**IIS 7.0 Hosting Deployment Guide [Draft v1]**

**Abstract**

This whitepaper provides an overview of the features and benefits of IIS 7.0. Hosters utilizing IIS 6.0 will be able to use the setup and configuration recommendations to implement IIS 7.0 into their hosting environment. This whitepaper also provides hosters who have not yet implemented IIS 7.0 the information they need to evaluate requirements for their hosting environment.

**Note:** The steps and guidance in this document have been tested using Windows Server 2008 Beta 3 and Windows Server 2003.

The information contained in this document represents the current view of Microsoft Corporation on the issues discussed as of the date of publication. Because Microsoft must respond to changing market conditions, it should not be interpreted to be a commitment on the part of Microsoft, and Microsoft cannot guarantee the accuracy of any information presented after the date of publication.

This white paper is for informational purposes only. MICROSOFT MAKES NO WARRANTIES, EXPRESS OR IMPLIED, IN THIS   
DOCUMENT.

Complying with all applicable copyright laws is the responsibility of the user. Without limiting the rights under copyright, no part of this document may be reproduced, stored in, or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording, or otherwise), or for any purpose, without the express written permission of Microsoft Corporation.

Microsoft may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. Except as expressly provided in any written license agreement from Microsoft, the furnishing of this document does not give you any license to these patents, trademarks, copyrights, or other intellectual property.

© 2005 Microsoft Corporation. All rights reserved.

The example companies, organizations, products, domain names, e-mail addresses, logos, people, places, and events depicted herein are fictitious. No association with any real company, organization, product, domain name, e-mail address, logo, person, place, or event is intended or should be inferred.

Microsoft, Windows, and Windows Server, are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.

The names of actual companies and products mentioned herein may be the trademarks of their respective owners.

Contents

[Overview 4](#_Toc168116679)

[Recommended Shared Hosting Configuration 4](#_Toc168116680)

[Machines and Architecture Used In This Whitepaper 4](#_Toc168116681)

[Fault-Tolerance 5](#_Toc168116682)

[Performance 5](#_Toc168116683)

[Additional Resources 5](#_Toc168116684)

[Remote File Server 6](#_Toc168116685)

[Configuring Separate Partitions for Improved Performance 6](#_Toc168116686)

[Configuring Share and NTFS Permissions 6](#_Toc168116687)

[Configuring SMB Commands for UNC (Network BIOS Commands) 8](#_Toc168116688)

[Front-End Web Server 10](#_Toc168116689)

[Installation 10](#_Toc168116690)

[Backing Up the Web Server 13](#_Toc168116691)

[Recommended IIS Configuration for Shared Hosting 14](#_Toc168116692)

[Isolating Web Sites by Application Pool 14](#_Toc168116693)

[Configure IIS 7.0 to run 32-bit mode worker processes 14](#_Toc168116694)

[Configure the Anonymous User to be the Application Pool Identity 14](#_Toc168116695)

[Setting Default Documents 15](#_Toc168116696)

[Configure Logging and Diagnostics 16](#_Toc168116697)

[Configure Caching 16](#_Toc168116698)

[Configure Compression 17](#_Toc168116699)

[Delegated Configuration 17](#_Toc168116700)

[Additional Resources 21](#_Toc168116701)

[Configuring Dynamic Content 22](#_Toc168116702)

[Recommended ASP.NET Configuration for Shared Hosting 22](#_Toc168116703)

[Trust Levels 22](#_Toc168116704)

[Configuring ASP.NET to Serve Content from a Remote UNC Share 24](#_Toc168116705)

[Configuring ASP.NET Temporary Compilation Directories 24](#_Toc168116706)

[Configuring ASP.NET to Use the WorkStation Garbage Collector 25](#_Toc168116707)

[FastCGI with PHP 26](#_Toc168116708)

[Recommended Settings for FastCGI with PHP 26](#_Toc168116709)

[ASP 28](#_Toc168116710)

[Recommended Settings for ASP 28](#_Toc168116711)

[Additional Resources 28](#_Toc168116712)

[Configuring Web Farm Support 29](#_Toc168116713)

[Shared Configuration 29](#_Toc168116714)

[Configuring Network Load Balancing 31](#_Toc168116715)

[Deploying Step-by-Step 33](#_Toc168116716)

[IIS Server Pre-Setup 33](#_Toc168116717)

[IIS 7.0 Server Setup and Deployment 33](#_Toc168116718)

[Code Samples for Provisioning and Management 36](#_Toc168116719)

[Microsoft.Web.Administration 36](#_Toc168116720)

[IIS 7 WMI Provider 38](#_Toc168116721)

[AppCmd.exe 39](#_Toc168116722)

[Other Managed Code Samples 40](#_Toc168116723)

# Overview

Hosting multiple Web sites on a single machine can be challenging. Hosters may struggle with quality of service, site density, security and performance configuration. IIS 7.0 is built into Microsoft Windows Server 2008 and makes many improvements in these areas.

It is a secure, easy to manage platform for developing and reliably hosting Web applications and services. With IIS 7.0, you can minimize patching and security risks with fine grained control over the Web server footprint and reduce administrative costs by managing Web infrastructure more efficiently.

IIS 7.0 also provides benefits for shared hosting environments, including support for throttling the number of active Web sites and centralizing your Web server configuration for multiple servers.

Some of the key differences or features include:

* Ability to share configuration between multiple servers
* Centralized management for servers and web farms
* Remote administration for non-administrators
* Custom delegation of features for each site or application
* Improved performance and security for CGI applications on Windows
* Isolated configuration for application pools

## Recommended Shared Hosting Configuration

Every hoster has a different way of running their shared hosting environment. Some hosters use an application pool for every site or place multiple sites in a single application pool. Some hosters use a remote file server or Network Appliance Server (NAS) for content storage while others place content on the local disk of the Web server. There are many different options available, so the IIS team decided to optimize and test IIS 7.0 for a specific shared hosting setup. This setup is intended as a baseline that can be changed in order to work better for different hosting environments.

**Shared Hosting Highlights**

* Configured and tested with up to 5,000 Web sites (not running concurrently)
* A traffic pattern of 90/10 was assumed (90% of traffic to 10% of sites)
* An Active Directory domain is used to provide accounts
* The application pool account is used as the anonymous identity (instead of requiring a separate account for the anonymous identity)
* Content is stored on a remote UNC share
* This guidance includes information about configuring ASP, ASP.NET and PHP

## Machines and Architecture Used In This Whitepaper

The following infrastructure and architecture was used to test the shared hosting deployment recommended in this whitepaper:

|  |  |  |
| --- | --- | --- |
| **Name** | **Function** | **Comments** |
| IIS7SHDC | Domain Controller | Active Directory domain accounts are used to ensure security across machines. |
| IIS7SHWS | Web Server | Web server running IIS 7.0 on Windows Server 2008. |
| IIS7SHRC | Remote File Server | File server running Windows. |

### Fault-Tolerance

The shared hosting architecture includes UNC for storage of content, to prevent the web server from being a single point of failure and additionally to allow easier migration of web servers. But this scenario does not provide a fully redundant solution.

For instance, the remote file server is a single server and a point of failure for the content. For a highly available hosting deployment, it is recommended that more than one back-end is configured using File System Replication v2 (FRSv2) or greater, available in Windows Server 2008 and Windows Server 2003 R2.

### Performance

The performance testing done by the IIS team covered only the specific scenario and configuration outlined in this whitepaper. The testing was performed on x64 machines, which are recommended for performance benefits. These performance benefits translate into increased scalability, as both the Web server and the Remote file server can take advantage of increased addressable memory.

The performance testing was performed using a Windows Server 2008 domain controller, but hosters may use a Windows Server 2003 domain controller.

## Additional Resources

Additional information about Windows Server 2008 and other information discussed in this overview can be found at the following locations:

* Microsoft Windows Server 2008 Home at <http://www.microsoft.com/windowsserver2008/default.mspx>.
* Microsoft IIS home page at <http://www.iis.net>.
* Benefits of Microsoft Windows x64 Editions at <http://www.microsoft.com/windowsserver2003/techinfo/overview/x64benefits.mspx>.

# Remote File Server

Many hosting scenarios involve a remote file server that provides content storage for a number of Web sites. These Web sites need a location for their content while also being isolated from each other and prevented from using too much disk space. In this section, we discuss certain recommendations for server administrators running in such environments with Windows-based file servers.

## Configuring Separate Partitions for Improved Performance

On the file server, performance may be improved by separating the system and content partitions and physical hard drives.

The following setup is recommended:

* Use a unique partition and physical drive(s) for the system.
* Use a separate partition and physical drive(s) for the content.

## Configuring Share and NTFS Permissions

The file server permissions should be carefully implemented to provide appropriate access to content. This involves locking down permissions on the share and physical folders.

The following table lists permissions that were used for the UNC share and folders. Based on the hosting environment used, server administrators should develop their own custom permissions that meet their needs.

| **Path** | **Permissions** | **Reason** |
| --- | --- | --- |
| \\server\share$  (*share*) | Domain Administrators – Full Control  Domain Users – Change  MachineAccounts$ - Full Control | The share permissions need to allow the administrators and site accounts to access the content. The physical path will be restricted to actual needed permissions. |
| E:\Content (*physical path of share*) | Administrators – Full Control  System – Full Control | This is the folder that is shared. It does not need permissions for any accounts aside from the built-in Administrators group and System account. |
| E:\Content\<sitename>  (the container for a specific site or user) | Administrators – Full Control  System – Full Control  Site Owner – List Folder Contents | This folder is used as a container for folders like the site’s home directory and its logfiles. The Site Owner should be able to read this folder but does not need write. |
| E:\Content\<sitename> \wwwroot  (the IIS home directory for the site) | Administrators – Full Control  System – Full Control  Site Owner – Modify  App Pool Username – Read | This is the root of a Web site belonging to the user account. App Pool Username is used as both the application pool identity and the anonymous username for the Web site. |
| E:\Content\<sitename>\Logs  (the container for logs) | Administrators – Full Control  System – Full Control  Site Owner – Read | Please note that this folder for logs is stored ABOVE the root of the site, so that it is not accessible by a visitor browsing the site. It is not recommended that you put this folder any location accessible from a web browser, for security purposes. |
| E:\Content\<sitename>\Logs\FailedReqLogs  (the container for failed request tracing logs) | Administrators – Full Control  System – Full Control  App Pool Username - Full Control | This is the folder used to store Failed Request log files, which allow a site owner to diagnose problems with their Web site. These logs are written by the worker process identity, App Pool Username. |
| E:\Content\<sitename>\Logs\W3SVCLogFiles  (the container for W3SVC traffic logs) | Administrators – Full Control  System – Full Control  MachineAccount$ - Full Control | This is the folder used to store the log files for the Web site, which allow a site owner to see their traffic patterns. If the server administrator doesn’t wish to share these files or wants to provide an alternate method for determining traffic, these files can be stored elsewhere.  MachineAccount$ is the Web server’s machine account, as these logs are written by HTTP.SYS. |

**To configure permissions for the share**

1. In Windows Explorer, right-click the folder you want to share, and then click **Properties**.
2. Click the **Sharing** tab and click the **Advanced Sharing** button.
3. Click **Continue** to accept the prompt that Windows needs your permission to perform the action.
4. Check the **Share this folder** option.
5. Set the **Share name** and **Comment** as appropriate. To make the share hidden, add a $ to the end of the share name.   
     
   Note: Hiding a share means that when you connect to [\\server](file:///\\server) you will not see the share unless you specifically enter the path [\\server\share$](file:///\\server\share$).
6. Click the **Permissions** button and remove the Everyone group, if it exists.
7. Add the appropriate user or group that should have access to the share.
8. Click **OK** to close the window(s).

**To configure permissions for the folder structure**

1. In Windows Explorer, right-click the folder you want to share, and then click **Properties**.
2. Click the **Security** tab and click the **Edit** button.
3. Add the appropriate users or groups that should have access at each level of the folder structure.
4. Click **OK** to close the window(s).

## Configuring SMB Commands for UNC (Network BIOS Commands)

On Windows servers, each connection as a unique user to another server uses an SMB connection. A connection is used when you open a remote file share in Windows Explorer or via a net use command. The same is true when you browse to a web page and IIS impersonates the configured user account to access the files.

For each connection, there are a number of work items or work contexts used. By default, this limit is set low and when storing IIS content or configuration on a UNC share, you may run out quickly. This results in the error “The network BIOS command limit has been reached”. There may be other errors, such as an ASP.NET or ASP 500 error.

There are a few registry key that control how many commands and work items are permitted. On the IIS servers, a registry key is used to control the number of connections or commands that can be made to a file server. On the file server, two registry keys control the number of connections and work items that are allowed.

When using UNC shares for content or configuration files, you need to set the registry keys on both the file server if it is running Windows, and the Web server.

**To configure the registry key on the file server**

1. From the command prompt, run regedt32 to open the registry.
2. Navigate to HKEY\_LOCAL\_MACHINE\SYSTEM\CurrentControlSet\Services\lanmanserver\parameters.
3. If it doesn’t exist, create a DWORD “MaxMpxCt” and set the value to 800 hexidecimal. This will specify a value of 2,048 decimal.
4. If it doesn’t exist, create a DWORD “MaxWorkItems” and set the value to 2000 hexidecimal. This will specify a value of 8,192 decimal, or 4x the MaxMptCt.

**To configure the registry key on the Web server**

1. From the command prompt, run regedt32 to open the registry.
2. Navigate to HKEY\_LOCAL\_MACHINE\SYSTEM\CurrentControlSet\Services\lanmanworkstation\parameters.
3. If it doesn’t exist, create a DWORD “MaxCmds” and set the value to 800 hexidecimal. This will specify a value of 2,048 decimal.

For more information, go to “IIS runs out of work items and causes RPC failures when connection to a re mote UNC path” at <http://support.microsoft.com/kb/221790/>.

# Front-End Web Server

IIS 7.0 is a secure and easy to manage platform for reliably hosting Web sites. To deploy a Web server and host thousands of Web sites on it, server administrators need to configure and isolate sites from each other. In this section, we discuss certain recommendations for server administrators running in such environments.

## Installation

IIS 7.0 provides a modular architecture that enables server administrators to customize exactly which features are installed and running on the Web server. It is factored into more than 40 features modules that can be independently installed on the box, dramatically reducing the potential attack surface, and lowering the footprint requirements on the machine.

For more information about the new modular architecture, go to <http://www.iis.net/default.aspx?tabid=7&subtabid=71>.

The following table lists the components that were installed on the Web server. Based on the hosting environment used, server administrators should develop their own set of components to install that meet their needs.

| **Component** | **Installed?** | **Reason** |
| --- | --- | --- |
| **Web Server** | Yes | This is the core Web server. |
| **Common HTTP Features** |  | Includes commonly used features such as enabling static content (HTML, jpeg, etc.) or default documents. |
| Static Content | Yes | Static content, such as .html, .css and .jpeg files, can be served by the Web server if this is enabled. |
| Default Document | Yes | When a visitor enters www.site.com but does not provide a document, such as default.aspx, this feature allows them to seamlessly be directed to the default document defined for the Web site. |
| HTTP Errors | Yes | Allows error messages sent to a visitor’s browser to be customized and for the server administrator to see the new detailed errors on the local Web server. |
| HTTP Redirection | Yes | Allows hosted customers to redirect requests for one URL to be redirected to another. |
| **Application Development** |  | Allows applications to be developed and run on the server, such as ASP.NET. |
| ASP.NET | Yes | ASP.NET ISAPI and modules for managed code applications (.aspx pages). |
| .NET Extensibility | Yes | Infrastructure required for ASP.NET. Allows developers to change and extend Web server functionality in the new request pipeline. |
| ASP | Yes | Required if customers use classic ASP applications. |
| CGI | Yes | Required for CGI applications, such as for PHP so that it can use the new FastCGI component. |
| ISAPI Extensions | Yes | Required for ASP.NET and other ISAPI extensions. |
| ISAPI Filters | Yes | Required for ASP.NET 1.1 ISAPI filter. |
| Server Side Includes | No | Not recommended unless customers are using Service Side Include files. |
| **Health and Diagnostics** |  | Provides infrastructure to monitor and troubleshoot the health of the Web server and sites. |
| HTTP Logging | Yes | Allows logging of Web site activity or traffic. |
| Logging Tools | No | Infrastructure for managing Web server logs. |
| Request Monitor | Yes | Allows requests to be monitored in-flight. Can be used to determine why a worker process is unresponsive or slow. |
| Tracing | Yes | Infrastructure to diagnose problems using Event Tracing in Windows and Failed Request Tracing. |
| Custom Logging | No | Support for logging activity in a different format from IIS generated log files. Not needed unless a custom format is used. |
| ODBC Logging | No | Infrastructure for logging activity to an ODBC compliant database. Not needed unless ODBC logging is used. |
| **Security** |  | Infrastructure for securing requests and filtering incoming requests based on security rules. |
| Basic Authentication | No | Clear-text authentication method that can be secured with SSL. Not needed if only anonymous authentication is used. |
| Windows Authentication | No | Hash-based authentication method. Not needed if only anonymous authentication is used. |
| Digest Authentication | No | Hash-based authentication method. Not needed if only anonymous authentication is used. |
| Client Certificate Mapping Authentication | No | Allows client certificates to be used to authenticate users. This type of client certificate mapping uses Active Directory. |
| IIS Client Cert Mapping Authentication | No | Allows client certificates to be used to authenticate users. This type of client certificate mapping uses IIS natively. |
| URL Authorization | Yes | Allows users to create rules that restrict access to content. |
| Request Filtering | Yes | Recommended security feature. Screens incoming requests based on rules sets. |
| IP and Domain Restrictions | No | Content can be allowed or denied to users based on the originating IP address or domain name of the request. |
| **Performance** |  |  |
| Static Content Compression | Yes | Allows static content to be compressed and unlike dynamic responses, compressed static responses can be cached without degrading CPU resources. |
| Dynamic Content Compression | Yes | Allows dynamic compression, utilizing bandwidth more efficiently, but may add a CPU load. |
| **Management Tools** |  |  |
| IIS Management Console | Yes | Needed to manage IIS locally, provides a user interface (UI) for server management. |
| IIS Management Scripts and Tools | Yes | Needed to script tasks, allows programmatic management of the server using scripts. |
| Management Service | Yes | Needed to allow remote management of IIS and to allow delegated users to administer their sites using a remote manager tool. |
| IIS 6 Management Compatibility | No | Do not install unless compatibility with scripts and management tools for IIS 6.0 is required. |
| IIS Metabase Compatibility | No | Provides compatibility for scripts based on IIS 6 interfaces for ADSI (Active Directory Service Interface) and ABO (Admin Base Object). |
| IIS 6 WMI Compatibility | No | Provides compatibility for scripts based on the IIS 6 WMI (Windows Management Instrumentation) provider. |
| IIS 6 Scripting Tools | No | Provides compatibility for scripts based on IIS 6 interfaces for ADO (ActiveX Data Objects) or ADSI APIs. |
| IIS 6 Management Console | No | Provides the IIS 6 based management tool. |
| **FTP Publishing Service** | No |  |
| FTP Server | No | Only needed if users upload using FTP. |
| FTP Management Console | No | Only needed if users upload using FTP. |

**Note:** Install only the components required for the hosting environment. For example, if Digest Authentication is not needed, do not enable it.

**To install the Web Server role using Server Manager**

1. Navigate to **Administrative Tools** and click on **Server Manager**.
2. Click on the **Go to Manage Roles** link.
3. Click **Next** on the Before You Begin page.
4. Select the **Web Server** role.
5. Click **Next**.
6. Click **Add Required Features** button on the **Add Roles Wizard** prompt that appears.
7. Click **Next**.
8. Add the appropriate role services per the installation table.
9. Continue through the installation wizard and click **Next** or **Finish** as appropriate.

## Backing Up the Web Server

Once you have installed IIS 7.0, it is a good time to backup your configuration and become more familiar with the built-in command-line tool, appCmd.exe. You can run appCmd.exe to create a backup of your Web server before you’ve changed any configuration.

**To create a backup using appCmd**

1. Open a command prompt and CD to the %windir%\system32\inetsrv\ directory.
2. Type *appcmd add backup “FirstBackup”* and hit **Enter**.

This will create a backup with the name “FirstBackup”. For more information about appCmd.exe, go to <http://www.iis.net/default.aspx?tabid=2&subtabid=25&i=954>.

## Recommended IIS Configuration for Shared Hosting

The following settings were used on the Web server to ensure security and performance in a hosted environment.

### Isolating Web Sites by Application Pool

Each application in its own application pool, configured with a unique process identity, offers the highest level of isolation. Each application has its own process and if the application stops responding, it is unable to affect other sites on the server. For these reasons, this is the recommended configuring for a shared hosting environment.

IIS 7.0 introduces a configuration isolation services that prevents application pools from needing to access to the applicationHost.config. At startup time, IIS reads the applicationHost.config and writes a temporary file for each application pool that contains the configuration for that pool. The worker process only has access to that file, and not to other pool configuration files. This feature was introduced to provide both security and performance improvements.

### Configure IIS 7.0 to run 32-bit mode worker processes

Another one of the performance benefits of the x64 platform is that it increases virtual address space, so more memory is available. It is recommended that server administrators configure IIS to use 32-bit worker processes on 64-bit Windows. Not only is compatibility better than native 64-bit, performance and memory consumption are also better.

The server should be set to enable this globally so that administrators do not need to set it for each application pool. But unlike previous versions of IIS, server administrators can now configure specific worker processes to run in 32-bit or 64-bit mode independently of each other on the same server.

**To configure 32-bit mode for the server**

1. Open a command prompt and run the following:

%windir%\system32\inetsrv\appcmd set config -section:applicationPools -applicationPoolDefaults.enable32BitAppOnWin64:true

### Configure the Anonymous User to be the Application Pool Identity

In prior versions of IIS, the anonymous user allowed classic ASP content to be isolated by using a different anonymous user for each site. Isolation was accomplished by preventing ASP scripts from reverting back to the process identity, called the “impersonation sandbox”. In mixed environments where classic ASP, ASP.NET and other dynamic content is used, the impersonation sandbox is not as helpful. For instance, ASP.NET uses a different isolation technique called Trust Levels and always runs as the process identity, unless a different type of impersonation is used.

To achieve isolation in IIS 7.0, the application pools can simply be run as separate identities. If no anonymous user account is configured for a Web site, IIS can be configured to automatically use the application pool identity. This can greatly reduce the number of accounts needed for Web sites and make management of the accounts easier.

**To configure the Web server to use the application pool identity as the anonymous identity**

1. Open a command prompt and run the following:  
     
   %windir%\system32\inetsrv\appcmd set config -section:anonymousAuthentication /username:"" --password

**Note:** Enabling this feature is done by setting the anonymous user attribute to a blank string, i.e., username=””. Removing the element from the applicationHost.config file will result in a different behavior than having a blank string.

### Setting Default Documents

The default document is for client requests to a Web site that do not specify a document name, such as www.microsoft.com. The default document specifies what file to serve. Multiple default documents can be configured for a Web site, but there is a performance benefit to using fewer documents.

For each request, IIS must get the list of default documents and look for each file in the content path until it finds the first match. If 5 documents are configured and the last document is the one used by the site, this increases the length of time spent. However, if you use one default document or the first in the list, this speeds up the request.

Using the new delegation functionality of IIS 7.0, server administrators can delegate this property to the site owner and encourage the site owner to use only 1 document.

**To configure the default documents globally using the IIS Manager**

1. Navigate to **Administrative Tools** and click on **Internet Information Services (IIS) Manager**.
2. Click on the server name node.
3. Scroll down to locate the **Default Document** icon and double click on it.
4. In the **File name(s)** field, enter your default documents, separated by commas.   
     
   **Note:** Set this to as few default documents as possible. If the property is delegated, users can add specific documents and limit it to 1 document to get better performance for their sites.
5. Click **Apply** in the Actions pane.

To configure the default document or other properties for delegation, please see the topic “*Delegating Configuration*” later in this section.

### Configure Logging and Diagnostics

IIS 7.0 provides the Web site activity or traffic logging that was provided in IIS 6.0, but also includes the new Failed Request Tracing. Request-based tracing provides a way to determine what is happening with a request, allowing server administrators to look at each step of the request as it is processed by IIS.

Problems such as poor performance or authenticated failures, even the Server Error 500 from ASP or ASP.NET, can be difficult to troubleshoot. Failed Request Tracing is designed to buffer the trace events for a request and write them to disk when a failure occurs. A failure is defined by the server administrator using the tracing rules. The tracing rules can be as broad as all content and status codes, or as specific as a single status and substatus code like 404.2.

For shared hosting, it is recommended that server administrators delegate Failed Request Tracing to site owners, enabling site owners to determine problems with their sites and helping prevent support calls. The tracing logs can be stored in the logs\FailedRequestLogFiles as discussed in the earlier topic “*Configuring Share and NTFS Permissions*” in the “*Remote File Server*” section.

**To configure the Failed Request Tracing logfile location using the command line**

1. Open a command prompt and run the following:

%windir%\system32\inetsrv\AppCmd set config <sitename> -section:traceFailedRequestsLogging.directory:\\remotefileserver\content$\<sitename>\logs\failedRequestLogfiles

It is also recommended that server administrators configure all logging fields for W3SVC logging, as it may be needed when a problem has occurred.

**To configure the W3SVC logfile location using the command line**

1. Open a command prompt and run the following:

%windir%\system32\inetsrv\AppCmd set config <sitename> -section:–logfile.directory:\\remotefileshare\content$\<sitename>\logs\logfiles

For more information, go to the Diagnostics overview for IIS 7 at <http://www.iis.net/default.aspx?tabid=7&subtabid=76>.

### Configure Caching

The IIS 7.0 output cache is a new feature that makes it possible to cache entire responses in memory, even from dynamic content. Unlike existing caches that IIS 6.0 has, it has a “smart” cache, which allows site owners and developers to configure it to cache separate copies of responses based on query string value. It is also integrated with the HTTP.sys kernel cache, allowing for fast performance. Kernel caching is unlocked by default, developers can take advantage of this feature by configuring caching profiles within their application. Server administrators can run a command line tool that shows what content is in the HTTP.sys.

**To view the HTTP response cache**

1. Open a command prompt and run the following:  
     
   netsh http show cache

### Configure Compression

Dynamic compression can save a signifcant amount of bandwidth with a tradeoff in using extra CPU. If bandwidth savings are more important than additional CPU, and the server is not currently bound by CPU, dynamic compression should be enabled.

The amount of CPU utilized by dynamic compression can be controlled by a setting on the server called dynamicCompressionDisableCpuUsage. This ensures that if the system reaches the specified level of CPU usage, dynamic compression will not cause additional load. The default value for disabling the CPU usage is 90%. Depending on requirements, server admins should determine a number that works for their environment.

**To configure dynamic compression**

1. Open a command prompt and run the following:   
     
   %windir%\system32\inetsrv\appcmd set config -section:urlCompression /doDynamicCompression:true  
     
   %windir%\system32\inetsrv\AppCmd set config section:–httpCompression /dynamicCompressionDisableCpuUsage:50  
     
   %windir%\system32\inetsrv\AppCmd set config section:–httpCompression /dynamicCompressionEnableCpuUsage:30

## Delegated Configuration

The IIS 7.0 administrative user interface, called the IIS Manager, offers a new, more efficient tool for managing the Web server. It provides support for IIS and ASP.NET configuration settings. It also enables those who host or administer Web sites to delegate administrative control to developers or content owners, thus reducing cost of ownership and administrative burden for the server administrator. It supports remote connections over HTTP and can be used through a firewall.

For more information about the new administration features in IIS 7.0, go to <http://www.iis.net/default.aspx?tabid=7&subtabid=73>.

Before enabling the remote delegation, server administrators should consider what features and properties they wish to delegate to site owners.

The following table lists the features that were delegated on the Web server. Based on the hosting environment used, server administrators should develop their own set of delegated features that meet their needs.

| **Feature** | **Delegated Setting** | **Reason** |
| --- | --- | --- |
| .NET Compilation | Read Only  (changed from Read/Write) | Specifies settings for ASP.NET compilation processing directives like the temporary compilation directory.  Prevent users from setting the temporary compilation directory manually. |
| .NET Globalization | Read/Write | Specifies settings for default culture and globalization properties for Web requests. |
| .NET Profile | Read/Write | Specifies settings for user selected options in ASP.NET applications. |
| .NET Roles | Read/Write | Specifies settings for groups for use with .NET users and forms authentication. |
| .NET Trust Levels | Read Only  (changed from Read/Write) | Specifies the trust level. By locking down the trust level when you follow the ASP.NET guidance in this document, you will be setting this to Read Only and locking it for the server.  Prevent Web site owners from setting the trust level to a higher level than set by the server administrator. Ex: If a custom trust level is set by the administrator, this setting should be set to Read Only so it cannot be overridden. |
| .Net Users | Configuration R/W | Specifies settings for management of users who belong to roles and use forms authentication. |
| Application Settings | Read/Write | Specifies settings for storing data (name and value pairs) that managed code applications can use at runtime. |
| ASP | Read Only | Specifies Classic ASP settings. |
| ASP.NET Impersonation | Read/Write | Specifies impersonation settings. Site owners can use this to run their site under a different security context. |
| Authentication – Anonymous | Read Only | Specifies anonymous authentication settings. |
| Authentication – Forms | Read/Write | Specificies forms authentication settings. |
| Authentication – Windows | Read Only | Specifies Windows authentication settings. |
| Authorization Rules | Read/Write | Specifies the list of Allow or Deny rules that control access to content. |
| CGI | Read Only | Specifies properties for CGI applications.  Should be left Read Only to prevent users from changing settings. |
| Compression | Read/Write | Specifies settings to configure compression. |
| Connection Strings | Read/Write | Specifies connection strings that applications can use. |
| Default Document | Read/Write | Specifies default documents for the web site.  By leaving this Read/Write, users will be able to specify a custom default document for their site without contacting the server admin. |
| Directory Browsing | Read/Write | Specifies directory browsing settings. |
| Error Pages | Read Only | Specifies what HTTP error responses are returned. |
| Failed Request Tracing Rules | Read/Write | Specifies settings for failed request tracing rules. Enables users to create rules for tracing requests based on parameters like time taken or status code, and diagnose problems with their site. |
| Feature Delegation | Remove Delegation  (changed from Read/Write) | Specifies settings for delegating features to applications.  It can be turned off unless server admins wish to enable this feature for site owners. |
| Handler Mappings | Read/Write |  |
| HTTP Response Headers | Read/Write | Specifies HTTP headers that are added to responses from the web server. |
| ISAPI Filters | Read Only | Specifies ISAPI filters that process requests made to the site or server, such as ASP.NET. |
| Logging | Remove Delegation |  |
| Machine Key | Read/Write | Specifies hashing and encryption settings for applications services, such as view state, forms authentication and membership and roles. |
| MIME Types | Read Only | Specifies what file types can be served as static files. |
| Modules | Read/Write | Specifies native and managed code modules that process requests made to the site or server. |
| Output Caching | Read/Write | Specifies rules for caching output. |
| Pages and Controls | Read/Write | Specifies page and control settings for applications. |
| Redirect Rules | Read/Write | Specifies settings for redirecting requests to another file or URL. |
| Session State | Read/Write | Specifies session state and forms authentication cookie settings. |
| SMTP E-mail | Read/Write | Specifies email address and delivery options for email sent from the site. |
| SSL Settings | Read Only | Specifies settings for SSL. |

**To enable the Remote Delegation Service using the IIS Manager**

1. Navigate to **Administrative Tools** and click on **Internet Information Services (IIS) Manager**.
2. Click on the server name node.
3. Scroll down to locate the **Feature Delegation** icon and double click on it.
4. Change any properties that should or should not be delegated.
5. Click the Back button or select the server name node to return to the server feature list.
6. Double-click on the **Management Service** icon.
7. Start the service to enable config.
8. Stop the service to make changes.
9. Check **Allow remote connections**.
10. Select whether you wish to allow only Windows users or both Windows and membership users.
11. Change the port or certificate if desired.
12. Click **Start** in the right-hand task pane to enable the Remote Delegation service.

For more information, go to the *Remote Administration for IIS Manager* overview at <http://www.iis.net/articles/view.aspx/IIS7/Use-IIS7-Administration-Tools/IIS-Manager-Administration-Tool/Remote-Administration-for-IIS-Manager>.

**To enable a delagated user for a Web site using the IIS Manager**

1. Navigate to **Administrative Tools** and click on **Internet Information Services (IIS) Manager**.
2. Expand the server name node.
3. Click on the Sites node.
4. Double-click on the name of the site you want to enable the user on.
5. Under the **Management** grouping at the bottom of the page, double-click on the **IIS Manager Permissions** icon.
6. Click on **Allow User…** task in the **Actions** pane.
7. In the dialog, choose whether to use a Windows user or an IIS Manager user. In this case, we’ll select a Windows user, enter the name and click **OK**.

For more information, go to the *Configuring Remote Administration and Feature Delegation* overview at <http://www.iis.net/articles/view.aspx/IIS7/Managing-IIS7/Delegation-in-IIS7/Delegating-permissions-in-IIS-Manager/Configuring-Remote-Administration-and-Feature-Dele?Page=4>.

## Additional Resources

Additional information about IIS 7.0 and other topics discussed in this section can be found at the following resources:

* For information about the new modular architecture, go to <http://www.iis.net/default.aspx?tabid=7&subtabid=71>.
* For more information about appCmd, go to <http://www.iis.net/default.aspx?tabid=2&subtabid=25&i=954>.
* For more information about the new administration features in IIS 7.0, go to <http://www.iis.net/default.aspx?tabid=7&subtabid=73>.

# Configuring Dynamic Content

IIS 7.0 offers enhancements for dynamic content, especially ASP.NET. In IIS 7.0, the ASP.NET runtime extensibility model is integrated into the core server. This allows developers to fully extend IIS with the rich functionality of ASP.NET 2.0 and the .NET Framework, instead of using lower-level C++ APIs. Existing ASP.NET applications will benefit from the closer integration by being able to use existing features like forms authentication, roles and output caching for all types of content, not just .aspx pages.

To learn more about some of the key features and benefits, go to <http://www.iis.net/default.aspx?tabid=2&subtabid=25&i=928>.

## Recommended ASP.NET Configuration for Shared Hosting

The following settings were used on the Web server to ensure security and performance in a hosted environment for ASP.NET. This guidance is provided for IIS 6.0 in the *ASP.NET 2.0 Deployment Guide*.

To learn more about why these settings are recommended and additional details on how to configure them, download the *ASP.NET 2.0 Deployment Guide* at <http://www.microsoft.com/downloads/details.aspx?FamilyID=9E33EA25-666C-47FA-AC52-8D04785C4BD2&displaylang=en>.

### Trust Levels

Trust levels allow you to define security rules for ASP.NET. They define what types of operations an application can perform, such as reading from disk or accessing the registry. The recommended trust level is Medium, which allows many common operations and restricts many of the dangerous applications. If more operations need to be permitted than Medium trust allows, you should create a custom trust level.

**To configure a custom trust level and configure additional permissions**

1. Navigate to the %windir%\Microsoft.NET\Framework\{version}\CONFIG directory.
2. Copy the Medium trust policy file, web\_mediumtrust.config, to create a new policy file in the same directory (%windir%\Microsoft.NET\Framework\{version}\CONFIG\).
3. Give it a name that indicates that it is a custom trust level. For instance, it could be named web\_CustomTrust.config.
4. Open the new file, web\_CustomTrust.config and location the WebPermission.

...

<IPermission class="WebPermission"

version="1"

<ConnectAccess>

<URI uri="$OriginHost$" />

</ConnectAccess>

</IPermission>

...

1. Edit WebPermission so that the ConnectAccess element is removed and Unrestricted is set to true.

...

<IPermission class="WebPermission"

version="1"

**Unrestricted="true"/>**

...

1. Comment out PrintingPermission, EnvironmentPermission and DnsPermission unless it is known they are needed for the hosting environment.
2. Add the new <trustLevel> element to the <securityPolicy> section of the Web.config file, as shown below:

<location allowOverride="true">

<system.web>

<securityPolicy>

<trustLevel name="Full" policyFile="internal" />

<trustLevel name="High" policyFile="web\_hightrust.config" />

<trustLevel name="Medium" policyFile="web\_mediumtrust.config" />

<trustLevel name="Low" policyFile="web\_lowtrust.config" />

<trustLevel name="Minimal"

policyFile="web\_minimaltrust.config" />

**<trustLevel name="Custom" policyFile="web\_CustomTrust.config" />**

</securityPolicy>

<trust level="Full" originUrl="" />

</system.web>

</location>

1. Lock the trust level so that it cannot be changed by applications on the server, by setting the location allowOverride element to false.

**<location allowOverride="false">**

<system.web>

<securityPolicy>

<trustLevel name="Full" policyFile="internal" />

<trustLevel name="High" policyFile="web\_hightrust.config" />

<trustLevel name="Medium" policyFile="web\_mediumtrust.config" />

<trustLevel name="Low" policyFile="web\_lowtrust.config" />

<trustLevel name="Minimal"

policyFile="web\_minimaltrust.config" />

<trustLevel name="Custom" policyFile="web\_CustomTrust.config" />

</securityPolicy>

<trust level="Full" originUrl="" />

</system.web>

</location>

### Configuring ASP.NET to Serve Content from a Remote UNC Share

When configuring ASP.NET work with content stored on a remote UNC share, a few additional steps are required.

When IIS gets content from a remote UNC share, it impersonates the authenticated user. This is either the user specifically set as the virtual directory user, or if one is not specified, the anonymous user. For our IIS 7.0 guidance, we state that the application pool user should be used as the anonymous user, so in this case the impersonated identity is the application pool identity. As you can see, using one identity for all access (anonymous, application pool and virtual directory) greatly simplifies a deployment.

In order to run some ASP.NET applications from a remote UNC share, the impersonated user will need access to the Windows temp and the temporary compilation directories.

**To serve ASP.NET content from a UNC share**

1. Grant permission for all application pool identities to the %windir%\Temp directory by editing the ACLs for the folder.
2. Add a Code Access Security (CAS) policy to allow ASP.NET appropriate access to the UNC share. To do this, open a command prompt and run the following:  
     
   caspol -m -ag 1. -url "file://\\remotefileserver\content$\\*" FullTrust

### Configuring ASP.NET Temporary Compilation Directories

By default, the temporary compilation directory is   
“%windir%\Microsoft.NET\Framework\{version}\Temporary ASP.NET Files”. This directory contains all the temporary files that get generated by ASP.NET. This directory is shared between all application pools. Although not a requirement when Medium Trust is used it is a good security practice to create a compilation directory for each Application Pool.

**To create and set a unique temporary compilation directory**

1. Open a command prompt and run the following command to create a folder and set permissions:  
     
   md %systemdrive%\inetpub\tempcompilation\aspnettemp\<sitename>  
     
   cacls %systemdrive%\inetpub\tempcompilation\aspnettemp\<sitename> /G AppPoolUsername:F Administrators:F System:F
2. Run the following command to set the temporary compilation directory a site:  
     
   %windir%\system32\inetsrv\appcmd set config "<sitename>" -section:compilation -tempDirectory:%systemdrive%\inetpub\tempcompilation\aspnettemp\site%1 /commit:webroot

Configuring ASP.NET to Use the WorkStation Garbage Collector

By default, the Server Garbage Collector (GC) is used by the .NET Framework to optimize speed and CPU load. For shared hosting, where memory is more often the constraint for number of active sites, the Workstation GC should be used. The Workstation GC optimizes for memory.

**To use the Workstation GC**

1. Navigate to the %windir%\Microsoft.NET\Framework\{version} directory.
2. Open the aspnet.config file and locate the <gcServer> element.

...

<configuration>   
<runtime>   
<gcServer enabled="true"/>   
</runtime>   
</configuration>  
...

1. Set the gcServer enabled mode to false:

...

<configuration>   
<runtime>   
**<gcServer enabled="false"/>**</runtime>   
</configuration>  
...

1. Save the file and close it.

To learn more about this setting, go to the KB article “*You may receive an error message, or the computer stops responding, when you host Web…”* at <http://support.microsoft.com/kb/911716>.

## FastCGI with PHP

The FastCGI is a new component in IIS 7.0 that enables popular application frameworks like PHP to be hosted on IIS in a high-performance and reliable way. FastCGI provides a high-performance alternative to the Common Gateway Interface (CGI), a standard way of interfacing external applications with Web servers.

CGI programs are executables launched by the web server for each request in order to process the request and generate dynamic responses that are sent back to the client. Because many of these frameworks do not support multi-threaded execution, CGI enables them to execute reliably on IIS by executing exactly one request per process. Unfortunately, it provides poor performance due to the high cost of starting and shutting down a process for each request.

FastCGI overcomes this performance penalty by re-using CGI processes to service subsequent requests, while continuing to ensure single request concurrency. The non-threadsafe version of PHP 5 is substantially faster than the threadsafe version.

### Recommended Settings for FastCGI with PHP

Only a few FastCGI settings need to be changed in order for PHP to run for shared hosting. It is recommended that server administrators do not change any settings other than any documented here, unless necessary for their environment.

However, it is recommended that you use PHP’s php.ini file to lock down PHP. By default, builds from Zend such as 5.2.x include a php.ini-recommended file that includes PHP’s recommended settings. Server administrators should explore this file and the PHP Web site for more information about PHP security.

If the environment is following recommended settings for using a unique application pool and identity, PHP FastCGI impersonation does not offer any additional benefits and does not need to be used.

**To install PHP**

1. Download the non-thread safe version of PHP and copy the files to a local folder, such as c:\{php\_version}.

**To configure the PHP / FastCGI handler mapping from IIS Manager**

1. Navigate to **Administrative Tools** and click on **Internet Information Services (IIS) Manager**.
2. Click on the server name node.
3. Scroll down to locate the **Handler Mappings** icon and double click on it. There is a list of existing handler mappings for the server, which define the extensions that the server knows how to process and what handler is responsible for processing them.
4. In the **Request Path** field, enter \*.php to enable this handler for all PHP requests.
5. Select FastCGIModule in the **Module** drop-down list.
6. In the **Executable (optional)** field, enter the path to php-cgi.exe, such as c:\{php\_version}\php-cgi.exe. Note: If you’re using PHP 4, this would be php.exe.
7. Enter a name in the **Name** field and click OK. You will be prompted by a dialog box to create a FastCGI application pool. Click Yes, so that the configuration needed for php-cgi.exe is created automatically.

**To configure the PHP / FastCGI handler mapping from command line**

1. Run the following commands from a command prompt:   
     
   %windir%\system32\inetsrv\appcmd set config /section:system.webServer/fastCGI /+[fullPath='c:\{php\_version}\php-cgi.exe']  
     
   %windir%\system32\inetsrv\appcmd set config /section:system.webServer/handlers /+[name='PHP-FastCGI',path='\*.php',verb='\*',modules='FastCgiModule',scriptProcessor='c:\{php\_version}\php-cgi.exe',resourceType='Either']

Note: If you’re using PHP 4, php-cgi.exe would be php.exe.

**To configure PHP maximum instances**

1. Run the following command from a command prompt:

%windir%\system32\inetsrv\appcmd set config -section:fastCgi /[fullPath='c:\{php\_version}\php-cgi.exe'].maxInstances:2

**Configuring PHP on a web farm**

If you have a web farm with multiple front-end Web servers and you’re using shared configuration, you will need to ensure that PHP exists on every front-end Web server in the same location.

For more information, go to the *Using FastCGI to host PHP applications on IIS7* walkthrough at <http://www.iis.net//articles/onepagearticle.ashx/IIS7/Hosting-Web-Applications/PHP/Using-FastCGI-to-host-PHP-applications-on-IIS7>.

## ASP

Classic ASP is included in IIS 7.0, but there is only one setting that should be changed. Because the ASP disk template cache is not shared hosting aware, it should be disabled. This should not pose a problem because up to 500 ASP pages per worker process are cached in memory before anything is written to the disk.

### Recommended Settings for ASP

It is recommended that the following settings are turned off for shared hosting environments.

**To turn off the ASP Template Cache**

1. Open a command prompt and run the following:

%windir%\system32\inetsrv\appcmd set config -section:asp -cache.maxDiskTemplateCacheFiles:0

## Additional Resources

Additional information about dynamic content on Windows and other topics discussed in this section can be found at the following resources:

* To learn more about why these ASP.NET settings are recommended and additional details on how to configure them, download the *ASP.NET 2.0 Deployment Guide* at <http://www.microsoft.com/downloads/details.aspx?FamilyID=9E33EA25-666C-47FA-AC52-8D04785C4BD2&displaylang=en>.

# Configuring Web Farm Support

IIS 7.0 has a new feature called shared configuration that improves the web farm experience for server administrators. This section covers how to configure shared configuration and network load balancing (NLB) for IIS 7.0.

## Shared Configuration

In IIS 6.0, synchronizing and replicating configuration between servers was difficult because no built-in functionality existed. Although the server administrator could use the script iiscnfg.vbs to copy configuration between servers, this would still leave ACLs on the metabase and local paths like the Windows directory to be fixed.

In IIS 7.0, there is a new feature called shared configuration that enables administrators to store the IIS server configuration files applicationHost.config and administration.config on a remote share. It can be used when building a web farm to resolve issues related to replication and synchronization of configuration.

**What is copied by shared configuration?**

* applicationHost.config
* administration.config

**What is not copied?**

* machine.config
* root Web.config
* Web.config files for sites and applications (these are part of the Web site content and can be stored on a share)
* Web site content (can be stored on a share)
* Custom modules
* Installed modules (the installed modules should be configured the same way on each server)

**To setup shared configuration**

1. Create a user that will access the share containing the configuration. This can be a domain user or a local user, if the machine is not joined to a domain.   
     
   Note: If the machine is not joined to the domain, the account will need to be created on the Web server(s) and the file server.
2. Create a share for storing the configuration files and set permissions.This can be done using the following commands:  
     
   md %SystemDrive%\centralconfig  
   net share centralconfig$=%SystemDrive%\centralconfig /grant:ConfigUser,Read /grant:Administrators,Full
3. Navigate to **Administrative Tools** and click on **Internet Information Services (IIS) Manager**.
4. Click on the server name node.
5. Scroll down to locate the **Shared Configuration** icon and double click on it.
6. To export the configuration files from the local machine to another location, click on the **Export Configuration** task in the **Actions** pane.
7. Enter the path to the share created in step 2, and a password that will be used to protect the encryption keys, then click **OK**.
8. Check the **Enable shared configuration** checkbox and enter the share path.
9. Click **Connect As…** and enter the user account created in step 1. This account will be used to access the share.
10. Click **Apply** in the **Actions** pane to save the settings.
11. Enter the encryption key password in the shared configuration dialog that appears and click **OK**.

By clicking the on the Export Configuration task, the UI will export the applicationHost.config and administration.config files to the path, as well as the encryption keys that are used by IIS to encrypt properties like custom application pool identity passwords. By default, there are no encrypted properties in the applicationHost.config file, unless the server administrator has entered custom identities for an application pool or web site.

Shared configuration itself works by using a new file called redirection.config, located in the %windir%\system32\inetsrv\config folder, to specify a path and username for accessing the configuration files. When the shared configuration checkbox is enabled, the UI updates the redirection.config file with the path and username.

While this task can be performed manually (or programmatically) by xcopying the files and updating the redirection.config file, the server administrator will also need to export and import the encryption keys unless they intend not to use encryption. The UI handles all of these tasks seamlessly for the administrator.

For more information, go to the *Centralized Configuration* walkthrough at <http://www.iis.net/articles/view.aspx/IIS7/Use-IIS7-Administration-Tools/Using-XML-Configuration/Centralized-Configuration>.

## Configuring Network Load Balancing

Network load balancing in Windows Server 2008 provides traffic distribution using TCP/IP and can be used with IIS 7.0 shared configuration feature to create a web farm that provides redundancy and fault-tolerance. NLB works by balacning traffic between the nodes in a web farm or cluster. Servers emit a heartbeat message to other hosts in the cluster and listen for the hearbeat of other hosts. If a host fails, the remaining hosts adjust and redistribute the workload.

NLB does not monitor the health of your application. Instead, it allows the application developer to determine how healthy a load-balanced application is. Since each application has its own notion of load and health, measuring and monitoring these quantities is best achieved by the application itself. By using collected measurements from your application and NLB’s public WMI provider, it is a relatively simple task to add load and health monitoring to your load-balanced application.

Note: It is recommended that the servers have two networks, one for the public cluster IP address that is shared by every host or node in the cluster. This will likely be used as the IP for web sites on the server. The second one is for private traffic between the nodes of a web farm. For example, if the server is using an internal network for traffic to internal resources like the file server or Active Directory serrvers, this could be used for traffic between the nodes.

**To install NLB**

1. Navigate to **Administrative Tools** and click on **Server Manager.**
2. Scroll down to the **Features** section or click on the **Features** node in the left-hand tree view.
3. Click on the **Add Features** link.
4. In the wizard, select **Network Load Balancing** from the list of available optional components.
5. Click **Next** and **Finish** as applicable to complete the wizard.

**To configure NLB**

1. Navigate to to **Administrative Tools** and click on **Network Load Balancing Manager**, or run nlbmgr from a command prompt.
2. Right-click **Network Load Balancing Clusters** and click **New Cluster**.
3. Connect to the host that will be part of the cluster, in this case the Web server. In the **Host** field, enter the name of the host and click **Connect**.
4. Select the interface you want to use with the cluster and click **Next**.
5. In the **Host Parameters** field, type a value in the **Priority (Unique host identifier)**.
6. In the **Cluster IP Addresses** field, click **Add** to enter the IP address that is shared by every host in the cluster. NLB will add this IP address to the TCP/IP stack on the selected interface of all hosts chosen to be part of the cluster. Click **Next** to continue.
7. In **Cluster Parameters** dialog, type the IP address and subnet mask.
8. In **Cluster operation mode**, click **Unicast** to specify that a unicast media access control (MAC) address should be used for cluster operations. Click **Next** to continue.
9. In **Port Rules**, click **Edit** to modify the default port rules if you need advanced rules. Otherwise, you may leave this default.
10. Click **Finish** to create the cluster.  
      
    To add more hosts to the cluster, right-click the new cluster and then click **Add Host to Cluster**.

# Deploying Step-by-Step

This section provides a basic overview of the steps to deploy a front-end Web server running IIS 7.0 and a back-end file server running Windows Server 2008. This setup will be using the information in this document to install the servers and configure recommended settings.

## IIS Server Pre-Setup

Before configuring permissions on the file server, it is necessary to join the web server.

Note: It is possible not to use domain accounts, but the setup in this whitepaper assumes a domain-joined machine. Certain things need to be changed for non-domain scenarios (if you’re using content on UNC, the same user account with same password must exist on both the web server and file server, for instance).

1. Join the web server to the domain.
2. If you will be storing content or configuration on a UNC share, configure the MaxCmds registry key on the Web server.
3. If you will be using a Windows-based file server, MaxMptCt and MaxWorkItems on the file server.

## IIS 7.0 Server Setup and Deployment

1. Install IIS 7.0.
2. Run appCmd to create an initial backup.  
     
   %windir%\system32\inetsrv\appcmd add backup "FirstBackup"
3. Configure 32bit mode globally on the server.

%windir%\system32\inetsrv\appcmd set config -section:applicationPools -applicationPoolDefaults.enable32BitAppOnWin64:true

1. Configure the anonymous user to be the worker process identity globally on the server.  
     
   %windir%\system32\inetsrv\appcmd set config -section:anonymousAuthentication /username:"" --password
2. Configure default docs at the server level in IIS Manager.
3. Configure logging fields at the server level in IIS Manager.
4. Configure the logfile directory per site.  
     
   %windir%\system32\inetsrv\AppCmd set config <sitename> -section:–logfile.directory:\\remotefileshare\content$\<sitename>\logs\logfiles
5. Configure failed request tracing directory per site.

%windir%\system32\inetsrv\AppCmd set config <sitename> -section:traceFailedRequestsLogging.directory:\\remotefileserver\content$\<sitename>\logs\failedRequestLogfiles

1. Configure delegated properties.
2. Enable Shared Configuration, if needed for your environment.
3. Enable Remote Web Service manager in IIS Manager.   
     
   Note: If you are using Shared Configuration, you will need to set WMSVC to run as a custom identity that has read access to the configuration share.
4. Enable the WMSVC port within the local firewall on your Web server by opening the Control Panel and double-clicking the Windows Firewall icon. Then select “Change firewall settings” and add the port for the service.  
     
   Note: The firewall is enabled by default and you will need to add the default port (8172) or custom port if you chose one.
5. Create a custom trust level based on Medium trust.
   * Set unrestricted true for webpermission
   * Add custom trust to web.config and lock it
6. Enable Full trust to the remote file share.  
     
   caspol -m -ag 1. -url "file://\\remotefileserver\content$\\*" FullTrust
7. Open %windir%\Microsoft.NET\Framework\{version}\aspnet.config and set the gcServer enabled to false instead of true.  
     
   <configuration>   
   <runtime>   
   <gcServer enabled="false"/>   
   </runtime>   
   </configuration>
8. Disable ASP template cache.

%windir%\system32\inetsrv\appcmd set config -section:asp -cache.maxDiskTemplateCacheFiles:0

1. Run appCmd to create a backup after all successful changes.  
     
   %windir%\system32\inetsrv\appcmd add backup "AfterWalkthroughBackup"
2. Create a backup schedule.

# Code Samples for Provisioning and Management

Once the servers are configured, the next step is creating Web sites and configuring them. This section provides samples for various provisioning steps.

**The provisioning may include:**

* Create a Web site
* Set multiple host headers (bindings)
* Create an application pool
* Set a custom identity on the application pool
* Assign a Web site to an application pool
* Set the failed request tracing log files path
* Set the W3SVC log files path
* Configure a delegated admin for the Web site

## Microsoft.Web.Administration

Managed code can be used provision Web sites and edit configuration.

**To create a site and an application pool**

using System;

using System.Collections.Generic;

using System.Text;

using System.Diagnostics;

using Microsoft.Web.Administration;

namespace IIS7Demos

{

class CreateSites

{

const int NUMBEROFSITES = 100;

const int SITEBASENUMBER = 1000;

const string POOLPREFIX = "POOL\_";

const string SITENAMEPREFIX = "SITE";

const string ROOTDIR = "e:\\content";

static void Main(string[] args)

{

ServerManager mgr = new ServerManager();

SiteCollection sites = mgr.Sites;

for (int i = SITEBASENUMBER; i < NUMBEROFSITES+SITEBASENUMBER; i++)

{

if (!CreateSitesInIIS(sites, SITENAMEPREFIX, i, ROOTDIR))

{

Console.WriteLine("Creating site {0} failed", i);

}

}

mgr.CommitChanges();

watch.Stop();

Console.WriteLine("Creating {0} sites took {1} seconds", NUMBEROFSITES, ((double)watch.ElapsedMilliseconds) / 1000f);

}

static bool CreateSitesInIIS(SiteCollection sites, string sitePrefix, int siteId, string dirRoot)

{

string siteName = sitePrefix + siteId;

// site gets set to Poolname using the following format. Example: 'Site\_POOL10'

string poolName = POOLPREFIX + sitePrefix + siteId;

try

{

Site site = sites.CreateElement();

site.Id = siteId;

site.SetAttributeValue("name", siteName);

sites.Add(site);

Application app = site.Applications.CreateElement();

app.SetAttributeValue("path", "/");

app.SetAttributeValue("applicationPool", poolName);

site.Applications.Add(app);

VirtualDirectory vdir = app.VirtualDirectories.CreateElement();

vdir.SetAttributeValue("path", "/");

vdir.SetAttributeValue("physicalPath", dirRoot + @"\" + siteName);

app.VirtualDirectories.Add(vdir);

Binding b = site.Bindings.CreateElement();

b.SetAttributeValue("protocol", "http");

b.SetAttributeValue("bindingInformation", ":80:" + siteName);

site.Bindings.Add(b);

}

catch (Exception ex)

{

Console.WriteLine("Create site {0} failed. Reason: {1}", siteName, ex.Message);

return false;

}

return true;

}

}

}

**To set a unique temporary compilation directory for each site**

using System;

using Microsoft.Web.Administration;

public class setASPNETCompilationDirectory

{

static void Main()

{

ServerManager manager = new ServerManager();

Configuration rootConfig = manager.GetWebConfiguration(new WebConfigurationMap(), null);

ConfigurationSection section = rootConfig.GetSection("system.web/compilation");

section.Attributes["tempDirectory"].Value = @"e:\inetpub\temp\temporary asp.net files\site1";

section.SetMetadata("lockAttributes", "tempDirectory");

manager.CommitChanges();

}

}

## IIS 7 WMI Provider

WMI scripts can be used to provision Web sites and run many commands to edit configuration.

**To create a site and application pool**

If WScript.Arguments.Count < 4 Then

                WScript.Echo "Not enough parameters. Enter: username | web site name | app pool password | site ID"

                WScript.Quit

End If

userName = WScript.Arguments(0)

siteName = WScript.Arguments(1)

appPoolPassword = WScript.Arguments(2)

siteID = WScript.Arguments(3)

appPoolName = "apppool\_" & siteName

physicalPath = "[\\server\share](file:///\\server\share)"  & userName & "\" & siteName

Set oIIS = GetObject("winmgmts:root\WebAdministration")

Set oBinding = oIIS.Get("BindingElement").SpawnInstance\_

oBinding.BindingInformation = "\*:80:" & siteName

oBinding.Protocol = "http"

Set oBinding2 = oIIS.Get("BindingElement").SpawnInstance\_

oBinding2.BindingInformation = "\*:80:www." & siteName

oBinding2.Protocol = "http"

arrBindings = array(oBinding, oBinding2)

Set oSiteDefn = oIIS.Get("Site")

oSiteDefn.Create siteName, arrBindings, physicalPath

WScript.Echo "Site created"

WScript.Sleep(100)

Set oSite = oIIS.Get("Site.Name='" & siteName & "'")

oSite.ID = siteID

oSite.Put\_

Set oSite = oIIS.Get("Site.Name='" & siteName & "'")

appPoolUserName = "poolname\_" & siteID

Set oAppDefn = oIIS.Get("ApplicationPool")

oAppDefn.Create appPoolName

WScript.Echo "App pool created"

WScript.Sleep(3000)

Set oAppDefn = oIIS.Get("ApplicationPool.Name='" & appPoolName & "'")

oAppDefn.ProcessModel.IdentityType = 3

oAppDefn.ProcessModel.Username = appPoolUserName

oAppDefn.ProcessModel.Password = appPoolPassword

oAppDefn.Put\_

WScript.Echo "Identity set for App Pool"

Set oSiteDefn = oIIS.Get("Site.Name='" & siteName & "'")

oSiteDefn.ApplicationDefaults.ApplicationPool = appPoolName

oSiteDefn.Put\_

WScript.Echo "Site assigned to pool"

Set oSite = oIIS.Get("Site.Name='" & siteName & "'")

oSite.Start

## AppCmd.exe

appCmd.exe can be used to provision Web sites and run many commands to edit configuration.

**To create a site and an application pool (with failed request tracing and W3svc logfile locations)**

%windir%\system32\inetsrv\Appcmd add AppPool -name:%poolname% -processModel.username:%poolaccount% -processModel.password:%poolaccountpwd% -enable32BitAppOnWin64:true

%windir%\system32\inetsrv\AppCmd add site -name:%sitename% -bindings:http/\*:80:%sitename% -physicalPath:%sitepath% –logfile.directory:%W3svclogpath% -traceFailedRequestsLogging.directory:%FREBlogpath%

%windir%\system32\inetsrv\Appcmd set app -app.name:%sitename%/ -applicationPool:%poolname%

**To configure the Failed Request Tracing logfile location**

%windir%\system32\inetsrv\AppCmd set config <sitename> -section:traceFailedRequestsLogging.directory:\\remotefileserver\content$\<sitename>\logs\failedRequestLogfiles

**To configure the W3SVC logfile location**

%windir%\system32\inetsrv\AppCmd set config <sitename> -section:–logfile.directory:\\remotefileshare\content$\<sitename>\logs\logfiles

## Other Managed Code Samples

**To set permissions on a folder using managed code**

using System;

using System.IO;

using System.DirectoryServices;

using System.Security.AccessControl;

using System.Security.Principal;

class Program

{

static void Main(string[] args)

{

String dir = @"e:\content";

DirectorySecurity dirsec = Directory.GetAccessControl(dir);

dirsec.SetAccessRuleProtection(true, false);

foreach (AuthorizationRule rule in dirsec.GetAccessRules(true, true, typeof(NTAccount)))

{  
 dirsec.RemoveAccessRuleAll

(

new FileSystemAccessRule

(

rule.IdentityReference,

FileSystemRights.FullControl,

AccessControlType.Allow

)

);

}

dirsec.AddAccessRule

(

new FileSystemAccessRule

(

@"BUILTIN\Administrators",

FileSystemRights.FullControl,

AccessControlType.Allow

)

);

dirsec.AddAccessRule

(

new FileSystemAccessRule

(

@"BUILTIN\Administrators",

FileSystemRights.FullControl,

InheritanceFlags.ObjectInherit,

PropagationFlags.InheritOnly,

AccessControlType.Allow

)

);

dirsec.AddAccessRule

(

new FileSystemAccessRule

(

@"BUILTIN\Administrators",

FileSystemRights.FullControl,

InheritanceFlags.ContainerInherit,

PropagationFlags.InheritOnly,

AccessControlType.Allow

)

);

Directory.SetAccessControl(dir, dirsec);

}

}

**To create a user account using managed code**

using System;

using System.DirectoryServices;

class Program

{

static void Main(string[] args)

{

DirectoryEntry AD =

new DirectoryEntry("WinNT://" + Environment.MachineName + ",computer");

DirectoryEntry NewUser =

AD.Children.Add("PoolID1", "user");

NewUser.Invoke("SetPassword", new object[] { "PoolIDPwd1" });

NewUser.Invoke("Put", new object[] { "Description", "AppPool Account" });

NewUser.CommitChanges();

}

}