following steps:
  1. Create a new application pool.
  2. Create a service account to be used as the identity for the application pool.

     For more information about how to create a service account to be used as an identity for an application pool, see "Create a Service Account" in "IIS Deployment Procedures" in this book.

3. Configure the application pool identity to use the service account.

   For more information about how to configure the identity for an application pool, see "Configure Application Pool Identity" in "IIS Deployment Procedures" in this book.

4. Place the Web site or application in the new application pool.

Example: Converting to Worker Process Isolation Mode

A fictitious organization, Contoso, has an existing IIS 5.0 Web server that hosts four Web applications. The administrator plans to upgrade the Web server to IIS 6.0, and has tested the applications for compatibility with IIS 6.0 worker process isolation mode and Windows Server 2003. Table 5.4 lists the existing configuration of the Web applications before upgrading to IIS 6.0.

Table 5.4 Configuration Before Upgrade

Application Name	Request Processing Model	Identity
Application-A	**Isolated**	**AppIdent-01**
Application-B	**In-process**	**LocalSystem**
Application-C	**Isolated**	**AppIdent-02**
Application-D	**In-process**	**LocalSystem**
Application-E	**In-process**	**LocalSystem**

After the upgrade, the administrator verified that the Web applications continued to run properly in IIS 5.0 isolation mode. Then the administrator configured IIS 6.0 to run in worker process isolation mode. Table 5.5 lists the configuration of the Web applications immediately after configuring IIS to run in worker process isolation mode.

Table 5.5 Configuration After Converting to Worker Process Isolation Mode

Application Name	Application Pool	Application Pool Identity
Application-A	**Default Application Pool**	**NetworkService**
Application-B	**Default Application Pool**	**NetworkService**
Application-C	**Default Application Pool**	**NetworkService**
Application-D	**Default Application Pool**	**NetworkService**
Application-E	**Default Application Pool**	**NetworkService**

To approximate the original configuration of the Web applications in worker process isolation mode, the administrator does the following:

1. Creates a new application pool for each application that was configured for isolation.
2. Configures each application pool with the identity assigned previously to the application configured for isolation.
3. Ensures that the identity assigned to each newly created application pool is added to the IIS_WPG local user group.
4. Assigns each application to the corresponding application pool.
5. Continues hosting all other applications in the default application pool.
6. Verifies that the applications in the default application pool properly run under the NetworkService account identity.

Table 5.6 Final Configuration of Web Sites and Applications

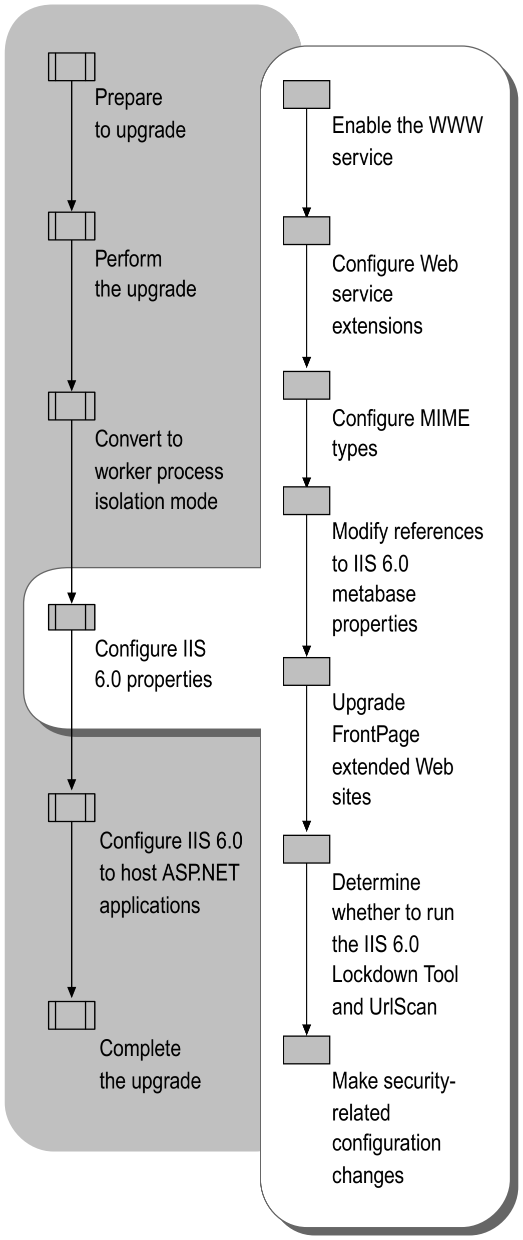
Application Name	Application Pool	Application Pool Identity
Application-A	**AppPool-01**	**AppIdent-01**
Application-B	**Default**	**NetworkService**
Application-C	**AppPool-02**	**AppIdent-02**
Application-D	**Default**	**NetworkService**
Application-E	**Default**	**NetworkService**

Configuring IIS 6.0 Properties

Up to this point in the upgrade process, you have upgraded the operating system and all of the operating system components, including IIS 6.0, on the Web server. However, you might need to further configure the IIS 6.0 properties on the Web server so that the Web sites run as they did before the server was upgraded. In addition, you can configure your Web server to take advantage of the enhanced security and availability capabilities of IIS 6.0.

Figure 5.5 illustrates the process for configuring the IIS 6.0 properties on your Web server.

Figure 5.5 Configuring IIS 6.0 Properties


Prepare to upgrade
Perform the upgrade
Convert to worker process isolation mode
Configure IIS 6.0 properties
Configure IIS 6.0 to host ASP.NET applications
Complete the upgrade
Enable the WWW service
Configure Web service extensions
Configure MIME types
Modify references to IIS 6.0 metabase properties
Upgrade FrontPage extended Web sites
Determine whether to run the IIS 6.0 Lockdown Tool and UrlScan
Make security-related configuration changes

Enabling the WWW Service

When you upgrade a Web server running Windows 2000 Server and IIS 5.0, the World Wide Web Publishing Service (WWW service) is disabled unless, before upgrading, you elected to run the IIS Lockdown Tool or make the appropriate changes to the registry. However, if you did not choose either of those methods, you must now enable the WWW service.

Note

If you are upgrading a Web server that is currently running Windows NT Server 4.0 and IIS 4.0, the WWW service is not disabled. Therefore, you can continue to the next step in the deployment process.

For more information about how to enable the WWW service after upgrade see “Enable the WWW Service After Upgrade” in “IIS Deployment Procedures” in this book.

Configuring Web Service Extensions

Many Web sites and applications hosted on IIS include dynamic content and other enhanced capabilities. Providing dynamic content and other enhanced capabilities requires executable code, such as ASP, ASP.NET, and ISAPI extensions. The handlers that extend IIS functionality beyond serving static pages are known as *Web service extensions*.

Because of the enhanced security features in IIS 6.0, you can enable or disable individual Web service extensions. After upgrade, all of the Web service extensions are enabled except for the extensions that are mapped to 404.dll by the IIS Lockdown Tool. If you did not run the IIS Lockdown Tool prior to upgrade, all of the Web service extensions are enabled.

The Windows Server 2003 upgrade creates a permission entry for Web service extensions, which enables all of the Web service extensions that are not explicitly prohibited. Enabling all of the Web service extensions ensures the highest possible compatibility with your Web sites. However, doing this creates a security risk by enabling functionality that might not be necessary for your server, which increases the attack surface of the server.

Note

Web service extensions allow you to enable and disable the serving of dynamic content. *MIME types* allow you to enable and disable the serving of static content. For more information about enabling and disabling the serving of static content, see “Configuring MIME Types” later in this chapter.

Configure the Web service extensions after upgrade by completing the following steps:

1. Configure the Web service extensions list so that the following entries, which enable all Web service extensions, are set to **Prohibited**:

   - **All Unknown CGI Extensions**
   - **All Unknown ISAPI Extensions**

   For information about how to prohibit a Web service extension, see "Configure Web Service Extensions" in "IIS Deployment Procedures" in this book.

2. Enable the essential predefined Web service extensions based on the information in Table 5.7.

   Table 5.7 Predefined Web Service Extensions

Web Service Extension	Enable When
Active Server Pages	One or more of the Web sites or applications contains ASP content.
ASP.NET version 1.1.4322	One or more of the Web sites or applications contains ASP.NET content.
FrontPage Server Extensions 2002	One or more of the Web sites are FrontPage extended.
Internet Data Connector	One or more of the Web sites or applications uses the IDC to display database information (content includes .idc and .idx files).
Server-Side Includes	One or more of the Web sites uses server-side include (SSI) directives to instruct the Web server to insert various types of content into a Web page.
WebDAV	You want to support WebDAV on the Web server. Not recommended for dedicated Web servers.

3. For each Web service extension that is used by your applications and is not a one of the default Web service extensions, add a new entry to the Web service extensions list and configure the status of the new entry to **Allowed**.

   For example, one of your applications might use an ISAPI extension to provide access to a proprietary database. Add the ISAPI extension to the Web service extensions list, and then configure the status of the ISAPI extension to **Allowed**.

   For information about how to add a Web service extension and enable the extension, see "Configure Web Service Extensions" in "IIS Deployment Procedures" in this book.

4. Use a Web browser on a client computer to verify that the Web sites and applications run on the server.

Configuring MIME Types

IIS serves only the static files with extensions that are registered in the Multipurpose Internet Mail Extensions (MIME) types list. IIS is preconfigured to recognize a default set of global MIME types, which are recognized by all configured Web sites. MIME types can also be defined at the Web site and directory levels, independent of one another or the types defined globally. IIS also allows you to change, remove, or configure additional MIME types. For any static content file extensions used by the Web sites and applications hosted by IIS that are not defined in the MIME types list, you must create a corresponding MIME type entry.

Configure the MIME types after upgrade by completing the following steps:

1. Remove the entry **.* application/octet-stream**, which enables all MIME types. Removing this entry allows you to restrict the static content served by IIS.

   For information about how to remove a MIME type from the list, see "Configure MIME Types" in "IIS Deployment Procedures" in this book.

2. For each static file type that is used by your applications, ensure that an entry exists in the MIME types list.

   When your application uses the standard MIME types included in IIS, no new MIME type entry is required. For information about how to add a MIME type, see "Configure MIME Types" in "IIS Deployment Procedures" in this book.

3. Use a Web browser on a client computer to verify that the Web sites and applications run on the server.

Modifying References to IIS 6.0 Metabase Properties

There are metabase properties that were used to configure features in earlier versions of IIS that are no longer supported in IIS 6.0. Because some features are eliminated, or implemented differently in IIS 6.0, the corresponding unused metabase properties are not referenced by any code in IIS 6.0. In cases where the feature is implemented differently in IIS 6.0, new metabase properties have been created to replace the obsolete, or unused, property.

In addition, there is one IIS 5.0 metabase property — **CPUResetInterval** — whose behavior has changed because of architectural changes made to IIS 6.0.

To determine whether any of your Web sites, applications, or setup programs reference these changed or unsupported IIS metabase properties, see “Changes to Metabase Properties in IIS 6.0” in this book. You can follow the recommendations associated with each changed metabase property to accommodate functionality changes.

Tip
The metabase properties that are no longer supported in IIS 6.0 are not available in IIS 6.0, even when IIS 6.0 is configured to run in IIS 5.0 isolation mode.

Upgrading FrontPage Extended Web Sites

When you upgrade a Web server that has FrontPage 2000 Server Extensions, the upgrade process automatically installs FrontPage 2002 Server Extensions. After upgrade, both FrontPage 2000 Server Extensions and FrontPage 2002 Server Extensions are installed. After upgrade, you must upgrade each of the FrontPage extended Web sites and configure IIS 6.0 so that FrontPage 2000 Server Extensions is prohibited.

Upgrade your FrontPage extended Web sites by completing the following steps:

1. Set the status of the **FrontPage Server Extensions 2000** entry in the Web service extensions list to **Prohibited**.

   For more information about configuring Web service extensions, see “Configure Web Service Extensions” in “IIS Deployment Procedures” in this book.

2. For each FrontPage extended Web site, upgrade the Web site to FrontPage 2002 Server Extensions.

   You can upgrade individual Web sites by using the FrontPage 2002 Server Extensions Server Administration Web page. You can also upgrade individual sites, or multiple sites in a batch, by using Owsadm.exe.

   For more information about how to upgrade FrontPage extended Web sites to FrontPage 2002 Server Extensions, see “Upgrade FrontPage Extended Web Sites” in “IIS Deployment Procedures” in this book.

Determining Whether to Run the IIS Lockdown Tool and UrlScan

UrlScan and the IIS Lockdown Tool are IIS security related programs designed for IIS 5.1 and earlier. Each tool provides different types of protection for earlier versions of IIS.

IIS Lockdown Tool

The IIS Lockdown Tool is provided to assist administrators in configuring optimal security settings for existing IIS servers. You cannot install the IIS Lockdown Tool after migration because all of the default configuration settings in IIS 6.0 meet or exceed the security configuration settings made by the IIS Lockdown Tool.

UrlScan

UrlScan is a tool that is provided to reduce the attack surface of Web servers running earlier versions of IIS. By default, IIS 6.0 has features that significantly improve security by reducing the attack surface of the Web server. UrlScan provides flexible configuration for advanced administrators, while maintaining the improved security in IIS 6.0. When you need this flexibility in configuring your Web server, you can run UrlScan on IIS 6.0.

For more information about determining whether to run UrlScan after migrating your server to IIS 6.0, see the Using UrlScan link on the Web Resources page at http://www.microsoft.com/windows/reskits/.

Making Security-Related Configuration Changes

After upgrading your server to IIS 6.0, you can make additional security-related configuration changes on the Web server. If you ran the IIS Lockdown Tool before upgrading the Web server, most of these changes are already in place. The IIS Lockdown Tool removes unnecessary IIS components, including virtual directories, to reduce the attack surface available to malicious users. Otherwise, make these security-related configuration changes to help reduce the attack surface and increase the security of the Web server.

Make the security-related configuration changes by completing the following steps:

1. Enable essential IIS components and services.
2. Remove unnecessary IIS virtual directories.
3. Configure the anonymous user identity.

Enabling Essential IIS Components and Services

IIS 6.0 includes other components and services in addition to the WWW service, such as the FTP service and SMTP service. IIS components and services are installed and enabled by means of the **Application Server** subcomponent in **Add or Remove Windows Components**. After installing IIS, you must enable the IIS 6.0 components and services required by the Web sites and applications running on the Web server.

Enable only the essential IIS 6.0 components and services required by your Web sites and applications. For more information about enabling the essential IIS protocols and services see “Enabling Only Essential IIS Components and Services” in “Securing Web Sites and Applications” in this book.

Removing Unnecessary IIS Virtual Directories

Table 5.8 lists the virtual directories that can be removed from IIS 6.0. Compare the virtual directories in IIS Manager to the virtual directories in Table 5.8, and then remove any of the virtual directories listed in Table 5.8.

Table 5.8 Virtual Directories that Can Be Removed After Upgrade

Virtual Directory	Description
IISAdmin	**Provides an HTML-based administration tool. It is designed primarily for administrators who administer the Web server through a Web interface, but IIS Manger is the recommended method for Web server administration. The virtual directory is not installed as part of IIS 6.0, but it is not removed when a Web server running an earlier version of IIS is upgraded to IIS 6.0.**
IISADMPWD	**This virtual directory allows you to reset passwords from Windows NT Server 4.0 and Windows 2000 Server. It is designed primarily for intranet scenarios and is not installed as part of IIS 6.0, but it is not removed when a Web server running IIS 4.0 is upgraded to IIS 6.0. For more information about this functionality, see the Microsoft Knowledge Base link on the Web Resources page at http://www.microsoft.com/windows/reskits/webresources, and search for article Q184619.**
IISHelp	**Provides an HTML version of the IIS documentation. It is designed primarily for application developers and is not installed as part of IIS 6.0, but it is not removed when a Web server running an earlier version of IIS is upgraded to IIS 6.0.**
MDAC	**Contains sample applications that illustrate the use of Microsoft Data Access Components that are not required on production Web servers. It is designed primarily for application developers and is not installed as part of IIS 6.0, but it is not removed when a Web server running an earlier version of IIS is upgraded to IIS 6.0.**
IISSamples	**Contains sample applications that are not required on production Web servers. It is designed primarily for application developers and is not installed as part of IIS 6.0, but it is not removed when a Web server running an earlier version of IIS is upgraded to IIS 6.0.**
Printers	**Provides an HTML-based printer administration tool. It is designed primarily for administrators who administer printers through a Web interface, but using the Windows Server 2003 printer administration interface is the recommended method for printer administration. The virtual directory is not installed as part of IIS 6.0, but it is not removed when a Web server running an earlier version of IIS is upgraded to IIS 6.0.**

(continued)

Table 5.8 Virtual Directories that Can Be Removed After Upgrade *(continued)*

Virtual Directory	Description
Scripts	**Contains scripts that are used for the sample applications in other virtual directories and is required on production Web servers. It is designed primarily for application developers and is not installed as part of IIS 6.0, but it is not removed when a Web server running an earlier version of IIS is upgraded to IIS 6.0.**

For more information about how to remove unnecessary virtual directories after upgrade, see "Remove Virtual Directories" in "IIS Deployment Procedures" in this book.

Configuring the Anonymous User Identity

When the Web sites and applications running on the Web server require anonymous access, IIS is configured with a user account specifically for anonymous access. When a user connects to the Web server anonymously, IIS creates a process token for the user based on the user account that you configured in the anonymous user identity. The user account can be stored in the local account database on the Web server, or in a domain.

If, before upgrade, the anonymous user identity is configured to use a domain-based user account, after upgrade you need to configure the anonymous user account to the same domain-based user account. This is because the upgrade process automatically configures IIS to user the default anonymous account IUSR_*computername*, where *computername* is the name of the computer on which IIS is running. You can configure the anonymous user identity to the domain-based user account in IIS Manager.

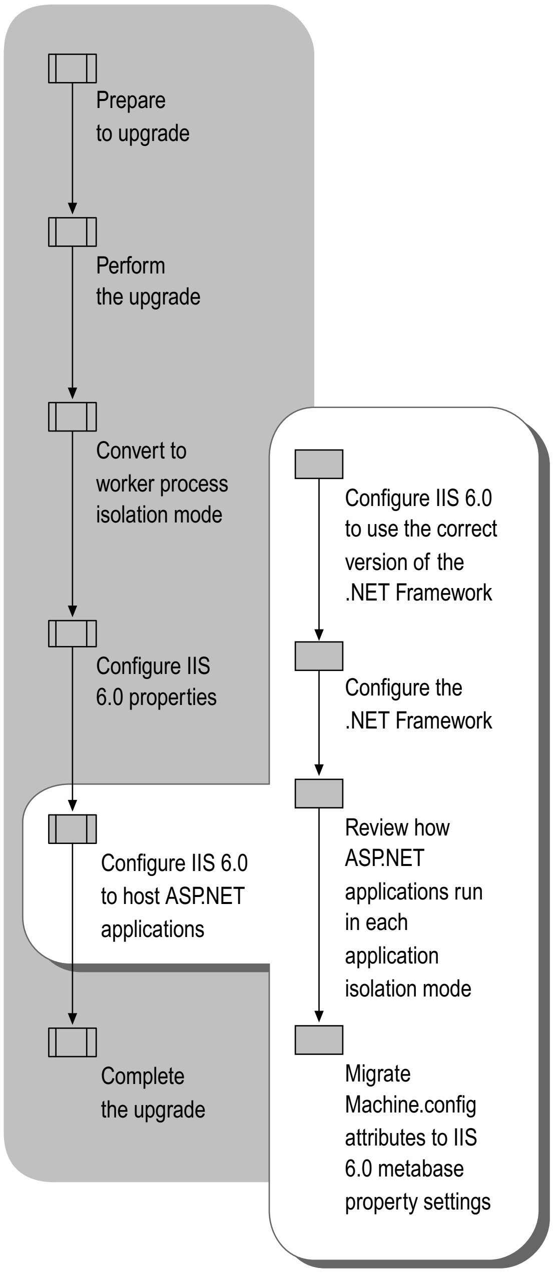
For more information about how to configure the anonymous user identity, see "Configure Anonymous User Identity" in "IIS Deployment Procedures" in this book.

Configuring IIS 6.0 to Host ASP.NET Applications

If the newly upgraded IIS 6.0 server includes existing ASP.NET applications, you need to configure IIS to use the correct version of the .NET Framework, and you must configure the .NET Framework to support your applications. If IIS 6.0 is configured to run in worker process isolation mode, you need to migrate ASP.NET configuration settings from the Machine.config file to the equivalent settings in IIS 6.0.

Figure 5.6 illustrates the process for configuring IIS for ASP.NET applications

Figure 5.6 Configuring IIS for ASP.NET Applications


Prepare to upgrade
Perform the upgrade
Convert to worker process isolation mode
Configure IIS 6.0 properties
Configure IIS 6.0 to host ASP.NET applications
Complete the upgrade
Configure IIS 6.0 to use the correct version of the .NET Framework
Configure the .NET Framework
Review how ASP.NET applications run in each application isolation mode
Migrate Machine.config attributes to IIS 6.0 metabase property settings

Configuring IIS 6.0 to Use the Correct Version of the .NET Framework

Before upgrading a Web server that hosts ASP.NET applications, version 1.0 of the .NET Framework is installed on the server and the applications are configured to use that version of the .NET Framework. After upgrade, version 1.1 of the .NET Framework is installed and the applications are configured to use version 1.1 of the .NET Framework. However, version 1.0 of the .NET Framework is still installed, which is referred to as side-by-side support.

Running versions 1.0 and 1.1 of the .NET Framework side-by-side is only supported when IIS is configured to run in IIS 5.0 isolation mode. If you have already configured IIS to run in worker process isolation mode, then you can only use version 1.1 of the .NET Framework. In most cases, ASP.NET applications function correctly with version 1.1 of the .NET Framework. For more information about possible application incompatibilities when upgrading from version 1.0 to version 1.1 of the .NET Framework, see “Determining Application Compatibility with the .NET Framework” earlier in this chapter. When your ASP.NET application is incompatible with version 1.1 of the .NET Framework, configure the application to use version 1.0 of the .NET Framework and configure IIS to run in IIS 5.0 isolation mode.

You can configure each ASP.NET application to use a specific version of the .NET Framework by registering a *script map* in IIS for the application. A script map associates a file name extension and HTTP verb with the appropriate ISAPI for script handling. For example, when IIS receives a request for a .aspx file, the script map for the corresponding application directs IIS to forward the requested file to the appropriate version of the ASP.NET ISAPI for processing.

The script map for each ASP.NET application can be applied directly to an application, or inherited from a parent application. However, ASP.NET supports only one version of the .NET Framework for each application pool. For more information about how to configure the script map for an ASP.NET application, see “Configure an ASP.NET Application for ASP.NET” in “IIS Deployment Procedures” in this book.

Configuring the .NET Framework

The configuration method for the .NET Framework is determined by the application isolation mode that you use to configure IIS 6.0. Table 5.9 lists the methods for configuring the .NET Framework that are associated with each IIS 6.0 application isolation mode.

Table 5.9 Methods for Configuring the .NET Framework

Application Isolation Mode	Configuration Method for the .NET Framework
IIS 5.0 isolation mode	**Configured by making changes to the Machine.config file in the *systemroot*\Microsoft.NET\Framework\*VersionNumber*\Config folder.**
Worker process isolation mode	**Configured by making changes to the IIS 6.0 metabase.**

When IIS 6.0 is configured to run in IIS 5.0 isolation mode, the .NET Framework uses the processModel section of the Machine.config file (in the *systemroot*\Microsoft.NET\Framework\versionNumber\Config folder) for its configuration and no additional steps are required.

However, if you configured IIS 6.0 to run in worker process isolation mode, the .NET Framework largely ignores the **<processModel>** section of the Machine.config file, and instead gets its process configuration from the IIS 6.0 metabase. Because the upgrade process does not migrate the existing settings in the Machine.config file, you must manually convert any settings required by the ASP.NET applications.

For information about how to convert the Machine.config attribute settings to IIS 6.0 metabase property settings, see “Migrating Machine.config Attributes to IIS 6.0 Metabase Property Settings” later in this chapter. For more information about configuring IIS 6.0 for ASP.NET applications, see “Deploying ASP.NET Applications in IIS 6.0” in this book.

Reviewing How ASP.NET Applications Run In Each Application Isolation Mode

When running in worker process isolation mode, ASP.NET applications use the W3wp.exe worker process and application pool properties, which are stored in the IIS 6.0 metabase. When you configure IIS 6.0 to run in IIS 5.0 isolation mode, ASP.NET applications use the ASP.NET request processing model, Aspnet_wp.exe, and configuration settings. These configuration settings are stored in the Machine.config file.

Behavior of ASP.NET Applications that Are Running in IIS 5.0 Isolation Mode

By default, ASP.NET applications are configured to run in worker process isolation mode. If your application can only run in the ASP.NET process model, you must configure the server to run in IIS 5.0 isolation mode to be able to run the application on IIS 6.0. When IIS 6.0 is configured to run in IIS 5.0 isolation mode, ASP.NET applications behave as follows:

- The applications run within Aspnet_wp.exe.

  Aspnet_wp.exe is a request processing model similar to worker process isolation mode and contains similar worker process management capabilities as the WWW service in IIS 6.0. Aspnet_wp.exe includes most of the IIS application management features, such as recycling, health detection, and *processor affinity* — the ability to force worker processes to run on specific microprocessors.

- The configuration settings are stored in the Machine.config file.

  When configured in IIS 5.0 isolation mode, the configuration settings for ASP.NET applications are managed by modifying the Machine.config file, not the IIS metabase file. Because there is no administrative console for the Machine.config settings, any configuration settings for ASP.NET must be made directly to the Machine.config file.

Important

> In IIS 5.0 isolation mode, the .NET Framework ignores any configuration changes made in the IIS metabase. Administrative consoles, such as IIS Manager, make changes to the IIS 6.0 metabase, but those changes are not read by the .NET Framework.

When IIS 6.0 is configured in IIS 5.0 isolation mode, your ASP.NET applications should behave as they did in IIS 5.0. However, incompatibilities can result when running version 1.1 of the .NET Framework. For more information about configuring IIS to support ASP.NET applications that use version 1.0 of the .NET Framework, see "Configuring IIS 6.0 to Use the Correct Version of the .NET Framework" earlier in this chapter.

Behavior of ASP.NET Applications that Are Running in Worker Process Isolation Mode

When IIS 6.0 is configured to run in worker process isolation mode, ASP.NET applications behave as follows:

- The process model within the ASP.NET ISAPI extension is disabled and ASP.NET applications run using worker process isolation mode in IIS 6.0.

  In this configuration, the ASP.NET application runs in worker process isolation mode like any other application, such as an ASP application. In addition, IIS 6.0 provides all of the management features, such as recycling, health detection, and processor affinity.

- The ASP.NET ISAPI extension is configured by a combination of configuration settings that are stored in the IIS metabase (MetaBase.xml) and configuration settings in the Machine.config file.

  When configured in worker process isolation mode, the majority of the configuration settings are stored in the IIS metabase. You can adjust these settings directly in MetaBase.xml or from administrative consoles, such as IIS Manager, or scripts.

  However, if there are existing settings in the **<processModel>** section of the Machine.config file, those configuration settings must be converted to the appropriate application pool settings when the Web server is configured to run in worker process isolation mode Additionally, there are other configuration settings that are still made through the Machine.config file, regardless of the application mode. For more information about converting Machine.config attributes to worker process isolation mode settings, see "Migrating Machine.config Attributes to IIS 6.0 Metabase Property Settings" later in this chapter.

When IIS 6.0 is configured in worker process isolation mode, your ASP.NET applications should behave as they did on IIS 5.0. Before deploying your ASP.NET applications on your production Web servers, test compatibility with IIS 6.0 running in worker process isolation mode and version 1.1 of the .NET Framework. For more information about determining application compatibility with IIS 6.0 running in worker process isolation mode, see “Determining Application Compatibility with Worker Process Isolation Mode” earlier in this chapter. For more information about determining compatibility with version 1.1 of the .NET Framework see “Determining Application Compatibility with the .NET Framework” earlier in this chapter.

If you determine that your ASP.NET applications are incompatible with IIS 6.0 running in worker process isolation mode, reconfigure IIS to run in IIS 5.0 isolation mode. If your ASP.NET applications are incompatible with version 1.1 of the .NET Framework, configure IIS to use version 1.0 of the .NET Framework with your ASP.NET application.

For more information about configuring IIS 6.0 to run in IIS 5.0 isolation mode, see “Configure Application Isolation Modes” in “IIS Deployment Procedures” in this book.

Migrating Machine.config Attributes to IIS 6.0 Metabase Property Settings

When you upgrade a server running IIS 5.0 and version 1.0 of the .NET Framework, the ASP.NET applications can depend on specific attribute settings in the **<processModel>** section of the Machine.config file. The **<processModel>** section configures the ASP.NET process model settings and therefore affects all ASP.NET applications running on an IIS Web server. If any of the following Machine.config attributes exist, you must migrate the attribute configuration settings to their equivalent metabase property settings in IIS 6.0.

Note

Follow the deployment steps in this section only when IIS 6.0 is configured to run in worker process isolation mode. When IIS 6.0 is configured to run in IIS 5.0 isolation mode, IIS 6.0 reads the ASP.NET settings from the Machine.config file, and you do not need to migrate the Machine.config attribute settings. Thus, you can continue to the next step in the upgrade process.

Migrate the Machine.config attributes by adjusting the configuration settings in the Application Pools node in IIS Manager. Table 5.10 lists the Machine.config attributes and where to find more information about how to migrate the attributes.

Table 5.10 Machine.config Attributes and How To Migrate Them

Machine.config Attribute	Migrating This Attribute
cpuMask	**See "Migrating Performance Related Attributes later in this chapter.**
idleTimeout	**See "Migrating Performance Related Attributes" later in this chapter.**
memoryLimit	**See "Migrating Recycling Related Attributes later in this chapter.**
password	**See "Migrating Identity Related Attributes later in this chapter.**
pingFrequency	**See "Migrating Health Related Attributes later in this chapter.**
pingTimeout	**See "Migrating Health Related Attributes later in this chapter.**
requestLimit	**See "Migrating Recycling Related Attributes later in this chapter.**
restartQueueLimit	**See "Migrating Performance Related Attributes later in this chapter.**
shutdownTimeout	**See "Migrating Health Related Attributes later in this chapter.**
timeout	**See "Migrating Recycling Related Attributes later in this chapter.**
username	**See "Migrating Identity Related Attributes later in this chapter.**
webGarden	**See "Migrating Performance Related Attributes later in this chapter.**

The Machine.config attribute settings in Table 5.10 must be migrated to IIS 6.0 settings. However, you must still configure the following attribute settings in the Machine.config file:

- **maxWorkerThreads**
- **maxIoThreads**
- **responseDeadlockInterval**

When configured to run in worker process isolation mode, IIS 6.0 and the .NET Framework ignore any **<processModel>** section Machine.config attribute settings that are not in the previous list or in Table 5.10.

Note

The <**processModel**> section of the Machine.config file configures the ASP.NET process model settings on an IIS Web server. The <**processModel**> section provides global configuration settings for all of the ASP.NET applications running on the server.

Migrating Recycling-Related Attributes

In worker process isolation mode, you can assign one or more Web sites or applications to an application pool. Each application pool has one or more worker processes that process client requests for the Web sites and applications in the application pool. You can configure IIS to periodically recycle worker processes assigned to an application pool, which in turn recycles the Web sites and applications running in that application pool. Recycling worker processes keeps problematic Web sites and applications running smoothly, especially when it is not feasible to modify the application code. Recycling worker processes within an application pool ensures that the Web sites and applications in the application pools remain healthy, and that system resources can be recovered.

The Machine.config attributes that affect recycling-related metabase properties include:

- **timeout**
- **requestLimit**
- **memoryLimit**

Migrating the timeout Attribute

The **timeout** Machine.config attribute, default value infinite, specifies the time limit after which ASP.NET starts a new worker process to take the place of the current one. The attribute is a string value in the format of hr:min:sec. To configure this value in IIS 6.0, you need to convert this format to time in minutes.

Configure the **Recycling Worker Processes (in minutes)** setting in IIS 6.0 to the same time value, in minutes, that is configured in the **timeout** Machine.config attribute.

For more information about how to configure the **Recycling Worker Processes (in minutes)** setting, see “Configure Application Pool Recycling” in “IIS Deployment Procedures” in this book.

Migrating the requestLimit Attribute

The **requestLimit** Machine.config attribute, default value infinite, specifies the number of requests after which ASP.NET starts a new worker process to take the place of the current one.

Configure the **Recycling Worker Processes (in requests)** setting in IIS 6.0 to the same value, in number of requests, that is configured in the **requestLimit** Machine.config attribute.

For more information about how to configure the **Recycling Worker Processes (in requests)** setting, see “Configure Application Pool Recycling” in “IIS Deployment Procedures” in this book.

Migrating the memoryLimit Attribute

The **memoryLimit** Machine.config attribute, default value 60 percent, specifies the percentage of physical memory that the worker process can consume before ASP.NET starts a new worker process to take the place of the current one.

Recycling based on memory consumption works slightly different in IIS 6.0, where worker processes can be recycled based on both virtual memory and physical memory. Also, the memory limit is not a percentage value, but is specified as a value in megabytes.

Configure the **Maximum used memory (in megabytes)** setting in IIS 6.0 to the amount of physical memory represented by the percentage specified in the **memoryLimit** Machine.config attribute.

Also, adjust the **Maximum virtual memory (in megabytes)** settings in IIS 6.0 if necessary. The **Maximum virtual memory (in megabytes)** is the maximum amount of virtual memory, which includes the used memory plus the reserved memory,

For more information about how to configure the **Maximum used memory (in megabytes)** setting, see "Configure Application Pool Recycling" in "IIS Deployment Procedures" in this book.

Migrating Performance-Related Attributes

In worker process isolation mode, IIS 6.0can be configured to optimize the performance of an application pool, allowing you optimize the performance of your Web applications. Migrating the performance-related Machine.config attributes helps ensure that your Web applications perform as they did on IIS 5.0.

Note

In addition to the configuration settings mentioned in this section, IIS 6.0 provides CPU utilization monitoring for worker processes. For more information, see "Configuring Applications Pools and Worker Processes for ASP.NET Applications" in "Deploying ASP.NET Applications in IIS. 6.0" in this book.

The Machine.config attributes that affect performance-related metabase properties include:

- **webGarden and cpuMask**
- **idleTimeout**
- **restartQueueLimit**

Migrating the webGarden and cpuMask Attributes

The combination of the **webGarden** and **cpuMask** Machine.config attributes provide configuration for ASP.NET on IIS 5.0 for *Web gardens*. Web gardens are created when there is more than one worker process servicing applications. In IIS 5.0, Web gardens are created for use by all applications running on the Web server. In IIS 6.0, Web gardens are created within each application pool.

Web gardens in IIS 5.0 allow ASP.NET to schedule a separate worker process for each microprocessor on a Web server with multiple microprocessors. In contrast, you can create a Web garden on IIS 6.0 by specifying multiple worker processes for an application pool, regardless of the number of microprocessors in the Web server. For more information about Web gardens in IIS 6.0, see “Configuring Web Gardens” in “Ensuring Application Availability” in this book.

The **webGarden** Machine.config attribute, which has a default value of **False**, controls CPU affinity when used in conjunction with the **cpuMask** attribute. When the **webGarden** attribute is set to:

- **True.** ASP.NET creates a separate worker process for each processor specified in the **cpuMask** Machine.config attribute.
- **False.** ASP.NET creates only one worker process and the **cpuMask** Machine.config attribute is ignored.

The **cpuMask** Machine.config attribute, default value 0xffffffff, specifies which microprocessors on a multiprocessor Web server are able to run worker processes initiated by ASP.NET. By default, all processors are able to run worker processes initiated by ASP.NET. The **cpuMask** attribute also contains a bitmask value that indicates the microprocessors that are able to run ASP.NET processes. When a bit in the **cpuMask** bitmask value is set to 1, the corresponding microprocessor is able to run ASP.NET processes. You need to record the number of bitmasks enabled to convert this to the correct number of application pools.

For example, if **cpuMask** is set to a value of 0x0d, the equivalent of the binary bit pattern is 1101. If your Web server has four microprocessors, the bitmask value 1101 indicates that you need three worker processes.

Configure the **Maximum number of worker processes** setting in IIS 6.0 to the number of microprocessors that are enabled in the **cpuMask** Machine.config attribute. The default value for **Maximum number of worker processes** is 1, indicating that Web gardens are not enabled.

For more information about how to configure the **Maximum number of worker processes** setting, see “Configure Application Pool Performance” in “IIS Deployment Procedures” in this book.

Migrating the restartQueueLimit Attribute

The **restartQueueLimit** Machine.config attribute, which has a default value of 10, specifies the maximum number of requests that will be queued in ASP.NET while waiting for the worker process to recycle after an abnormal termination.

Configure the **Limit the kernel request queue to** setting in IIS 6.0 to the same value that is configured in the **restartQueueLimit** Machine.config attribute.

For more information about how to configure the **Limit the kernel request queue to** setting, see “Configure Application Pool Performance” in “IIS Deployment Procedures” in this book.

Note
When application pool queue length limits are enabled, IIS monitors the number of requests for a designated application pool queue before queuing a new request. If adding a new request to the queue causes it to exceed the queue length limit, the server rejects the request and sends a 503-error response (that cannot be customized) to the client. However, requests that are already queued remain in the queue even if the limit is changed to a value that is less than the current queue length.

Migrating the idleTimeout Attribute

The **idleTimeout** Machine.config attribute, default value infinite, specifies the maximum number of minutes that a worker process can be inactive before it is automatically shut down by ASP.NET. The attribute is a string value in the format of hr:min:sec. To configure this value in IIS 6.0, you need to convert this format to time in minutes.

Configure the **Shutdown worker processes after being idle for** setting in IIS 6.0 to the same time value, in minutes, that is configured in the **idleTimeout** Machine.config attribute.

For more information about how to configure the **Shutdown worker processes after being idle for** setting, see “Configure Application Pool Performance” in “IIS Deployment Procedures” in this book.

Migrating Health-Related Attributes

By detecting the degree of stability (or health) of an ASP.NET application, IIS determines whether corrective action is required. In IIS 5.0, ASP.NET detects an unhealthy worker process. In IIS 6.0, the WWW service detects an unhealthy worker process that has terminated abnormally if all of the available IIS threads in the worker process assigned to the application pool are blocked.

The Machine.config attributes that affect health-related metabase properties include:

- **shutdownTimeout**
- **pingFrequency** and **pingTimeout**

Migrating the shutdownTimeout Attribute

The **shutdownTimeout** Machine.config attribute, default value five seconds, specifies the maximum number of seconds allotted for a worker process to shut down before ASP.NET automatically shuts it down. The attribute is a string value in the format of hr:min:sec. To configure this value in IIS 6.0, you need to convert this format to time in seconds.

Configure the **Worker process must shutdown** setting in IIS 6.0 to the same time value, in minutes, that is configured in the **shutdownTimeout** Machine.config attribute.

For more information about how to configure the **Worker process must shutdown** setting, see "Configure Application Pool Health" in "IIS Deployment Procedures" in this book.

Migrating the pingFrequency and pingTimeout Attributes

In IIS 5.0, the combination of the **pingFrequency** and **pingTimeout** Machine.config attributes specifies whether ASP.NET should monitor the health of worker processes. If IIS 6.0 is running in worker process isolation mode, the WWW service periodically monitors the health of a worker process.

The **pingFrequency** Machine.config attribute, default value 30 seconds, specifies the interval at which ASP.NET queries the worker processes to determine if each worker process is still operating correctly. The **pingFrequency** Machine.config attribute is a string value in the format of hr:min:sec. To configure this value in IIS 6.0, you need to convert this format to time in seconds.

The **pingTimeout** Machine.config attribute, default value five seconds, specifies the interval after which ASP.NET restarts a worker process if there was no response to the most recent query. If a worker process does not respond to the "ping" within the value set by the **pingTimeout** Machine.config attribute, the worker process is restarted. The **pingTimeout** Machine.config attribute is a string value in the format of hr:min:sec. To configure this value in IIS 6.0, you need to convert this format to time in seconds.

You can configure the equivalent settings in IIS 6.0 by making changes to the equivalent IIS metabase properties, listed in Table 5.11.

Table 5.11 Equivalent IIS 6.0 Metabase Properties for pingFrequency and pingTimeout

Machine.config Attribute	Equivalent IIS Metabase Property
pingFrequency	**PingInterval**
pingTimeout	**PingResponseTime**

Note

Unless required by your ASP.NET applications, use the default settings in IIS 6.0. In IIS 6.0, the default setting for **PingInterval** is 30 seconds and the default setting for **PingResponseTime** is 90 seconds.

You can configure the **PingInterval** metabase property through the **Ping worker process** setting for application pools in IIS 6.0 Manager. For more information about how to configure the **Ping worker process** setting, see "Configure Application Pool Health" in "IIS Deployment Procedures" in this book.

You can only configure **PingResponseTime** by directly modifying the metabase or programmatically through Windows Management Instrumentation (WMI) or ADSI. For more information about directly modifying the metabase, see “Modify the IIS Metabase Directly” in “IIS Deployment Procedures” in this book.

Migrating Identity-Related Attributes

In IIS 5.0, specifying an identity allowed a worker process in ASP.NET to use a Windows identity other than that of the default *process identity*. Process identity is an operating system term used to denote the account that a process runs under. Every running process on a Windows Server 2003 has a process identity that is used to control access to resources on the Web server.

Note
In IIS 6.0, the application pool identity allows worker processes servicing the application pool to use an identity other than the default identity NetworkService. In IIS 5.0, the default identity is LocalSystem.

The Machine.config attributes that affect identity-related metabase properties include:

- **username**
- **password**

Migrating the username Attribute

The **username** Machine.config attribute specifies the user account used by ASP.NET as an identity for worker processes. The attribute is a string value and does not exist in Machine.config by default.

In IIS 5.0, the **username** Machine.config attribute has the following values:

- **No entry.** When the **username** attribute does not exist in the Machine.config file, this causes ASP.NET to run worker processes under the identity of LocalSystem.
- **Machine.** Causes ASP.NET to run worker processes under a user account named ASPNET that is created automatically when ASP.NET is installed. This is the default configuration.
- **System.** Causes ASP.NET to run worker processes under a user account named System that is created automatically when ASP.NET is installed and allows ASP.NET processes to have full administrative privileges. Applications running under the identity of the System account have unconstrained privileges on the Web server. Run applications under the identity of the System account only when required by your applications.
- **Configured account.** When the **username** attribute contains a service account, this causes ASP.NET to run worker processes under the identity of the service account.

Configure the equivalent settings in IIS 6.0 by selecting an IIS 6.0 configuration setting that allows the ASP.NET application to run properly while not compromising the security of the Web server. This selection is done in an order of preference, from most secure to least secure.

Configure the User name settings in IIS 6.0 by selecting one of the following in order of security preference:

- **Option 1.** Configure IIS 6.0 to use NetworkService.
- **Option 2.** Configure IIS 6.0 to use a new service account and grant the account the minimal user rights or group membership to allow the applications to run successfully.
- **Option 3.** Configure IIS 6.0 to use a new service account that belongs to the local Administrators group.

Option 1 Configure IIS 6.0 to use NetworkService

This is the default identity for IIS 6.0 and is the recommend identity. Most ASP.NET applications can run by using this configuration. Test your ASP.NET application to ensure proper operation with NetworkService as the identity.

For more information about how to configure the **username** setting, see “Configure Application Pool Identity” in “IIS Deployment Procedures” in this book.

Option 2 Configure IIS 6.0 to use a new service account and grant the account the minimal user rights or group membership to allow the applications to run successfully

Select this option when the ASP.NET application is unable to run under the NetworkService identity. You might need to do this when the **username** Machine.config attribute does not exist or is set to Machine, System, or to a configured account.

For more information about how to create an account see “Create a Service Account” in “IIS Deployment Procedures” in this book. For more information about how to grant user rights, see “Grant User Rights to a Service Account” in “IIS Deployment Procedures” in this book. For more information about how to configure the **username** setting, see “Configure Application Pool Identity” in “IIS Deployment Procedures” in this book.

Option 3 Configure IIS 6.0 to use a new service account that belongs to the local Administrators group

As a last resort, select this option when the ASP.NET application requires an identity that is a member of the local Administrators group. You might need to do this when the **username** Machine.config attribute does not exist, is set to System, or is set to a configured account that is a member of the local Administrators group.

For more information about how to create an account see “Create a Service Account” in “IIS Deployment Procedures” in this book. For more information about how to make a service account a member of the local Administrators group, see “Make a Service Account a Member of the Local Administrators Group” in “IIS Deployment Procedures” in this book. For more information about how to configure the **username** setting, see “Configure Application Pool Identity” in “IIS Deployment Procedures” in this book.

Migrating the password Attribute

The **password** Machine.config attribute specifies the password for the user account used by ASP.NET as an identity for worker processes. The attribute is a string value and does not exist in Machine.config by default, which indicates that the worker processes should run under the default identity of LocalSystem in IIS 5.0.

If you selected NetworkService for the identity for the worker processes, then no password is required. If you decided to create a service account, then configure the **Password** setting in IIS 6.0 with the password for the service account.

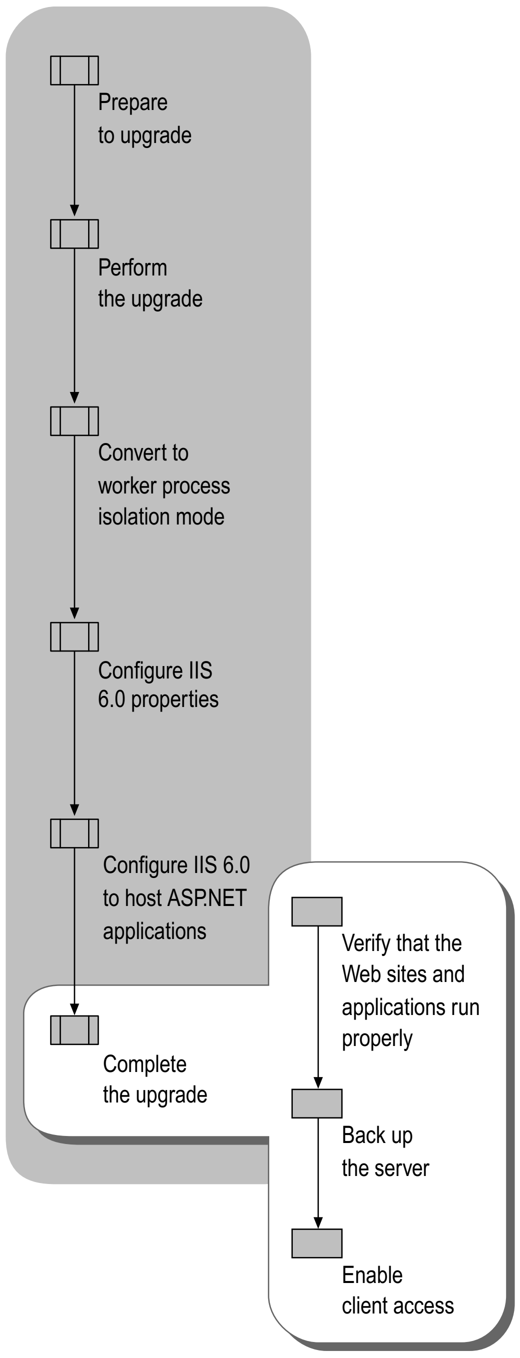
For more information about how to configure the **Password** setting, see “Configure Application Pool Identity” in “IIS Deployment Procedures” in this book.

Completing the Upgrade

At this point in the process, your Web sites and applications have been upgraded on the server. Also, the IIS 6.0 properties have been configured so that you are now ready to verify that the Web sites and applications are running properly. After you confirm that the Web sites and applications are running properly, you can back up the Web server and enable client access. When you complete these steps, the upgrade process is complete.

Figure 5.7 illustrates the process for completing the upgrade to IIS 6.0.

Figure 5.7 Completing the Upgrade to IIS 6.0


Prepare to upgrade
Perform the upgrade
Convert to worker process isolation mode
Configure IIS 6.0 properties
Configure IIS 6.0 to host ASP.NET applications
Complete the upgrade
Verify that the Web sites and applications run properly
Back up the server
Enable client access

Verifying That the Web Sites and Applications Run Properly

Before deploying your server to a production environment, verify that the Web site content and application configuration information upgraded successfully.

Verify that your Web sites and applications are running properly by completing the following steps:

1. Review the system log in Windows Server 2003 on the Web server to determine if any of the Web sites did not start.

   IIS 6.0 creates entries in the system log when a Web site fails to start for any reason. Search the System log on the target server to determine if any errors occurred. For more information about how to troubleshoot Web sites that fail to start, see "Troubleshooting" in IIS 6.0 Help, which is accessible from IIS Manager.

2. Run your existing validation procedures against the Web sites and applications that run on the server.

   Because the network adapter that connects the Web server to the clients is disabled, you might need to directly connect the Web server to a client computer and enable the network adapter to validate the Web sites and applications.

   After running your existing validation procedures, disable the network adapter that connects the Web server to the clients and reconnect the Web server as it was originally.

Tip

See "Troubleshooting" in IIS 6.0 Help, which is accessible from IIS Manager, to assist you in resolving any difficulties in running the applications.

Backing Up the Server

Before you enable client access to the Web server, perform a complete image backup. The purpose of performing this image backup is to provide a point-in-time snapshot of the Web server. If you need to restore the target Web server in the event of a failure, you can use this backup to restore the Web server to a known configuration.

Important

Do not continue to the next step until you have a successful backup of the entire Web server. Otherwise, you can lose Web sites, applications, or data that you changed after upgrading the Web server.

For more information about how to back up the Web server, see “Back Up and Restore the Web Server to a File or Tape” in “IIS Deployment Procedures” in this book.

Enabling Client Access

After you have upgraded the Web server, you are ready to enable client access to the Web sites. During the upgrade process, you disabled the network adapter on the Web server to prevent users from accessing the Web server during the upgrade process. Now that you know the upgrade is completed successfully, you can re-enable the network adapters.

Enable client access to the Web server by completing the following steps:

1. Enable the network adapter used by clients to access the Web server.

   For more information about how to enable the network adapter used by clients, see “Enable Network Adapters” in “IIS Deployment Procedures” in this book.

2. Monitor client traffic to determine if clients are accessing the Web server.

   For more information about how to monitor client traffic to Web sites on the Web server, see “Monitor Active Web and FTP Connections” in “IIS Deployment Procedures” in this book.

3. Establish a monitoring period, such as a few hours or a day, to confirm that clients that are accessing Web sites and applications on the Web server are experiencing the response times and application responses that you expected.

Additional Resources

These resources contain additional information and tools related to this chapter.

Related Information

- "Deploying ASP.NET Applications in IIS 6.0" in this book for information about deploying ASP.NET applications in IIS.
- "Ensuring Application Availability" in this book for information about improving the availability of Web sites and applications in IIS.
- "Migrating IIS Web Sites to IIS 6.0" in this book for information about migrating Web sites to a new Web server running IIS 6.0.
- "Securing Web Sites and Applications in IIS 6.0" in this book for information about securing Web sites and applications in IIS.
- "Designing a Test Environment" in *Planning, Testing, and Piloting Deployment Projects* of the *Windows Server 2003 Deployment Kit*.
- The Compatibility Considerations and Version Changes link on the Web Resources page at http://www.microsoft.com/windows/reskits/webresources for a current list of possible compatibility issues when upgrading from version 1.0 to version 1.1 of the .NET Framework.
- The *Hardware Compatibility List* on the product disc or the Hardware Driver Quality link on the Web Resources page at http://www.microsoft.com/windows/reskits/webresources for information about the hardware devices that are supported on Windows Server 2003.
- The Using UrlScan link on the Web Resources page at http://www.microsoft.com/windows/reskits/ for information about determining whether to run UrlScan after migrating your server to IIS 6.0.

Related IIS 6.0 Help Topics

- "Troubleshooting" in IIS 6.0 Help, which is accessible from IIS Manager, for information about how to troubleshoot Web sites that fail to start.

Related Tools

- The IIS Lockdown Tool link on the Web Resources page at http://www.microsoft.com/windows/reskits/webresources to download the latest version of the IIS Lockdown Tool.
- The Windows Application Compatibility link on the Web Resources page at http://www.microsoft.com/windows/reskits/webresources to download the latest version of the Windows Application Compatibility Toolkit.