Redbooks Paper

Phillip Dundas
David Watts

Tuning Windows Server 2003 on IBM System x Servers

Windows® Server 2003[1] is Microsoft's mainstream server operating system and has been now for almost four years. Over previous versions of the Windows server operating system, including Windows 2000 Server and Windows NT® Server, Windows Server® 2003 offers considerable improvements in stability, performance, security, scalability and manageability.

Since the last iteration of this redpaper, Microsoft have announced three important enhancements to the core Windows 2003 server operating system. These are:

- Windows Server 2003, Service Pack 1 for 32-bit (x86) Editions
- Windows Server 2003, x64 (64-bit) Editions
- Windows Server 2003, Release 2 (R2) for 32-bit (x86) & 64-bit (x64) Editions

This redpaper builds upon the performance tuning techniques detailed in its previous release to also emphasize the performance benefits that can be realized from these important product releases.

Introduction

Windows Server 2003 is designed to be a largely "self-tuning" operating system. A standard "vanilla" installation of the operating system will yield sound performance results in most circumstances. In some instances however, specific settings and parameters can be tuned to optimize server performance even further. This chapter describes in detail many tuning techniques, any of which can become another weapon in your arsenal of methods to extract the best performance possible from your Windows server.

Tip: As with all configuration changes, you need to implement the following suggestions one at a time to see what performance improvements are offered. If system performance decreases after making a change, then you need to reverse the change.

[1] Product screen captures and content reprinted with permission from Microsoft® Corporation.

Many of the changes listed in this chapter might only offer marginal performance improvements in and of themselves. The real benefits however of server tuning are realized when multiple tuning improvements are made and combined with one another. For a given server function, not all tuning techniques listed in this chapter will be appropriate. The challenge for the server engineer or architect is in determining which of these techniques, when combined, will yield the biggest performance enhancements. Many factors will play into this, including the server function, the underlying hardware and how this has been configured, and the network and storage infrastructure that the server is connected to.

It is also well worth noting that some of the performance tuning techniques outlined in this chapter might no longer be relevant in the x64 (64-bit) versions of Windows Server 2003. Several of these techniques described throughout are used to tweak and work around the limitations of the x86 (32-bit) architecture. As a result, in the x64 versions, they are no longer relevant. Where known, this has been outlined.

The Windows Server 2003 family - 32-bit (x86) Editions

Windows Server 2003 comes in four different 32-bit (x86) versions. These are:

- Windows Server 2003, Web Edition
- Windows Server 2003, Standard Edition
- Windows Server 2003, Enterprise Edition
- Windows Server 2003, Datacenter Edition

Each of these has support for different hardware configurations and features and largely determines how scalable the server is. Table 1 compares the capabilities of the various versions available in the 32-bit (x86) versions of Windows Server 2003.

Note that the maximum amount of memory and the number of CPUs supported in the Enterprise and Datacenter editions of the 32-bit editions of Windows Server 2003 has increased with the release of Service Pack 1 (SP1) and Release 2 (R2).

Table 1 Windows Server 2003 Family - 32-bit (x86) Editions

Requirement	**Web Edition**	**Standard Edition**	**Enterprise Edition**	**Datacenter Edition**
Maximum supported RAM	2 GB	4 GB	32/64* GB	64/128* GB
Number of supported processors	1 to 2	1 to 4	1 to 8	8 to 32/64** 8-way capable***
Server clustering	No	No	Up to 8 node	Up to 8 node
Support for /3GB switch	No	No	Yes	Yes
Support for /PAE switch	No	No	Yes	Yes

* Maximum physical memory (RAM) supported has increased from 32 GB to 64 GB for Enterprise Edition with R2 and from 64 GB to 128 GB for Datacenter Edition with R2.

** Maximum CPU support has increased from 32 to 64 CPUs for Datacenter Edition with R2

*** Windows Server 2003 Datacenter Edition requires a server that is eight-way capable but only requires a minimum of four-processors in the actual system.

The Windows Server 2003 family - 64-bit (x64) Editions

Microsoft have not released the Web Edition of Windows Server 2003 in the 64-bit (x64) family of server operating systems. The editions that are available are:

- Windows Server 2003, Standard x64 Edition

- Windows Server 2003, Enterprise x64 Edition
- Windows Server 2003, Enterprise Itanium® Edition
- Windows Server 2003, Datacenter x64 Edition
- Windows Server 2003, Datacenter Itanium Edition

As it is a considerably later release, much of the code-base for the 64-bit (x64) editions of Windows Server 2003 are based on the same code the makes up the Service Pack 1 editions of Windows Server 2003. As a result, Service Pack 1 is not an option for the 64-bit (x64) editions. Release 2 (R2) is an optional extra for the 64-bit (x64) editions of Windows Server 2003, though it is expected that most customers would install R2 by default to access the many extra features available within this latest product offering.

Due to the fundamental architectural differences of 64-bit computing, vastly higher memory thresholds are available in the 64-bit (x64) editions of Windows Server 2003, as evidenced in Table 2.

Table 2 Windows Server 2003 Family - 64-bit (x64) Editions

Requirement	Standard x64 Edition	Enterprise x64 Edition	Datacenter x64 Edition	Enterprise Itanium Edition	Datacenter Itanium Edition
Maximum supported RAM	32 GB*	1 TB	1 TB	1 TB	1 TB
Number of supported processors	1 to 4	1 to 8	8 to 64	1 to 8	8 to 64 8-way capable*
Server clustering	No	Up to 8 node	Up to 8 node	Up to 8 node	Up to 8 node
* Windows Server 2003 Datacenter Edition requires a server that is eight-way capable but only requires a minimum of four-processors in the actual system					

A more thorough comparison of all feature differences between the various versions of the Windows Server 2003 operating system for both 32-bit and 64-bit editions can be found at:

http://www.microsoft.com/windowsserver2003/evaluation/features/comparefeatures.mspx

Windows Server 2003, 64-bit (x64) Editions

After years of working around the limitations of the 32-bit processor architecture, in April, 2005, Microsoft released the much awaited 64-bit editions of Windows Server 2003. While the Itanium version has been available for some time, it is the release of the editions of Windows Server 2003 to support the *x64* processors that will see 64-bit computing finally transition into the Windows server mainstream.

In a relatively short-period of time, it is expected that the 64-bit editions of Windows Server 2003 will displace the now seemingly archaic 32-bit cousin. This is largely assisted by the high-level of compatibility that Microsoft have built into the 64-bit (x64) operating system, offering true backward compatibility for 32-bit applications with little to no degradation in performance.

32-bit limitations

The amount of virtual memory that can be addressed by the 32-bit versions of Windows Server 2003 is 4 GB, through a virtual address space. On a standard implementation, this 4 GB is divided into 2 GB for kernel mode processes and 2 GB for application (user) mode processes.

In Windows Server 2003, 32-bit editions, it is possible to increase the amount of memory available from 2 GB to 3 GB for 32-bit applications that have been designed to use more than 2 GB, through the use of the /3GB and /PAE switches, as explained in , “The /3GB BOOT.INI parameter (32-bit x86)” on page 30 and , “Using PAE and AWE to access memory above 4 GB (32-bit x86)” on page 31.

This increase of available user memory from 2 GB to 3 GB presents problems, however:

- It imposes limitations on the amount of memory available to kernel mode processes to 1 GB
- It does not work around the architectural limit of the total 4 GB virtual address space.
- It increases the difficulty of developing applications as they need to make use of the Addressable Windows Extensions (AWE) application programming interface (API) to take advantage of Physical Address Extensions (PAE).
- It has not removed the physical memory constraint of 64 GB.

With the upgrade to Windows Server 2003 64-bit (x64) editions, these limitations no longer exist and there are opportunities for significant improvements in server performance.

64-bit benefits

The single biggest performance increase of the 64-bit architecture is the amount of memory that can now be addressed. With Windows Server 2003 x64 Editions, the addressable virtual memory space increases from the 32-bit limit of just 4 GB to 16 TB. Entire databases, data sets, directory stores, indexes, Web caches and applications can now be loaded completely into memory, delivering often staggering processing performance improvements and vastly increased scalability.

It is worth noting that the current Windows Server 2003 x64 editions actually only use 40 bits for addressing memory, offering an address space of 2^40, or 16 TB. 16 Exabytes is that actual theoretical maximum of a full 64-bit address space.

This virtual address space is divided evenly between user mode and kernel mode, as with 32-bit Windows. This provides native 64-bit applications with 8 TB of virtual address space. Even 32-bit applications that have been written to take advantage of memory greater than the 2 GB limitation of the 32-bit architecture can benefit from this immediately in that they can now address 4 GB of virtual address space as this space no longer needs to be shared with kernel mode processes.

Table 3 highlights some of the key difference between 32-bit and 64-bit memory limits. Each of the notable improvements in these memory limits for the 64-bit (x64) platform offers real scalability and performance enhancements.

Table 3 Memory limitations of 32-bit (x86) and 64-bit (x64) Windows Server 2003

Memory Limit	32-bit (x86)	64-bit (x64)
Total virtual address space	4 GB	16 TB
Virtual address space per 32-bit process	2 GB	4 GB
Virtual address space per 64-bit process	Not applicable	8 TB
Paged Pool	491 MB	128 GB
Non-paged Pool	256 MB	128 GB

Memory Limit	32-bit (x86)	64-bit (x64)
System Page Table Entry (PTE)	660 MB to 990 MB	128 GB

The Windows on Windows 64 emulator (WOW64) allows 32-bit applications to run on Windows Server 2003 x64 Editions exactly as they might run on a 32-bit edition. This has been written so optimally however than any overhead imposed is in emulation activities is very marginal and in many cases imperceptible. Even with the emulation between 32-bit and 64-bit, in several cases 32-bit applications will run faster on Windows Server 64-bit (x64) Editions due to other improvements in the operating system, offering another notable benefit to this new operating system version.

One of the other most notable advantage of the 64-bit (x64) editions is the greatly increased amount of physical RAM than can be supported, offering huge scalability benefits. With the Standard Edition of Windows Server 2003 x64, the maximum supported memory is 32 GB. With the Enterprise and Datacenter Editions however, this blows out to a considerably larger 1 TB of RAM. When compared to the previous memory maximums of 4 GB, 64 GB and 128 GB of the Standard, Enterprise and Datacenter editions of Windows Server 2003 R2, the increase in supportable memory, and hence performance, is significant.

Overall performance improvements in disk and network I/O throughput and efficiency should be evident in the 64-bit (x64) editions of Windows Server 2003. Greater physical memory support and addressable memory space means that caches can be considerably larger, improving I/O performance. An increased number of larger (wider) processor registers will also deliver notable performance gains to applications as data does not need to be written out to memory as frequently and function calls can process more arguments.

Windows Server 2003 64-bit (x64) Editions also delivers improved reliability over previous versions. Based on the exactly the same source code as Windows Server 2003 Service Pack 1 32-bit editions (x86), the newer edition will offer the same reliability that this platform has offered to date. In addition, Windows Server 2003 64-bit (x64) editions include Patch-Guard, a technology which protects poorly-written code third-party code from patching the Windows kernel which in turn could destabilize or crash a server.

Security improvements are also available with the 64-bit (x64) edition of Windows Server 2003. Building on the benefits of Data Execution Prevention (DEP) released first in Windows Server 2003, Service Pack 1 (32-bit x86), the 64-bit (x64) editions all include this feature as a standard. DEP prevents Windows against buffer overflows, effectively stopping malicious code from being able to execute from memory locations it should not.

All these features of Windows Server 2003 64-bit (x64) editions serve to make this new version of the operating system the most high-performing, scalable, stable and secure version released to date.

The transition to 64-bit computing

With the release of Intel® 64 Technology (previously known as EMT64T) and AMD AMD64, server hardware vendors have made the transition from 32-bit (x86) to 64-bit (x64) processing a very straightforward process. These processors support both 32-bit and 64-bit operating systems and applications, making the migration path an easy one.

With the 64-bit (x64) versions of Windows Server 2003 able to run 32-bit applications directly, often at a much higher level of performance, the move to the optimal native 64-bit computing should present few hurdles.

Acknowledgements

Much of this material for this section on Windows Server 2003 64-bit (x64) editions has been collated from two key articles available from the Microsoft Web site. More detail and case studies of the benefits of Windows Server 2003 64-bit (x64) computing can be found by referring to these two papers:

- *Benefits of Microsoft Windows x64 Editions*

  `http://www.microsoft.com/windowsserver2003/techinfo/overview/x64benefits.mspx`
- *Windows Server 2003 x64 Editions Deployment Scenarios*

  `http://www.microsoft.com/windowsserver2003/64bit/x64/deploy.mspx`

Windows Server 2003, Release 2 (R2)

Windows Server 2003, R2 is an update release for the Windows Server 2003 operating system. This release brings an impressive array of additional features to the native operating system.

This release is different from a Service Pack in that it brings new features and functionality to the operating system while as Service Pack is a rollup of fixes, updates and patches at a given point in time. That said, the installation of R2 is dependent on Windows Server 2003, Service Pack 1 already being installed.

R2 offers enhancements to Windows Server 2003 in the following main areas:

- Simplified branch office server management
- Improved identity and access management
- Reduced storage management costs
- Rich Web platform
- Cost effective server virtualization
- Seamless UNIX® / Windows Interoperability

For more detail on the features delivered by R2, visit the following links:

`http://www.microsoft.com/windowsserver2003/R2/whatsnewinr2.mspx`
`http://download.microsoft.com/download/7/f/3/7f396370-86ba-4cb5-b19e-e7e518cf53ba/WS03R2RevGuide.doc`

The components of R2 that offer notable performance benefits are those included to improve branch office server manageability, such as the following:

- Robust file replication

  The replication engine for the Distributed File System (DFS™) has been completely rewritten in Windows Server 2003 R2. DFS is multimaster file replication service, significantly more scalable and efficient in synchronizing file servers than File Replication Services (FRS), its predecessor. DFS schedules and throttles replication processes, supports multiple replication topologies, and utilizes Remote Differential Compression (RDC) to increase WAN efficiency. If WAN connections fail, data can be stored and forwarded when WAN connections become available. Through the efficiency gains of these new features in R2 DFS, the performance of core user-facing processes improves.
- Advanced compression technologies

  Remote Differential Compression (RDC) is a WAN-friendly compression technology that replicates only the changes needed to ensure global file consistency.Any WAN performance improvements often serve to improve the user experience.

Processor scheduling

Windows uses *preemptive multitasking* to prioritize process threads that the CPU has to attend to. *Preemptive multitasking* is a methodology whereby the execution of a process is halted and another is started, at the discretion of the operating system. This prevents a single thread from monopolizing the CPU.

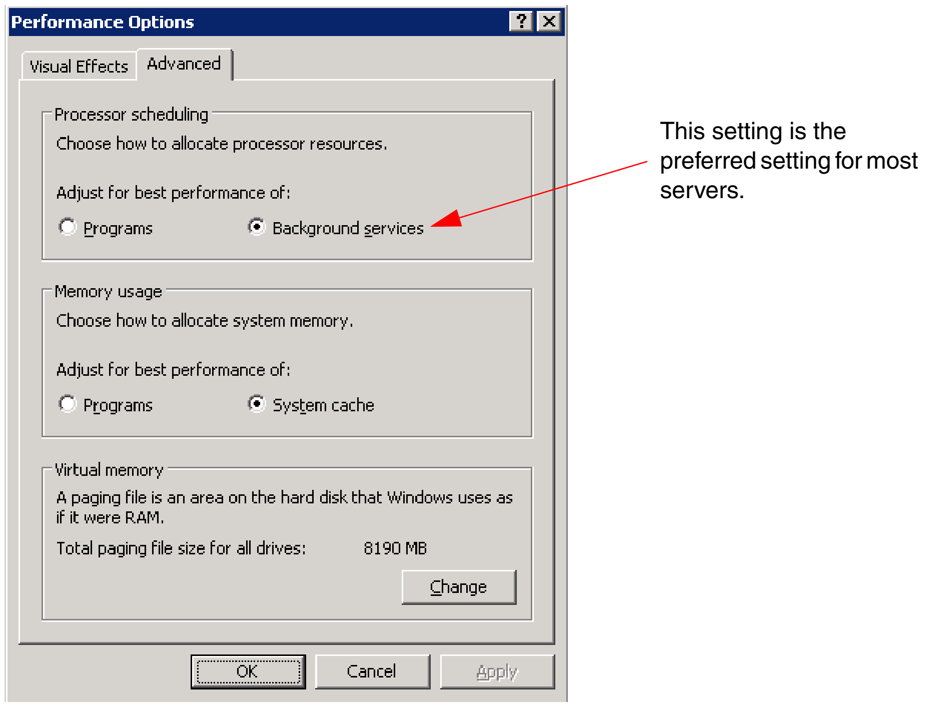
Switching the CPU from executing one process to the next is known as *context-switching*. The Windows operating system includes a setting that determines how long individual threads are allowed to run on the CPU before a context-switch occurs and the next thread is serviced. This amount of time is referred to as a *quantum*.

This setting lets you choose how processor quanta are shared between foreground and background processes. Typically for a server, it is not desirable to allow the foreground program to have more CPU time allocated to it than background processes. That is, all applications and their processes running on the server should be given equal contention for the CPU.

We recommend selecting **Background services** so that all programs receive equal amounts of processor time.

To change this:

1. Open the **System** Control Panel.
2. Select the Advanced tab.
3. Within the Performance frame, click **Settings**.
4. Select the Advanced tab. The window shown in Figure 1 opens.



Processor scheduling
Choose how to allocate processor resources.
Adjust for best performance of:
Programs [✓ Background services]

Memory usage
Choose how to allocate system memory.
Adjust for best performance of:
Programs [✓ System cache]

Virtual memory
A paging file is an area on the hard disk that Windows uses as if it were RAM.
Total paging file size for all drives: 8190 MB

Figure 1 Configuring processor scheduling

Modifying the value using the control panel applet as described above modifies the following registry value to affect the duration of each quanta:

```
HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\PriorityControl\
Win32PrioritySeparation
```

The Win32PrioritySeparation Registry values in Windows Server 2003 are:

- 0x00000026 (38) for best performance of Programs
- 0x00000018 (24) for best performance of Background services.

These values remain the same between the 32-bit (x86) and 64-bit (x64) editions of the Windows Server 2003 operating system.

We strongly recommend you use only the control panel applet shown in Figure 1 on page 7 for these settings in order to always get valid, appropriate, operating system revision-specific, and optimal values in the registry.

Virtual memory

Memory *paging* occurs when memory resources required by the processes running on the server exceed the physical amount of memory installed. Windows, like most other operating systems, employs *virtual memory* techniques that allow applications to address greater amounts of memory than what is physically available. This is achieved by setting aside a portion of disk for paging. This area, known as the *paging file,* is used by the operating system to *page* portions of physical memory contents to disk, freeing up physical memory to be used by applications that require it at a given time. The combination of the paging file and the physical memory installed in the server is known as *virtual memory*. Virtual memory is managed in Windows by the Virtual Memory Manager (VMM).

Physical memory can be accessed at exponentially faster rates than disk. Every time a server has to move data between physical memory and disk will introduce a significant system delay. While some degree of paging is normal on servers, excessive, consistent memory paging activity is referred to as *thrashing* and can have a very debilitating effect on overall system performance. Thus, it is always desirable to minimize paging activity. Ideally servers should be designed with sufficient physical memory to keep paging to an absolute minimum.

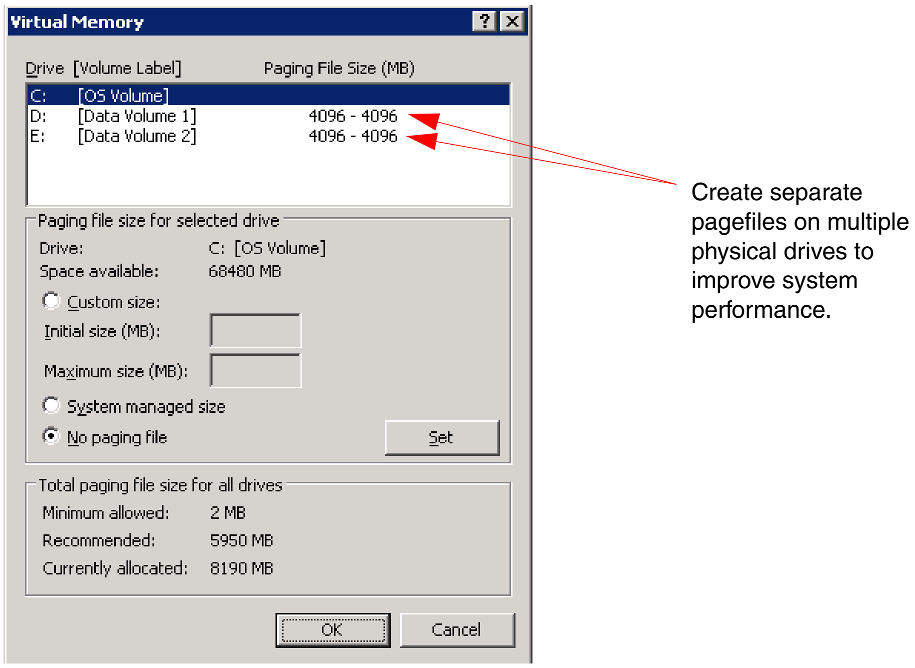
The paging file, or pagefile, in Windows, is named PAGEFILE.SYS.

Virtual memory settings are configured through the System control panel.

To configure the page file size:

1. Open the System Control Panel.
2. Select the Advanced tab.
3. Within the Performance frame, click **Settings**.
4. Select the Advanced tab.
5. Click **Change**. The window shown in Figure 2 on page 9 opens.

Windows Server 2003 has several options for configuring the page file that previous versions of Windows did not. Windows Server 2003 has introduced new settings for virtual memory configuration, including letting the system manage the size of the page file, or to have no page file at all. If you let Windows manage the size, it will create a pagefile of a size equal to physical memory + 1 MB. This is the minimum amount of space required to create a memory dump in the event the server encounters a STOP event (blue screen).


Virtual Memory

Drive [Volume Label] Paging File Size (MB)

```
C:   [OS Volume]
D:   [Data Volume 1]      4096 - 4096
E:   [Data Volume 2]      4096 - 4096
```

Paging file size for selected drive

```
Drive:             C: [OS Volume]
Space available:   68480 MB
```

Custom size: ○ Initial size (MB): Maximum size (MB):

System managed size ○

No paging file ●

Total paging file size for all drives

```
Minimum allowed:       2 MB
Recommended:           5950 MB
Currently allocated:   8190 MB
```

Set
OK
Cancel
Create separate pagefiles on multiple physical drives to improve system performance.

Figure 2 Virtual memory settings

A pagefile can be created for each individual volume on a server, up to a maximum of sixteen page files and a maximum 4 GB limit per pagefile. This allows for a maximum total pagefile size of 64 GB. The total of all pagefiles on all volumes is managed and used by the operating system as one large pagefile.

When a pagefile is split between smaller pagefiles on separate volumes as described above, when it needs to write to the pagefile, the virtual memory manager optimizes the workload by selecting the least busy disk based on internal algorithms. This ensures best possible performance for a multiple-volume pagefile.

While not best practice, it is possible to create multiple page files on the same operating system volume. This is achieved by placing the page files in different folders on the same volume. This change is carried out through editing the system registry rather than through the standard GUI interface. The process to achieve this is outlined in Microsoft KB article 237740:

http://support.microsoft.com/?kbid=237740

We do not recommend this approach as no performance gain will be achieved by splitting the page file into segments on the same volume regardless of the underlying physical disk or array configuration.

Configuring the pagefile for maximum performance gain

Optimal pagefile performance will be achieved by isolating pagefiles to dedicated physical drives running on RAID-0 (striping) or RAID-1 (mirroring) arrays, or on single disks without RAID at all. Redundancy is not normally required for pagefiles, though performance might be improved through the use of some RAID configurations. By using a dedicated disk or drive array, this means PAGEFILE.SYS is the only file on the entire volume and risks no fragmentation caused by other files or directories residing on the same volume. As with most disk-arrays, the more physical disks in the array, the better the performance. When distributed between multiple volumes on multiple physical drives, the pagefile size should be kept uniform between drives and ideally on drives of the same capacity and speed.

We strongly recommend against the use of RAID-5 arrays to host pagefiles as pagefile activity is write intensive and thus not suited to the characteristics of RAID-5.

Where pagefile optimization is critical, do not place the pagefile on the same physical drive as the operating system, which happens to be the system default. If this must occur, ensure that the pagefile exists on the same volume (typically C:) as the operating system. Putting it on another volume on the same physical drive will only increase disk seek time and reduce system performance as the disk heads will be continually moving between the volumes, alternately accessing the page file, operating system files and other applications and data.

Remember too that the first partition on a physical disk is closest to the outside edge of the physical disk, the one typically hosting the operating system, where disk speed is highest and performance is best.

Note if you do remove the paging file from the boot partition, Windows cannot create a crash dump file (MEMORY.DMP) in which to write debugging information in the event that a kernel mode STOP error message (“blue screen of death”) occurs. If you do require a crash dump file, then you will have no option but to leave a page file of at least the size of physical memory + 1 MB on the boot partition.

We recommend setting the size of the pagefile manually. This normally produces better results than allowing the server to size it automatically or having no page file at all. Best-practice tuning is to set the initial (minimum) and maximum size settings for the pagefile to the same value. This ensures that no processing resources are lost to the dynamic resizing of the pagefile, which can be intensive. This is especially given this resizing activity is typically occurring when the memory resources on the system are already becoming constrained. Setting the same minimum and maximum page file size values also ensures that the paging area on a disk is one single, contiguous area, improving disk seek time.

Windows Server 2003 automatically recommends a total paging file size equal to 1.5 times the amount of installed RAM. On servers with adequate disk space, the pagefile on all disks combined should be configured up to twice (that is, two times) the physical memory for optimal performance. The only drawback of having such a large pagefile is the amount of disk space consumed on the volumes used to accommodate the page file(s). Servers of lesser workloads or those tight on disk space should still try to use a pagefile total of at least equal to the amount of physical memory.

Creating the pagefile to optimize performance

Creating the whole pagefile in one step reduces the possibility of having a partitioned pagefile and therefore improves system performance.

The best way to create a contiguous static pagefile in one step is to follow this procedure for each pagefile configured:

1. Remove the current page files from your server by clearing the Initial and Maximum size values in the Virtual Memory settings window or by clicking **No Paging File**, then clicking **Set** (Figure 2 on page 9).
2. Reboot the machine and click **OK**. Ignore the warning message about the page file.
3. Defragment the disk you want to create the page file on. This step should give you enough continuous space to avoid partitioning of your new page file.
4. Create a new page file with the desired values as described is , “Creating the pagefile to optimize performance” on page 10.
5. Reboot the server.

An ever better approach is to re-format the volume entirely and create the pagefile immediately before placing any data on the disk. This ensures the file is created as one large contiguous file as close to the very outside edge of the disk as possible, ensuring no fragmentation and best disk access performance. The work and time involved in moving data to another volume temporarily to achieve this outcome often means, however, that this procedure is not always achievable on a production server.

Measuring pagefile usage

A good metric for measuring pagefile usage is *Paging file: %Usage Max* in the Windows System Monitor. If this reveals consistent use of the page file, then consider increasing the amount of physical memory in the server by this amount. For example, if a pagefile is 2048 MB (2 GB) and your server is consistently showing 10% usage, it would be prudent to add an additional say 256 MB RAM.

While today it is often considered an easy and relatively inexpensive way to upgrade a server and improve performance, adding additional memory should only be done after investigating all other tuning options and it has been determined that memory is indeed the system bottleneck. There is simply no benefit to be gained by having more physical memory in a server than it can put to use. Additional, unused memory might be better put to use in another server.

File system cache

The file system cache is an area of physical memory set aside to dynamically store recently accessed data that has been read or written to the I/O subsystem, including data transfers between hard drives, networks interfaces, and networks. The Windows Virtual Memory Manager (VMM) copies data to and from the system cache as though it were an array in memory.

The file system cache improves performance by reducing the number of accesses to physical devices attached to the I/O subsystem of the server. By moving commonly used files into system cache, disk and network read and write operations are reduced and system performance is increased. You can optimize Windows server performance by tuning the file system cache.

Performance of the file system cache is greatly improved in the 64-bit (x64) editions of Windows Server 2003. The default 2 GB kernel maximum virtual memory address space in the 32-bit (x86) editions of Windows is a major bottleneck as this same space is shared by the system page table entries (PTE), page pool memory, non-paged pool memory and file system cache. Using the /3GB switch on 32-bit (x86) systems can improve application performance (as described in , “Optimizing the protocol binding and provider order” on page 19 and , “Optimizing network card settings” on page 20) but forces the Windows kernel to operate in only 1 GB of virtual address space, potentially making the situation worse. These same constraints no longer apply in the 64-bit (x64) editions of Windows, greatly enhancing system performance.

Tip: The method for managing the file system cache has changed in Windows Server 2003. There are now two applets, not one as in previous versions of Windows.

In previous versions of the Windows server operating system, one Control Panel applet was used for managing the file system cache. Now, in Windows Server 2003, two configuration options exist that determine how much system memory is available to be allocated to the working set of the file system cache versus how much memory is able to be allocated to the working set of applications, and the priority with which they are managed against one another.

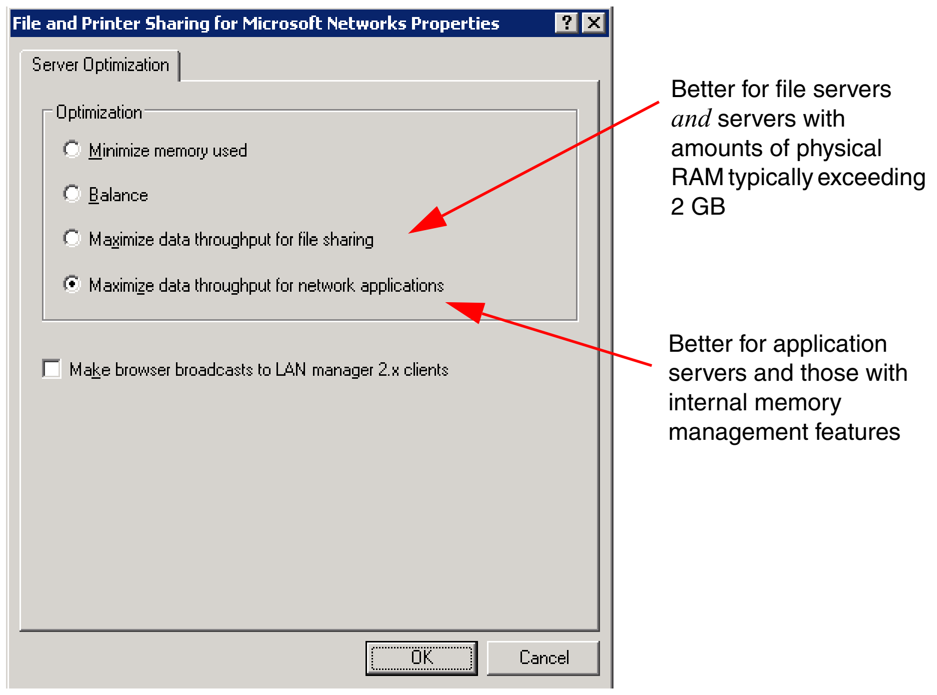
The selection made in these dialogs will depend on the intended server function.

The configuration options are:

- File and Printer Sharing for Microsoft Networks (Network Control Panel applet)
- System (System Control Panel applet), also referred to in Microsoft documentation as "Performance Options" as it has consolidated several performance applets into one location.

The File and Printer Sharing dialog can be accessed as follows:

1. Click **Start** → **Control Panel** → **Network Connections**.
2. While still in the Start menu context, right-click Network Connections and choose **Open**.
3. Select any of Local Area Connections including any teamed network interface. This setting affects all LAN interfaces, so which LAN connection you choose in the above steps is not important.
4. Right-click the selected connection object and choose **Properties**.
5. Select **File and Printer Sharing for Microsoft Networks**.
6. Click **Properties**. The window that is shown in Figure 3 opens.


File and Printer Sharing for Microsoft Networks Properties

Server Optimization

Optimization

- ○ Minimize memory used
- ○ Balance
- ○ Maximize data throughput for file sharing
- ● Maximize data throughput for network applications

☐ Make browser broadcasts to LAN manager 2.x clients

OK Cancel

Better for file servers *and* servers with amounts of physical RAM typically exceeding 2 GB

Better for application servers and those with internal memory management features

Figure 3 File and Print Sharing for Microsoft Networks applet: server optimization

The four options here modify two registry entries:

- HKLM\System\CurrentControlSet\Services\LanmanServer\Parameters\Size
- HKLM\System\CurrentControlSet\Control\Session Manager\Memory Management\LargeSystemCache

The value of the registry entries will be set depending on the option selected in the control panel as shown in Table 4.

Table 4 Registry effect of Server Optimization option selected

Server optimization option selected	LanmanServer Size	LargeSystemCache
Minimize memory used	1	0
Balance	2	0
Maximize data throughput for file sharing	3	1
Maximize data throughput for network applications	3	0

These values are the same for both the 32-bit (x86) and 64-bit (x64) editions of Windows Server 2003.

The file system cache has a working set of memory like any other process. The option chosen in this dialog effectively determines how large the working set is allowed to grow to and with what priority the file system cache is treated by the operating system relative to other applications and processes running on the server.

Typically only one of the bottom two options in the control panel is employed for an enterprise server implementation and are thus the only two detailed here:

- **Maximize throughput for file sharing**

  This option is the default setting. It instructs the operating system to give the working set of the file system cache a higher priority for memory allocation than the working sets of applications. It will yield the best performance in a file server environment that is not running other applications.

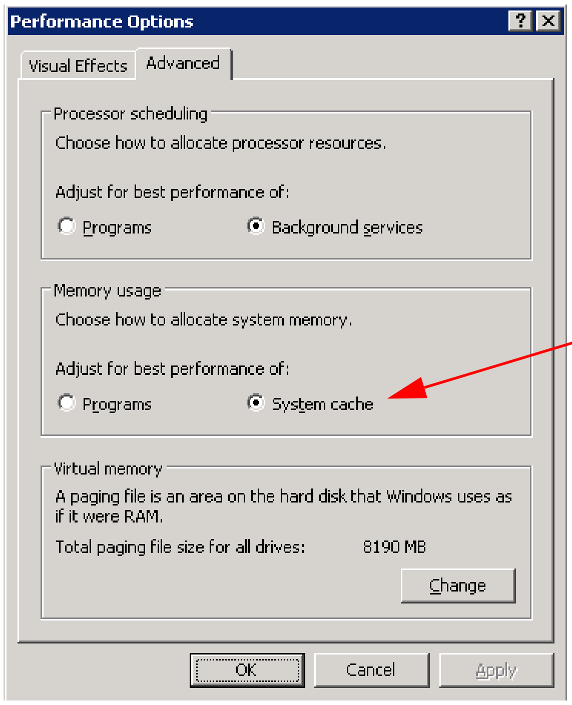
  if other applications are running on the server, it will require sufficient physical memory to obtain the maximum performance gain as more memory is set aside for the file system cache than for applications. As a result the “maximize throughput for network applications” is typically used. This “file sharing” option might, however, be the best option to use on servers with large quantities of physical memory as described below.

- **Maximize throughput for network applications**

  This choice is the recommended setting for machines running applications that are memory-intensive. With this option chosen, the working set of applications will have a priority over the working set of the file system cache. This setting is normally the best setting to use for all servers except those with (a) dedicated file servers or with applications exhibiting file server-like characteristics or (b) those with significant amounts of memory (see below).

The second control panel used for managing the file system cache in Windows Server 2003 is within the System applet:

1. Click **Start** → **Control Panel** → **System**.
2. Select the **Advanced** tab.
3. Within the **Performance** frame, click **Settings**.
4. Select the **Advanced** tab. The window shown in Figure 4 on page 14 opens.


Performance Options

Visual Effects | Advanced

Processor scheduling

Choose how to allocate processor resources.

Adjust for best performance of:

( ) Programs [✓ Background services]

Memory usage

Choose how to allocate system memory.

Adjust for best performance of:

( ) Programs [✓ System cache]

Virtual memory

A paging file is an area on the hard disk that Windows uses as if it were RAM.

Total paging file size for all drives: 8190 MB

Change
OK
Cancel
Apply
Memory optimization settings for the file system cache are controlled here

Figure 4 System applet: memory usage

The System applet reflects and modifies the value included within the same LanmanServer "**LargeSystemCache**" registry key as described above for File and Printer Sharing in the Network applet in Figure 3 on page 12. Making a change to the LargeSystemCache through this applet however does so without impacting the MemoryManagement "Size" value that File and Printer Sharing does.

Given that most users only use the **Maximize throughput for network applications** or the **Maximize throughput for file sharing** options for enterprise servers, the **Size** value remains same, a value of 3. This setting means that using the System applet to adjust the LargeSystemCache value is redundant as it is just as easily set using File and Print Sharing. As a result we recommend using the first control panel as described above and leave this second control panel untouched.

It would seem that the only advantage to using both Control Panel applets in conjunction would allow you to have the applets actually indicate **Maximize throughput for network applications** and simultaneously indicate memory usage favors **System cache**. This same effect to the registry is achieved by selecting **Maximize throughput for file-sharing** (as per Table 4 on page 13)—visually, it simply does not say "Maximize throughput for network applications." If you do desire this change purely for aesthetic reasons then make sure you set the first Network applet *before* the second System applet as the first overrides the second selections, but the reverse *does not* occur.

Servers with large amounts of free physical memory

How much memory is able to be allocated to the file system cache depends on how much physical memory exists in the server and the file system cache option selected in the dialogs above.

With Windows Server 2003, when **Maximize data throughput for file sharing** is selected (LargeSystemCache set to "1"), the maximum the file system cache can grow to is 960 MB. When **Maximize throughput for network applications** is selected (LargeSystemCache set to “0”) the maximum the file system cache can grow to is 512 MB. (See Microsoft KB 837331, URL below) Depending on the selection made here it is possible that adding more physical memory up to point will allow the file system cache to grow even larger, up to these stated maximums.

On a server with physical memory from say 2 GB and upwards, it might be preferable to leave **Maximize data throughput for file sharing** option selected. That is, providing the total amount of memory used by the operating system and server applications does not exceed the amount of physical RAM minus 960 MB. In fact any application server that can be determined to have 960 MB or more of RAM unused will likely be given a performance boost by enabling the large system cache.

By enabling this, all the disk and network I/O performance benefits of using a large file system cache are realized and the applications running on the server continue to run without being memory constrained.

Some applications have their own memory management optimizers built into them, including Microsoft SQL Server and Microsoft Exchange. In such instances, the setting above is best set to **Maximize throughput for network applications** and let the application manage memory and their own internal system cache as it sees appropriate.

See Microsoft Knowledge Base article 837331 for more information:

`http://support.microsoft.com/?kbid=837331`

Note well that the maximum size of the file system cache increases from 960 MB in the 32-bit (x86) edition of Windows Server 2003 to 1 TB in the 64-bit (x64) editions. This has potential to yield enormous performance improvements on systems where the file system cache is actively used.

Disabling or removing unnecessary services

When Windows is first installed, many services are enabled that might not be necessary for a particular server. While in Windows Server 2003 many more services are disabled by default than in previous editions of the server operating system, there still remains on many systems an opportunity for improving performance further by examining running services.

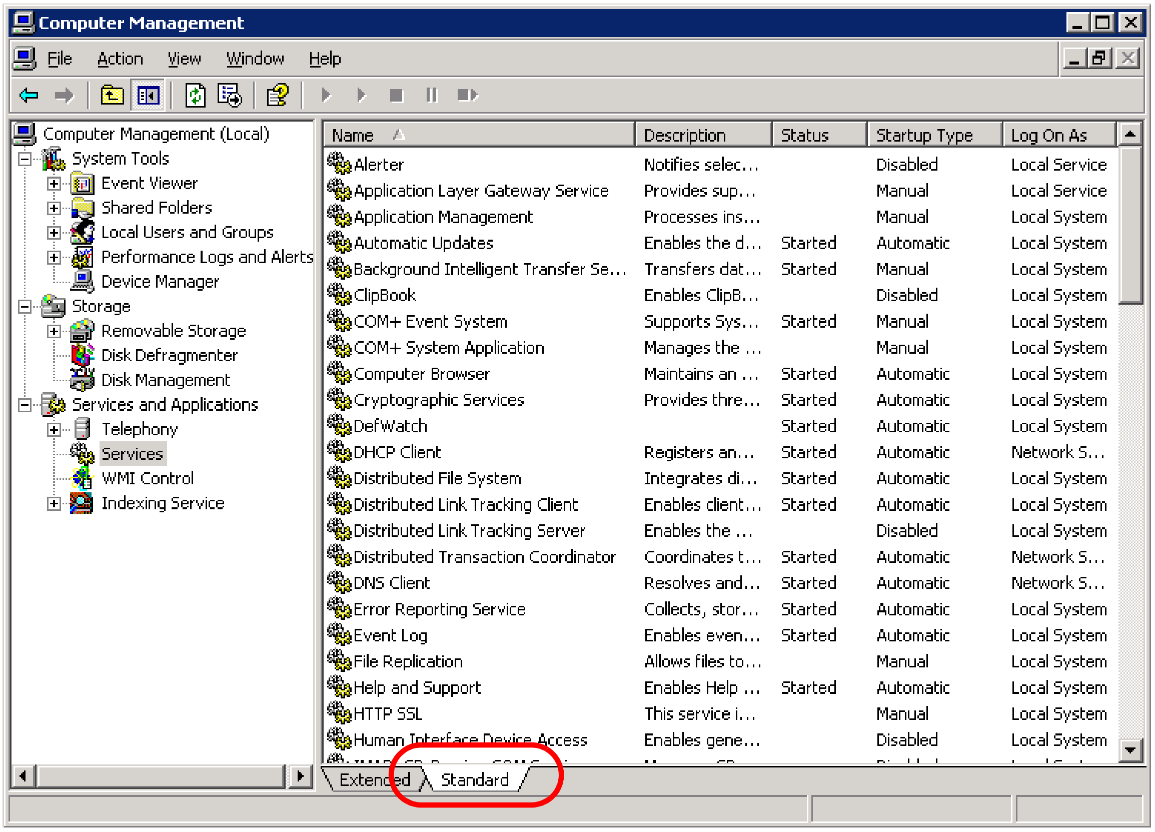
Inexperienced users might also inadvertently add additional services when installing or updating the operating system that are not actually required for a given system. Each service requires system resources and as a result is best to disable unnecessary services to improve performance.

Care does need to be taken when disabling services. Unless you are completely certain of the purpose of a given service it is recommended to research it further before choosing to disable it. Disabling some services that the operating system requires to be running can render a system inoperable and possibly unable to boot.

To view the services running in Windows, complete the following steps:

1. Right-click **My Computer** and select **Manage**.
2. Expand the Services and Applications icon.
3. Select the Services icon.
4. Click the Standard tab at the bottom of the right-pane. A window similar to Figure 5 on page 16 opens. All the services installed on the system are displayed. The status, running or not, is shown in the third column.

5. Click twice on **Status** at the top of the third column shown. This sorts together all running (Started) services from those that are not running.



Name	Description	Status	Startup Type	Log On As
Alerter	Notifies selec...		Disabled	Local Service
Application Layer Gateway Service	Provides sup...		Manual	Local Service
Application Management	Processes ins...		Manual	Local System
Automatic Updates	Enables the d...	Started	Automatic	Local System
Background Intelligent Transfer Se...	Transfers dat...	Started	Manual	Local System
ClipBook	Enables ClipB...		Disabled	Local System
COM+ Event System	Supports Sys...	Started	Manual	Local System
COM+ System Application	Manages the ...		Manual	Local System
Computer Browser	Maintains an ...	Started	Automatic	Local System
Cryptographic Services	Provides thre...	Started	Automatic	Local System
DefWatch		Started	Automatic	Local System
DHCP Client	Registers an...	Started	Automatic	Network S...
Distributed File System	Integrates di...	Started	Automatic	Local System
Distributed Link Tracking Client	Enables client...	Started	Automatic	Local System
Distributed Link Tracking Server	Enables the ...		Disabled	Local System
Distributed Transaction Coordinator	Coordinates t...	Started	Automatic	Network S...
DNS Client	Resolves and...	Started	Automatic	Network S...
Error Reporting Service	Collects, stor...	Started	Automatic	Local System
Event Log	Enables even...	Started	Automatic	Local System
File Replication	Allows files to...		Manual	Local System
Help and Support	Enables Help ...	Started	Automatic	Local System
HTTP SSL	This service i...		Manual	Local System
Human Interface Device Access	Enables gene...		Disabled	Local System

Figure 5 Windows Services

From this dialog, all services that are not required to be running on the server should be stopped and disabled. This will prevent the service from automatically starting at system boot time. To stop and disable a service, do the following:

1. Right-click the service and click **Properties**.
2. Click **Stop** and set the **Startup type** to **Disabled**.
3. Click **OK** to return to the Services window.

If a particular service has been installed as part an application or Windows component and is not actually required on a given server, a better approach is to remove or uninstall this application or component altogether. This is typically performed through the **Add or Remove Programs** applet in Control Panel.

Some services might not be required at system boot time but might be required to start by other applications or services at a later time. Such services should be set to have a startup type of **Manual.** Unless a service is explicitly set to have a startup type of **Disabled**, it can start at any time and perhaps unnecessarily use system resources.

Windows Server 2003 comes installed with many services that Windows 2000 Server and Windows NT Server did not. Designed as a significantly more secure operating system than its predecessors, many of the services have their startup type set to Disabled or Manual by default. Nonetheless, there remains several services enabled on a standard installation that can likely be disabled on many servers. For example, the Print Spooler service is enabled by default but is not usually required if the server is not functioning as a print server or does not have local printing requirements.

Table 5 lists services on a standard Windows Server 2003 installation that should be reviewed for their requirement on your systems. This is not a definitive list of all services, just those that should be considered for their applicability on an enterprise server. Note well that this list includes only those services that are not already disabled by default on Windows Server 2003 and might be candidates for disabling.

These services might still be required for your environment, depending on the particular function of the server and the applications it is running. For example, the File Replication service (FRS) is normally required on an Active Directory® domain controller, but its inclusion with other server types should be questioned. Each server is different and implementing the following recommendations should be tested before changing.

Table 5 Windows service startup recommendations

Service	Default startup type	Recommended setting
Application Management	Manual	Disabled
Alerter	Automatic	Disabled
Clipbook	Disabled	Disabled
Computer Browser	Automatic	Disabled
Distributed file system	Automatic	Disabled
Distributed link tracking client	Automatic	Disabled
Distributed transaction coordinator	Automatic	Manual
Error Reporting Service	Automatic	Disabled
Fax Service	Manual	Disabled
File Replication	Manual	Disabled
Help and Support	Automatic	Disabled
HTTP SSL	Manual	Disabled
License Logging	Manual	Disabled
Logical Disk Manager	Automatic	Manual
Messenger	Automatic	Disabled
Portable Media Serial Number Service	Manual	Disabled
Shell Hardware Detection	Automatic	Disabled
Windows Audio	Automatic	Disabled
Wireless Configuration	Automatic	Disabled

Removing unnecessary protocols and services

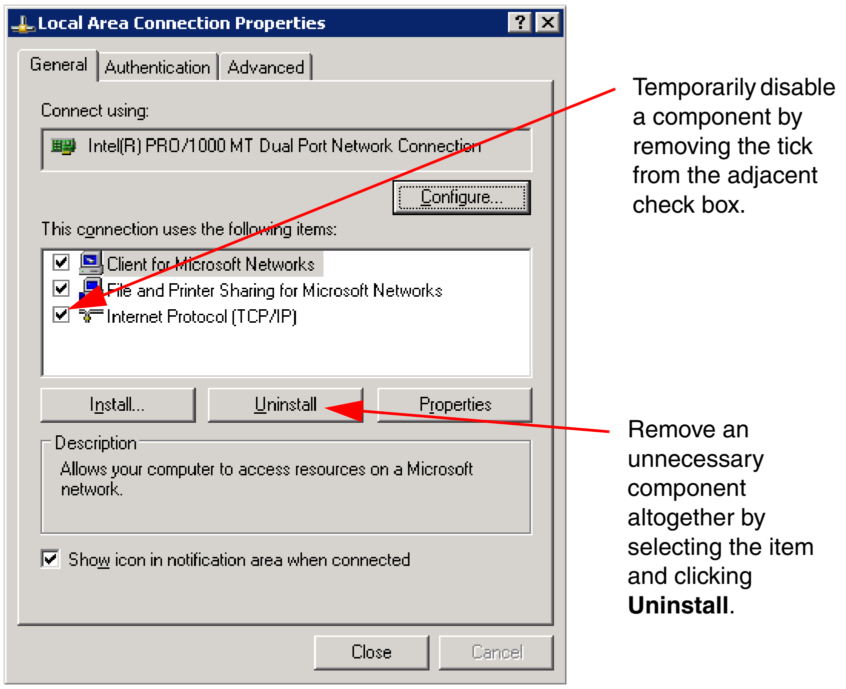
Windows servers often have more network services and protocols installed than are actually required for the purpose or application for which they have been implemented. Each additional network client, service or protocol places additional overhead on system resources.

In addition, each protocol generates network traffic. By removing unnecessary network clients, services and protocols, system resources are made available for other processes, excess network traffic is avoided and the number of network bindings that must be negotiated is reduced to a minimum.

TCP/IP is largely viewed as the *de facto* enterprise network protocol in modern networks. Unless integration with other systems is required it is likely sufficient now to just have TCP/IP loaded as the only network protocol on your server.

To view the currently installed network clients, protocols and services:

1. Click **Start** → **Control Panel** → **Network Connections**.
2. While still in the Start menu context, right-click **Network Connections** and choose **Open**.
3. Click **Properties**.
4. Right-click **Local Area Connection** (or the entry for your network connection).
5. Click **Properties.** The window shown in Figure 6 opens.


Local Area Connection Properties
General
Authentication
Advanced
Configure...
Install...
Uninstall
Properties
Close
Cancel
Temporarily disable a component by removing the tick from the adjacent check box.
Remove an unnecessary component altogether by selecting the item and clicking **Uninstall**.

Connect using:
Intel(R) PRO/1000 MT Dual Port Network Connection

This connection uses the following items:
- [✓] Client for Microsoft Networks
- [✓] File and Printer Sharing for Microsoft Networks
- [✓] Internet Protocol (TCP/IP)

Description
Allows your computer to access resources on a Microsoft network.

[✓] Show icon in notification area when connected

Figure 6 Network clients, services and protocols

To remove an unnecessary item, select it and click **Uninstall.** To **disable** the item temporarily without completely uninstalling it, simply remove the tick from the check box beside it. This latter approach (disable rather than uninstall) might be a more appropriate method in determining which services, protocols and clients are actually required on a system. When is has been determined that disabling an item has no adverse affect on the server, it can then be uninstalled.

In many instances, the three components listed in Figure 6 are often sufficient for a file and print server on a standard TCP/IP based network. That is:

- Client for Microsoft Networks
- File and Printer Sharing for Microsoft Networks
- Internet Protocol (TCP/IP)

Optimizing the protocol binding and provider order

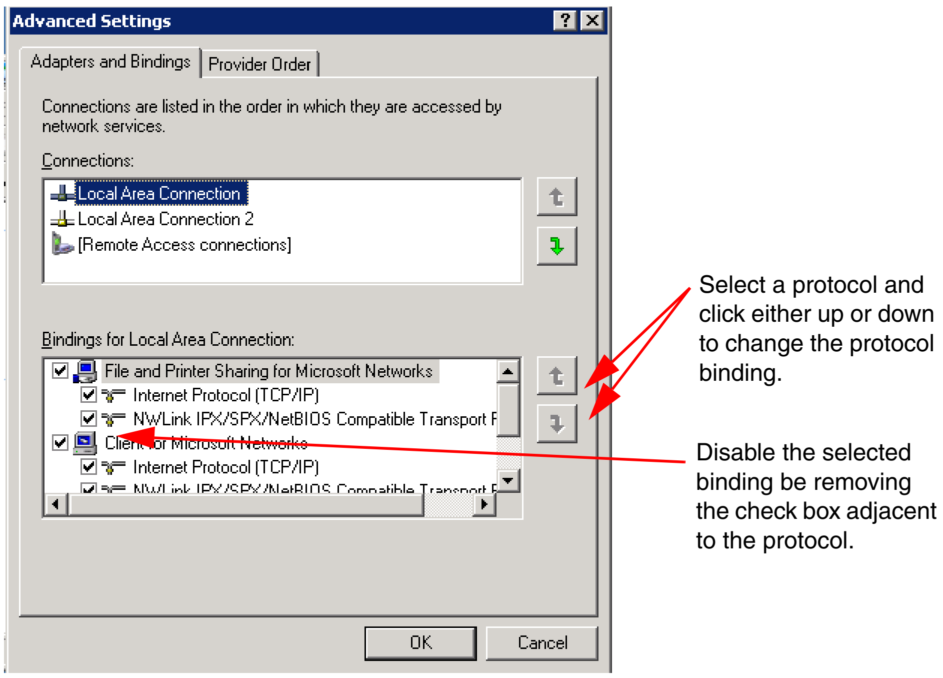
Optimizing the protocol order and the provider order can also make a difference to performance.

Protocol binding order

On a system supporting more than one network protocol, the order in which they are bound to the network clients and services running on the server is important. All network communications for a given service or client start with the protocol listed at the top of the binding list. If after a given period, no response is received, communications are routed to the next protocol in the list until all protocols are exhausted. As a result it is crucial to ensure the most frequently used protocol for a given client or service is moved to the top of the binding list to offer the best network I/O performance possible.

To view the order of network bindings, do the following:

1. Click **Start** → **Control Panel** → **Network Connections**.
2. While still in the Start menu context, right-click Network Connections and choose **Open**.
3. Click **Properties**.
4. From the menu bar, click **Advanced** → **Advanced Settings**. The window shown in Figure 7 opens.


Advanced Settings
Adapters and Bindings
Provider Order
Connections:
Local Area Connection
Local Area Connection 2
[Remote Access connections]
Bindings for Local Area Connection:
File and Printer Sharing for Microsoft Networks
Internet Protocol (TCP/IP)
NWLink IPX/SPX/NetBIOS Compatible Transport P
Client for Microsoft Networks
Internet Protocol (TCP/IP)
NWLink IPX/SPX/NetBIOS Compatible Transport
OK
Cancel

Connections are listed in the order in which they are accessed by network services.

Select a protocol and click either up or down to change the protocol binding.

Disable the selected binding be removing the check box adjacent to the protocol.

Figure 7 Windows protocol binding order

By selecting a protocol and clicking the up and down buttons, you can change the binding priority of your protocols.

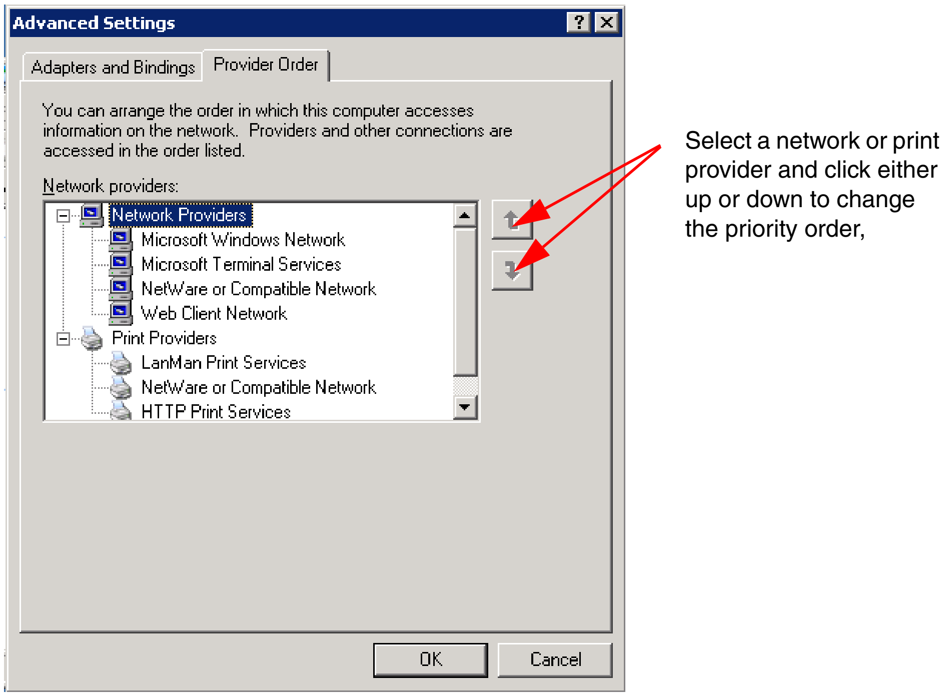
If an installed protocol is not required by a particular service or client, it should be disabled. Do so by removing the tick in the check box beside the protocol in question. This will improve system performance and possibly improve security.

Network and print provider order

Servers will often have multiple network and print providers installed. Similar to network bindings, the order in which these are configured will determine how quickly they respond to client requests for services running on the server. It will also affect how quickly the server itself connects to hosts when functioning as a client. The most commonly used network providers should be moved to the top of the list with the remaining ones ordered down in order of decreasing priority.

To access the network provider order configuration:

1. Click **Start** → **Control Panel** → **Network Connections**.
2. While still in the Start menu context, right-click **Network Connections** and choose **Open**.
3. Click **Properties**.
4. From the menu bar, click **Advanced** → **Advanced Settings**.
5. Select the **Network Provider** tab. The window shown in Figure 8 opens.


Advanced Settings
Adapters and Bindings
Provider Order
Network providers:
OK
Cancel
Select a network or print provider and click either up or down to change the priority order,

You can arrange the order in which this computer accesses information on the network. Providers and other connections are accessed in the order listed.

- Network Providers
  - Microsoft Windows Network
  - Microsoft Terminal Services
  - NetWare or Compatible Network
  - Web Client Network
- Print Providers
  - LanMan Print Services
  - NetWare or Compatible Network
  - HTTP Print Services

Figure 8 Windows network provider order

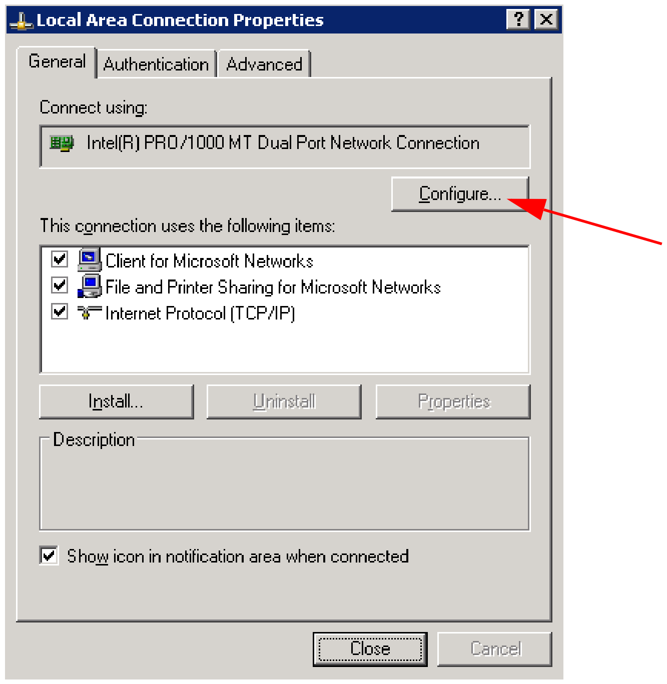
By selecting a network or print provider and clicking the up and down buttons, you can change the order in which the computer responds to client requests.

Optimizing network card settings

Many network interface cards in servers today have settings that can be configured through the Windows interface. Setting these optimally for your network environment and server configuration can significantly affect the performance of network throughput. Of all the performance tuning features outlined in this chapter, it is the ones in this section that have been noted to have the biggest improvement on system performance and throughput.

To access this range of settings, following these steps:

1. Click **Start** → **Settings** → **Network Connections**.
2. Click **Properties**.
3. Right-click **Local Area Connection** (or the name of your network connection).
4. Click **Properties**. The window shown in Figure 9 opens.


Local Area Connection Properties

General | Authentication | Advanced

Connect using:

Intel(R) PRO/1000 MT Dual Port Network Connection

This connection uses the following items:

- [✓] Client for Microsoft Networks
- [✓] File and Printer Sharing for Microsoft Networks
- [✓] Internet Protocol (TCP/IP)

Description

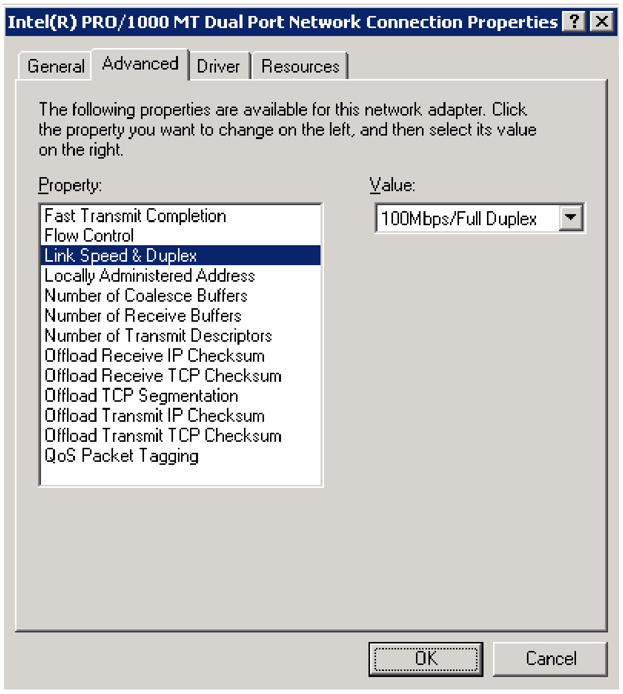
[✓] Show icon in notification area when connected

Click **Configure** to access the configuration settings available for the network interface card.

Figure 9 Accessing the network interface card configuration

5. Click **Configure**.

6. Click the **Advanced** tab. A dialog box similar to that in Figure 10 opens, depending on the network adapter your system is using.


Intel(R) PRO/1000 MT Dual Port Network Connection Properties
General
Advanced
Driver
Resources
OK
Cancel

The following properties are available for this network adapter. Click the property you want to change on the left, and then select its value on the right.

Property:

Fast Transmit Completion
Flow Control
Link Speed & Duplex
Locally Administered Address
Number of Coalesce Buffers
Number of Receive Buffers
Number of Transmit Descriptors
Offload Receive IP Checksum
Offload Receive TCP Checksum
Offload TCP Segmentation
Offload Transmit IP Checksum
Offload Transmit TCP Checksum
QoS Packet Tagging

Value: 100Mbps/Full Duplex

Figure 10 Network interface card advanced settings configuration

The exact configuration settings available differ from one network interface card to another. However, a handful of settings are common between most Intel-based cards in the IBM® System x™ range of servers.

Note: You apply these settings for each physical network interface, including the individual cards within a set *teamed* of interfaces that are configured for aggregation, load balancing, or fault tolerance. With some teaming software, you might need to apply these settings to the team also. Note also that some network interface cards are largely self-tuning and do not offer the option to configure parameters manually.

The following settings are the ones that can have the most dramatic impact to performance:

- **Link Speed and Duplex**

  Experience suggests that the best practice for setting the speed and duplex values for each network interface in the server is to configure them in one of two ways:

  – Set to auto-negotiation if, and only if, the switch port is also set to auto negotiation also. The server and switch should then negotiate the fastest possible link speed and duplex settings.

  – Set to the same link speed and same duplex settings as those of the switch. These settings will, of course, normally yield the best performance if set to the highest settings that the switch will support.

  We do *not* recommend the use of auto-negotiation the server network interface combined with manually setting the parameter on the switch, or vice-versa. Using such a combination of settings at differing ends of the network connection to the server has often found to be the culprit of poor performance and instability in many production environments and should definitely be avoided.

To repeat, use either auto-negotiation at both interfaces, or hard-code the settings at both interfaces, but not a mix of both of these.

For more information, see the following Cisco Web site:

`http://www.cisco.com/warp/public/473/46.html#auto_neg_valid`

- **Receive Buffers**

  This setting specifies the number of memory buffers used by the network interface driver when copying data to the protocol memory. It is normally set by default to a relatively low setting. We recommend setting this value as high as possible for maximum performance gains. On servers low on physical memory, this can have a negative impact as these buffers are taken from available physical memory on the system. On most modern systems however, the maximum setting can be implemented without any notable impact to memory resources.

  The amount of memory used by modifying this setting can easily be determined by watching the appropriate metrics in the Task Manager or System Monitor before and after making the changes. Monitor this impact before making the change permanent.

- **Coalesce Buffers**

  Map registers are system resources used in physical to virtual address conversion with bus mastering cards like the ones in some IBM System X servers. Coalesce buffers are those available to the network driver if the driver runs out of map registers. We recommend setting this value as high as possible for maximum performance gains. Note the same impact to memory as with receive buffers is possible when increasing the number of coalesce buffers.

- **Transmit Descriptors / Transmit Control Blocks**

  This setting specifies how many transmit control buffers the driver allocates for use by the network interface. This directly reflects the number of outstanding packets the driver can have in its "send" queue. We recommend setting this value as high as possible for maximum performance gains. Note the same impact to memory as with receive buffers is possible when increasing the number of transmit descriptors / transmit control blocks.

- **Offload features**

  In almost all instances there will be benefit derived from enabling network interface offload features. In some instances the network interface might *not* be able to handle the offload capabilities at high throughput however as a general rule enabling offload his will benefit overall system performance. Some network interfaces have separate options or parameters to enable or disable offloading for send and receive traffic.

Other advanced settings are often available with network interface cards than just those described here. The documentation for the network interface should be consulted to detail the meaning and impact of changing each setting.

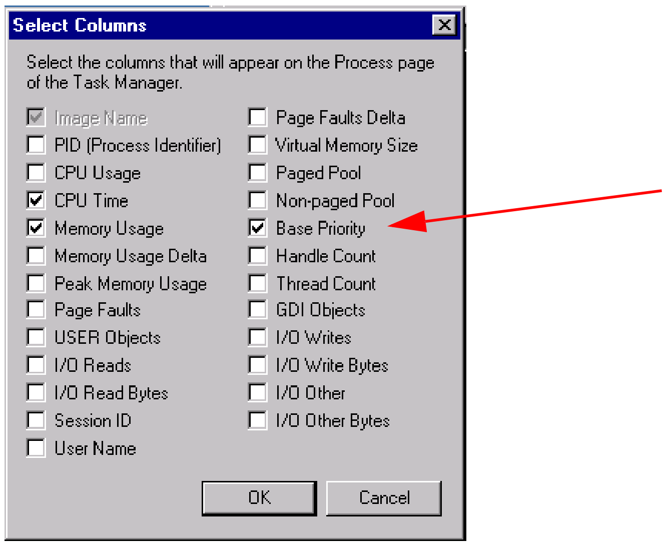
Where possible, use these settings to take network processing requirements away from the server itself and to the network interface. That is, "offload" the network requirements from the server CPU where possible and try to have the network interface do as much of the processing as it can. This will ensure optimal performance.

Process scheduling, priority levels, and affinity

The scheduler is a component of the Windows operating system kernel. It is the scheduler that coordinates the servicing of the processes and their threads waiting and ready to use the system CPUs. The kernel schedules ready threads based upon their individual *dynamic priority*. The dynamic priority is a number between 0 and 31 that determines the importance of threads relative to one another. The higher the priority value, the higher the priority level. For example, a thread with a priority of 15 is serviced more quickly than a thread with a priority of 10.

Even if it requires preempting a thread of lower priority, threads with the highest priority always run on the processor. This activity ensures Windows still pays attention to critical system threads required to keep the operating system running. A thread will run on the processor for either the duration of its CPU quantum (or time slice, described in , “Windows Server 2003, 64-bit (x64) Editions” on page 3) or until it is preempted by a thread of higher priority.

Task Manager allows you to easily see the priority of all threads running on a system. To do so, open Task Manager, and click **View** → **Select Columns**, then select **Base Priority,** as shown in Figure 11.



Select the columns that will appear on the Process page of the Task Manager.

- [✓] Image Name
- [ ] PID (Process Identifier)
- [ ] CPU Usage
- [✓] CPU Time
- [✓] Memory Usage
- [ ] Memory Usage Delta
- [ ] Peak Memory Usage
- [ ] Page Faults
- [ ] USER Objects
- [ ] I/O Reads
- [ ] I/O Read Bytes
- [ ] Session ID
- [ ] User Name
- [ ] Page Faults Delta
- [ ] Virtual Memory Size
- [ ] Paged Pool
- [ ] Non-paged Pool
- [✓] Base Priority
- [ ] Handle Count
- [ ] Thread Count
- [ ] GDI Objects
- [ ] I/O Writes
- [ ] I/O Write Bytes
- [ ] I/O Other
- [ ] I/O Other Bytes

Select **Base Priority** to ensure that you can see the priority of all running processes.

Figure 11 Selecting Base Priority in Task Manager

This displays a column in Task Manager as shown in Figure 12 that allows you to see the relative priority of processes running on the system.

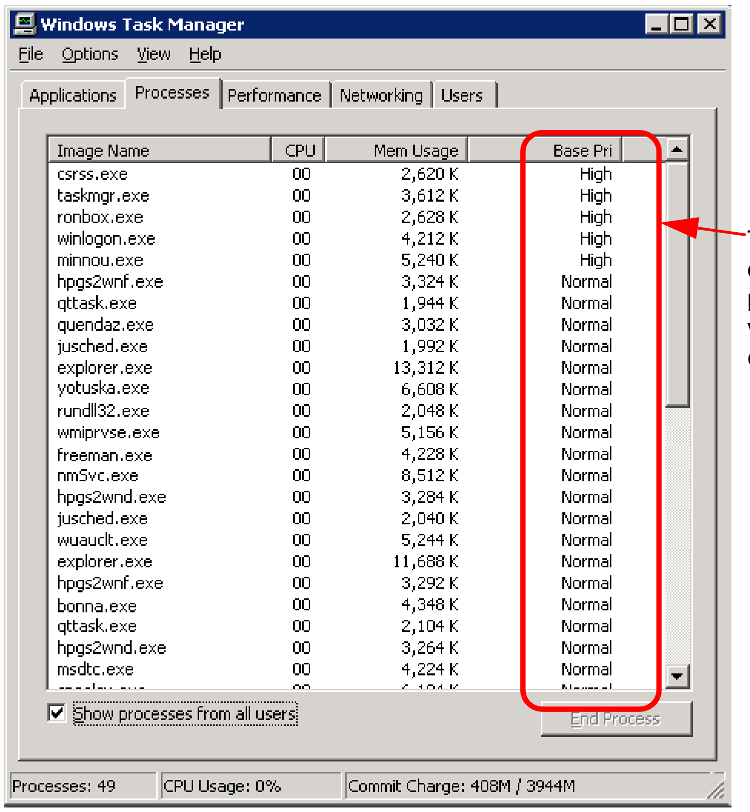

Windows Task Manager
File Options View Help
Applications Processes Performance Networking Users
Show processes from all users
End Process

Image Name	CPU	Mem Usage	Base Pri
csrss.exe	00	2,620 K	High
taskmgr.exe	00	3,612 K	High
ronbox.exe	00	2,628 K	High
winlogon.exe	00	4,212 K	High
minnou.exe	00	5,240 K	High
hpgs2wnf.exe	00	3,324 K	Normal
qttask.exe	00	1,944 K	Normal
quendaz.exe	00	3,032 K	Normal
jusched.exe	00	1,992 K	Normal
explorer.exe	00	13,312 K	Normal
yotuska.exe	00	6,608 K	Normal
rundll32.exe	00	2,048 K	Normal
wmiprvse.exe	00	5,156 K	Normal
freeman.exe	00	4,228 K	Normal
nmSvc.exe	00	8,512 K	Normal
hpgs2wnd.exe	00	3,284 K	Normal
jusched.exe	00	2,040 K	Normal
wuauclt.exe	00	5,244 K	Normal
explorer.exe	00	11,688 K	Normal
hpgs2wnf.exe	00	3,292 K	Normal
bonna.exe	00	4,348 K	Normal
qttask.exe	00	2,104 K	Normal
hpgs2wnd.exe	00	3,264 K	Normal
msdtc.exe	00	4,224 K	Normal

Processes: 49   CPU Usage: 0%   Commit Charge: 408M / 3944M

The Base Priority column shows process priority values relative to each other.

Figure 12   Windows Task Manager displaying the Base Priority column

Most applications loaded by users run at a *normal* priority which has a base priority value of 8. Task Manager also allows the administrator the ability to change the priority of a process, either higher or lower.

To do so, right-click the process in question, and click **Set Priority** from the drop-down menu as shown in Figure 13 on page 26. Then click the new priority that you want to assign to the process.

Note: This procedure changes the priority of actual processes running on the system, but the change only lasts as long as the life of the selected process.

If you want to launch a process with a non-normal priority, you can do so using the START command from a command-prompt. Type `START /?` for more information about how to do this.

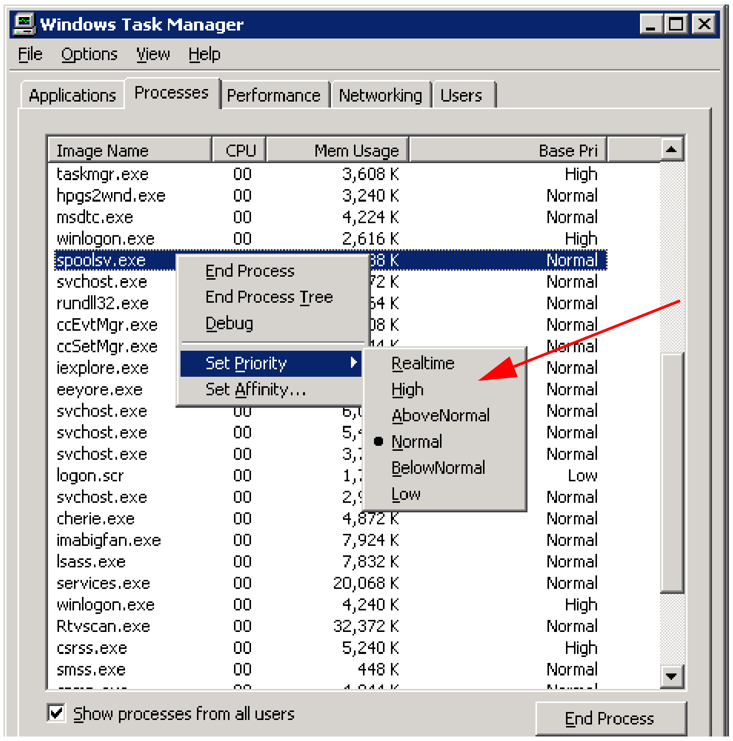


Image Name	CPU	Mem Usage	Base Pri
taskmgr.exe	00	3,608 K	High
hpgs2wnd.exe	00	3,240 K	Normal
msdtc.exe	00	4,224 K	Normal
winlogon.exe	00	2,616 K	High
spoolsv.exe		38 K	Normal
svchost.exe		72 K	Normal
rundll32.exe		54 K	Normal
ccEvtMgr.exe		08 K	Normal
ccSetMgr.exe		[illegible]	Normal
iexplore.exe			Normal
eeyore.exe			Normal
svchost.exe	00	6,0	Normal
svchost.exe	00	5,4	Normal
svchost.exe	00	3,7	Normal
logon.scr	00	1,7	Low
svchost.exe	00	2,9	Normal
cherie.exe	00	4,872 K	Normal
imabigfan.exe	00	7,924 K	Normal
lsass.exe	00	7,832 K	Normal
services.exe	00	20,068 K	Normal
winlogon.exe	00	4,240 K	High
Rtvscan.exe	00	32,372 K	Normal
csrss.exe	00	5,240 K	High
smss.exe	00	448 K	Normal

Task Manager allows you to **change the priority** of a given process from six options.

Figure 13 Changing the priority of a process using Task Manager

Threads, as a sub-component of processes, inherit the base priority of their parent process. The four priority classes are:

- Idle
- Normal
- High
- Realtime

Each process's priority class sets a range of priority values (between 1 and 31) and the threads of that process have a priority within that range. If the priority class is Realtime (priorities 16 to 31), the thread's priority can never change while it is running. A single thread running at priority 31 will prevent all other threads from running.

Conversely, threads running in all other priority classes are variable, meaning the thread's priority can change while the thread is running. For threads in the Normal or High priority classes (priorities 1 through 15), the thread's priority can be raised or lowered by up to a value of 2 but cannot fall below its original, program-defined base priority.

When should you modify the priority of a process? In most instances, you should do this as rarely as possible. Windows normally does a very good job of scheduling processor time to threads. Changing process priority is not an appropriate long-term solution to a bottleneck on a system. If you are suffering performance problems related to processes not receiving sufficient processing time, eventually additional or faster processors will be required to improve the situation.

Normally the only conditions under which the priority of a process should be modified are when the system is CPU-bound. Processor utilization, queue length and context switching can all be measured using System Monitor to help identify processor bottlenecks.

In a system with plenty of spare CPU capacity, testing has shown that changing the base priority of a process offers marginal, if any, performance improvements. This is because the processor is comfortable with the load it is under and able to schedule time appropriately to threads running on the system. Conversely, on a system suffering heavy CPU-load, CPU time being allocated to nominated processes will likely benefit from changing the base priority. On extremely busy systems, threads with the lowest priority will be serviced infrequently, if at all.

Modifying process priorities can be an effective troubleshooting technique for solving short-term performance problems however it is rarely a long-term solution.

Important: Changing priorities might destabilize the system. Increasing the priority of a process might prevent other processes, including system services, from running. In particular, be careful not to schedule many processes with the High priority and avoid using the Realtime priority altogether. Setting a processor-bound process to Realtime could cause the computer to stop responding altogether.

Decreasing the priority of a process might prevent it from running at all, not merely force it to run less frequently. In addition, lowering priority does not necessarily reduce the amount of processor time a thread receives; this happens only if it is no longer the highest-priority thread.

Process affinity

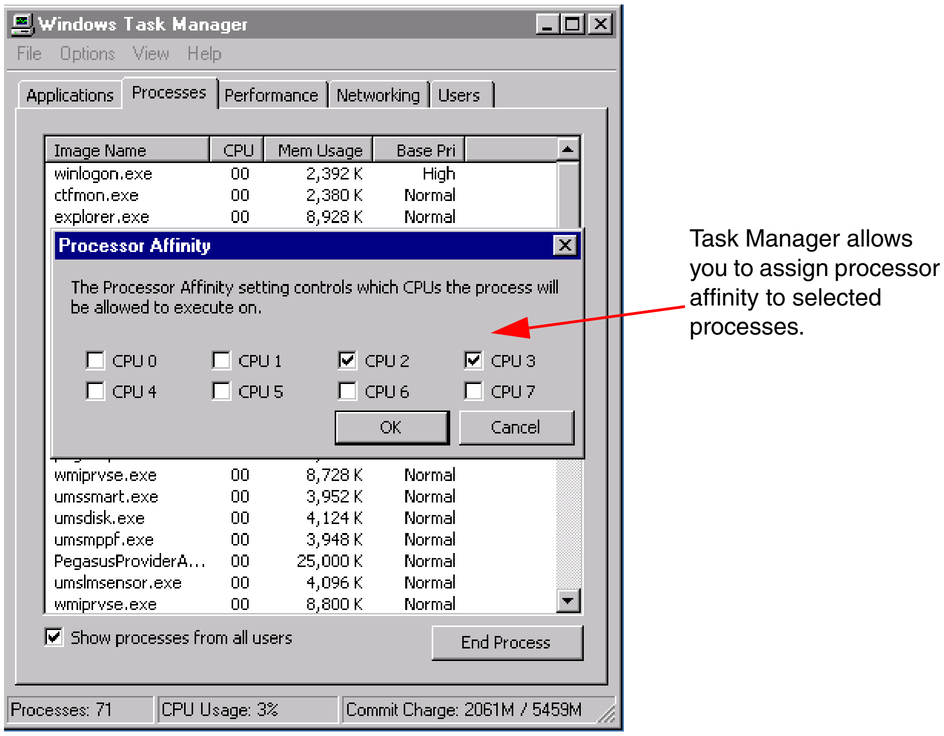
On symmetric multi-processing (SMP) systems, the Windows scheduler distributes the load of ready threads over all available processors based on thread priority. Even though Windows will often try to associate known threads with a specific CPU (called *soft affinity*), threads invariably end up distributed among multiple processors.

Hard affinity can be applied to permanently bind a process to a given CPU or set of CPUs, forcing the designated process to always return to the same processor. The performance advantage in doing this is best seen in systems with large Level 2 caches as the cache hit ratio will improve dramatically.

Assigning hard processor affinity to assign processes to CPUs is not typically used as a method for improving system performance. The only circumstances under which it will occasionally be employed are those servers where multiple instances of the same application are running on the same system, such as SQL Server or Oracle®. As with all tuning techniques, the performance of the system should be measured to determine whether using process affinity has actually offered any tangible benefit. Determining the correct mix of processes assigned to the CPUs in the system can be time-consuming.

Some applications, like SQL Server, provide internal options to assign themselves to specific CPUs. The other method for setting affinity is through Task Manager:

1. Right-click the process in question
2. Click **Set Affinity** as shown in Figure 14 on page 28.
3. Select the CPUs to which you want to restrict the process and click **OK**.



Windows Task Manager

File Options View Help

Applications | Processes | Performance | Networking | Users

```
Image Name          CPU   Mem Usage   Base Pri
winlogon.exe        00     2,392 K    High
ctfmon.exe          00     2,380 K    Normal
explorer.exe        00     8,928 K    Normal
```

Processor Affinity

The Processor Affinity setting controls which CPUs the process will be allowed to execute on.

☐ CPU 0 ☐ CPU 1 ☑ CPU 2 ☑ CPU 3

☐ CPU 4 ☐ CPU 5 ☐ CPU 6 ☐ CPU 7

OK Cancel

```
wmiprvse.exe        00     8,728 K    Normal
umssmart.exe        00     3,952 K    Normal
umsdisk.exe         00     4,124 K    Normal
umsmppf.exe         00     3,948 K    Normal
PegasusProviderA... 00    25,000 K    Normal
umslmsensor.exe     00     4,096 K    Normal
wmiprvse.exe        00     8,800 K    Normal
```

☑ Show processes from all users End Process

Processes: 71 CPU Usage: 3% Commit Charge: 2061M / 5459M

Task Manager allows you to assign processor affinity to selected processes.

Figure 14 Assigning Processor Affinity to a selected process

Note that like changing the process's priority, changing process affinity in this manner will only last for the duration of the process. If the process ends or the system is rebooted, the affinity will need to be reallocated as required. Note also that not all processes permit affinity changes.

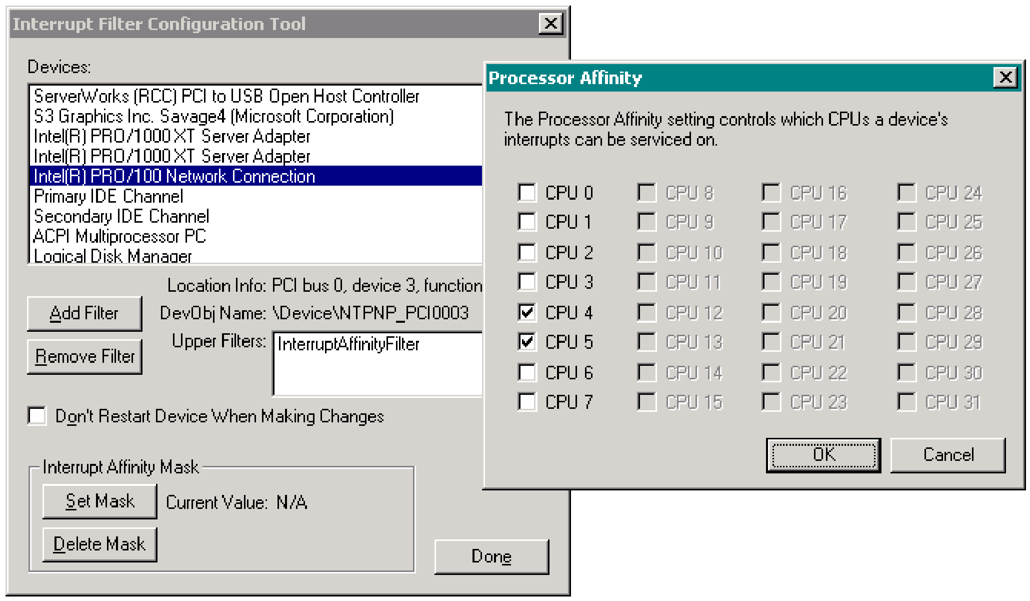
Note too that the **Set Affinity** option described above in the Task Manager application will only display in the context menu on a system with multiple logical or physical processors, including processors with multiple cores.

Assigning interrupt affinity

Microsoft offers a utility called *Intfiltr* that allows the binding of device interrupts to specific system processors. This partitioning technique can be employed to improve system performance, scaling and the partitioning of large servers.

Specifically for server performance tuning purposes, Intfiltr allows you to assign the interrupts generated by each network adapter to a specific CPU. Of course, it is only useful on SMP systems with more than one network adapter installed. Binding the individual network adapters in a server to a given CPU can offer large performance efficiencies.

Intfiltr uses plug-and-play features of Windows that permits affinity for device interrupts to particular processors. Intfiltr binds a *filter* driver to devices with interrupts and is then used to set the affinity mask for the devices that have the filter driver associated with them. This permits Windows to have specific device interrupts associated with nominated processors.


Interrupt Filter Configuration Tool

Devices:

- ServerWorks (RCC) PCI to USB Open Host Controller
- S3 Graphics Inc. Savage4 (Microsoft Corporation)
- Intel(R) PRO/1000 XT Server Adapter
- Intel(R) PRO/1000 XT Server Adapter
- Intel(R) PRO/100 Network Connection
- Primary IDE Channel
- Secondary IDE Channel
- ACPI Multiprocessor PC
- Logical Disk Manager

Location Info: PCI bus 0, device 3, function

DevObj Name: \Device\NTPNP_PCI0003

Upper Filters: InterruptAffinityFilter

Don't Restart Device When Making Changes

Interrupt Affinity Mask — Current Value: N/A

Processor Affinity

The Processor Affinity setting controls which CPUs a device's interrupts can be serviced on.

CPU 0, CPU 1, CPU 2, CPU 3, ✓ CPU 4, ✓ CPU 5, CPU 6, CPU 7, CPU 8 – CPU 31

Figure 15 Assigning processor affinity using the INTFILTR tool

Interrupt filtering can affect the overall performance of your computer, in both a positive and negative manner. Under normal circumstances, there is no easy way to determine which processor is best left to handle specific interrupts. Experimentation and analysis will be required to determine whether interrupt affinity has yielded performance gains. To this end, by default, without tools like Intfiltr, Windows directs interrupts to any available processor.

Note some consideration needs to be made when configuring Intfiltr on a server with CPUs that supports Hyper-Threading to ensure that the interrupts are assigned to the correct physical processors desired, not the logical processors. Assigning interrupt affinity to two logical processors that actually refer to the same physical processor will obviously offer no benefit and can even detract from system performance.

Interrupt affinity for network cards can offer definite performance advantages on large, busy servers with many CPUs. Our recommendation is to trial Intfiltr in a test environment to associate specific interrupts for network cards with selected processors. Note that this test environment should simulate as close to your production environment as possible, including hardware, operating system and application configuration. This will allow you to determine if using interrupt affinity is going to offer a performance advantage for your network interfaces.

Note: You can use Intfiltr to create an affinity between CPUs and devices other than network cards also, such as disk controllers. Experimentation again is the best way to determine potential performance gains. To determine the interrupts of network cards or other devices, use Windows Device Manager or, alternately, run System Information (WINMSD.EXE).

The Intfiltr utility and documentation is available free of charge from Microsoft:

`ftp://ftp.microsoft.com/bussys/winnt/winnt-public/tools/affinity/intfiltr.zip`

For more information, see:

`http://support.microsoft.com/?kbid=252867`

The /3GB BOOT.INI parameter (32-bit x86)

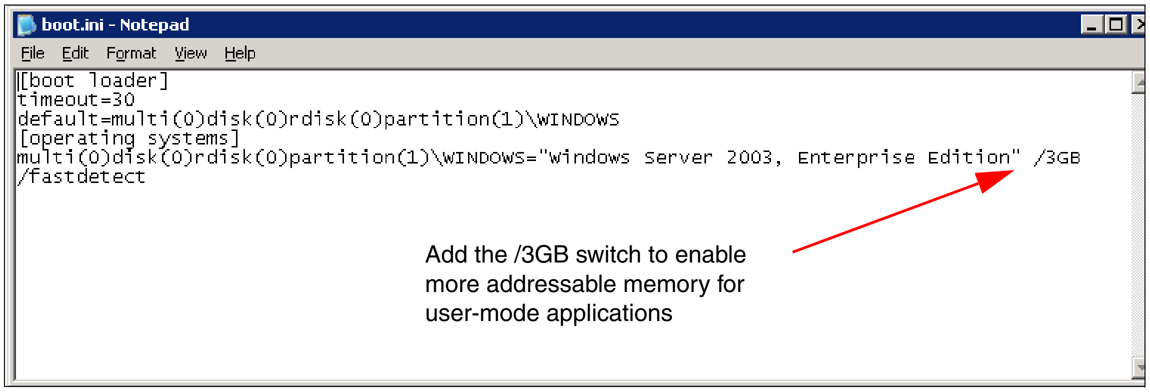
By default, the 32-bit (x86) editions of Windows can address a total of 4 GB of virtual address space. This is a constraint of the 32-bit (x86) architecture. Normally, 2 GB of this is reserved for the operating system kernel requirements (privileged-mode) and the other 2 GB is reserved for application (user-mode) requirements. Under normal circumstances, this creates a 2 GB *per-process* address limitation.

Windows provides a /3GB parameter to be added to the BOOT.INI file that reallocates 3 GB of memory to be available for user-mode applications and reduces the amount of memory for the system kernel to 1 GB. Some applications, such as Microsoft Exchange and Microsoft SQL Server, written to do so, can derive performance benefits from having large amounts of addressable memory available to individual user-mode processes. In such instances, having as much free space for user-mode processes is desirable.

Given the radically increased memory capability of 64-bit (x64) operating systems, the /3GB switch nor the /PAE switch (described in , "Using PAE and AWE to access memory above 4 GB (32-bit x86)" on page 31) are not for use on the 64-bit (x64) editions of the Windows Server 2003 operating system.

To edit the BOOT.INI file to make this change, complete the following steps:

1. Open the System Control Panel.
2. Select the Advanced tab.
3. Within the Startup and Recovery frame, click **Settings**.
4. Click **Edit**. Notepad opens, and you can edit the current BOOT.INI file.
5. Edit the current ARC path to include the /3GB switch, as shown in Figure 16.
6. Restart the server for the change to take effect.


boot.ini - Notepad
File Edit Format View Help

```
[boot loader]
timeout=30
default=multi(0)disk(0)rdisk(0)partition(1)\WINDOWS
[operating systems]
multi(0)disk(0)rdisk(0)partition(1)\WINDOWS="Windows Server 2003, Enterprise Edition" /3GB
/fastdetect
```

Add the /3GB switch to enable more addressable memory for user-mode applications

Figure 16 Editing the BOOT.INI to include the /3GB switch

You normally use this switch only when a specific application recommends its use. Typically, you use it where applications have been compiled to use more than 2 GB per process, such as some components of Exchange.

For more information, see:

http://support.microsoft.com/kb/291988
http://support.microsoft.com/kb/851372
http://support.microsoft.com/kb/823440

Using PAE and AWE to access memory above 4 GB (32-bit x86)

As described in , “The /3GB BOOT.INI parameter (32-bit x86)” on page 30, the native 32-bit architecture of the x86 processor allows a maximum addressable memory space of 4 GB. The Intel Physical Address Extension (PAE) is a 36-bit memory addressing mode that allows 32-bit (x86) systems to address memory above 4 GB.

PAE requires appropriate hardware and operating system support to be implemented. Intel introduced PAE 36-bit physical addressing with the Intel Pentium® Pro processor. Windows has supported PAE since Windows NT Server 4.0, Enterprise Edition and is supported with the Advanced and Datacenter Editions of Windows 2000 Server and the Enterprise and Datacenter Editions of Windows Server 2003.

Windows uses 4 KB pages with PAE to map up to 64 GB of physical memory into a 32-bit (4 GB) virtual address space. The kernel effectively creates a “map” in the privileged mode addressable memory space to manage the physical memory above 4 GB.

The 32-bit (x86) editions of Windows Server 2003 allow for PAE through use of a /PAE switch in the BOOT.INI file. This effectively allows the operating system to use physical memory above 4 GB. As the 64-bit (x64) editions of Windows are not bound by this same memory architecture constraint, the PAE switch is not used in these versions of the Windows Server 2003 operating system.

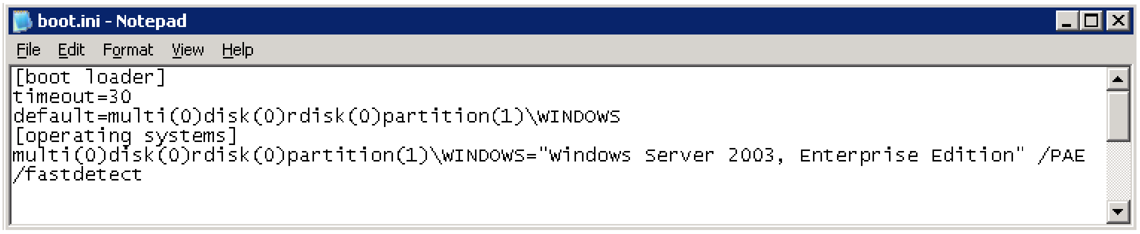
Even with PAE enabled, the underlying architecture of the system is still based on 32-bit linear addresses. This effectively retains the usual 2 GB of application space per user-mode process and the 2 GB of kernel mode space because only 4 GB of addresses are available. However, multiple processes can immediately benefit from the increased amount of addressable memory because they are less likely to encounter physical memory restrictions and begin paging.

Address Windowing Extensions (AWE) is a set of Windows APIs that take advantage of the PAE functionality of the underlying operating system and allow applications to directly address physical memory above 4 GB.

Some applications like SQL Server 2000, Enterprise Edition, have been written with these APIs and can harness the significant performance advantages of being able to address more than 2 GB of memory per process. More recent applications, like SQL Server 2005 x64 Editions, have been written to take advantage of the 64-bit (x64) memory architecture and can deliver enormous performance gains over their 32-bit (x86) counterparts. They do not need to use AWE to address memory above 4 GB.

To edit the BOOT.INI file to enable PAE, complete the following steps:

1. Open the System Control Panel.
2. Select the Advanced tab.
3. Within the Startup and Recovery frame, click **Settings**.
4. Click **Edit**. Notepad opens, and you can edit the current BOOT.INI file.
5. Edit the current ARC path to include the /PAE switch as shown in Figure 17 on page 32.
6. Restart the server for the change to take effect.



```
[boot loader]
timeout=30
default=multi(0)disk(0)rdisk(0)partition(1)\WINDOWS
[operating systems]
multi(0)disk(0)rdisk(0)partition(1)\WINDOWS="Windows Server 2003, Enterprise Edition" /PAE
/fastdetect
```

Figure 17   Editing the BOOT.INI to include the /PAE switch

For more information, see:

http://support.microsoft.com/kb/283037
http://support.microsoft.com/kb/268363
http://support.microsoft.com/kb/823440

Interaction of the /3GB and /PAE switches

There is often confusion between when to use the /3GB switch and when to use the /PAE switch in the BOOT.INI file. In some cases it is desirable to use both.

Recall the following information previously covered:

- The /3GB switch reallocates the maximum 4 GB addressable memory from the normal 2 GB for user-mode applications and 2 GB for the kernel to allow 3 GB of physical memory to be used for applications, leaving 1 GB for the system kernel.
- PAE permits the operating system to see and make use of physical memory above and beyond 4 GB. This is achieved through the use of the 1 or 2 GB of kernel addressable memory (depending on the use of the /3GB switch) to “map” and manage the physical memory above 4 GB.
- Applications written using AWE make use of PAE to allow individual applications (processes) to use more than the 2 GB limitation per process.

On a server with between 4 GB and 16 GB of RAM hosting applications that have been compiled or written with AWE to use more than 2 GB of RAM per process *or* hosting many applications (processes), each contending for limited physical memory, it would be desirable to use both the /3GB and /PAE switches. This will deliver the best performance possible for such a system.

Servers with more than 16 GB of physical memory should *not* use both the /3GB switch and the /PAE switch. The /PAE switch is obviously required to make use of all physical memory above 4 GB. Remember however that PAE uses the kernel addressable memory to manage the physical memory above 4 GB. When physical memory exceeds 16 GB, the 1 GB of memory allocated to the kernel when the /3GB switch is used is not sufficient to manage all the additional physical memory above 4 GB. Thus, only the /PAE switch should be used in such a case to avoid the system running out of kernel memory.

For more information, see:

http://msdn2.microsoft.com/en-us/library/ms175581.aspx

TCP/IP registry optimizations

Windows has several registry parameters that can be modified to optimize the performance of the TCP/IP protocol for your environment. In most networks, Windows TCP/IP is tuned by the operating system to achieve maximum performance, however there might be settings that can be made to improve performance on your network.

When changing these parameters, you should understand what the parameters are doing and what will be the ramification of changing them. If you find that any of these changes cause throughput of your server to decrease, then you should reverse the changes made.

Tip: Several of the registry values described in this chapter do not exist in the registry by default and must be added manually. For these values, there is a system default which is overridden when you create the registry value.

This section and the next list a series of modifications that can, however, be made to the system registry that under certain circumstances might offer system performance improvements. Almost all of these modifications are ones for which there is no corresponding control panel or other GUI utility built natively into Windows to allow these values to be tuned without using the registry directly.

Modifying the registry is not for the faint-hearted and should only be done by experienced technical people. Incorrectly modifying the registry can cause serious system instability and in some circumstances, stop a server from booting correctly. Before implementing any of the suggestions made here, ensure you completely understand their impact and downstream effect they might have. Further reading from the Microsoft Web site or elsewhere is recommended to gain more understanding of the parameter changes listed in this section. Where useful further references exist, these have been provided.

Note that several of these changes might not take effect until the server is rebooted subsequent to making the change. Note too that when several of these changes are made and hard-coded into the registry, they restrict Windows from self-tuning this value to what might, under different circumstances, be a more optimal setting. Thus, any modifications made should only be to affect the server performance for the role it is currently hosting.

Note: Tuning Windows TCP/IP settings can have a significant impact on memory resources. Monitoring memory usage is very important if you choose to implement any of the settings suggested this section.

TCP window size

The TCP receive window specifies the maximum number of bytes that a sender can transmit without receiving an acknowledgment from the receiver. The larger the window size, the fewer acknowledgements are sent back, and the more optimal the network communications are between the sender and receiver. Having a smaller window size reduces the possibility that the sender will time-out while waiting for an acknowledgement, but will increase network traffic and reduce throughput.

TCP dynamically adjusts to a whole multiple of the maximum segment size (MSS) between the sender and receiver. The MSS is negotiated when the connection is initially set up. By adjusting the receive window to a whole multiple of the MSS, the number of full-sized TCP segments used during data transmission is increased, improving throughput.

By default, TCP will try to automatically negotiate the optimal window size depending on the maximum segment size. It initially starts at 16 KB and can range to a maximum of 64 KB. The TCP window size can also be statically specified in the registry, permitting potentially larger or more efficient values than can be negotiated dynamically.

The maximum TCP window size is normally 65535 bytes (64 KB). The maximum segment size for Ethernet networks is 1460 bytes. The maximum multiple of increments of 1460 that can be reached before exceeding this 64 KB threshold is 62420 bytes. This value of 62420 can thus be set in the registry for optimal performance on high-bandwidth networks. The value does not ordinarily exist in the registry and must be added.

The `TcpWindowSize` registry value can be set at a global or per-interface level. Interface-level settings will override the global value. To achieve the maximum window size, we recommended to set this only at the global level.

The registry value recommendation is as follows:

Key:	`HKLM\SYSTEM\CurrentControlSet \Services\Tcpip\Parameters`
Value:	`TCPWindowSize`
Data type:	`REG_DWORD`
Range:	`0x0` to `0xFFFF`
Default:	`0x4470` (17520 bytes, the Ethernet MSS (1470)) multiple closest to 16 K)
Recommendation:	`0xFAF0` (62420)
Value exists by default:	No, needs to be added.

For more information, see:

`http://support.microsoft.com/?kbid=263088`
`http://support.microsoft.com/?kbid=224829`

Large TCP window scaling and RTT estimation (timestamps)

The following TCP features are described in RFC 1323.

For more efficient use of high bandwidth networks, an even larger TCP window size can be used than described above in , “TCP window size” on page 33. This feature is new to Windows 2000 and Windows Server 2003 and is referred to as *TCP window scaling* and is used to increase the maximum TCP window size from the previous limit of 65535 bytes (64 KB) up to 1073741824 bytes (1 GB).

With large window (scaling window) support enabled, Windows can dynamically recalculate and scale the window size. This enables more data to be transmitted between acknowledgements, increasing throughput and performance.

The amount of time used for round-trip communications between a sender and receiver is referred to by TCP as the round-trip time (RTT). A time stamp option available to TCP improves the accuracy of RTT values by calculating it more frequently. This option is particularly helpful in estimating RTT over longer round-trip WAN links and will more accurately adjust TCP retransmission time-outs. This time stamp option provides two time stamp fields in the TCP header, one to record the initial transmission time and the other to record the time on the receiver.

The time stamp option is particularly valuable when window scaling support is enabled to ensures the integrity of the much larger packets being transmitted without acknowledgement. Enabling timestamps might actually have a slight impact to throughput as it adds 12 bytes to the header of each packet. The value of data integrity versus maximum throughput will thus need to be evaluated. In some instances, such as video streaming, where large TCP window size might be advantageous, data integrity might be secondary to maximum throughput. In such an instance, window scaling support can be enabled without timestamps.

Support for scaling window size and timestamps is negotiated at connection setup between the sender and receiver. Only if both the sender and receiver have support for these features enabled, will they be used during data transmission.

A small TCP window size will be negotiated initially and over time, using internal algorithms, the window size will grow to the maximum specified size. To enable support for scaling Windows or improved time stamps (RTT) estimation, make the following changes to the registry.

Key:	`HKLM\SYSTEM\CurrentControlSet \Services\Tcpip\Parameters`
Value:	`TCP1323Opts`
Data type:	`REG_DWORD`
Range:	`0x0` - `0x3` (see Table 6)
Default:	`0x0`
Recommendation:	`0x3`
Value exists by default:	No, needs to be added.

For more information, see:

http://support.microsoft.com/?kbid=224829

Note: The registry entry for this value is a 2-bit bitmask. The lower-bit determines whether scaling is enabled, and the higher-bit determines whether timestamps are enabled.

Table 6 Possible entries for TCP1323Opts registry value

TCP1323Opts registry value	Result
0x0	Disable windows scale and timestamps
0x1	Windows scaling enabled only
0x2	Timestamps scaling enabled only
0x3 (recommended)	Windows scaling and timestamps enabled

After TCP1323Opts has been used to enabled TCP window scaling, the registry value TCPWindowSize described in , “TCP window size” on page 33 can be increased to value from 64K (65535 bytes) right through to 1 GB (1,073,741,824 bytes). For best performance and throughput, the value set here should be a multiple of the maximum segment size (MSS).

As the optimal value for TCPWindowSize with window scaling support enabled will be different for each implementation, no specific recommendation is made here. This should be determined through careful testing in your environment. Note that as the window size increases, so does the risk of data corruption and subsequent resends as fewer acknowledgements will be issued from the receiver. This might actually then have a negative impact on performance.

TCP connection retransmissions

The number of times that TCP will retransmit an unacknowledged connection request (SYN) before aborting is determined by the registry value `TcpMaxConnectRetransmissions`. For each given attempt, the retransmission time-out is doubled with each successive retransmission. For Windows Server 2003, the default number of time-outs is 2 and the default time-out period is 3 seconds (set by the `TCPInitialRTT` registry entry).

The parameter for connections retransmissions can be incremented to prevent a connection from timing out across slow WAN links. As the optimal value will be different for each implementation, no specific recommendation is made. This should be determined through careful testing in your environment. Note: this parameter should not be set so high that the connection will not time out at all.

Key:	`HKLM\SYSTEM\CurrentControlSet \Services\Tcpip\Parameters`
Value:	`TCPMaxConnectRetransmissions`
Data type:	`REG_DWORD`
Range:	`0x0 - 0xFF`
Default:	`0x2`
Recommendation:	None made - environment specific
Value exists by default:	No, needs to be added.

For more information, see:

http://support.microsoft.com/?kbid=120642

TCP data retransmissions

The number of times that TCP will retransmit an unacknowledged data segment before aborting is specified by the registry value `TcpMaxDataRetransmissions`. The default value is 5 times.

TCP establishes an initial interval by measuring the round trip for a given connection. With each successive retransmission attempt, the interval doubles until responses resume or time-out altogether - at which time the interval is reset to the initially calculated value.

As the optimal value will be different for each implementation, no specific recommendation is made here. This should be determined through careful testing in your environment.

Key:	`HKLM\SYSTEM \CurrentControlSet \Services\Tcpip\Parameters`
Value:	`TCPMaxDataRetransmissions`
Data type:	`REG_DWORD`
Range:	`0x0 - 0xFFFFFFFF`
Default:	`0x5`
Recommendation:	None made, environment specific
Value exists by default:	No, needs to be added.

For more information, see:

http://support.microsoft.com/?kbid=120642

TCP TIME-WAIT delay

By default, TCP will normally allocate a port with a value between 1024 and 5000 for a socket request for any available short-lived (ephemeral) user port. When communications over a given socket have been closed by TCP, it waits for a given time before releasing it. This is known as the TIME-WAIT delay. The default setting for Windows Server 2003 is two minutes, which is appropriate for most situations. However, some busy systems that perform many connections in a short time might exhaust all ports available, reducing throughput.

Windows has two registry settings that can be used to control this time-wait delay:

- `TCPTimedWaitDelay` adjusts the amount of time that TCP waits before completely releasing a socket connection for re-use.
- `MaxUserPort` sets the number of actual ports that are available for connections, by setting the highest port value available for use by TCP.

Reducing TCPTimedWaitDelay and increasing MaxUserPort can increase throughput for your system.

Note: These changes only optimize performance on exceptionally busy servers hosting thousands of simultaneous TCP connections, such as a heavily loaded LDAP, FTP or Web servers.

Key:	`HKLM\SYSTEM\CurrentControlSet \Services\Tcpip\Parameters`
Value:	`TCPTimedWaitDelay`
Data type:	`REG_DWORD`
Range:	`0x0` - `0x12C` (0 - 300 seconds)
Default:	`0x78` (120 seconds)
Recommendation:	`0x1E` (30 seconds)
Value exists by default:	No, needs to be added.

Key:	`HKLM\SYSTEM\CurrentControlSet \Services\Tcpip\Parameters`
Value:	`MaxUserPort`
Data type:	`REG_DWORD`
Range:	`0x1388` - `0xFFFE` (5000 - 65534)
Default:	`0x1388` (5000)
Recommendation:	`0xFFFE`
Value exists by default:	No, needs to be added.

Note: The value name is `MaxUserPort`, not `MaxUserPorts`.

For more information, see *Microsoft Windows Server 2003 TCP/IP Implementation Details*, which is available from:

http://www.microsoft.com/technet/prodtechnol/windowsserver2003/technologies/networking/tcpip03.mspx

TCP Control Block (TCB) table

For each active TCP connection on a system, various control variables about the connection are stored in a memory block called a TCP control block (TCB). These TCB values are initialized when the connection is established and then continually updated throughout the lifetime of the connection. Each of these TCBs are maintained in a hash table called the TCB table.

The size of the TCB table is controlled by the registry value `MaxHashTableSize`. On a large system with many active connections, having a larger table reduces the amount of time spent the system must spend to locate a particular TCB.

By partitioning the TCB table, contention for table access is minimized. TCP performance can be optimized by increasing the number of partitions. This is particularly so on multi-processor systems. The registry value `NumTcbTablePartitions` value controls the number of partitions. By default, the value is the square of the number of processors in the system.

The MaxHashTableSize value should always be of a power of two to correctly function. You might consider using the maximum value for large servers that might host a high number of connections. Bear in mind however that the table uses non-paged pool memory so do not set too high a value if the system is constrained on non-paged pool memory of course if the system simply will not support a high load of connections.

With a server that has more than one CPU, the NumTcbTablePartitions parameter should be four times the number of processors installed in the system In most cases this will perform equally or better than the default square of the number of CPUs in the system, especially on servers with 8 or 16 CPUs, where too high a value of NumTcbTablePartitions can impact CPU performance.

Key:	`HKLM\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters`
Value:	`MaxHashTableSize`
Data type:	`REG_DWORD`
Range:	`0x40` - `0x10000` (`1` - `65536`), should be a power of 2 (2n)
Default:	`0x200` (`512`)
Recommendation:	`0x10000` (`65536`)
Value exists by default:	No, needs to be added.

Key:	`HKLM\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters`
Value:	`NumTcbTablePartitions`
Data type:	`REG_DWORD`
Range:	`0x1` - `0xFFFF` (1 - 65535), should be a power of 2 (2n)
Recommendation:	4 x the number of processors in the system
Default:	*n2*, where n is the number of CPUs installed

Tip: Because this value does not exist in the registry by default, be careful to ensure the value set is `NumTcbTablePartitions`, *not* `NumTcpTablePartitions`.

For more information, see *Microsoft Windows Server 2003 TCP/IP Implementation Details*, which is available from:

http://www.microsoft.com/technet/prodtechnol/windowsserver2003/technologies/networking/tcpip03.mspx

TCBs are normally pre-allocated in memory to avoid spending time allocating and de-allocating TCBs every time TCP connections are established and closed. The reuse or caching of TCBs improves memory management but also restricts how many active connections TCP can support at a given time.

The registry value `MaxFreeTcbs` configures the threshold number of connections required before TCBs in the TIME-WAIT state (that is, the connection has been closed but is not free for reuse yet), are re-used. This value was often set higher than the default to optimize performance in older Windows NT implementations to ensure there was always sufficient pre-allocated TCBs.

Since Windows 2000, a feature was added to decrease the chance of running out of pre-allocated TCBs. If more than the number of TCBs specified in the new registry value `MaxFreeTWTcbs` are in the TIME-WAIT state then all connections that have been in the TIME-WAIT state for longer than 60 seconds are forcibly closed and made available for use again.

With this feature now incorporated into Windows 2000 Server and now Windows Server 2003, modification of the `MaxFreeTcbs` registry value is no longer deemed valuable in optimizing TCP performance.

TCP acknowledgement frequency

TCP uses delayed acknowledgements to reduce the number of packets transmitted in the network, thereby improving performance. Normally, an acknowledgement (ACK) is sent every second TCP segment. This value can be increased to reduce the cost of processing of network traffic, especially in the case of large data uploads from the client to the server.

Note: This value is configured *at the interface level* for each network interface installed in the system and differs depending on the speed of the interface.

The default value is 2.

For Fast Ethernet (100 Mbps) network interfaces, use a value of `5` (`0x5`).

For Gigabit (1000 Mbps) network interfaces, use a value of `13` (`0xD`).

Key:	`HKLM\System\CurrentControlSet\Services\Tcpip \Parameters\Interface\xx` (xx depends on network interface)
Value:	`TcpAckFrequency`
Data type:	`REG_DWORD`
Range:	`0x1` - `0xD` (1-13)
Default:	`2`
Recommendation:	`0x5` (`5`) for FastEthernet or `0xD` (`13`) for Gigabit interfaces
Value exists by default:	No, needs to be added.

For more information, see *Performance Tuning Guidelines for Windows Server 2003*, which is available from:

http://www.microsoft.com/windowsserver2003/evaluation/performance/tuning.mspx

Maximum transmission unit

The TCP/IP maximum transmission unit (MTU) defines the maximum size of an IP datagram that can be transferred in one frame over the network over a specific data link connection. The MTU might differ for network segments between the sender and receiver. Too small a packet size means data transmission is inefficient however too large a packet size means that data might exceed the MTU of the links over which the packet is transmitted.

One method of determining the most efficient packet size allows routers to divide packets as they encounter a network segment with a smaller MTU than that of the packet being transmitted. This is referred to as IP *segmentation* or *fragmentation*. This method places extra overhead on routers in need to divide and reassemble packets.

A preferred and more common option is for a client to determine the maximum MTU that can be used on all segments between the sender and receiver. The client communicates with routers along the path as required to determine the smallest MTU that can be used in all segments from start to end. This process is known as calculating the path maximum transmission unit (PMTU) and will result in the most efficient packet size that will not need to be fragmented.

TCP/IP normally determines the optimal MTU dynamically, as described in , “Path Maximum Transmission Unit (PMTU) Discovery” on page 41. Windows does however allow the MTU to be statically configured. This is not normally recommended but might be suitable in some environments. Under most circumstances, allowing TCP/IP to dynamically determine the MTU is preferred, unless you can be certain of the MTU for every segment in your environment that a host might need to communicate over. By setting it statically, the MTU will not need to be negotiated and can offer a a performance improvement.

The MTU for your environment can be determined by using the PING command on one of your servers and issuing the following command:

```
PING -f -l <MTUsize> <remote host IP address>
```

Tip: The **`f`** and **`l`** parameters must be in lower-case for this command to work.

The **`MTUSize`** parameter is one you will use to determine the PMTU between the server and a remote host that it communicates with frequently. This might be a host on the same segment or one across multiple inter-continental WAN links. The command should be issued repeatedly using different values for MTUSize until the highest possible PMTU setting is achieved without receiving a “packet needs to be fragmented” response.

A good value to start with for MTUSize is **`1500`**, the Windows default. Work up or down from there (normally down) until the maximum setting is determined. Example 1 shows an optimal setting of a 1472 byte MTU before the packet starts being fragmented at 1473 bytes.

Example 1 Determining the MTU size

```
C:\>ping -f -l 1472 w3.ibm.com

Pinging w3ibm.southbury.ibm.com [9.45.72.138] with 1472 bytes of data:

Reply from 9.45.72.138: bytes=1472 time=26ms TTL=245
Reply from 9.45.72.138: bytes=1472 time=26ms TTL=245
Reply from 9.45.72.138: bytes=1472 time=26ms TTL=245
Reply from 9.45.72.138: bytes=1472 time=26ms TTL=245

Ping statistics for 9.45.72.138:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 26ms, Maximum = 26ms, Average = 26ms

C:\>ping -f -l 1473 w3.ibm.com

Pinging w3ibm.southbury.ibm.com [9.45.72.138] with 1473 bytes of data:

Packet needs to be fragmented but DF set.
Packet needs to be fragmented but DF set.
Packet needs to be fragmented but DF set.
Packet needs to be fragmented but DF set.

Ping statistics for 9.45.72.138:
    Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),
```

Tip: This registry value is set at the interface level, not as an overall TCP/IP parameter.

After you determine this optimal MTU value, you set it in the registry as follows:

Key:	`HKLM\SYSTEM \CurrentControlSet \Services\Tcpip\Parameters\Interface\xxxxxxx` (depends on network interface)
Value:	`MTU`
Data type:	`REG_DWORD`
Range:	`0x44` (68), determined dynamically MTU OR `0xFFFFFFFF`
Default:	`0xFFFFFFFF` (determine dynamically PMTU)
Recommendation:	`0xFFFFFFFF`
Value exists by default:	No, needs to be added.

For more information, see

`http://www.microsoft.com/windows2000/techinfo/reskit/en-us/regentry/58792.asp`

Important: There is a close interaction between the `MTU` registry setting and the registry setting described in the next section, `EnablePMTUDiscovery`.

To use a statically determined value for MTU as described above, `EnablePMTUDiscovery` should be disabled (that is, set to 0). If `EnablePMTUDiscovery` is disabled and no value is set for MTU as described above, TCP/IP will configure a default MTU size of 576 bytes. This packet size usually avoids any packet fragmentation but is far from optimal for most environments. This means that in most instances, if `EnablePMTUDiscovery` is disabled, a value should be set for MTU as well.

If `EnablePMTUDiscovery` is enabled (the default), then the `MTU` registry value should be set to 0xFFFFFFFF, or it can be removed altogether.

Path Maximum Transmission Unit (PMTU) Discovery

Under most circumstances, the maximum transmission unit (MTU) of every network segment that a server might possibly communicate over will not be known. Remote networks will often have an entirely different MTU to that of local networks.

When enabled, the registry value `EnablePMTUDiscovery` lets TCP/IP automatically determine the MTU for all networks along the path to a remote host. When the MTU has been determined for all network segments on a path, it will use the highest, and thus most efficient MTU value that can used without packet fragmentation occurring.

PMTU detection is enabled by default in Windows. As the `EnablePMTUDiscovery` does not ordinarily exist in the registry, it would normally only be created with the intention of disabling PMTU detection. If you do choose to disable PMTU detection, a default MTU of 576 bytes will be used for all communications, unless a value is also set for the MTU registry value as described in , “Maximum transmission unit” on page 39.

Note: This registry value applies to all network interfaces.

The following registry value controls the use of PMTU Discovery:

Key:	`HKLM\SYSTEM \CurrentControlSet \Services\Tcpip\Parameters`
Value:	`EnablePMTUDiscovery`
Data type:	`REG_DWORD`
Range:	`0` or `1`
Default:	`1`
Recommendation:	`1`
Value exists by default:	No, needs to be added.

For more information, see

`http://www.microsoft.com/windows2000/techinfo/reskit/en-us/regentry/58752.asp`

Memory registry optimizations

For most purposes, Windows operates very well in its native self-tuning capacity. Nonetheless there are many other registry changes relating to the memory subsystem that can be modified to improve system performance under specific circumstances. Some of the ones that have been noted to improve performance in production environments are listed here.

Note that several of the memory specific tuning parameters listed here hold relevance only for the 32-bit (x86) versions of the Windows Server 2003 operating system. They are no longer valid for the 64-bit (x64) editions given the greatly expanded memory architecture.

As with all changes, ensure you have a working and tested backup of the registry and entire server before making the change. Changes should be made and tested only one at a time. If system performance is negatively affected by making such a change, it should be reversed immediately.

Disable kernel paging

Servers with sufficient physical memory might benefit from disabling portions of the Windows operating system kernel and user-mode and kernel-mode drivers from being paged to disk. This registry setting forces Windows to keep all components of the kernel (or executive) and drivers in memory and, thus, allows much faster access to them when required.

Key:	`HKLM\SYSTEM \CurrentControlSet \Control \Session Manager\Memory Management`
Value:	`DisablePagingExecutive`
Data type:	`REG_DWORD`
Range:	`0x0` (default) or 0x1
Recommendation:	`0x1`
Value exists by default:	No, needs to be added

For more information, see:

`http://support.microsoft.com/?kbid=184419`

Optimizing the Paged Pool Size (32-bit x86)

Windows allocates memory in *pools* for the operating system and its components, which processes access through the use of *kernel mode*. Two pools of kernel mode memory exist:

- The paged pool (which can be paged to the pagefile)
- The non-paged pool (which can never be paged)

Performance and system stability can be seriously impacted if Windows experiences memory resource constraints and is unable to assign memory to these pools.

The amount of physical memory assigned to these two pools is assigned dynamically at system boot time. The maximum default size on 32-bit (x86) editions of Windows Server 2003 for the paged memory pool is 491 MB, and 256 MB for the non-paged pool. In the 64-bit (x64) editions of Windows Server 2003, both the paged pool and non-paged pool have a limit of 128 GB. As a result the following values do not apply for the 64-bit (x64) editions.

Some applications and workloads can demand more pooled memory than the system has been allocated by default. Setting the PagedPoolSize registry value as listed in Table 7 can assist in ensuring sufficient pooled memory is available.

Changing this setting requires a restart of the operating system.

Table 7 PagedPoolSize values - 32-bit (x86) Editions of Windows Server 2003

PagedPoolSize value	Meaning
0x0 (default)	Requests that the system will dynamically calculate an optimal value at system startup for the paged pool based on the amount of physical memory in the computer. This value will change if more memory is installed in the computer. The system typically sets the size of the paged pool to approximately twice that of the nonpaged pool size.
Range: 0x1 - 0x20000000 (512 MB)	Creates a paged pool of the specified size, in bytes. This takes precedence over the value that the system calculates, and it prevents the system from adjusting this value dynamically. Limiting the size of the paged pool to 192 MB (or smaller) lets the system expand the file system (or system pages) virtual address space up to 960 MB. This setting is intended for file servers and other systems that require an expanded file system address space (meaning slightly faster access) at the expense of being able to actually cache less data. This only makes sense if you know the files your server frequently accesses already fit easily into the cache.
0xFFFFFFFF	With this value, Windows will calculate the maximum paged pool allowed for the system. For 32-bit systems, this is 491 MB. This setting is typically used for servers that are attempting to cache a very large number of frequently used small files, some number of very large size files, or both. In these cases, the file cache that relies on the paged pool to manage its caching it able to cache more files (and for longer periods of time) if more paged pool is available.

Setting this value to `0xB71B000` (192 MB) provides the system with a large virtual address space, expandable to up to 960 MB. Note that a corresponding entry of zero (0) is required in the SystemPages registry value for this to take optimal effect.

Key:	`HKLM\SYSTEM \CurrentControlSet \Control \Session Manager\Memory Management`
Value:	`PagedPoolSize`
Data type:	`REG_DWORD`
Range:	`0x0` (default) to `0xFFFFFFFF` (4294967295)
Recommendation:	`0xB71B000` (192000000) for native SMB shares. For NFS shares when using Services for UNIX, use a value of `0xFFFFFFFF` (4294967295)
Value exists by default:	Yes

Key:	`HKLM\SYSTEM\CurrentControlSet\Control \SessionManager\Memory Management`
Value:	`SystemPages`
Data type:	`REG_DWORD`
Range:	`0x0` (default) to `0xFFFFFFFF`
Recommendation:	`0x0`
Value exists by default:	Yes

For more information, see:

http://www.microsoft.com/resources/documentation/windows/2000/server/reskit/en-us/core/fnec_evl_fhcj.asp

Increase memory available for I/O locking operations

By default, the 32-bit (x86) editions of Windows Server 2003 sets a limit to the amount of memory that can be set aside for I/O locking operations at 512 KB. Depending on the amount of physical memory in a server, this can be increased in various increments as per Table 8. You can insert the values listed in this table into the registry to increase the amount of memory available for locking operations on 32-bit (x86) systems. These values hold no validity for the 64-bit (x64) editions of Windows Server 2003.

Table 8 Maximum I/O lock limit values

Amount of physical RAM	Maximum lock limit (IoPageLockLimit value)
Less than 64 MB	Physical memory minus 7 MB
64 MB - 512 MB	Physical memory minus 16 MB
512 MB upwards	Physical memory minus 64 MB

These value ranges listed in Table 8 equate to those calculated in Table 9, depending on the exact amount of physical RAM in the machine. As almost all servers today have more than 512 MB RAM, the calculations in Table 9 take into account only 512 MB RAM and above.

The appropriate value should be determined from the Table 8 and then entered into the registry value `IoPageLockLimit`. This value then takes precedence over the system default of 512 KB and specifies the maximum number of bytes that can be locked for I/O operations:

Key:	`HKLM\SYSTEM \CurrentControlSet \Control \Session Manager\Memory Management`
Value:	`IoPageLockLimit`
Data type:	`REG_DWORD`
Range:	`0` (default) to `0xFFFFFFFF` (in bytes, *do not exceed this maximum*!)
Recommendation:	depends on RAM, see Table 9
Default:	`0x80000` (512 KB)
Value exists by default:	No, needs to be added.

Table 9 Recommended settings for IoPageLockLimit

Amount of physical RAM	IoPageLockLimit Setting (hex)
512 MB	0x1C000000
1 GB (1024 MB)	0x3C000000
2 GB (2048 MB)	0x80000000
4 GB (4096 MB)	0xFC000000
8 GB (8096 MB) and above	0xFFFFFFFF

For more information, see:

http://www.microsoft.com/windows2000/techinfo/reskit/en-us/regentry/29932.asp

Increasing available worker threads

At system startup, Windows creates several server threads that operate as part of the System process. These are called *system worker threads*. They exist with the sole purpose of performing work on the behalf of other threads generated by the kernel, system device drivers, the system executive and other components. When one of these components puts a work item in a queue, a thread is assigned to process it.

The number of system worker threads would ideally be high enough to accept work tasks as soon as they become assigned. The trade off of course is that worker threads sitting idle are using system resources unnecessarily.

The `DefaultNumberofWorkerThreads` value sets the default number of worker threads created for a given work queue. Having too many threads needlessly consumes system resources. Having too few slows the rate at which work items are serviced.

Key:	`HKLM\SYSTEM \CurrentControlSet \Services\RpcXdr\Parameters`
Value:	`DefaultNumberofWorkerThreads`
Data type:	`REG_DWORD`
Range:	`0x0` (default) to `0x40` (64)
Recommendation:	16 times the number of CPUs in the system
Value exists by default:	No, needs to be added

Delayed worker threads process work items that are not considered time-critical and can have their memory stack paged out while waiting for work items. The value for `AdditionalDelayedWorkerThreads` increases the number of delayed worker threads created for the specified work queue. An insufficient number of threads slows the rate at which work items are serviced; a value too high will consume system resources unnecessarily.

Key:	`HKLM\SYSTEM \CurrentControlSet\Control \SessionManager\Executive`
Value:	`AdditionalDelayedWorkerThreads`
Data type:	`REG_DWORD`
Range:	`0x0` (default) to `0x10` (16)
Recommendation:	`0x10` (16)
Value exists by default:	Yes

Critical worker threads process time-critical work items and have their stack present in physical memory at all times. The value for `AdditionalCriticalWorkerThreads` increases the number of critical worker threads created for a specified work queue. An insufficient number of threads slows the rate at which time-critical work items are serviced. A value that is too high consumes system resources unnecessarily.

Key:	`HKLM\SYSTEM\CurrentControlSet\Control \SessionManager\Executive`
Value:	`AdditionalCriticalWorkerThreads`
Data type:	`REG_DWORD`
Range:	`0x0` (default) to `0x10` (16)
Recommendation:	`0x10` (16)
Value exists by default:	Yes

Prevent the driver verifier from running randomly

The driver verifier at random intervals verifies drivers for debugging randomly. Disabling this functionality might improve system performance.

Key:	`HKLM\SYSTEM\CurrentControlSet\Control \SessionManager\Memory Management`
Value:	`DontVerifyRandomDrivers`
Data type:	`REG_DWORD`
Range:	`0x0` (default) or `0x1`

Recommendation:	`0x1`
Value exists by default:	No

For more information, see:

`http://www.microsoft.com/windowsserver2003/evaluation/performance/tuning.mspx`

File system optimizations

Several registry tuning parameters are available in Windows Server 2003. These will assist with performance in both 32-bit (x86) and 64-bit (x64) editions of the server operating system.

Increase work items and network control blocks

The maximum number of concurrent outstanding network requests between a Windows Server Message Block (SMB) client and server is determined when a session between the client and server is negotiated. The maximum value that is negotiated is determined by registry settings on both the client and server. If these values are set too low on the server, they can restrict the number of client sessions that can be established with the server. This is particularly a problem in a Terminal Server environment where clients are typically launching many simultaneous application instances on the server itself and using many local resources.

The three values that can adjusted to improve system performance for work items exist in the LanmanServer and LanmanWorkstation registry keys and are:

- MaxWorkItems
- MaxMpxCt
- MaxCmds

Each of these values do not exist by default in the registry. The default settings for the first two values are determined by the hardware configuration of the server combined with the value of the Server dialog (File & Print Sharing setting discussed in , “File system cache” on page 11). MaxCmds has a default of 50.

The MaxWorkItems value specifies the maximum number of receive buffers, or work items, that the Server service is permitted to allocate at one time. If this limit is reached, then the transport must initiate flow control, which can significantly reduce performance.

The MaxMpxCt value sets the maximum number of simultaneous outstanding requests on a client to a particular server. During negotiation of the Server Message Block dialect, this value is passed to the client's redirector where the limit on outstanding requests is enforced. A higher value can increase server performance but requires more use of server work items (MaxWorkItems).

The MaxCmds value specifies the maximum number of network control blocks that the redirector can reserve. The value of this entry coincides with the number of execution threads that can be outstanding simultaneously. Increase this value will improve network throughput, especially if you are running applications that perform more than 15 operations simultaneously.

Take care not to set any of these values too high. The more outstanding connections that exist, the more memory resources will be used by the server. If you set the values too high, the server could run out of resources such as paged pool memory.

Tip: The MaxWorkItems value must be at least *four times* as large as MaxMpxCt.

Key:	`HKLM\SYSTEM\CCS\Services\LanmanServer\Parameters`
Value:	`MaxWorkItems`
Data type:	`REG_DWORD`
Value:	`1 - 65535`
Default:	Configured dynamically based on system resources and server setting
Recommendation:	`32768`
Value exists by default:	No, needs to be added

Key:	`HKLM\SYSTEM\CCS\Services\LanmanWorkstation\Parameters`
Value:	`MaxCmds`
Data type:	`REG_DWORD`
Value:	`50 - 65535`
Default:	`50`
Recommendation:	`4096`
Value exists by default:	No, needs to be added

Key:	`HKLM\SYSTEM\CCS\Services\LanmanServer\Parameters`
Value:	`MaxMpxCt`
Data type:	`REG_DWORD`
Value:	`8192`
Default:	Configured dynamically based on system resources and server setting
Recommendation:	`1`
Value exists by default:	No, needs to be added

For more information, see:

http://support.microsoft.com/?kbid=232476
http://support.microsoft.com/?kbid=271148

Disable NTFS last access updates

Each file and folder on an NTFS volume includes an attribute called Last Access Time. This attribute shows when the file or folder was last accessed, such as when a user performs a folder listing, adds files to a folder, reads a file, or makes changes to a file. Maintaining this information creates a performance overhead for the file system especially in environments where a large number of files and directories are accessed quickly and in a short period of time, such as by a backup application. Apart from in highly secure environments, retaining this information might add a burden to a server that can be avoided by updating the following registry key:

Key:	`HKLM\SYSTEM \CurrentControlSet\Control\FileSystem`
Value:	`NTFSDisableLastAccessUpdate`
Data type:	`REG_DWORD`
Value:	`0` or `1`
Default:	`0`
Recommendation:	`1`
Value exists by default:	No, needs to be added

For more information, see:

http://www.microsoft.com/resources/documentation/WindowsServ/2003/all/deployguide/en-us/46656.asp

In Windows Server 2003, this parameter can also be set by using the command:

```
fsutil behavior set disablelastaccess 1
```

Disable short-file-name (8.3) generation

By default, for every long file name created in Windows, NTFS generates a corresponding short file name in the older 8.3 DOS file name convention for compatibility with older operating systems. In many instances this functionality can be disabled, offering a performance increase.

Note that before disabling short name generation, ensure that there is no DOS or 16-bit application running on the server that requires 8.3 file names or that are there any users accessing the files on the server through 16-bit applications or older file systems or operating systems. Note too that even some recent applications have been known to exhibit problems at installation and run time if this setting is made.

Key:	`HKLM\SYSTEM \CurrentControlSet\Control\FileSystem`
Value:	`NTFSDisable8dot3NameCreation`
Data type:	`REG_DWORD`
Value:	`0` or `1`
Default:	`0`
Recommendation:	`1`
Value exists by default:	Yes

In Windows Server 2003, this parameter can also be set by using the command:

```
fsutil behavior set disable8dot3 1
```

Use NTFS on all volumes

Windows offers multiple file system types for formatting drives, including NTFS, FAT, and FAT32. NTFS is always be the file system of choice for servers.

NTFS offers considerable performance benefits over the FAT and FAT32 file systems and should be used exclusively on Windows servers. In addition, NTFS offers many security, scalability, stability and recoverability benefits over FAT.

Under previous versions of Windows, FAT and FAT32 were often implemented for smaller volumes (say <500 MB) because they were often faster in such situations. With disk storage relatively inexpensive today and operating systems and applications pushing drive capacity to a maximum it is unlikely that such small volumes will be warranted. FAT32 scales better than FAT on larger volumes but is still not an appropriate file system for Windows servers.

FAT and FAT32 have often been implemented in the past as they were seen as more easily recoverable and manageable with native DOS tools in the event of a problem with a volume. Today, with the various NTFS recoverability tools built both natively into the operating system and as third-party utilities available, there should no longer be a valid argument for not using NTFS for file systems.

Do not use NTFS file compression

While it is an easy way to reduce space on volumes, NTFS file system compression is not appropriate for enterprise file servers. Implementing compression places an unnecessary overhead on the CPU for all disk operations and is best avoided. Think about options for adding additional disk, near-line storage or archiving data before seriously considering file system compression.

Monitor drive space utilization

The less data a disk has on it, the faster it will operate. This is because on a well defragmented drive, data is written as close to the outer edge of the disk as possible, as this is where the disk spins the fastest and yields the best performance.

Disk seek time is normally considerably longer than read or write activities. As noted above, data is initially written to the outside edge of a disk. As demand for disk storage increases and free space reduces, data is written closer to the center of the disk. Disk seek time is increased in locating the data as the head moves away from the edge, and when found, it takes longer to read, hindering disk I/O performance.

This means that monitoring disk space utilization is important not just for capacity reasons but for performance also. It is not practical nor realistic to have disks with excessive free space however.

Tip: As a rule of thumb, work towards a goal of keeping disk free space between 20% to 25% of total disk space.

Use disk defragmentation tools regularly

Over time, files become fragmented in non-contiguous clusters across disks, and system performance suffers as the disk head jumps between tracks to seek and re-assemble them when they are required.

Disk defragmentation tools work to ensure all file fragments on a system are brought into contiguous areas on the disk, improving disk I/O performance. Regularly running a disk defragmentation tool on a server is a relatively easy way to yield impressive system performance improvements.

Tip: We recommend you defragment your drives daily if possible.

Most defragmentation tools work the fastest and achieve the best results when they have plenty of free disk space to work with. This provides another good reason to monitor and manage disk space usage on production servers.

In addition, try to run defragmentation tools when the server is least busy and if possible, during scheduled system downtime. Defragmentation of the maximum number of files and directories will be achieved if carried out while applications and users that typically keep files open are not accessing the server.

Windows Server 2003 includes a basic disk defragmentation tool. While offer good defragmentation features, it offers little in the way of automated running—each defragmentation process must be initiated manually or through external scripts or scheduling tools.

A number of high-quality third-party disk defragmentation tools exist for the Windows operating system, including tailorable scheduling, reporting and central management functionality. The cost and performance benefit of using such tools is normally quickly realized when compared to the ongoing operational costs and degraded performance of defragmenting disks manually, or not at all.

Review disk controller stripe size and volume allocation units

When configuring drive arrays and logical drives within your hardware drive controller, ensure you match the controller stripe size with the allocation unit size that the volumes will be formatted with within Windows. This will ensure disk read and write performance is optimal and gain better overall server performance.

Having larger allocation unit (or *cluster* or *block*) sizes means the disk space is not used as efficiently, but it does ensure higher disk I/O performance as the disk head can read in more data in one read activity.

To determine the optimal setting to configure the controller and format the disks with, you should determine the average disk transfer size on the disk subsystem of a server with similar file system characteristics. Using the Windows System (Performance) Monitor tool to monitor the Logical Disk object counters of *Avg. Disk Bytes/Read* and *Avg. Disk Bytes/Write* over a period of normal activity will help determine the best value to use.

Some server purposes might warrant a smaller allocation unit size where the system will be accessing many small files or records however with file sizes, file systems and disk storage every increasing today our testing has found setting an allocation unit size of 64 KB delivers sound performance and I/O throughput under most circumstances. Improvements in performance with tuned allocation unit sizes can be particularly noted when disk load increases.

Note that either the FORMAT command line tool or the Disk Management tool are needed to format volumes larger than 4096 bytes (4 KB). Windows Explorer will only format up to this threshold. Note also that CHKDSK can be used to confirm the current allocation unit size of a volume however it needs to scan the entire volume before the desired information is displayed (shown as Bytes in each allocation unit).

Tip: The default allocation unit size that Windows Server 2003 will format volumes with is 4096 bytes (4 KB). It is definitely worth considering whether this size really is the most appropriate for your purposes for the volumes on your servers.

Use auditing and encryption judiciously

Auditing and file encryption are two valuable security features that while very beneficial, can also add considerable overhead to system resources and performance. In particular, CPU and memory resources can be taxed by using auditing and / or encryption.

Auditing is often a required system function by security conscious organizations. The extent to which it impacts server resources will be determined by what level auditing servers are actually employed. To ensure the performance impact of auditing is kept as low as possible, ensure that only the system events, and areas of file systems and the registry that actually do require auditing are configured so.

Similarly, file encryption is required in some instances to improve system security. As with auditing, ensure that only directories and files that actually require the security brought about by encryption are set up with this feature.

Other performance optimization techniques

This section details various other methods that should be implemented on Windows servers to extract the best performance.

Dedicate server roles

Where budget and operational resources permit, use dedicated domain-controller servers, and dedicated member, or stand-alone, servers. Do not combine Active Directory, DNS, WINS, DHCP, Certificate Services or similar infrastructure services onto a member server that's primary function is for another purpose. Each of these services places additional overhead on the server, taking away valuable resources that should be dedicated to the primary function the member server is serving.

In the same manner, system resources are best dedicated to specific intensive functions such as file serving, line-of-business applications servers, mail servers, database servers or Web servers. The nature of each of these functions will each affect the server subsystems in a different way so to improve performance and increase stability, dedicate servers for different server functions.

Run system intensive operations outside peak times

Applications and jobs that are intensive on any server subsystem should be scheduled to run outside peak server times. When a server needs to be running fastest, it does not make sense to slow it down with applications such as virus scanners, backup jobs or disk fragmentation utilities.

Log off the server console

A simple but important step in maximizing your server's performance is to keep local users logged off the console. A locally logged-on unnecessarily consumes systems resources, potentially impacting the performance of applications and services running on the system.

Remove CPU-intensive screen savers

A server is not the place to run fancy 3D or OpenGL screen savers. Such screen-savers are known to be CPU-intensive and use important system resources when they are running. It is best to avoid installing these altogether as an option at server-build time, or to remove them if they have been installed. The basic "Windows Server 2003" or blank screen savers are the best choice.

In a similar vein, do not use memory-wasting fancy desktop wallpapers images on your server.

Use the latest drivers, firmware, and service packs

Installing the latest version of a device driver, patch, BIOS update, microcode, or firmware revision for hardware is a very important part of routine server maintenance. Newer device drivers not only fix bugs and increase system stability, but can also increase the performance and efficiency of a device, improving overall system performance. A good example of this with System x servers is the various models of ServeRAID™ adapters. Revisions in the firmware and drivers for these RAID controllers has often added significant performance and functionality benefits over previous releases.

Microsoft periodically issues service packs and hot fixes for their operating systems. After a period of testing in your environment, these should be deployed to production systems. Service packs and hot fixes often introduce updated code to key kernel and sub-system components of the operating system add can extra performance and functionality benefits.

In addition to the performance benefits that can be offered by latest version of device drivers, patches, firmware and service packs, many hardware and software vendors will not offer support for their products until this latest revision is installed regardless. It is good maintenance practice to do so periodically.

Avoid the use of NET SERVER CONFIG commands

As discussed, Windows is for the most part a self-tuning operating system. If a value does not ordinarily exist in the registry, creating it and manually setting it normally prevents Windows from self-tuning the value automatically. In some circumstances this is what you hope to achieve by setting values that you believe are more optimal for your environment that Windows normally auto-configures.

The core Windows *server* service is controlled by a set of values defined in registry at:

```
HKLM\System\CurrentControlSet\Services\LanmanServer\Parameters
```

In a standard *clean* installation, many values that can exist in this registry location do not. This allows Windows to tune these values dynamically as the operating system sees fit to optimize system performance of the server service.

A command exists to set several of these parameters to desired values to avoid direct manipulation of the registry. These include the following commands:

```
net config server /autodisconnect:time
net config server /srvcomment:"text"
net config server /hidden:yes|no
```

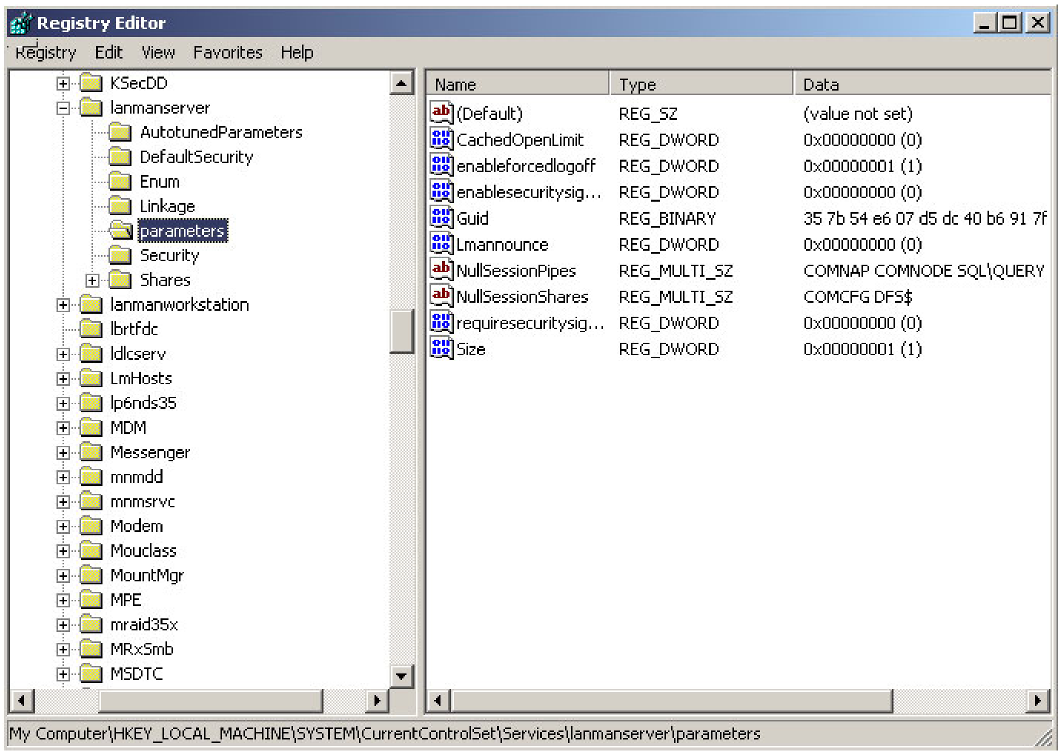
As expected, these three commands create and manipulate the contents of the registry values `autodisconnect`, `disc`, `srvcomment`, and `hidden` in the `LanmanServer\parameters` key to the desired values.

There is, however, a most unfortunate catch in using these commands. Not only do they create the necessary registry entry for the parameter requested, but they also create a whole series of other unrelated registry values that are extremely important to the functioning of the server service. Creating all these values sets these values to static settings and subsequently prevents them from further self-tuning by Windows. This is most undesirable and should be avoided.

For example, administrators who want to hide Windows computers from the network browse list might issue the command:

```
net config server /hidden:yes
```

Before issuing such a command, the registry parameters for the LanmanServer key on a typical Windows 2000 Server appear similar to Figure 18.

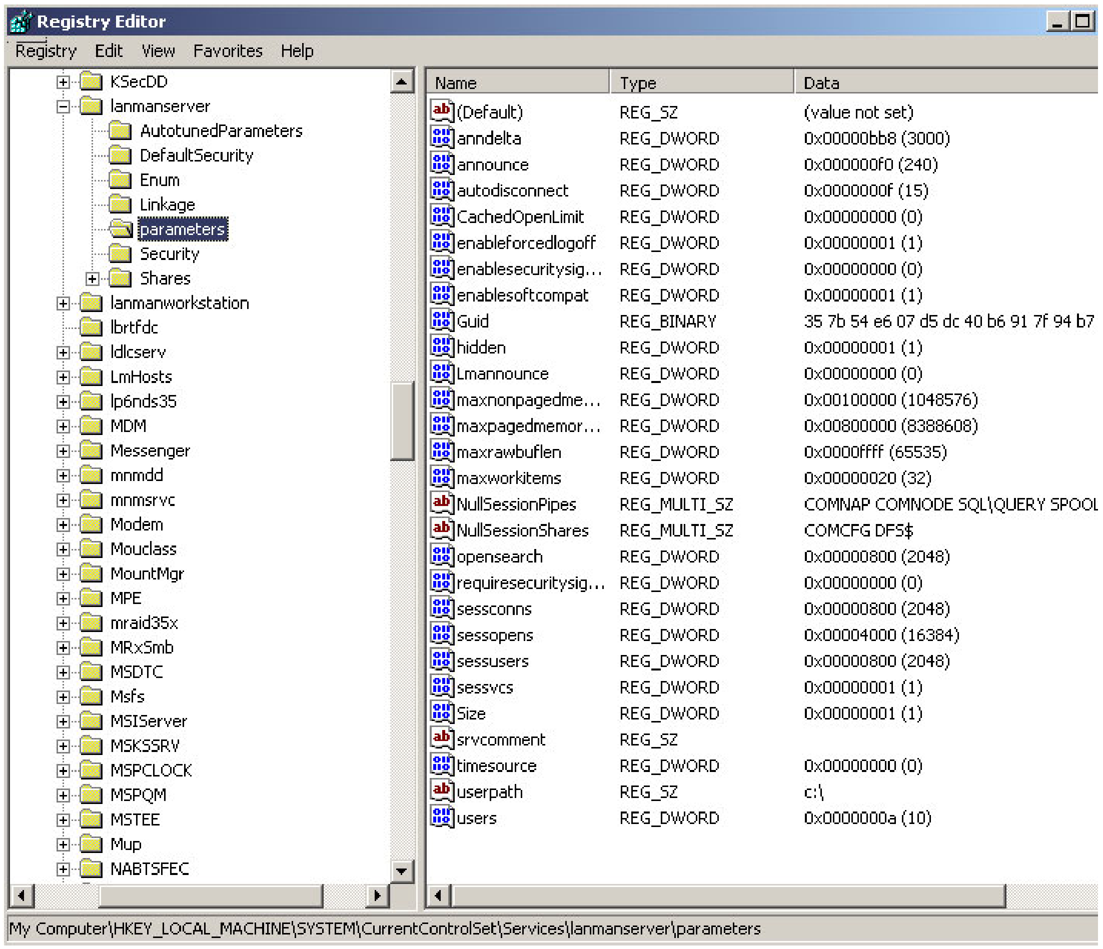


Name	Type	Data
(Default)	REG_SZ	(value not set)
CachedOpenLimit	REG_DWORD	0x00000000 (0)
enableforcedlogoff	REG_DWORD	0x00000001 (1)
enablesecuritysig...	REG_DWORD	0x00000000 (0)
Guid	REG_BINARY	35 7b 54 e6 07 d5 dc 40 b6 91 7f
Lmannounce	REG_DWORD	0x00000000 (0)
NullSessionPipes	REG_MULTI_SZ	COMNAP COMNODE SQL\QUERY
NullSessionShares	REG_MULTI_SZ	COMCFG DFS$
requiresecuritysig...	REG_DWORD	0x00000000 (0)
Size	REG_DWORD	0x00000001 (1)

My Computer\HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\lanmanserver\parameters

Figure 18 LanmanServer registry parameters - before *using NET CONFIG command*

After issuing this single, relatively simple command, however, Figure 19 shows the number of registry entries that have been inserted into the LanmanServer key. Each of one of these values is now statically set at the values displayed and is no longer able to be automatically tuned by Windows. Some of these values control important performance and scalability parameters of the Server service. They are now constrained to these static values and cannot be changed unless they are modified manually or remove altogether.



Name	Type	Data
(Default)	REG_SZ	(value not set)
anndelta	REG_DWORD	0x00000bb8 (3000)
announce	REG_DWORD	0x000000f0 (240)
autodisconnect	REG_DWORD	0x0000000f (15)
CachedOpenLimit	REG_DWORD	0x00000000 (0)
enableforcedlogoff	REG_DWORD	0x00000001 (1)
enablesecuritysig...	REG_DWORD	0x00000000 (0)
enablesoftcompat	REG_DWORD	0x00000001 (1)
Guid	REG_BINARY	35 7b 54 e6 07 d5 dc 40 b6 91 7f 94 b7
hidden	REG_DWORD	0x00000001 (1)
Lmannounce	REG_DWORD	0x00000000 (0)
maxnonpagedme...	REG_DWORD	0x00100000 (1048576)
maxpagedmemor...	REG_DWORD	0x00800000 (8388608)
maxrawbuflen	REG_DWORD	0x0000ffff (65535)
maxworkitems	REG_DWORD	0x00000020 (32)
NullSessionPipes	REG_MULTI_SZ	COMNAP COMNODE SQL\QUERY SPOOL
NullSessionShares	REG_MULTI_SZ	COMCFG DFS$
opensearch	REG_DWORD	0x00000800 (2048)
requiresecuritysig...	REG_DWORD	0x00000000 (0)
sessconns	REG_DWORD	0x00000800 (2048)
sessopens	REG_DWORD	0x00004000 (16384)
sessusers	REG_DWORD	0x00000800 (2048)
sessvcs	REG_DWORD	0x00000001 (1)
Size	REG_DWORD	0x00000001 (1)
srvcomment	REG_SZ	
timesource	REG_DWORD	0x00000000 (0)
userpath	REG_SZ	c:\
users	REG_DWORD	0x0000000a (10)

My Computer\HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\lanmanserver\parameters

Figure 19 LanmanServer registry parameters - after *using NET CONFIG command*

As shown in these figures, the undesirable impact of typing these commands. To avoid this issue, you need to create the values of `autodisconnect`, `disc`, `srvcomment`, and `hidden` manually as required when using the registry directly.

Windows Server 2003 does not populate the LanmanServer key with as many values as in Windows 2000 Server or Windows NT Server, but still introduces several parameters that are best left to auto-tune under most circumstances. This issue is clearly outlined in Microsoft KB 128167.

`http://support.microsoft.com/?kbid=128167`

Monitor system performance appropriately

Running the Windows System (Performance) Monitor directly on the server console of a server you are monitoring will impact the performance of the server and will potentially distort the results you are examining.

Wherever possible, use System Monitor from a remote system to avoid placing this extra load on the server itself. Similarly, do not use remote control software of the server console for performance monitoring nor should you use a remote client session using thin-client sessions such as Terminal Services or Citrix to carry out the task. Note however that monitoring a system remotely will affect the network and network interface counters which might impact your measurements.

Bear in mind that the more counters monitored, the more overhead is required. Don't monitor unnecessary counters. This tact should be taken regardless of whether carrying out "live" charted monitoring of a system or logging system performance over longer periods. In the same way, do not publish System Monitor as a Citrix application.

Along the same lines, reduce the monitoring interval to the minimum necessary to obtain the information you are looking to gather. When logging over an extended period, collecting data every five minutes or even more might still provide acceptable levels of information than the system default of every second. Monitoring more frequently will generate additional system overhead and create significantly larger log files than might be required.

The Future of Windows Server

This section discusses the current plans for follow-on products to Windows Server 2003.

Windows Server 2003, Service Pack 2

Windows continually seeks to improve its operating system through the issuing of Service Packs, which are a cumulative roll-up of all previous updates and fixes, including any previous service packs.

While the beta refresh 2 has just been realized at time of writing, the release date for the full version of Windows Server 2003, Service Pack 2 is currently scheduled for the first half of 2007.

Service Pack 2 will be applicable for both the Windows Server 2003 32-bit (x86) and 64-bit (x64) editions.

You can find the release notes for Service Pack 2 as it is released at:

`http://support.microsoft.com/kb/914961`

You can find the roadmap for Windows operating system Service Packs at:

`http://www.microsoft.com/windows/lifecycle/servicepacks.mspx`

Windows Server "Longhorn"

Microsoft Windows Server code name "Longhorn" is the next-generation of the Windows Server operating system. It is presently scheduled for release in late 2007, though its widespread adoption is not expected until well into 2008.

Longhorn is very significant upgrade of the Windows Server operating system. With many upgrades and changes, it is out of the scope of this chapter to discuss all the differences between Windows Server 2003 and Longhorn, especially in light of the fact that the product is still under final development and thus still subject to change. What is discussed briefly here however are those changes that should offer real server performance improvements.

The increased reliability, scalability, security and performance of Windows Server “Longhorn” make it a very impressive upgrade from its predecessors, including Windows Server 2003. Longhorn has been particularly tuned to deliver the best possible performance and integration with the last Windows desktop operating systems, most notably Vista, due for release in late 2006.

It is noteworthy that Longhorn is planned to be the last 32-bit release of Windows Server. Windows Server Longhorn R2 is planned to be a 64-bit operating system only.

Some notable performance changes in Longhorn include:

- **Server roles.** As it is shipped, Longhorn is very cut-back in functionality. You need to enable all server roles except file-serving as additional functions for a given server implementation. This functionality serves to tighten security and to improve performance because fewer idle processes are running on the server.
- **Server core.** Administrators can choose to install Windows Server with only core server functionality and without any extra overhead. This *server core* functionality limits the roles a server can perform but reduces management and improves system responsiveness. Server core is a minimal server installation option that provides an environment for running the following server roles:
  - Dynamic Host Configuration Protocol (DHCP) Server
  - Domain Name Systems (DNS) Server
  - File Server
  - Domain Controller

  The server core installation only installs a subset of the components of a “full” server operating system install. This includes removing the Windows Explorer GUI (shell), leaving only the command prompt to interface with the server.
- **IPv6.** Native IPv6 support across all client and server services creates a more scalable and reliable network, while the rewritten TCP/IP stack makes network communication much faster and more efficient.
- **SMB 2.0.** The new Server Message Block 2.0 protocol provides a number of communication enhancements, including greater performance when connecting to file shares over high-latency links and better security through the use of mutual authentication and message signing.
- **Memory Manager enhancements.** Longhorn includes enhanced support for NUMA (Non-Uniform Memory Architecture), allowing access to “closer” memory nodes, improving performance. Much faster large memory page allocations in both kernel AND user mode will be available.
- **Indexed Search**. Searching Windows Server “Longhorn” servers from a Windows Vista™ client avails of enhanced indexing and caching technologies on both to provide huge performance gains across the enterprise.
- **DNS background zone loading.** With Windows Server 2003, Active Directory integrated DNS servers for organizations with very large DNS zones can often take a long time to start as DNS data is retrieved from Active Directory, leaving the server unable to respond to user requests. In Longhorn, DNS loads zone data in the background so it can begin responding to user requests more quickly. It can query AD directly for any node record that might be unavailable while the zone loads directly onto the server.
- **Improved backup technology.** The use of Volume Shadow Copy Service and block-level backup technology improves performance and efficiency.
- **Windows Reliability & Performance Monitor.** This is a new MMC that combines the functionality of previous tools such as Performance Logs & Alerts, Server Performance Advisor and System Monitor.

- **Windows Systems Resource Manager (WSRM).** WSRM allows the server administrator to control how CPU and memory resources are allocated to applications, services and processes running on the server. Managing resources in this manner will offer improvements to system performance and reduces the opportunity for one application, service or process to take system resources away from another.
- **Internet Information Services 7.0 (IIS 7.0).** Improvements to the management tools and core feature set of IIS 7.0 yield definite performance benefits to those using IIS as a Web server.
- **Read-only domain controller (RODC).** Primarily aimed as a means of improving security for sites where physical security of domain controllers, cannot be guaranteed, the RODC also offers notable performance advantages to the server and wide-area-network (WAN). Local users experience faster logon time and more efficient access to WAN resources.

The 64-logical thread limitation

As the number of cores per physical processor increases, on large servers, the limit of 64 logical threads in Windows Server 2003 is becoming restrictive. Large systems, especially those with multiple nodes with multiple dual or even quad core processes and Hyper-Threading enabled have reached a threshold that is imposed by the operating system rather than the hardware.

An example of this is the IBM System x3950 systems which can host up to eight nodes each with four dual core processors with Hyper-Threading enabled. This configuration results in 128 logical processors which Windows Server 2003 cannot presently work around.

While there is no clear direction on a solution yet from Microsoft, they are well aware of this restriction and are prototyping support for more than 64 logical threads. This functionality will almost certainly not be delivered in the first release of Longhorn. Solving this particular problem is a particularly difficult one and means tackling many compatibility issues for applications and device drivers.

For more information, see the following PowerPoint® presentation:

http://download.microsoft.com/download/5/b/9/5b97017b-e28a-4bae-ba48-174cf47d23cd/SER058_WH06.ppt

The team that wrote this Redpaper

This Redpaper was produced by a team of specialists:

Phillip Dundas is the Technical Team Leader for the Windows Server Group at Macquarie Bank, based in Sydney, Australia. He has over 12 years of experience in network operating systems and server architecture, implementation, and support. His areas of technical expertise include Windows, VMWare, Citrix, Novell, server virtualization, server performance tuning, directory services, and establishing server standards. He also has extensive scripting experience and has developed several applications that are used by systems administrators to manage enterprise networks. Phillip holds a Bachelors degree in Applied Science (Computing) and a Masters degree in Commerce (Information Systems Management). Phillip holds many industry certifications, including Microsoft Certified Systems Engineer, VMWare Certified Professional, Certified Citrix Administrator, Master Certified Novell Engineer, and Certified Cisco Networking Associate.

David Watts is a Consulting IT Specialist at the IBM ITSO Center in Raleigh. He manages residencies and produces Redbooks™ on hardware and software topics related to System x servers and associated client platforms. He has authored over 80 Redbooks, Redpapers and Technotes. He has a Bachelor of Engineering degree from the University of Queensland (Australia) and has worked for IBM for over 15 years. He is an IBM Certified IT Specialist.

Notices

This document REDP-3943-01 was created or updated on February 7, 2007.

Send us your comments in one of the following ways:
- Use the online **Contact us** review redbook form found at:
  `ibm.com/redbooks`
- Send your comments in an email to:

redbooks@us.ibm.com

- Mail your comments to:
  IBM Corporation, International Technical Support Organization
  Dept. HZ8 Building 662
  P.O. Box 12195
  Research Triangle Park, NC 27709-2195 U.S.A.

Trademarks