Web Farm Configuration Guide

Joe Brinkman & Scott Willhite, Keivan Beigi

Version 2.0.0
Last Updated: March 26, 2010
Category: DotNetNuke v5.x

DOTNETNUKE™

Web Farm Configuration Guide

Information in this document, including URL and other Internet Web site references, is subject to change without notice. The entire risk of the use or the results of the use of this document remains with the user.

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, email address, logo, person, places, or events is intended or should be inferred.

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 DotNetNuke Corporation. DNN Corp. 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 DNN Corp., the furnishing of this document does not give you any license to these patents, trademarks, copyrights, or other intellectual property.

Copyright © 2009, DotNetNuke Corporation. All Rights Reserved.

DotNetNuke® and the DotNetNuke logo are either registered trademarks or trademarks of DotNetNuke 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.

DOTNETNUKE™

Web Farm Configuration Guide

Abstract

DotNetNuke is used for a broad range of websites; from single server installations to large enterprise class installs running in web-farms. Because of the issues that are unique to web-farms, the standard installation guidance is not sufficient. This document will outline the recommended and supported configuration for setting up your DotNetNuke installation in a web-farm environment.

Copyright © 2003-2010 DotNetNuke Corporation All Rights Reserved.

Contents

Chapter 1: Background 1

Chapter 2: Configure the App-Servers 3

Application Pool (UNC Share Only) 3

Permissions 4

Registry Updates (UNC Share Only) 7

Code Access Security (UNC Share Only) 8

Web Site Configuration 11

Chapter 3: Configure the File Server 13

Chapter 4: Database Configuration 15

Chapter 5: DotNetNuke Configuration 16

Caching Provider Overview 16

Web Server Configuration 17

DotNetNuke Scheduler 19

Additional Information 22

Errors and Omissions 22

Appendix A: Document History 23

Copyright © 2003-2010 DotNetNuke Corporation All Rights Reserved.

DOTNETNUKE™

Web Farm Configuration Guide

Copyright © 2003-2010 DotNetNuke Corporation All Rights Reserved.

Chapter 1: Background

The purpose of this document is to articulate the requirements for running DNN 5.1 and higher in a specific web-farm configuration. Conventions used in this document will assume some familiarity with IIS and Windows administrative functions. External sources will be noted throughout this document to provide addition information.

The configuration will utilize the following components:

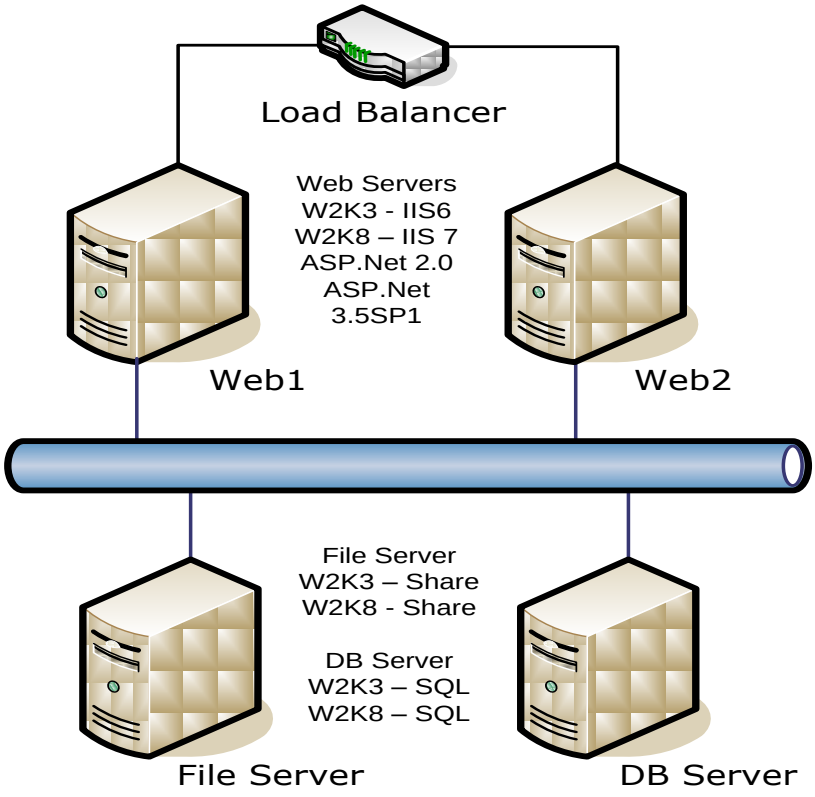


Figure 1 Standard Web-farm Configuration

Copyright © 2003-2010 DotNetNuke Corporation All Rights Reserved.

- All machines are members of an Active Directory Domain.
- Load Balancer managing traffic between 2 or more App-Servers.
- App-Servers running Windows 2003/2008/2008 R2 Server and ASP.NET 3.5 on IIS6/7.
- DB Server running SQL Server 2005/2008 (web-farm should have no impact on this configuration).
- File Server running Windows 2003/2008/2008 R2 Server to serve application files and content (Optional)

Assumptions:

This document assumes that an instance of DotNetNuke has already been configured and installed in a NON Load balanced configuration and is ready to be moved into the farm. The point of this assumption is to assure proper initial configuration of SQL and DotNetNuke and to avoid the possibility of corrupting the install process while the web-farm environment is being properly configured.

This document also assumes that your load balancer is correctly configured to direct traffic via an IP shared by both App-Servers. Further, that the individual loopback IP's of the App-Servers are also accessible (this is not a requirement, but is recommended to enable reaching the App-Servers individually).

Throughout this document we assume that the reader is familiar with the standard DotNetNuke configuration requirements. This document will focus on those settings and configuration items where the web-farm installation requires an alternative configuration.

Copyright © 2003-2010 DotNetNuke Corporation All Rights Reserved.

Chapter 2: Configure the App-Servers

Every web-farm consists of multiple App-Servers which are responsible for serving the content. App-Server configuration is the most complicated portion of setting up your DotNetNuke web-farm configuration so we will setup the App-Servers first.

Let's look at configuration of the Application Pool first. Bear in mind that the application pool will be configured identically on each App-Server in the farm with the exception of any scheduled app pool restarts. It is recommended that scheduled app pool restarts (if applicable) be staggered within the farm.

There are two available options to host the application files and content in a web-farm environment, hosting the file locally or in a UNC Share. There are pros and cons associated with each of these options so please use your discretion to make the choice that is more fitting to your requirements.

Application Pool (UNC Share Only)

The Identity of the Application Pool should be configured with a Network Username that can access each of the machines in your configuration. This name replaces the common usage of the "Network Service" user and will be granted a set of specific permissions in a variety of locations in the configuration. We will be using this domain user account throughout the remainder of this document as this account will need access to the UNC share, the application files and folders, and the database.

Configuring Application Pool Identity with IIS 6.0 (Windows Server 2003)

Configuring Application Pool Identity with IIS 7.0 (Windows Server 2008/R2)

Copyright © 2003-2010 DotNetNuke Corporation All Rights Reserved.

Permissions

We must give "domain\username" sufficient access to be used as an application pool security principal. There are a few places where this must be addressed. Proper access to the web root is required but we will defer this topic until we address the file server and remote file share where the web root will reside.

1. Required system folder access can be accomplished in two simple steps. First, add the "domain\username" user (or a group containing it) to the local `IIS_WPG` group in Windows Server 2003 or `IIS_IUSER` if you are using Windows Server 2008/2008R2

   IIS and Built-in Accounts (IIS 6.0)

   Understanding the Built-In User and Group Accounts in IIS 7.0 (IIS 7.0)

2. Execute the built in `aspnet_regiis` command with the `-ga` option by opening a command window to the .NET Framework 2.0 folder and running the command with the following syntax (see figure 3 below):

   ```
   aspnet_regiis -ga "domain\username"
   ```

   Expected output:

   ```
   Start granting domain\username access to the IIS metabase and other directories used by ASP.NET.
   Finished granting domain\username access to the IIS metabase and other directories used by ASP.NET.
   ```

   The key permissions to verify are that access is granted to the following system directories:

   - `%windir%\Microsoft.NET\Framework\v2.0.50727\Temporary ASP.NET Files (32Bit Systems)`
   - `%windir%\Microsoft.NET\Framework64\v2.0.50727\Temporary ASP.NET Files (64Bit Systems)`
   - `%windir%\Temp`

Copyright © 2003-2010 DotNetNuke Corporation All Rights Reserved.

This illustrates permissions applied to the ASP.NET Framework folder. The following KB article summarizes key permissions used by ASP.NET.

ASP.NET Required Access Control Lists (ACLs)

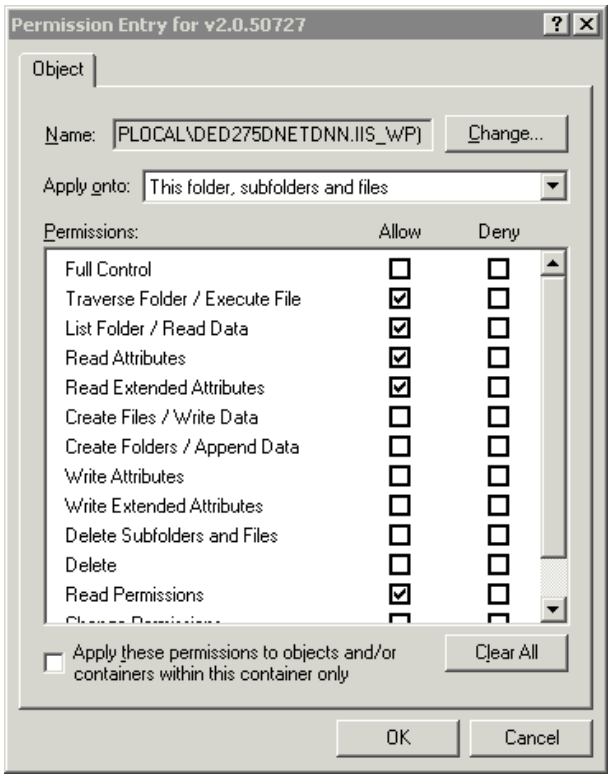


Figure 2 Framework folder permissions

Copyright © 2003-2010 DotNetNuke Corporation All Rights Reserved.

This illustrates permissions applied to the Windows Temp folder. Supporting information for can be found in the following Kb article.

Error message when ASP.NET 2.0 is configured to run with a user account: "Unable to generate a temporary class"

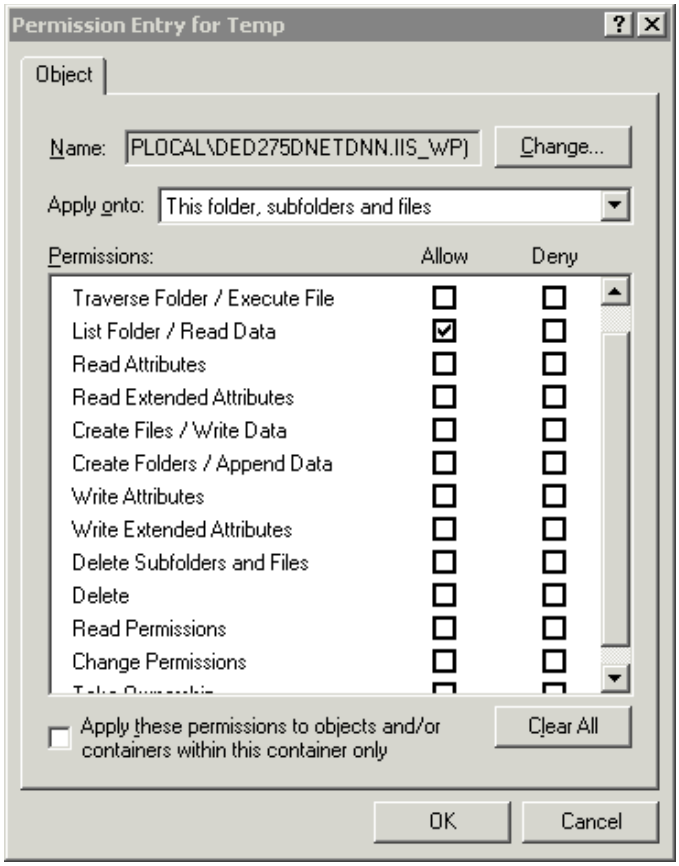


Figure 3 Framework folder permissions

Copyright © 2003-2010 DotNetNuke Corporation All Rights Reserved.

Registry Updates (UNC Share Only)

Out of the box, Windows Server is not setup to accommodate the file system monitoring traffic that is required when using a UNC share to host the application files and content. The following steps need to be performed on both the App-Servers and the File Server.

For more information, the following MS TechNet article provides excellent tuning advice including full documentation of the formulas used to calculate appropriate values for these registry settings based upon your usage scenario.

- Deploying and Configuring IIS 6.0 with Remotely Stored Content on UNC Servers and NAS Devices
- Tuning the Servers for UNC Content (IIS 6.0)

On the App-Server, you'll need to adjust (and may need to actually create) the value associated with the node:

```
HKLM\System\CurrentControlSet\Services\LanmanWorkstation\Parameters\MaxCmds
```

If you have to create the node, be sure to specify it as a DWORD value, and assign it in Decimal base.

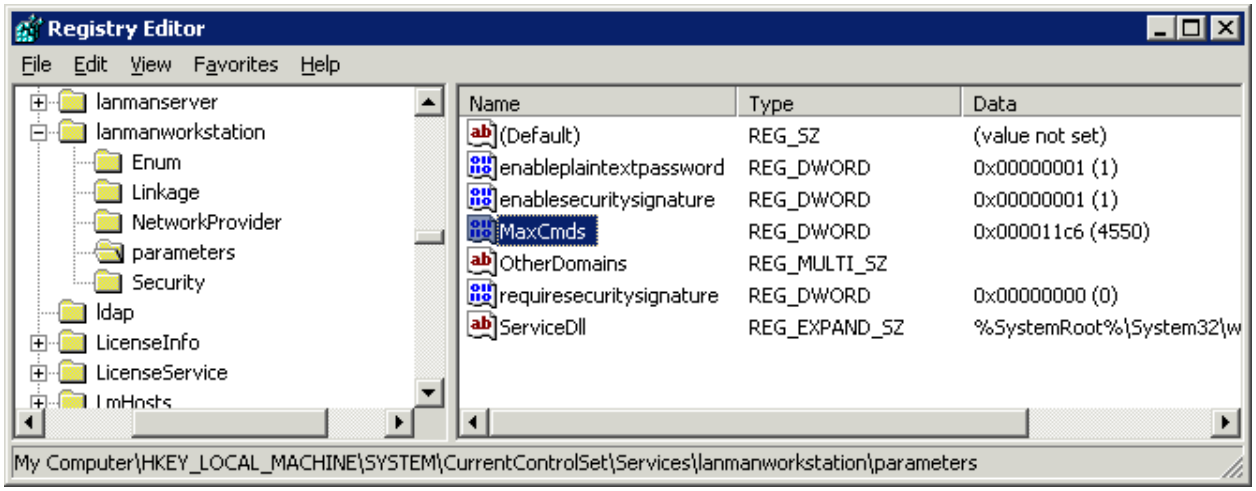


Figure 4 Making registry changes for Network accessed storage

Copyright © 2003-2010 DotNetNuke Corporation All Rights Reserved.

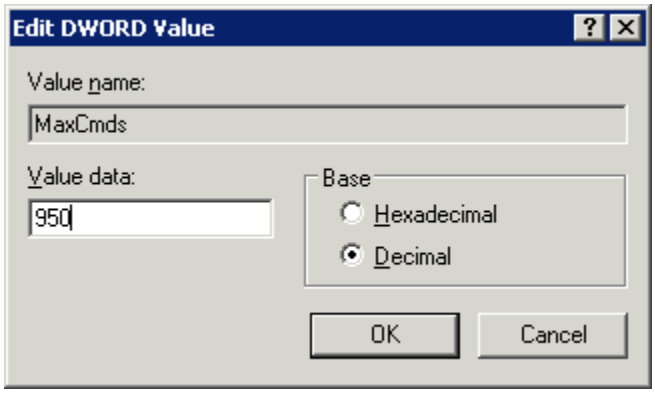


Figure 5 Edit the MaxCmds value

The appropriate value for `MaxCmds` depends upon your usage scenario. In a privately hosted environment where server density is not the first consideration, a default which anticipates both Wide Content (300 applications) and High Traffic may be desirable. In that configuration:

MaxCmds = 300 * (1(static) + 1(ASP) + 1(ASP.NET)) + 50 = 950

Even increasing the `MaxCmds` can still result in overloading ASP.NET with folder monitoring in a site with a large number of directories. By default, ASP.NET sets up a file change notification for each directory in the application. You can force ASP.NET to setup a single file change notification object on just the web application root folder by adding the following registry key to the IIS server (you will need to create the key if it doesn't exist):

```
HKLM\Software\Microsoft\ASP.NET\FCNMode
```

`FCNMode` should be set to a DWORD with the value of 2.

Code Access Security (UNC Share Only)

http://support.microsoft.com/kb/320268

ASP.NET, by default, does not "trust" content hosted remotely. So depending upon where you are in setup, you may be able to "see" the content on your remote share, but ASP.NET (at runtime) will terminate execution because it content location is not "trustworthy". To rectify this condition, we need to set appropriate security policy at the machine level.

This can be accomplished using either the Code Access Security Policy Tool (`caspol.exe`) or the .NET Framework 2.0 Configuration Tool. The appropriate command using `caspol.exe` would be:

```
caspol.exe -m -ag 1 -url "\\<ServerName>\<FolderName>\*" FullTrust -exclusive on
```

Copyright © 2003-2010 DotNetNuke Corporation All Rights Reserved.

If you are running in a 64bit version of Windows then you need to make sure you are running the 64bit version of `caspol`

64Bit: `%windir%\Microsoft.NET\Framework64`

32Bit: `%windir%\Microsoft.NET\Framework`

We will illustrate an alternative setup using the .NET Framework 2.0 Configuration Tool.

Note: .NET Framework 2.0 Configuration Tool is not installed by default with the .NET Framework and may require you to download and install the *Microsoft .NET Framework SDK*.

First, start the Microsoft .NET Framework 2.0 Configuration Tool from the Administrative Tools section of the Start Menu and navigate to the following node:

My Computer>Runtime Security Policy>Machine>Code Groups>All_Code

For the sake of clarity, I prefer to create a new Code Group called "`UNC_Zone`" where we will add a custom code group for each remote share the server will address.

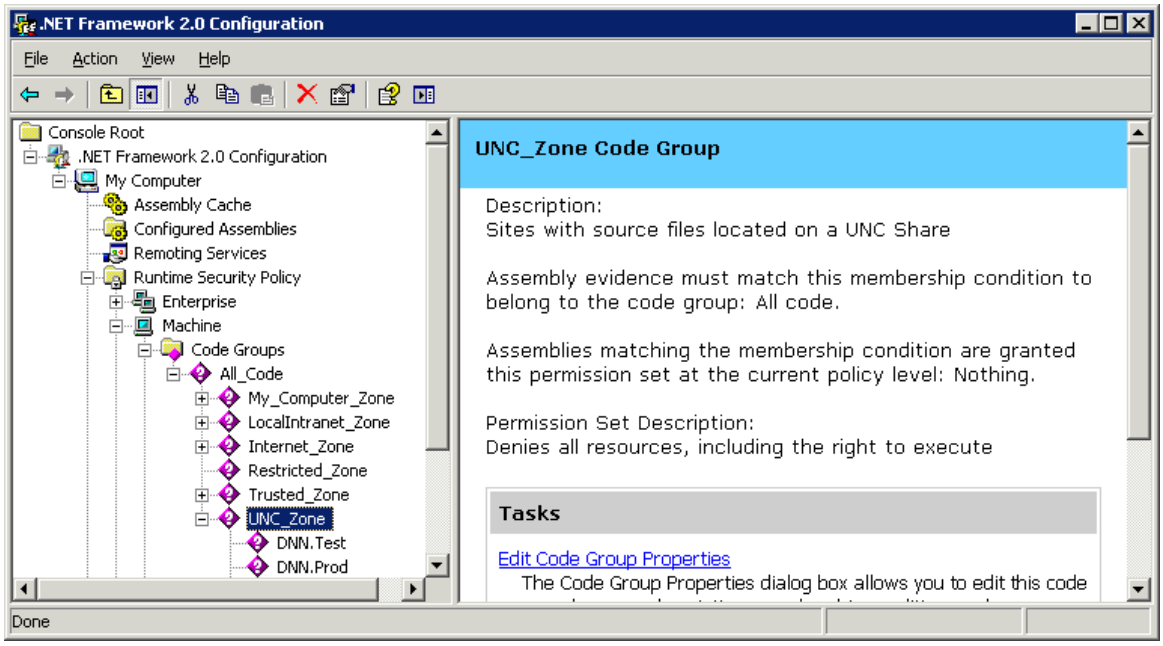


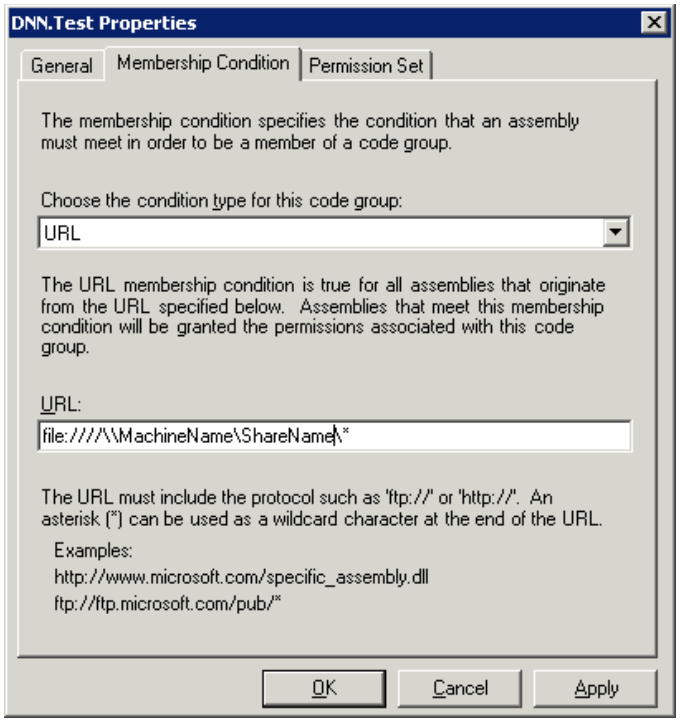
Figure 6 configure a new zone

Copyright © 2003-2010 DotNetNuke Corporation All Rights Reserved.

Since the "UNC_Zone" is merely a container, it will have a Membership Condition of "All Code" and a Permission Set of "Nothing".

Now add another Code Group within the UNC_Zone for each of your remote shares by clicking "Add A Child Code Group" in the main window. Give it a name (e.g. "DNN.Test") and description on the General tab.

Once your Runtime Security Policies have been established (using either this or the caspol method), you will need to restart your IIS services for them to take effect.

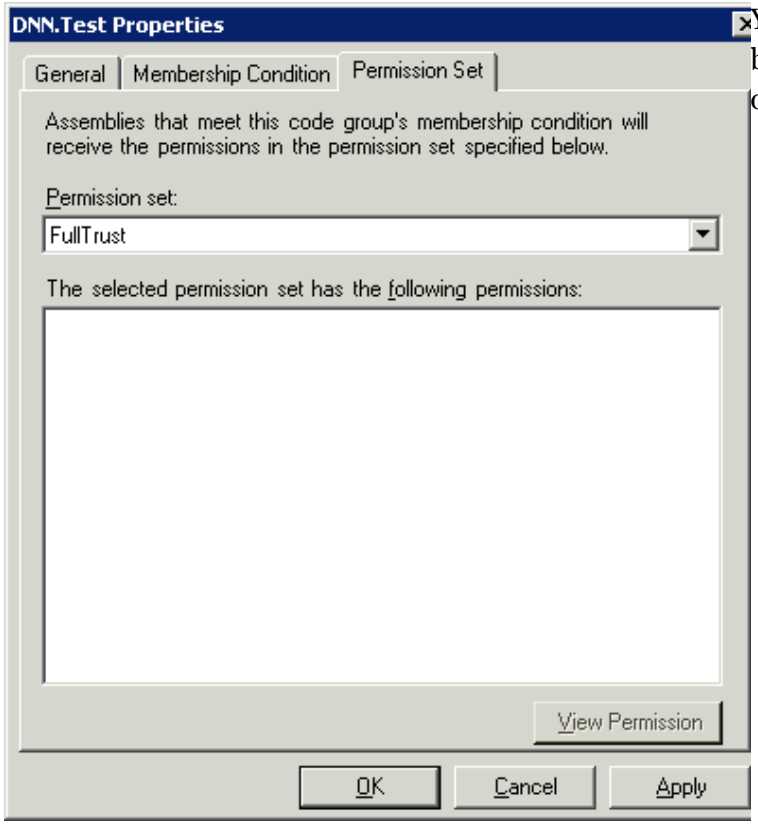


The Membership Condition should be specified as URL and the URL carefully formatted for this convention:

File:////\\{MachineName}\{ShareName}*

Figure 7 Configure the CAS code group Membership Condition

Copyright © 2003-2010 DotNetNuke Corporation All Rights Reserved.



Your circumstances may vary, but Full Trust is a valid choice of Permission Set.

Figure 8 Configure the code group permission set

Web Site Configuration

Finally, we must configure the website itself to reference the UNC Share. Please note that if the username you use to administer IIS on the webserver does not have permission to access the share, you will not see the files in the web root (although the IIS application will).

- Setting Home Directories (IIS 6.0)
- Change the Physical Path of Application Content (IIS 7.0)

After configuring the Network folder we need to ensure that site visitors will have access to the needed files.

Copyright © 2003-2010 DotNetNuke Corporation All Rights Reserved.

Enter the user name and password for the same account you configured for the App Pool.

Alternatively, you can create a local user with the same user name and password as a user account on the file server which will be used for file/share permissions.

- Changing the Account Used for Anonymous Authentication (IIS 6.0)
- Configure the Anonymous Authentication Identity (IIS 7.0)

At this point your App-Server is now completely configured. You will need to perform the same steps for each server in the web-farm.

Copyright © 2003-2010 DotNetNuke Corporation All Rights Reserved.

Chapter 3: Configure the File Server

With the App-Servers configured you are now ready to turn your attention to configuring the network storage (UNC Share). The file server requires two sets of permissions to be configured:

1. Network Share Permissions (SMB Permission)
2. File/Folder permissions (NTFS Permission)

Both sets of permissions will be applied to the account used by the Application Pool on App-Servers.

- How To Share Files and Folders over the Network in Windows Server 2003
- Configuring Windows Server 2008 File Sharing

Network Share Permissions (SMB Permission)

Full access permissions should be granted to the user used to configure the App-Servers.

Folder/File Permissions (NTFS Permission)

The next step in preparation for installing DotNetNuke is to set the NTFS file permissions for the ASP.NET worker process. This step is required as DotNetNuke needs to be able to create directories and files.

Copyright © 2003-2010 DotNetNuke Corporation All Rights Reserved.

To set the correct File permissions use File Explorer to locate the physical folder where the DotNetNuke application files exist. Right-click on the folder, and select Properties, and choose the Security Tab. Select the user account (or add it if necessary) and assign the necessary permissions.

Many sites will grant read, write and modify permissions from the Root folder down on all files and subfolders, but it's possible to restrict the set of permissions to enhance security.

The minimum required permissions for the various folders (and their included files) in DotNetNuke are as follows:

Folder	Required Permissions
Root (and all child folders)	Read Access
Root	Write Access (and Modify Access if expecting to create child portals)
App_Code	Write, Modify Access (if installing Modules)
Bin	Write, Modify Access (if installing Modules)
DesktopModules	Write, Modify Access (if installing Modules)
Install(and all child folders)	Write, Modify Access (to allow modules to be unpacked for installation)
Portals (and all child folders)	Write, Modify Access (to allow File Manager access and Skin/Container access)

Note: If you are likely to be installing Language Packs you will also need Write and Modify Access to all child folders of the root.

Copyright © 2003-2010 DotNetNuke Corporation All Rights Reserved.

Chapter 4: Database Configuration

The Database Server does not require any special configuration for a web-farm. As in all DotNetNuke configurations, you will need to provide permissions for the account running the App Pool to access the DotNetNuke database (Integrated Security). Follow the instructions in the DotNetNuke Installation Guide for complete database configuration instructions. The Installation Guide is included as part of the Documentation package available at http://downloads.dotnetnuke.com/

Note: If the configured user account does not have `dbo` set as the default schema, then you will need to configure the user permissions appropriately. You can find more information on the various database configuration options at

The Many Ways to Configure SQL Database Security for DNN

Copyright © 2003-2010 DotNetNuke Corporation All Rights Reserved.

Chapter 5: DotNetNuke Configuration

By now you should have more than one DotNetNuke App-server setup and pointed to the same database and possibly using the same physical path under IIS (UNC Share). The only functionality missing from this setup to be a complete Web-Farm is cache synchronization.

Caching Provider Overview

In the original web-farm implementation of DotNetNuke, the BroadcastPollingCachingProvider was configured as the default caching provider which was later replaced with FileBasedCachingProvider. These providers used Database and File System for keeping each of the App-Server caches in sync. While these mechanisms worked, they were highly inefficient (FileBasedCachingProvider was more efficient than BroadcastPolingCachingProvider) and resulted in a large increase in database and or file thrashing. This offset a significant amount of the performance gain that was obtained by adding multiple web servers to the farm.

Currently the recommended caching provider for web-farm configuration is WebRequestCachingProvider that uses web requests to keep the cached items synchronized across app-servers. This provider delivers higher levels efficiency since it doesn't rely on Database or the file system.

Note: The BroadcastPollingCachingProvider was officially deprecated on June 23, 2009, and is no longer supported.

Copyright © 2003-2010 DotNetNuke Corporation All Rights Reserved.

Web Server Configuration

To Configure App-Servers that will participate in the web-farm login as Host and to the Host > Professional Features > Manage Web Servers page.

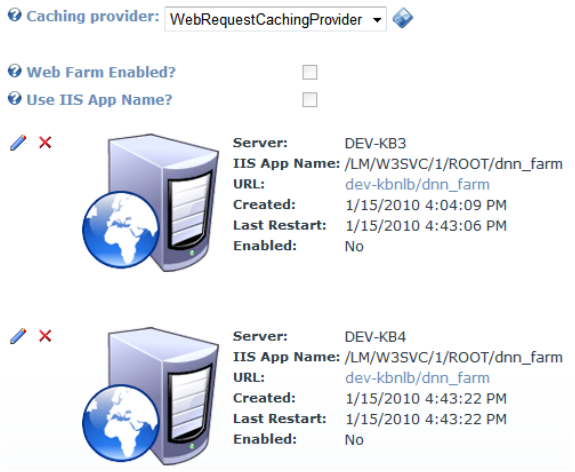


Figure 9 Web Servers

Make sure the Caching Provider is set to WebRequestCachingProvider. You will also notice that none of the web servers are enabled by default. Click the edit icon and check the enabled check box and save the changes. That will cause the servers to join the web farm.

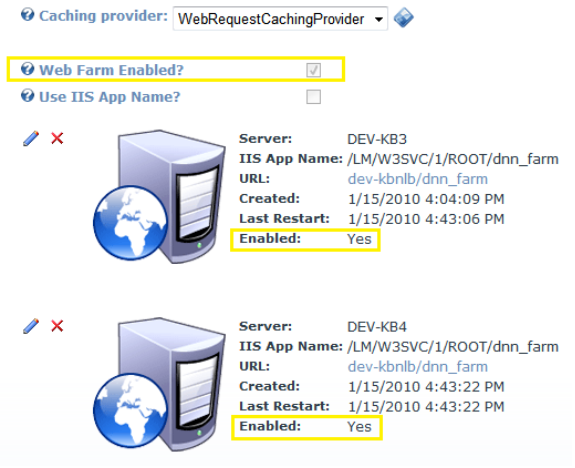


Figure 10 Web Servers (after joining web-farm)

Copyright © 2003-2010 DotNetNuke Corporation All Rights Reserved.

WebRequestCachingProvider will work with the most common webfarm setup, which utilizes a load balancer that sends requests to machines via a unique address e.g. requests come to www.site.com but get sent onto unique machine URL's such as 10.0.0.1, 10.0.0.2 etc. WebRequestCachingProvider is then able to use these unique addresses to maintain cache integrity. However some sites choose to use virtualized addresses - either the fully qualified domain name or a fixed IP, and the WebRequestCachingProvider can't do its job as it has no unique addresses. To resolve this issues there are a few additional steps to perform.

The first step is to assign a unique URL to each of the servers that are participating in the webfarm. As you can see in the previous figure both servers share the same URL which could cause problems with cache sync.

To resolve this we need to assign the unique URLs of each machine to the list of portal aliases.

Browse to the site settings page (i.e admin->site settings), and expand the "other settings" node and then the portal alias's node. You'll need to add unique portal alias's for both machines - typically the machines internal IP or internal name are the best options. In this example we use the internal machine IP.

Figure 11 Portal Alias with unique URLs added

Now we have unique URL's for each machine, we return to the Manage Web Servers page. Now we need to edit the servers to use the unique URL's, so click the pencil icon beside the first server and then select the relevant URL from the dropdown.

Copyright © 2003-2010 DotNetNuke Corporation All Rights Reserved.

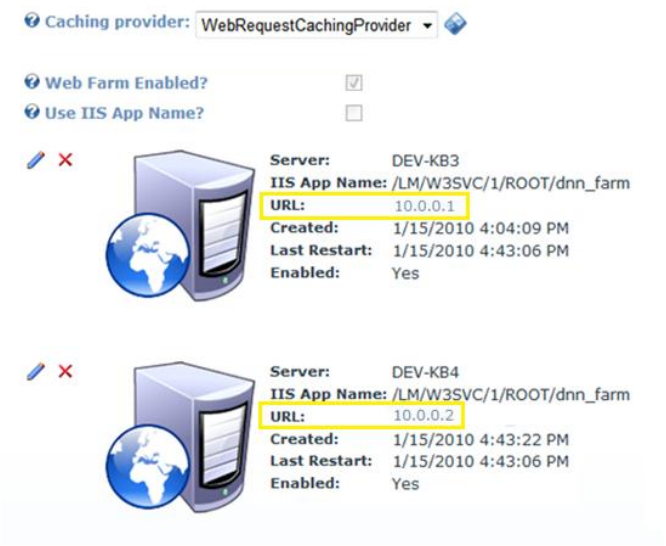


Figure 12 Final Result of Webfarm Configuration

Now you should have a fully functional Webfarm setup with synchronized caching.

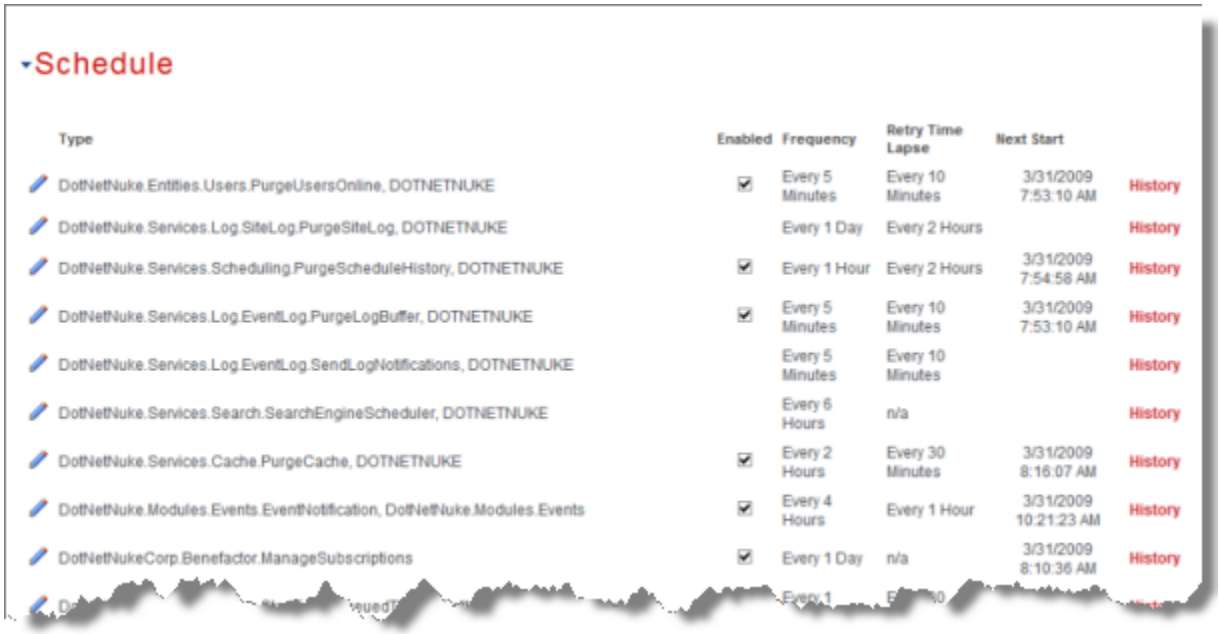
Note: configuration item [`EnableWebFarmSupport`] in web.config file is deprecated and is no longer used.

DotNetNuke Scheduler

The DotNetNuke Scheduler is able to run background tasks within your DotNetNuke installation. Scheduled tasks are recurring events that typically perform housecleaning tasks in the database or file system. Almost all scheduled tasks that are delivered with DotNetNuke are designed to run on a single App-Server.

To configure scheduled tasks, login as a Host user and go to the Host/Schedule page.

Copyright © 2003-2010 DotNetNuke Corporation All Rights Reserved.



Schedule

Type	Enabled	Frequency	Retry Time Lapse	Next Start	
DotNetNuke.Entities.Users.PurgeUsersOnline, DOTNETNUKE	☑	Every 5 Minutes	Every 10 Minutes	3/31/2009 7:53:10 AM	History
DotNetNuke.Services.Log.SiteLog.PurgeSiteLog, DOTNETNUKE		Every 1 Day	Every 2 Hours		History
DotNetNuke.Services.Scheduling.PurgeScheduleHistory, DOTNETNUKE	☑	Every 1 Hour	Every 2 Hours	3/31/2009 7:54:58 AM	History
DotNetNuke.Services.Log.EventLog.PurgeLogBuffer, DOTNETNUKE	☑	Every 5 Minutes	Every 10 Minutes	3/31/2009 7:53:10 AM	History
DotNetNuke.Services.Log.EventLog.SendLogNotifications, DOTNETNUKE		Every 5 Minutes	Every 10 Minutes		History
DotNetNuke.Services.Search.SearchEngineScheduler, DOTNETNUKE		Every 6 Hours	n/a		History
DotNetNuke.Services.Cache.PurgeCache, DOTNETNUKE	☑	Every 2 Hours	Every 30 Minutes	3/31/2009 8:16:07 AM	History
DotNetNuke.Modules.Events.EventNotification, DotNetNuke.Modules.Events	☑	Every 4 Hours	Every 1 Hour	3/31/2009 10:21:23 AM	History
DotNetNukeCorp.Benefactor.ManageSubscriptions	☑	Every 1 Day	n/a	3/31/2009 8:10:36 AM	History

Figure 13 View of the Scheduled tasks

Click on the edit button next to each scheduled task to change the settings. Edit the "Run on Servers" setting with the machine name of the server or servers that should run this task. Use a comma delimited list if you are running the task on multiple servers.

Copyright © 2003-2010 DotNetNuke Corporation All Rights Reserved.

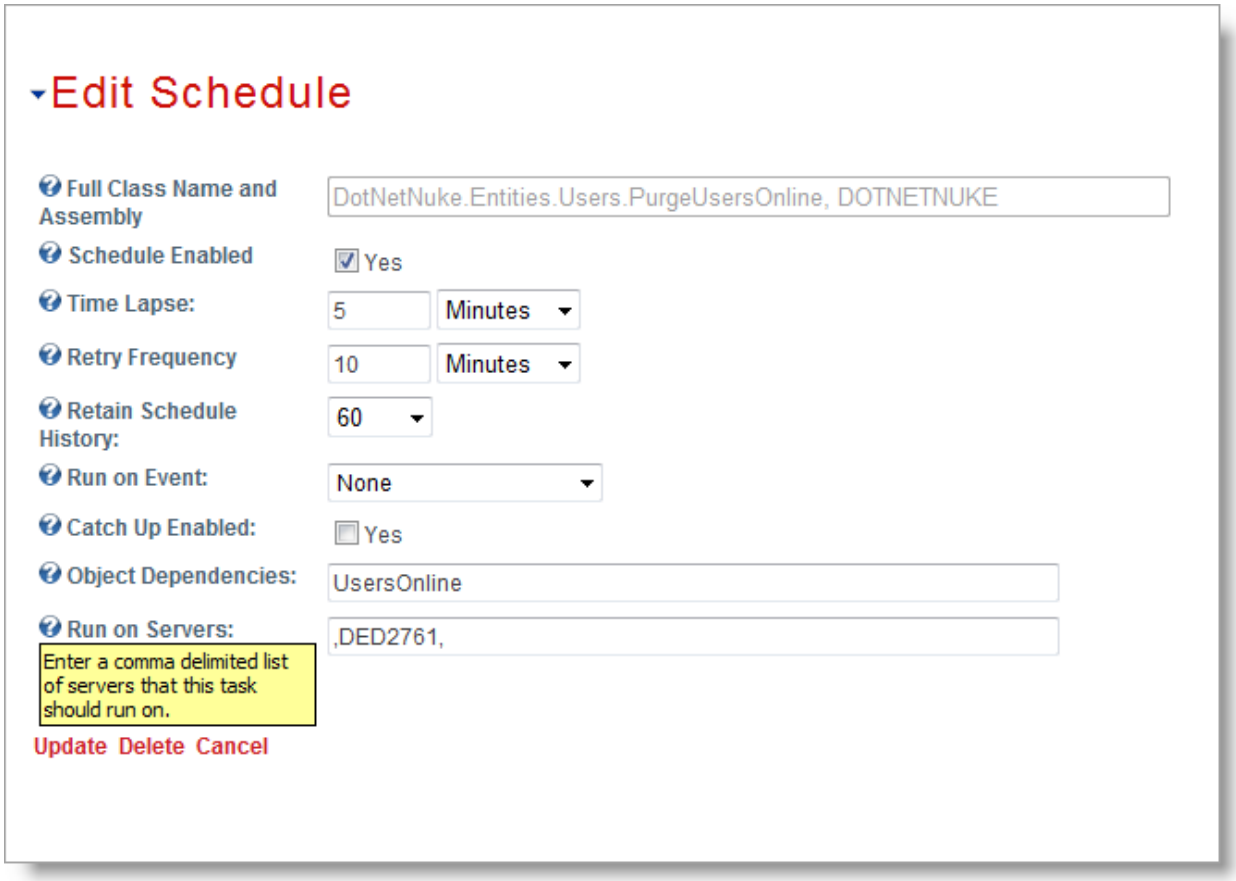


Figure 14 editing a scheduled task

At this point your DotNetNuke installation is fully configured and ready for operation in the web-farm

Copyright © 2003-2010 DotNetNuke Corporation All Rights Reserved.

Additional Information

The following additional websites provide helpful information about technologies and concepts related to DotNetNuke:

DotNetNuke Community Forums
http://forums.dotnetnuke.com

Microsoft® ASP.NET
http://www.ASP.NET

Errors and Omissions

If you discover any errors or omissions in this document, please email marketing@dotnetnuke.com. Please provide the title of the document, the page number of the error and the corrected content along with any additional information that will help us in correcting the error.

Copyright © 2003-2010 DotNetNuke Corporation All Rights Reserved.

Appendix A: Document History

Version	Last Update	Author(s)	Changes
1.0	Nov. 9, 2008	Scott Willhite	App-Server Configuration
1.1	Mar. 31, 2009	Joe Brinkman	File Server Configuration DB Configuration DotNetNuke Configuration
1.2	Jan. 15, 2010	Keivan Beigi	Added Support IIS 7.0/Windows Server 2008 Replaced FileBasedCachingProvider with WebRequestCachingProvider Fixed naming inconsistencies throughout the document

Copyright © 2003-2010 DotNetNuke Corporation All Rights Reserved.