DNN/Evoq Performance Configuration

Administrators Guide

Last Updated: July 16, 2015

IowaComputer GURUS

Contents

Overview

Document Revision History

Revision Date	Module Version	Minimum DNN Version	Notes
7/25/2009	1.00		First Document created
3/28/2010	1.01		Updated to include changes in DotNetNuke 5.x and DotNetNuke Professional Edition
7/02/2010	1.02		Updated with most recent DNN information and updated shared hosting information
7/12/2011	1.03		Updated with Dnn 6.x information
4/9/2012	2.00		Updated with detailed DNN 6.x information and updated information on hosting and other environment configuration recommendations
5/30/2012	6.00		Updated for DNN 6.2.x as well as incremented version number to align better to DNN versions
7/3/2012	6.20		Updated for additional information around Client Resource Management
9/19/2014	7.0		Updated with re-branding to DNN, inclusion of Evoq in the cloud and other offerings as well as general updates with new recommendations based on current best practices

Disclaimer

Information contained in this document has been compiled and is general recommendation information provided by IowaComputerGurus Inc. This information is provided "as-is" and we do not offer any guarantee or warranty on the information provided within. The reader is responsible for performing due-diligence verification that the recommendations in this document fit their needs as well as the specific DotNetNuke environment.

In addition, it is strongly recommended that a backup of all site files and database be completed and VERIFIED before testing any of the configuration recommendations outlined in this document. Although configuration changes only in this document if applied incorrectly, or if a third-party module interferes, it could cause a need to restore from a backup.

Copyright and Trademark Notices

DNN Performance Overview

The performance of a DNN based website is typically one of the biggest concerns to any site administrator. Sadly, there have been many sites and some history that have made statements such as "DNN is slow" and "getting performance out of DNN is hard" be the most common attitude when it comes to performance tuning and DotNetNuke.

However, that is not the case. You simply need to have a basic understanding of a few key items and how those can impact your performance. The remaining points in this section will cover the highlights of impact areas, with the remainder of the document outlining how you can work within those elements to properly configure sites for performance.

> ***NOTE:*** *It is recommended that you NEVER run DNN with the default configuration as it is not in any way configured for the best performance regardless of your hosting environment or utility setup.*

IIS/ASP.NET/Server ("Hosting Environment")

IIS and the ASP.NET runtime are the two most commonly overlooked areas that can impact the performance of a DotNetNuke site. IIS and ASP.NET control the "box" that DotNetNuke runs inside, being an ASP.NET application. It is controlled by ASP.NET and IIS. These two systems depending on configuration as well as hosting hardware can have varying impacts on the performance of a DotNetNuke® site. See Hosting Environment Configuration/Selection.

When working in a high performance environment, there are a number of key areas of ASP.NET configuration that should be optimized. An example may be a web-server with greater than 4 cores, or more than 4-8 GB of ram or working inside of a Web Farm (multi-server) environment.

The final section of this document is dedicated to the high-performance scenarios.

DotNetNuke Host Settings/Features

Once you have validated the configuration of the physical/virtual hardware that is used to host the DotNetNuke, site is time to start looking into the actual configuration of DotNetNuke Features and Functionality to ensure that it is optimized for your environment as well as for the best performance. At this time, the default configuration is typically not configured properly for most environments simply because it is really hard to get a baseline configuration that is optimized for each item.

DotNetNuke Scheduler

The final configuration item that is typically used to help ensure proper performance is to review/modify the scheduled tasks that are installed/used by DotNetNuke. These scheduled tasks can have varying impacts on performance based on site structures.

Skins, Modules, External Services, JavaScript and the Others

Simply setting the base configuration is only a piece of the larger puzzle. We discuss these base items in grave detail as the implementation and details are the same regardless of your site structure. However, it is important to take into consideration all of the other elements that can impact your site's performance such as the modules used, the way the skin is designed, external integrations and more.

Hosting Environment Configuration/Selection

It is best to start your performance configuration by looking at your hosting environment since this is the foundation of any DNN® installation. Even minor changes here can have dramatic effects on the overall site performance. In some cases, the selection of the vendor or configuration of your hosting environment can limit the performance that you are able to attain from your system. The following sub-sections will discuss hosting items in two areas. First, some general information around benchmarking and recommended environments, and then specific considerations based on the type of hosting.

> ***NOTE**: The below recommendations are ONLY around performance considerations with DotNetNuke and do not necessary represent all considerations needed when selecting a hosting provider.*

Basic Hosting Requirements

The minimum hosting requirements for DotNetNuke are easy to obtain, however, to get a truly unique setup there are a number of items to consider when it comes to hosting basics and running your DotNetNuke installation.

ASP.NET Version

Starting with DNN 7.0.0 and later, ASP.NET 4.0 is a minimum requirement. At this point in time, any installation of DNN will need to work with 4.0 and later, ASP.NET. 4.5 is strongly recommended as well.

IIS Version

This information is not necessarily the easiest to find out when working with certain hosting environments. Supporting the best overall flexibility with DotNetNuke with third-party vendors such as UrlRewriting can have performance impacts on the systems. It is recommended that individuals use IIS 7 or later to ensure that wildcard mappings and related functions are easily supported. (This means you are looking to be hosted on Windows Server 2008 or newer. Do not setup any new installations of DNN on Windows Server 2003)

SQL Server Version

DotNetNuke supports SQL Server 2005 and later. The specific edition does not have a direct impact on the performance of the system until your site becomes a specific size. It is important to note that if running DotNetNuke with a SQL Server Express edition background, that you are limited to 1 CPU core and 1GB of RAM for the SQL Server engine. This WILL become a limitation in databases that are heavy or overly large DotNetNuke Based sites.

It is recommended when working on dedicated/virtual environments that Web edition or greater is used to ensure that SQL Server itself will not become the performance bottleneck for your application.

NOTE: SQL Server Express is NOT recommended for ANY production usage due to its lack of ability to handle scheduled jobs such as database backups. If using SQL Express for a production server, please ensure that your environment is backed up on a regular basis and validate the success of these backups at regular intervals.

General DotNetNuke Application Size Memory & Disk

A standard DotNetNuke implementation will consume between 100-200MB of RAM within its application pool at system startup. When calculating system capacity, or reviewing limitations at hosting providers, this should be remembered as a site gets larger in terms of pages/content. This memory consumption can rise.

Original disk space for DNN is around 200MB and will grow as your site has more items added.

Shared Hosting Considerations

The shared hosting environment is one of the hardest to work with when it comes to application configuration. You will NOT be granted any configuration abilities over the environment in 99.9% of cases. However, we do have the following recommendations when working with a shared hosting environment.

IowaComputerGurus recommends any business to strongly consider the risks associated with shared hosting for their organization as it relates to impact from other users and sites. Costs are cheaper but with current cloud offerings for a reasonable cost increase, reliability and performance can be enhanced greatly for minimal cost.

Selecting a Good Hosting Provider

When it comes to shared hosting environments, not all hosting providers are the same. When looking for a DNN host to ensure that you have the best experience both in functionality and performance, there are a few key features that you want to be sure that you have.

- ASP.NET Process with Root Permissions – DNN is a powerful framework and installing modules, skins, and other extensions often requires that DNN be able to make changes to the file system. Some hosting providers do not allow the application to have full permissions at the root, without this you will start to experience issues.
- Full-Trust Support – Running DotNetNuke in full trust is going to get the best flexibility possible.
- Installation to site root – One of the most common limitations are sites that do not allow installations to the root, installing to the root is important for both SEO and maintenance/performance reasons.

- Known Support – It is important to work with a hosting provider that is known for good support/systems running DotNetNuke.
- ASP.NET Memory Limitations - Many shared hosting providers are starting to impose hard set limitations on ASP.NET. Application memory usage, depending on the action taken when the limit is hit, and the limit that is set for your site can be adversely affected in these situations. Obviously, you will want a hosting provider that has a very high limit, or a provider that takes a less-lethal action when the limit has been exceeded. See the section ASP.NET Memory Limitations for more detailed information.

Due to the above listed items, IowaComputerGurus has come across three hosting providers that we recommend for shared hosting. 3Essentials, PowerDNN, and Applied Innovations. The provider you choose depends on your needs. One or more of these providers should be able to provide the best product and service offering. Note: specific plans and recommendations might change at any time. (Referral links available at www.iowacomputergurus.com, referrals appreciated).

What is a Good Limit?

A default configuration for your average DotNetNuke installation will have a running footprint of anywhere between 100 and 200 megabytes of memory. In the case of IowaComputerGurus Inc. websites, our largest site with the most traffic typically uses between 150 and 160 megabytes of memory. Overall, a typical DotNetNuke site that is setup to run in a shared hosting environment should use a similar amount of ram. Keep in mind that the number and quality of installed third-party modules as well as the configuration of the site will impact the exact memory footprint of an application.

Therefore, it is our recommendation to ensure that a hosting provider will allow for usage of 180-200 megabytes of memory for any application. This will help to ensure that your site has room to grow before it has to get away from a shared hosting environment. Ideally, selecting a provider without memory limitations is best.

What Actions Can They Take?

The most common question that we receive regarding memory limitations is; what do they do to us if we exceed the limit? This is a question that is going to vary vendor to vendor. There are two typical types of action, application recycle or application lockout.

In the case of an application recycle, the hosting provider will simply stop your application and allow it to restart. This frees the memory that was used by your site and allows it to start fresh. In this scenario, your application will continue to serve content to all visitors, however, the users that are on the site when it is recycled will notice a delay when the next page loads due to the startup time for ASP.NET and pre-compilation. (Detailed discussion of this behavior in the Sequel Server Version). With this type of limitation, if a site gets out of control, the content is still there.

In the case of an application lockout, the application will be configured to show a "Service Unavailable" message for a pre-defined amount of time if the memory limit is exceeded. During this time your users will see an error message and NO content will be available. This is NOT a situation or action that you want a hosting provider to take. It is the safest from their perspective in that it helps them manage resources. As a website owner it causes you lost traffic.

How to Monitor

The final point when looking at ASP.NET memory limitations is that many hosting providers do NOT provide you default access to monitor the memory usage of your application. The limitation discussed when talking about the ASP.NET worker process is NOT that of your disk space but the actual running system memory used by the ASP.NET process. Most of the standard control panels, including Plesk, do NOT provide a method to analyze the usage patterns of application memory use. You will want to check with your hosting provider to see if any reporting/monitoring tool is available.

ASP.NET Keep Alive

When working in a shared hosting environment, you will not have control over the configuration of IIS. This can be a limiting factor from a performance perspective as IIS by default has an "Idle Timeout" that applies to ASP.NET applications. This timeout is by default set to 20 minutes, this means after 20 minutes of inactivity your site will unload from memory. The next page request will re-load the application; however, the user requesting the page can see a sometimes lengthy delay in application startup.

Recommendations to get around one of two items are talking to the hosting provider to have them extend the default timeout value or to employ a web based keepalive or monitoring service. IowaComputerGurus uses BinaryCanary.com to perform keepalive and monitoring activities.

Virtual Private Server Hosting (VPS) & Cloud Servers

A VPS hosting environment is a configuration where a user is given their own Windows Server installation that is on a virtualized hardware environment. Typically with higher quality VPS hosting plans you will see that 4-8 VPS accounts will exist on one single physical server.

In general, IowaComputerGurus has not encountered any issues with VPS hosting as long as you properly validate the plans being purchased and the load being placed on the server. The following are recommendations when working with VPS accounts.

Amount of RAM and Type of RAM

When working with a VPS, the best consideration should be the amount of RAM, both guaranteed and burst, as well as how that RAM is actually given. An example of this scenario may find one system that provides 1 GB of physical RAM and another system that provides virtual RAM. You will notice massive performance improvements using a system that provides physical RAM.

The amount of RAM is also important. IowaComputerGurus does NOT recommend using any configuration with less than 1GB of RAM for any DotNetNuke installation. If you are unsure of the specific RAM allotments and configurations, contact the hosting provider to validate their configurations. (NOTE: If SQL will be hosted on the same box we recommend a RAM minimum of 2GB or RAM.)

SQL Server Configuration

Many times when individuals opt for a VPS hosting plan it will start out with SQL Server being installed on the same system as the IIS server. It is very important that you properly configure SQL Server. By default, SQL Server will NOT limit the total memory that it can use, and therefore over time, it will keep taking more and more memory, eventually fighting IIS and other applications for available memory. Simply configuring a reasonable memory limitation on SQL Server will prevent a potential disastrous out of memory situation.

ASP.NET Keep Alive or IIS Configuration

As with shared hosting the default behavior of ASP.NET with a 20 minute idle timeout is still something to be concerned about. However, once you have reached the VPS level of hosting you are typically granted remote desktop access to the server which will allow you to simply modify the timeout value. In certain circumstances such as providers that use specific control panels such as PLESK this might not be an option. If you are not able to configure this value, it will be necessary to pursue a keepalive service outlined in the section APS.NET Keep Alive.

See the section Recommended IIS Application Pool Configurations for full Application Pool configuration recommendations.

Dedicated Hosting

Although a dedicated hosting environment may be the most costly of the major types of system, this type provides a DotNetNuke site administrator the best control and ability to tune the system for optimal system performance. The hard part here is in selecting the proper hardware and configuration. Many of the considerations for performance here are similar to that of the VPS environment.

Amount of RAM

When working in a dedicated hosting environment, it is possible that you will have IIS and SQL Server configured on the same machine. A minimum of 4GB of RAM would be recommended. If working with a dual server configuration, 2GB of RAM will be sufficient for your starting environment. However, when building a dedicated server, more RAM is better!!

SQL Server Configuration

Similar to VPS hosting, if you have both SQL Server and IIS configured on the same machine, it is very important to set memory limitations for SQL Server to ensure that it does not overrun IIS in a fight for memory. Typical baseline recommendations would not allow SQL Server to use more than 50% of the total available physical RAM in these types of situations.

IIS Configuration

As mentioned in earlier sections the default 20 minute idle timeout is a configuration setting that is recommended to be changed to prevent un-necessary application recycles.

See section Recommended IIS Application Pool Configurations for full Application Pool configuration recommendations.

Recommended IIS Application Pool Configurations

As previous sections have mentioned, at the server level one of the most important items to configure is the idle timeout value for the application pools to prevent IIS from unloading the application. In our experience, we have found that two specific configuration changes from the default result in the best overall server configuration.

The first step is to modify the idle timeout from 20 minutes to 120-240 minutes. This will still allow an application to shut down if it is incredibly low traffic. Secondly, we recommend to do a regular recycle of the application pool at an off peak hour. For our sites we recycle the application pool at 2AM. Doing this helps to ensure that memory use and overall server health is maintained.

In addition to the idle timeout, it is important to remember that a DNN application should have its own application pool, and in the case of multi-portal installations.

DO NOT setup the same home directory in two different IIS Websites or application pools. Doing this will result in all scheduled jobs running twice and sub optimal server performance.

DotNetNuke Host Settings Configuration

DotNetNuke is a very flexible platform and as such has a number of configuration options available. These configuration options grant a great flexibility but are often items that are considered "too complex" or are simply overlooked by site administrators. The following sub sections discuss items existing under the "Host" -> "Host Settings" menu option and how they can impact/modify performance of a DotNetNuke installation.

This document only covers the items relevant to performance. For a full overview of available host settings please see this article - http://www.mitchelsellers.com/blogs/articletype/articleview/articleid/189/dotnetnuke-host-settings-explained.aspx

Authentication Providers

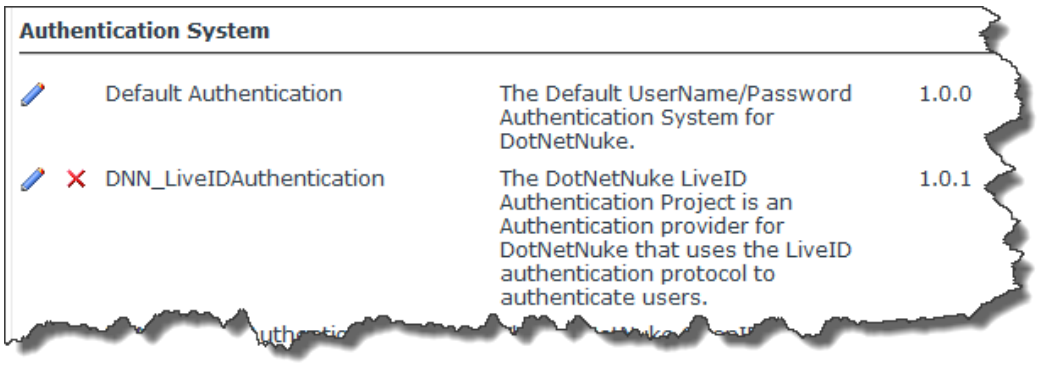
Starting with DotNetNuke 4.7.0, authentication provider extension types were introduced that allow website administrators to add functionality to authenticate users against different sources of record. Most DotNetNuke sites that we work with have only a single authentication provider in use, and that is the standard DotNetNuke configuration. However, behind the scenes you can have a performance hit when other providers enabled, even if they are not being used. This performance issue is typically limited to the login page itself where DotNetNuke polls all enabled providers to see if they must be rendered, but the performance change can be major.

It is IowaComputerGurus' recommendation that you disable all providers that are not being used. Disabling the other providers on our website reduced login page load time from 3.5 seconds to 0.75 seconds with no other system configuration changes.

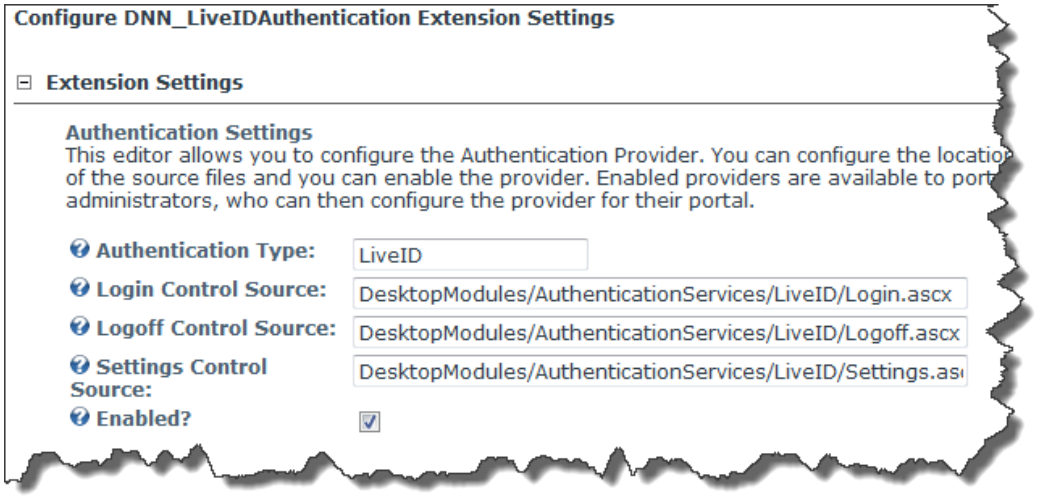
It is important to note that by default DotNetNuke installs three providers, so this configuration change is needed on ALL DotNetNuke portals.

Changing Enabled Providers (DNN 5.x and later)

In DotNetNuke 5.x and later the administration process to enable/disable authentication providers became more complicated requiring an edit to each providers definition. To enable/disable a provider start by going to "Host" -> "Extensions", on this page you will see an "Authentication System" section similar to the below image.



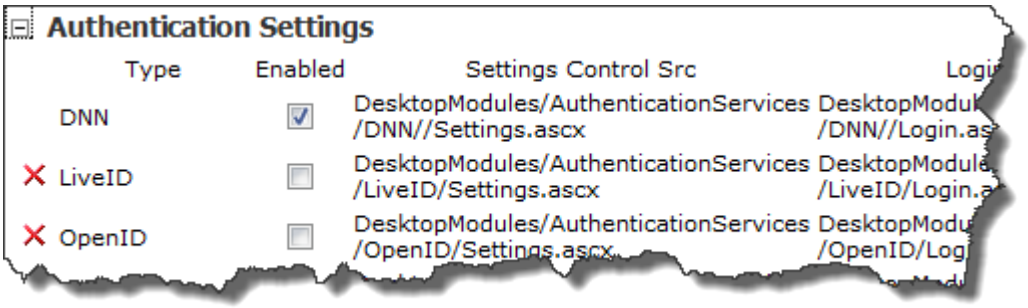
You might have more or less providers listed here. The standard DotNetNuke provider is labeled "Default Authentication". To disable a provider you must click "Edit" on each entry and you will be taken to a page similar to the following.



From here you can un-check the "Enabled" box. NOTE: This disables the provider at the host level and makes it not available to ANY child portal.

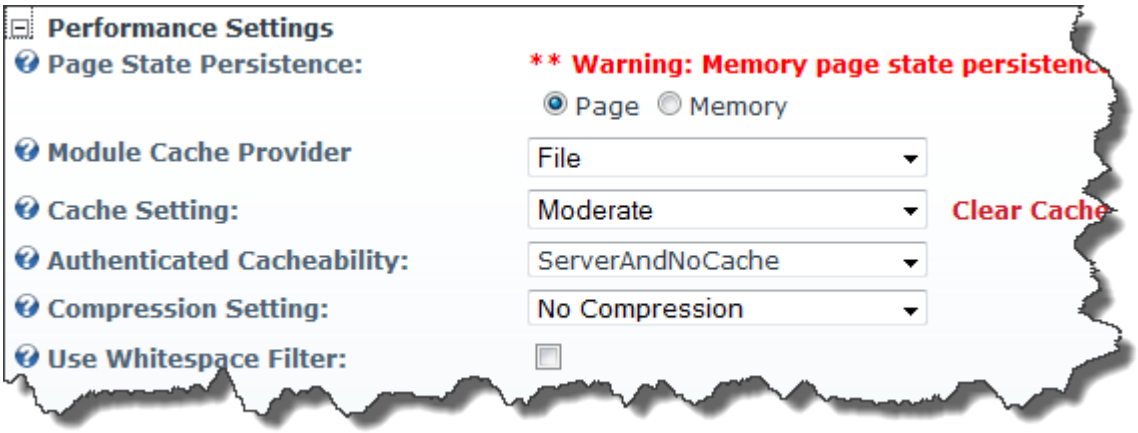
Changing Enabled Providers (DNN 4.7.0 - 4.9.6)

If you are working in a DotNetNuke 4.x environment this process is much similar, simply navigate to "Host" -> "Host Settings" and under "Advanced Settings" -> "Authentication Settings" you will see the following display. Simply un-check any providers that should be disabled. NOTE: this setting is saved as you make the changes, update is not necessary!

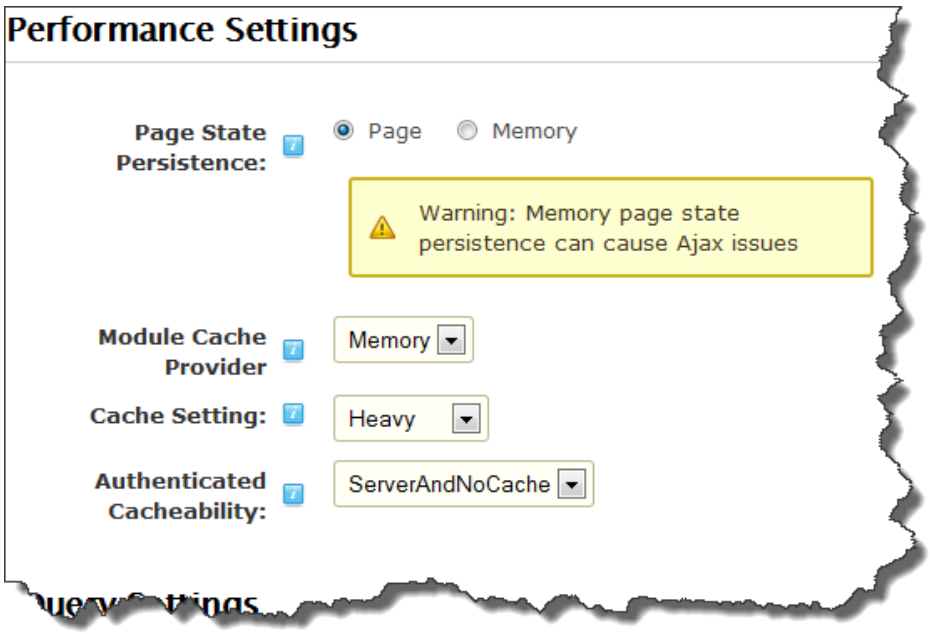


Performance Settings

Default DotNetNuke performance settings are OK for testing environments. Most other environments require items to change in at least one of these areas. The following sub sections will outline recommendations for configuration of these items. NOTE: The individual performance settings will be based on the version of DotNetNuke used, as well as the edition. Settings have evolved greatly since 4.x days. The following show basic representations of default configurations.



6.2.x and Later



Page State Persistence (Pre 7.x)

Although changing this option to "Memory" could reduce the overall size of the request sent to the user. Our experience has proven that this option most commonly causes other issues. We recommend that you NEVER set it to anything other than "Page". This feature has been removed in current versions of DNN.

Module Caching Method (4.x) / Module Cache Provider (5.x +)

The module caching method configures how DotNetNuke stores its cache of module objects. The proper configuration of this will vary depending on the specific environments that you are hosting in. With Shared Hosting and Dedicated Hosting, typically you will see better results using Memory caching. This is because you have memory available, and you want to reduce the amount of disk IO. When looking at some cloud computing environments, disk based caching makes sense. Website content is stored on a SAN or similar device with very good write speeds, and memory availability is either limited or sporadic.

Overall, starting with a baseline you can determine which option suits your needs the most after seeing the impacts.

Performance Setting (4.x) / Cache Setting (5.x+)

The performance setting is used to control how much other underlying data is cached by DotNetNuke. IowaComputerGurus recommendation is to always leave this set to ***Heavy***. Moderate caching simply does not offer enough caching.

Compression and Whitespace Filter (Pre 6.2.x+)

Prior to DNN 6.2.x there were options to have DotNetNuke perform compression operations for your website. This functionality was removed in 6.2 and the go-forward recommendation is to use IIS to handle this operation.

Output Cache Provider (5.x – Commercial Editions by default)

DNN Commercial editions added another type of caching provider into the available extension points. Specifically, the Output Cache Provider was created; this provider is used to cache the entire content of a generated page allowing for all database lookups and page generation activities to be skipped for un-changed content.

The functionality provided by this cache provider is very similar in nature to a portion of the functionality offered by the commercial PageBlaster module available from Snapsis. The extension point for this provider type is also available for DotNetNuke Community Edition, however at this time there are no implementations of the provider available.

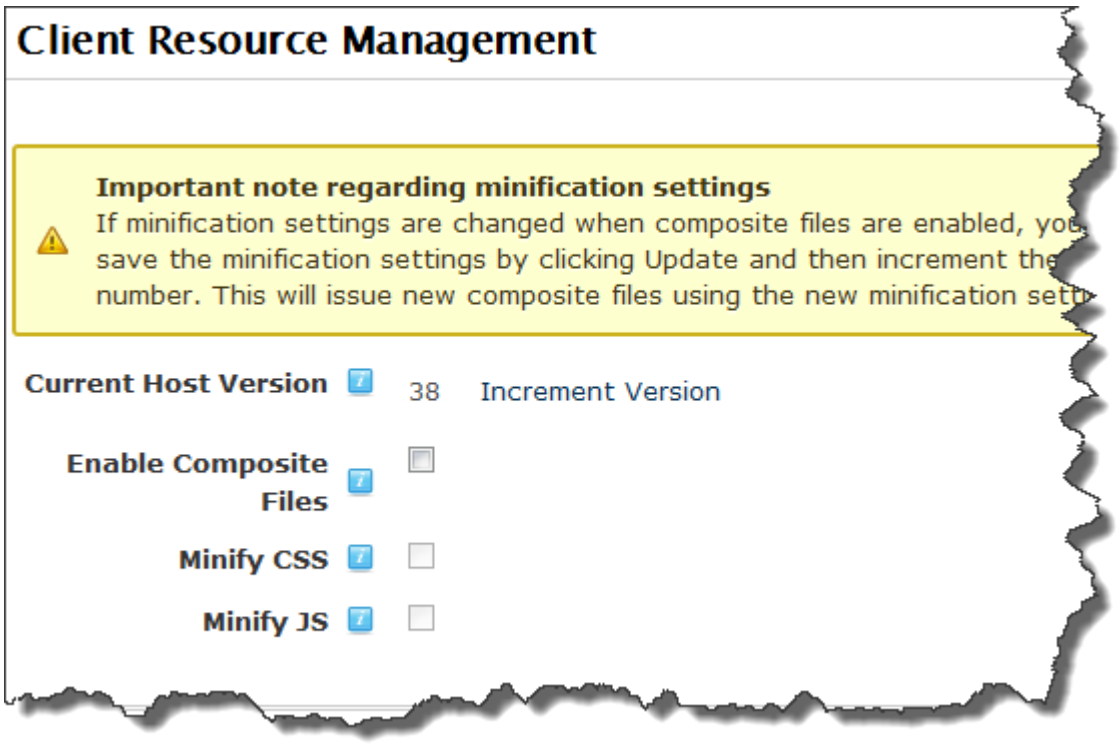
For users with this option enabled, IowaComputerGurus recommends using the same setting for this as what was used for "Module Caching". As the same rules and impacts apply to this type of caching. The Skins, Modules section of this document talks a bit more about the Output Cache Provider.

jQuery

Although not a direct performance contributor, you will want to be sure that you are not using the Debug version of jQuery as it adds a significant amount of page overhead to the application. If desired, you can enable the use of "Hosted jQuery" which offloads the serving of the jQuery libraries to third-party servers. On heavily trafficked sites, or sites with large image payloads, offloading these two requests to other servers can have a noticeable impact on your users.

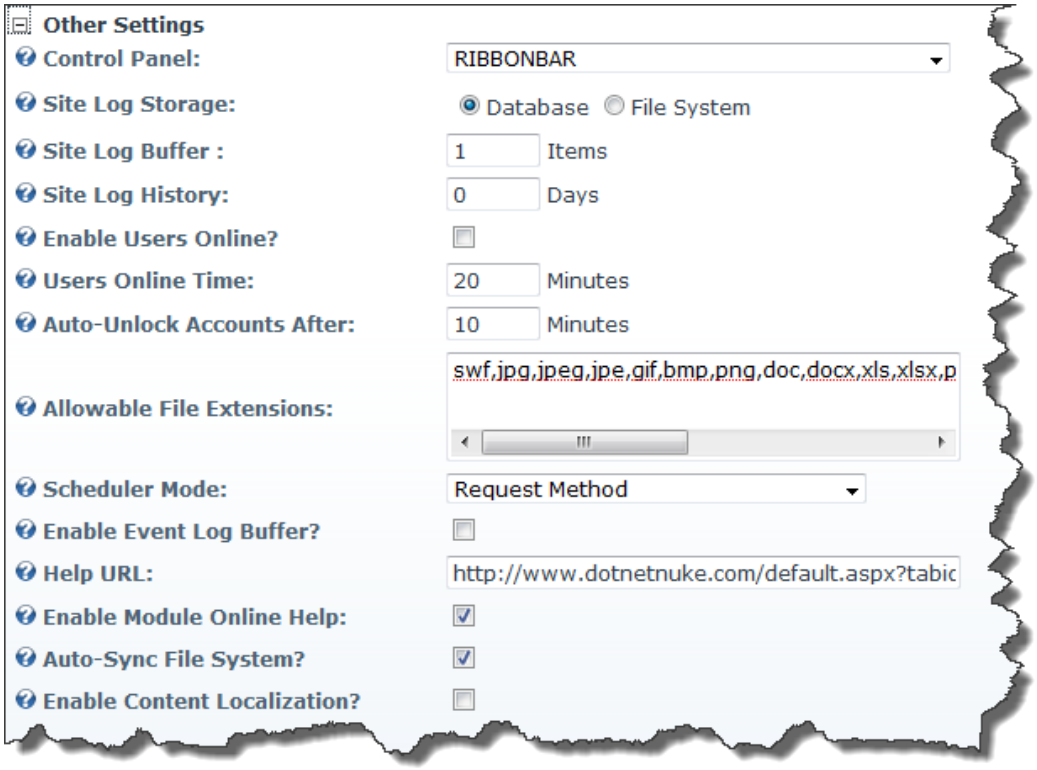
NOTE: If opting for hosted jQuery be sure to remember that you did this as you upgrade DotNetNuke in the fiture. As the core upgrades the minimum version of jQuery needed will change, however upgrades will NOT change the value for hosted jQuery. If you see unusual behavior on edit screens, validate that your hosted jQuery and "installed jQuery" versions are equal.

Client Resource Management



Client resource management is a newly added feature to the DotNetNuke 6.x platform. It is a very powerful function that allows DotNetNuke to create composite files combining your multiple CSS or JS files into a single file and optionally minifying the result.

The impacts of this to performance can be massive, improving page load time and decreasing page payload by a factor of 10 or more depending on the site and the number of included files. However, it should be noted that changing this setting CAN cause extreme layout issues and enabling should not be taken lightly. Our recommendation is to start small by enabling the composite file option and validating that the site functions as expected, both as a logged in user as well as a guest. If everything passes, continue the process enabling each minify option and re-testing along the way. If issues are encountered, research will be needed to identify the root cause of the issue.



This is another section of the configuration that is often overlooked. Not all settings here apply to performance optimization. The specific items that do apply are discussed in detail below.

Disable Users Online (4.x) / Enable Users Online (5.x+)

Users online is a facility that allows DotNetNuke to show you which users are online at the current time. This process adds a fair amount of overhead and it is recommended that you do NOT have users online enabled. (In 5.x the box should be unchecked, in 4.x the box should be checked).

Site Log History

Site Log History controls the data retention policy of the Site Log functionality available in the DotNetNuke core. For performance reasons IowaComputerGurus Inc recommends that the Site Log History setting be set to 0 days which disables the site log functionality.

This recommendation is due to the plethora of other processes available to obtain site statistics and the dramatic performance reduction that is realized when the log data fills up.

Scheduler Mode

This setting controls the manner in which DotNetNuke scheduled tasks are triggered. The default configuration of "Request Method" requires that on page requests a check is performance and any needed tasks are triggered. It has been our experience that this mode introduces additional request overhead and can delay end-user experiences. Our recommended configuration is to use the "Timer Method". This launches the scheduler on a separate thread rather than using the request to poll to tasks. With the other recommended configurations we have also experienced more consistent execution schedules on tasks.

Timer mode has become the default setting in early 7.x and later versions of DNN.

Enable Event Log Buffer?

Various items in DotNetNuke write entries to the event log, user login, page changes, etc. Without this option enabled, the insert is done right as the action occurs. Enabling the buffer allows DotNetNuke to queue these items and lets the user move on to other items faster. It is our recommendation that this setting be enabled on all installations.

Auto-Sync File System

This is the final configuration option in this section and relates to an automatic sync functionality that updates the file system information every time a user uses the UrlControl. It is our recommendation to disable this functionality as it is only needed if files are added to the portal via FTP on a regular basis.

DotNetNuke Scheduler

Another default DotNetNuke configuration that is worth reviewing to ensure that a site is optimized for the best performance is the collection of Scheduled tasks. Scheduled tasks provider vital functions for DotNetNuke, however, if a specific task runs too frequently it can cause an adverse reaction to the site performance. In some cases it can even render a site useless.

Search Process Items

Typically, the most common scheduled item to modify is the "Search Engine Scheduler" task. This scheduled task is used to index site content so that users can search the site. The default site configuration has this task running every 30 minutes. In most cases this is much too frequent. DotNetNuke when indexing content searches through ALL content, not just updated content. Therefore on large sites, the index process itself can take more than 30 minutes to process, having an adverse impact on the system.

Regardless of error conditions that might arise from long running tasks, this task also uses a high amount of SQL Server resources which then can slow the site down. Our recommendation is to modify the schedule from once every 30 minutes to once every 12-24 hours. This way the content still gets updated on a regular basis for search, but we limit the potential performance impact of the operation.

Search Tasks on Commercial Editions

There are additional search items when running on a commercial edition of DNN. These items will start with "Search" and vary depending on the specific version of the product you use. The key is to ensure that you do not have both the DNN Platform AND commercial items enabled. Additionally, the frequency of indexing should match your expected search freshness.

Messaging Dispatch

By default this service is enabled and runs once per minute on the site. This is important for those sites running social communities. However, if you are working on a site that does not have social aspects you will want to disable this feature to help avoid wasting resources.

Web Farm Considerations

If you are working with DotNetNuke in a web-farm environment it is also important to ensure that you are only running the scheduled jobs one at a time if they are database tasks. For example, since the search processes all update the database, they only need to run on ONE server in the farm or not all. To configure this edit the settings for the task, use the "Run on Serves" option to specify a single server rather than all.

Skins, Modules, External Services, JavaScript & the Others

Now that you have DNN configured at the most basic level, it is time to start digging into the other areas that can be a bit problematic. As you look at performance, there are many tools that can be used to review, score and otherwise recommend "what is best" for you. In this section we will discuss some known areas of maintenance, configuration, or simply using watchful eyes to ensure high performance.

Regular Clearing of Event Log

One of the built in functions within DotNetNuke is the "Event Log" (Admin -> Event Log). This is a log that records information such as user login success/failure, general module exceptions, and critical site level changes. The event log is a great resource. It is also very important that this table be managed. DotNetNuke by default does not clear this table, therefore, over time the log can grow to be very large.

As the size of the EventLog increases, performance will decrease due to the additional costs of inserting records. In a typical situation IowaComputerGurus recommends truncating this table on a daily basis. We make this recommendation because this allows for a 24 hour rolling history for any diagnostics purposes and avoids rapid transaction log growth when deleting records.

This can be accomplished a number of ways; SQL Server Scheduled Job, manual process, or using a module such as the free "Scheduled SQL Jobs" module which is available from our website.

NOTE: If you require auditing you might consider shipping this information outside of the database should you need login history or other information for historical purposes.

Regular Monitoring of Event Log Errors

Along with the regular cleaning of the event log it is important to review the error messages that are reported. If recurring errors are found it is important to resolve these items. The process of throwing exceptions and logging add un-necessary overhead to the system. Many recurring errors are common items that are easy to find. If you find a recurring error you, can easily go to the DotNetNuke Forums or our forums to ask for advice on how you might resolve the recurring errors.

Examples of common error messages would include regular messages each time the application starts that refer to a module that was not successfully installed, or a module that has invalid code at an extension point that is not readily visible to the site administrator. An example of this might be a module that integrates with the DotNetNuke search system and throws an error every time DNN tries to index the module's content. These errors are ONLY reported to the event log and would otherwise go un-noticed.

Skin Modifications

Another key piece of a site that can impact the performance of a site is the way in which the skin is built. Since the skin defines the user interface if the code here is in-efficient or overly heavy it can add additional time to download and render, thus slowing the site. There are two key recommendations that we have when it comes to skin design and performance.

Create an XHTML Compliant CSS Based Skin

The other item here that helps with site performance is to ensure that your skin is XHTML compliant and uses a CSS based layout. The XHTML compliance ensures that you have valid HTML which allows the browser to more quickly render the content once downloaded from the web server. A CSS based skin is typically lighter in terms of HTML markup which reduces the overall size of rendered content, reducing the time necessary to download the content.

Reduce Inline Images

Many modern skins employ Image Sprites to lower the number of items that are necessary to download each time the page loads. If you reduce the number of files that need to be downloaded you will improve the performance of your site. Work with your skin developer or internal design team to use sprites when possible to reduce the overall overhead of your skins.

Enable Static File Caching

Within ASP.NET by default items such as images, documents, and the like are not configured to be "cached' by the user on their local machine. This means that if your website has for sample 5 images on a page every time the page requires the image to be downloaded each time. If you enable static file caching which can be done via the web.config, you can significantly reduce the amount of time it takes for your site to load and reduce the bandwidth that your site will use.

For more information on this, please see Mitchel's detailed blog post http://mitchelsellers.com/blogs/2012/01/19/improve-performance-with-static-file-caching.aspx

Quality Third Party Modules

It is often the case when working with sites that are experiencing issues that we will find third-party modules are contributing or causing the performance issues identified. When selecting your third-party modules be sure to look into their history, and test them out. Ensure that your site performance is as good with the module as it is without. It can help identify things much sooner.

Seek Outside Assistance

Finally, if none of the items in this document have resolved your performance issues, there are many vendors in the DotNetNuke ecosystem that offer performance optimization services. If you are interested in IowaComputerGurus performance optimization services you may e-mail Mitchel directly at msellers@iowacomputergurus.com.

Although the processes are similar, each performance issue is a completely different journey. Reaching out to those with experience to lend a hand or to diagnose the issue for you will result in significant time savings.

Web Farms & High Load Situations

There are a number of new items to consider if you are thinking about supporting DNN in a situation where you will support high volume traffic or are working with a web farm configuration with multiple front-end web heads. The following are a few key items to remember. Specific implementation tasks require the consultation of an experienced individual due to the potential impacts.

Content Delivery Network (CDN)

Incorporating a CDN into your high volume situation can greatly improve throughput by allowing your web server(s) to focus on the more complicated tasks. As much as an 85% throughput improvement has been observed by simply adding a CDN to an existing environment.

ASP.NET Runtime Configuration

As you work with more capable servers, you will find that ASP.NET's default installed configurations are setup for smaller servers. Settings can be used to improve the throughput of your application by allowing ASP.NET to properly use all resources available. We have improved server-by-server configuration by more than 50% for clients by properly matching these settings to the server specifications AND application behaviors.

Ongoing Performance Improvement

Performance is not a "one and done" operation. It is important to not only validate your configuration but put items in place to monitor your application to be able to spot situations where the application health might not be as good as it used to be.

There are a number of tools and services available for this type of task. If you have your own server, we strongly recommend the use of NewRelic as it provides a plethora of information that helps with ongoing monitoring as well as with diagnostic processes.

Contact IowaComputerGurus

We welcome any feedback that you might have to this document as we have been constantly updating the document based on new findings and information as it comes available. If you have specific questions of the content of this document or would like to share any feedback we can be reached through any of the following communication channels.

IowaComputerGurus Inc.
5550 Wild Rose Lane, Suite 400
West Des Moines, Iowa 50266

Phone: (515) 270-7063
Fax: (866) 591-3679
Twitter: @IowaCompGurus
Email: webmaster@iowacomputergurus.com
Website: http://www.iowacomputergurus.com
Support: http://support.iowacomputergurus.com