



Operating System

Windows 2000 CHKDSK Management

By Elden Christensen

Microsoft Corporation

Draft: September 2002

Contributors: Terence Hosken, Wim van Wieren

Abstract

This document contains information spread across multiple knowledgebase articles, whitepapers, and other various documents. This contains a single source for all information regarding chkdsk, and specifically in regards to considerations on a Server Cluster. Topics covered are what chkdsk is, what it does, what your choices are, options, switches, registry keys, and cluster specifics.

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

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

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

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

© 2001 Microsoft Corporation. All rights reserved. Microsoft, Windows, Windows NT, and Active Directory are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.

Other product and company names mentioned herein may be the trademarks of their respective owners.

Microsoft Corporation • One Microsoft Way • Redmond, WA 98052-6399 • USA

Contents

| | |
|---|----|
| Introduction | 0 |
| File System Consistency..... | 1 |
| NTFS Recoverability | 1 |
| NTFS Transaction Log Recoverability | 2 |
| Windows 2000 Improvements | 3 |
| CHKDSK Performance | 4 |
| Understanding CHKDSK..... | 5 |
| Cluster Remapping | 6 |
| CHKDSK Phases | 6 |
| Phase One: Checking Files | 6 |
| Phase Two: Checking Indexes | 6 |
| Phase Three: Checking Security Descriptors | 7 |
| Phase Four: Checking Sectors | 7 |
| How Long will CHKDSK Take to Run | 8 |
| Variable 1: The "Indexes" Phase: | 8 |
| Variable 2: The Condition of the Volume: | 8 |
| Variable 3: Hardware Issues: | 8 |
| Variable 4: The CHKDSK Settings: | 8 |
| What are My Options | 10 |
| Running CHKDSK..... | 11 |
| CHKDSK Switches | 11 |
| Using the /C and /I Switches to Run an Abbreviated CHKDSK | 13 |
| The /C Switch: | 13 |
| The /I Switch: | 13 |
| AUTOCHK | 13 |
| CHKNTFS | 15 |
| CHKDSK and Server Clusters | 18 |
| How to Run CHKDSK on a Cluster | 19 |
| If Handles Remain Open or the Cluster Contains a Single Shared Disk | 20 |
| Windows NT 4.0: | 20 |

| | |
|---|----|
| Windows 2000: | 21 |
| Cluster resource | 21 |
| Windows NT 4.0 Cluster: | 22 |
| Windows 2000 Cluster: | 22 |
| New Private Properties in Windows 2000 | 22 |
| Chkdsk Status Codes in Cluster Log | 23 |
| Capturing Chkdsk Results for Server Clusters | 23 |
| Windows NT 4.0 | 24 |
| Windows 2000 | 24 |
| Windows 2000 Advanced Server SP1 and Datacenter | 25 |
| Conclusion | 28 |
| For More Information..... | 29 |
| Appendix | 30 |
| How to Automate CHKDSK | 30 |
| FSUtil.exe | 30 |
| FSUtil Syntax: | 30 |

Introduction

Microsoft is focused on ensuring that Windows 2000 continues to be a highly reliable Operating System. The focus is not only to extend functionality into new areas such as Active Directory, increase scalability, improve manageability etc., but to also address issues like the faster recovery of those ever-growing disk farms.

The following document summarises some of the improvements made in Windows 2000 CHKDSK and the way to manage corrupted volumes. It also discusses considerations of running CHKDSK on a Windows 2000 Advanced Server or Datacenter Server, Server Cluster.

CHKDSK execution performance has been significantly improved in Windows 2000 and in addition, enhancements have been made to the Windows 2000 File System (NTFS) with the aim of minimising failures. File system failures are predominately caused by hardware factors such as a malfunctioning/defective device, and/or incorrectly configured storage.

While Microsoft will continue to focus on improving disk recovery performance, the work being done does not remove the responsibility from an organization to apply "Best Practice" operational management, develop recovery procedures, and disaster recovery processes, which time and time again have been found to be key components in minimizing system outages of all types.

Microsoft continues improving the performance of CHKDSK to address the challenge put by our customers who are using new I/O hardware technology to put more and more data on "single" very large and growing volumes. The challenge for our customers is to maintain best practice and not only know how long a CHKDSK will take in their environments but to also have the practices in place that enable them to drive the recovery process, rather than being a victim of it.

This document contains some information already published in a number of TechNet articles, including some references, as well as some new information about the occasional reasons why a file system check is required. File system checks are not Microsoft platform specific

File System Consistency

NTFS is a recoverable file system that ensures volume consistency by using logging techniques. If the operating system crashes, the recoverable file system restores consistency by executing a recovery procedure that accesses information that has been stored in a log file. The NTFS file system does not guarantee protection of user file data. If the system crashes while an application is writing a file, the file can be lost or corrupted and a file system checker may be needed.

A file system's correctness and validity may be needed if there is catastrophic corruption of a meta-data file or corruption of user data and can be checked using a file systems check command, in Windows NT/2000 that is the CHKDSK.EXE command. CHKDSK.EXE is a command line utility that can be instructed to repair any problems it finds, and alert the administrator if there are any unrepairable issues. Commonly corruption is caused by power failures, failing hardware, or operator errors; for example, by not shutting down the system properly. Failure to shut down the system properly (or even power failures, for that matter) should not result in file-system corruption provided there are no underlying hardware issues.

NTFS Recoverability

NTFS is a recoverable file system that guarantees the consistency of the volume meta-data by using standard transaction logging and recovery techniques. In the event of a I/O failure committing data to the disk, NTFS restores consistency by running a recovery procedure that accesses information stored in a log file. The NTFS recovery procedure is exact, guaranteeing that the volume is restored to a consistent state.

NTFS ensures the integrity of all NTFS volumes by automatically performing disk recovery operations the first time Windows NT/2000 systems mounts an NTFS volume after the computer is restarted following a failure.

NTFS views each I/O operation that modifies a meta-data file on the NTFS volume as a transaction, and manages each one as an integral unit. Once started, the transaction is either completed or, in the event of a I/O operation failure, rolled back (such as when the NTFS volume is returned to the state it was in before the transaction was initiated).

When you format an NTFS volume, the format program creates a set of files that contains the data used to implement the file system structure. The NTFS file system reserves the first 16 records in the Master File Table (MFT) for the information about these files, called 'meta-data'. Meta-data is data stored on a volume in support of the file system format management, it isn't typically made accessible to applications. Meta-data includes the data that defines the placement of files and directories on a volume. In NTFS, all data stored on a volume is contained in files, including the data structures used to locate and retrieve files, bootstrap data, and the bitmap that records the allocation state of the entire volume. Meta-data files can be identified by the fact that the file names begin with a \$ sign and are hidden. For example: \$MFT, \$LogFile, \$Volume, \$Bitmap, \$BadClus, \$Secure to name a few.

To ensure that a transaction can be completed or rolled back, NTFS performs the sub-operations of the transactions on the volume. After NTFS updates the volume, it commits the transaction by recording in the log file that the entire transaction is complete. Both the log file entries and the volume updates are buffered by the system's file cache.

Once a transaction is committed, NTFS ensures that the entire transaction appears on the volume, even if the I/O operation fails due to a system shutdown or crash. During recovery operations that take place the next

time the volume is next mounted, NTFS redoes each committed transaction found in the log file. Then NTFS locates the transactions in the log file that were not committed at the time of the system failure and undoes each transaction sub-operation recorded in the log file. In this way, incomplete modifications to the volume meta-data are prohibited.

Important: *NTFS uses transaction logging and recovery to guarantee that the volume structure is not corrupted. For this reason, all meta data files remain accessible after a system failure. However, user data can be lost because of a system failure or a bad sector or a simple delete operations initiated by a user. NTFS does not protect user data, regular backups of data is highly recommended*

NTFS Transaction Log Recoverability

The file system processes each I/O operation that modifies metadata on an NTFS volume as a transaction. Each file on an NTFS volume is listed as a record in a special metadata file called the Master File Table (MFT). The first record in the table describes the MFT itself and the second record describes a second special file which mirrors the first few entries of the primary MFT. If the first MFT record is corrupted, NTFS uses the second record to find the MFT mirror file, the first record of which is identical to the first record of the MFT. The boot sector records the locations of both the MFT and MFT mirror files. The MFT mirror does not contain 100% of the entire MFT, rather it only contains the first few critical entries necessary for mounting the volume.

Windows 2000 stores a duplicate copy of the boot sector at the end of the logical disk.

The third record in the MFT is the log file which records all file transaction information. NTFS and the Log File Service use the DATA attribute of the log file to implement file system recoverability. The Log File Service is a component of the Windows NT Executive and the section below describes it in greater detail. Because the log file is a system file, it can be found early in the boot process and used to recover the disk volume, if necessary. When a user updates a file, the Log File Service records all redo and undo information for the transaction. For recoverability, redo information allows NTFS to roll the transaction forward (repeat the transaction if necessary), and undo allows NTFS to roll the transaction back if an error occurs. Here are the steps involved in committing data to the disk:

1. NTFS writes a log file record notating the volume update it intends to make
2. NTFS calls the cache manager to flush the log file record to disk
3. NTFS writes the volume update to the cache, modifying the cached metadata
4. The cache manager flushes the altered metadata to disk, updating the volume structure
5. NTFS writes a log file record which flags the transaction as having been completed.

If a transaction completes successfully, NTFS commits the file update to disk. If the transaction is not complete, NTFS ends or rolls back the transaction according to the undo information. If NTFS detects an error in the transaction, it rolls back the transaction. If NTFS cannot guarantee that a transaction completed successfully, it rolls the transaction back. Incomplete modifications to the volume are not allowed.

If the system crashes (due to power failure or other cause), NTFS performs three passes through the log on the disk: an analysis pass, a redo pass, and an undo pass. During the analysis pass, NTFS appraises the damage, if any, and determines which clusters it must update using the information in the log file.

The redo pass performs any steps logged from the last checkpoint. Then the undo pass rolls back any incomplete (uncommitted) transactions.

The NTFS recovery pass involves the following six steps:

1. When Windows NT recognizes an NTFS volume, it reads the MFT.
2. NTFS calls the Log File Service to open the log file. This causes the Log File Service Recovery to take place.
3. NTFS calls the Log File Service to read its restart area and reads all the data from the last checkpoint operation. This data initializes the transaction table, dirty pages table, and open file table so they can be used in the recovery process.
4. NTFS performs an analysis pass on its last checkpoint record. At the end of this pass, the transaction table contains only transactions that were active when the crash occurred.
5. NTFS performs a redo pass. At the end of this pass, the cache reflects the state of the volume when the crash occurred.
6. NTFS performs an undo pass. At the end of this pass, the volume is recovered to a stable state.

The Log File Service maintains two objects to support its functions:

- The restart area. A status area used to transfer information about a client's last checkpoint operation before a crash to the client's recovery procedure. The Log File Service maintains two restart areas to guarantee that at least one valid area is always available.
- The infinite log file. The Log File is a circularly reused file. When a new record is added, it is appended to the end of the file. When the Log File reaches its capacity, the Log File Service waits for writes to occur and frees space for new entries. This can be a very expensive (read: time-consuming) operation. When the log file gets "full" there is a perceptible pause in file-system activity that can significantly degrade performance if it happens repeatedly. Systems prone to this behavior should have their log file enlarged (see "chkdsk /L") and/or thought should be given to how I/O might be rebalanced.

Windows 2000 Improvements

The following is a summary of specific improvements made to CHKDSK:

1. The file (stage 1) and security (stage 3) verification stages were each speed up by as much as a factor of 3. Additionally both stage 1 & 3 of CHKDSK are now dual-threaded to maximise throughput.
2. The index verification stage was the most time consuming and complex stage in earlier versions of Windows NT. This has been improved from no change (special case) to at least nine times. Those volumes that take the longest time on NT4 CHKDSK will most likely hit the high end of the improvement.
3. The progress indicator for this index verification stage has been improved such that it is now more closely proportional to the amount of index entries checked. It will not be as likely to give the user the impression that CHKDSK is stuck because the progress indicator remains at a certain percentage for a long time when processing a directory containing the majority of the files in the volume.
4. Improvements have also been made in processing fairly fragmented/compressed files. In testing, CHKDSK was tasked to run on a single 0.5GB fragmented file and it took 30 seconds instead of 450 seconds, i.e. a fifteen fold (15) increase in speed.

5. Two options (/I, & /C) were added to CHKDSK to perform brief index verification and to skip cycle checking. When used together on a huge volume, a significant time saving can be achieved. This is particularly useful if the user does not care about the complete correction of directories
6. Improvements have also been made in processing files with huge amounts of data streams or huge attribute lists (usually due to huge file size and/or fragmentation/compression). In testing, CHKDSK was tasked to run on a single file with 5600 data streams and it took 208 seconds instead of 14,164 seconds. A sixty-eight (68) fold increase in speed.

Note: For more information on chkdsk, see the following Microsoft Knowledge Base article:

- Q187941 An Explanation of CHKDSK and the New /C and /I Switches
- Q314835 An Explanation of the New /C and /I Switches That Are Available

CHKDSK Performance

Microsoft has a dedicated team looking at all aspects of Windows 2000 performance. Tests were run on CHKDSK on 45GB (95% in use) volume with almost two million files and 236,500 indexes and obtained a 20-fold time decrease between NT4 SP3 and Windows 2000 from 33 hours down to 100 minutes.

Caveat: All figures were calculated from experiments run on fabricated test environments, and while every effort was made to replicate real world scenarios, these tests could not be made on the variety of hardware our customers have deployed or on the many combinations of disk and file configurations that exist and should only be used as a guide. It's the client's responsibility to apply best practice principles and evaluate the impact of the improvements in their specific environments.

Larger amounts of RAM will increase CHKDSK's performance in Windows 2000 and beyond.

CHKDSK Performance Results:

| | 2 Million Files | 3 Million Files |
|---------------------------|-----------------|-----------------|
| Windows NT 4.0 SP6 | 49.05 hrs | 103.7 hrs |
| Windows 2000 | 3.87 hrs | 6.03 hrs |
| Windows XP/.NET | .38 hrs | .696 hrs |

Important: To obtain the best CHKDSK performance, it is always recommended to apply the most recent Service Pack for your appropriate operating system.

Understanding CHKDSK

CHKDSK.EXE is the command-line interface for a program that verifies the logical integrity of a file system on Windows 2000. When CHKDSK encounters logical inconsistencies it takes actions to repair file system data, provided it is not in read-only mode.

The code that actually performs the verification when CHKDSK is run online resides in utility DLLs such as `Untfs.dll` and `Ufat.dll`. The verification routines invoked by CHKDSK.EXE are the same ones invoked when the Windows Explorer or Disk Administrator verifies a volume through the graphical user interface provided. When CHKDSK is scheduled to run at reboot, on the other hand, the binary module that contains the verification code is `AUTOCHK.EXE`.

`AUTOCHK.EXE` is a native Windows 2000 application that runs early enough in the system boot sequence that it does not have the benefit of Virtual Memory or other Win32 services. `AUTOCHK.EXE` generates the same kind of textual output that the utility DLLs invoked by CHKDSK.EXE does. But in addition to displaying this output on the screen during the boot process, `AUTOCHK.EXE` also logs an event to the Application Event Log for the system containing as much of the textual output as can fit into the event log's data buffer.

Because `AUTOCHK.EXE` and the verification code in the utility DLLs used by CHKDSK.EXE are based on the same source code, both will be referred to generically as "CHKDSK".

Since Windows NT 4.0 Service Pack 4 and in Windows 2000, two new switches (`/I` and `/C`) have been added to CHKDSK.EXE. These switches enable users to better manage downtime incurred by running CHKDSK or `AUTOCHK`. The switches are only valid when the target drive has the NTFS format. Each switch directs the CHKDSK routine to bypass certain actions it would otherwise take to validate the integrity of NTFS data structures.

Because the use of the `/C` and `/I` switches can result in a volume remaining corrupted even after CHKDSK completes, the use of these switches is not recommended except in situations where system downtime must be kept to an absolute minimum. These switches are intended to be used by users with exceptionally large volumes and who require flexibility in managing the downtime that is incurred when CHKDSK must be run on such volumes.

To understand when it might be appropriate to use these switches, it is important to have a basic understanding of some of the internal NTFS data structures, the kinds of corruption that can take place, what actions CHKDSK takes when it verifies a volume, and what the potential consequences are in circumventing CHKDSK's usual verification steps.

CHKDSK's activity is split into three major "passes" during which it examines all the "metadata" on the volume and an optional fourth pass. Metadata is "data about data." It is the file system overhead, so to speak, that is used to keep track of everything about all of the files on the volume. Metadata tells what allocation units make up the data for a given file, what allocation units are free, what allocation units contain bad sectors, and so on. The "contents" of a file, on the other hand, is termed "user data." NTFS protects its metadata using a transaction log. User data is not so protected.

Cluster Remapping

With transaction logging and recovery, NTFS ensures that the volume structure will not be corrupted, so all meta-data files remain available after a system crash. However, user data and meta-data can become at risk because of a system crash or a bad sector.

The NTFS file system implements a recovery technique called cluster remapping. When hardware returns a bad sector error to the file system, NTFS dynamically replaces the cluster containing the bad sector and allocates a new cluster for the data. If the error occurred during a read, NTFS returns a read error to the calling program, and the data is lost. When the error occurs during a write, NTFS writes the data to the new cluster, and no data is lost. The bad sector is recorded and not reused again. Bad sectors on the physical disk can result in data loss and corruption. Running a CHKDSK in a repair mode with the /F /R switches will manually initiate the checking of every sector on the disk, attempting data recovery, and remapping the bad sectors.

CHKDSK Phases

Phase One: Checking Files

During its first pass, CHKDSK displays a message that tells you that CHKDSK is verifying files and displays the percent of verification that is completed, counting from 0 to 100 percent. During this phase, CHKDSK examines each file record segment in the volume's master file table (MFT).

A specific file record segment in the MFT uniquely identifies every file and directory on an NTFS volume. The "percent completed" that CHKDSK displays during this phase is the percentage of the MFT that CHKDSK has verified. During this pass, CHKDSK examines each file record segment for internal consistency and builds two bitmaps, one representing the file record segments that are in use and the other representing the clusters on the volume that are in use.

At the end of this phase, CHKDSK has identified the space that is in use and the space that is available, both within the MFT and on the volume as a whole. NTFS keeps track of this information in bitmaps of its own, which are stored on the disk. CHKDSK compares its results with the bitmaps that NTFS keeps. If there are discrepancies, the discrepancies are noted in the CHKDSK output. For example, if a file record segment that was in use is found to be corrupted, the disk clusters that were associated with that file record segment are marked as "available" in the CHKDSK bitmap but are marked as "in use" in the NTFS bitmap. If discrepancies are found, as part of its repair operations CHKDSK replaces the NTFS bitmaps with the ones it generates.

Phase Two: Checking Indexes

During its second pass, CHKDSK displays a message that tells you that CHKDSK is verifying indexes and again displays the percent completed, counting from zero to 100 percent. During this phase, CHKDSK examines each of the indexes on the volume.

Indexes are essentially NTFS directories. The "percent completed" that CHKDSK displays during this phase is the percentage of the total number of the volume's directories and the objects in that directory that have been checked. During this pass, CHKDSK examines each directory that is on the volume, checking for internal consistency and verifying that every file and directory that is represented by a file record segment in the MFT is referenced by at least one directory. CHKDSK confirms that every file or subdirectory that is

referenced in a directory actually exists as a valid file record segment in the MFT and checks for circular directory references. Finally, CHKDSK confirms that the time stamps and file size information for the files are up-to-date in the directory listings for those files.

At the end of this phase, CHKDSK has made sure that there are no "orphaned" files and that all directory listings are for legitimate files. An orphaned file is a file for which there is a legitimate file record segment but for which there is no listing in any directory. An orphaned file often can be restored to its proper directory if that directory still exists. If the proper directory no longer exists, CHKDSK creates a directory in the root directory and places the file there. If CHKDSK finds directory listings for file record segments that are no longer in use, or for file record segments that are in use but that do not correspond to the file that is listed in the directory, CHKDSK simply removes the directory entry for the file record segment.

In Windows NT 4.0 CHKDSK does a directory-by-directory check of indexes, when CHKDSK encounters a directory with a large number of files, the CHKDSK percentage progress results can appear to hang. This has been improved in Windows 2000, and CHKDSK is not actually hung. A common mistake is to reboot the server with the assumption CHKDSK is hung. A good test to validate CHKDSK is still performing actions is to check drive activity. Some of the tasks CHKDSK performs can be CPU intensive, so even moderate to light disk activity is a sign that CHKDSK is still making progress. One factor that can significantly slow CHKDSK progress is the presence of bad sectors, since I/O operations that touch bad sectors will typically be retried multiple times with a delay between each retry.

Phase Three: Checking Security Descriptors

During its third pass, CHKDSK displays a message that tells you that CHKDSK is verifying security descriptors and, for the third time, displays "percent completed," counting from zero to 100 percent. During this phase, CHKDSK examines each security descriptor that are stored in the \$Secure metadata file for each files associations with files or directories that are on the volume.

Security descriptors contain information about ownership of a file or directory, about NTFS permissions for the file or directory, and about auditing for the file or directory. The "percent completed" that CHKDSK displays during this phase is the percentage of the volume's files and directories that have been checked. CHKDSK verifies that each security descriptor structure is well formed and is internally consistent, if they are not the permissions on the files will be set back to default. CHKDSK does not verify the actual existence of the users or groups that are listed or the appropriateness of the permissions that are granted.

Phase Four: Checking Sectors

The fourth pass of CHKDSK is only invoked if the /R switch is used. If the /R switch is in effect, CHKDSK runs a fourth pass and invokes 2 more phases to look for bad sectors in the volume's free space. CHKDSK attempts to read every sector on the volume to confirm that the sector is usable. Even without the /R switch, CHKDSK always reads sectors that are associated with metadata. Sectors that are associated with user data are read during earlier phases of CHKDSK if the /R switch is specified.

- Stage 4: Verifying file data: In this stage chkdsk attempts to read all the user data. If a read on a cluster fails, a fresh cluster is allocated in place of the bad cluster and the bad cluster is add to the list of bad clusters.
- Stage 5: Verifying free space: In this stage, chkdsk sends an IOCTL_DISK_VERIFY for the free sectors on the disk. Clusters containing bad sectors are added to the bad cluster list.

When CHKDSK finds an unreadable sector, it adds the cluster that contains that sector to the list of bad clusters. If the bad cluster is in use, CHKDSK allocates a new cluster to do the job of the bad cluster. If you are using a software fault-tolerant disk, NTFS recovers the bad cluster's data and writes the data to the newly allocated cluster. Otherwise, the new cluster is filled with a pattern of 0xFF bytes. Hardware fault-tolerant disk solutions are commonly capable of doing their own remapping of bad sectors, if this happens it is not necessary for NTFS to remap it as well.

If NTFS encounters unreadable sectors during the course of normal operation, NTFS remaps the sectors in the same way that it does when CHKDSK runs. Therefore, using the /R switch is usually not essential. However, using the /R switch is a convenient way to scan the entire volume if you suspect that a disk might have bad sectors.

How Long will CHKDSK Take to Run

The preceding description of the phases of running CHKDSK gives you only a broad outline of the most important tasks that CHKDSK performs to verify the integrity of an NTFS volume. CHKDSK also makes many additional specific checks during each pass and several quick checks between passes. However, even such a broad outline provides some basis for the following discussion of the variables that affect the amount of time that CHKDSK takes to run and of the impact of the new /C and /I switches that are available in Windows 2000.

Variable 1: The "Indexes" Phase:

During the first and third phases of running CHKDSK (checking files and checking security descriptors), the progress of the "percent completed" indicator is relatively smooth. Unused file record segments do require less time to process, and large security descriptors do take more time to process, but overall the "percent completed" is a fairly accurate reflection of the actual time that the phase requires. The second phase of CHKDSK is the one that typically takes the longest to run.

Variable 2: The Condition of the Volume:

Many factors that concern the state of a volume play a role in how long CHKDSK takes to run. A formula for predicting the time that is required to run CHKDSK on a given volume would have to include such variables as the number of files and directories, the degree of fragmentation of the volume in general and of the MFT in particular, the format of file names (long names, 8.3-formatted names, or a mixture), the number of bad sectors on the disk, and the amount of actual corruption that CHKDSK must repair.

Variable 3: Hardware Issues:

Hardware issues also affect how long it takes for CHKDSK to run. The variables include the amount of available memory, CPU speed, I/O throughput, (Fibre or SCSI, disk RPM speed, and so on).

Variable 4: The CHKDSK Settings:

If you do not use the /R switch, the biggest time concern on a given hardware platform is the number of files and directories that are on the volume, rather than the absolute size of the volume.

For example, without the /R switch, a 50-gigabyte (GB) volume that has only one or two large database files might take only seconds for CHKDSK to run. If you use the /R switch, CHKDSK has to read and verify every sector on the volume, which adds significantly to the time that is required for large volumes. On the other hand, running CHKDSK on even a relatively small volume might require hours if the volume has hundreds of thousands or even millions of small files--regardless of whether you specify the /R switch. Here is the answer to the question everyone asks:

There is no magical formula to be able to tell how long CHKDSK will take

No one has yet figured out a formula to be able to predict how long CHKDSK will take that accurately takes into account all the variables mentioned above. On the other hand, it is not unreasonable for customers to exercise "due diligence" by running CHKDSK against their own hardware configuration to eliminate a large number of variables. All else being equal, CHKDSK times will tend to increase at least linearly with respect to the total number of files and folders on a volume.

As you can see, running CHKDSK can take anywhere from a few seconds to several days, depending on your specific situation. The number one question Microsoft Product Support Services is asked is: "Exactly how long is CHKDSK going to take?", and the answer is we don't know. The best way to *predict* how long CHKDSK will take to run on a given volume is to actually do a trial run in read-only mode during a period of low system usage.

However, you must use this technique with great care, for the following reasons:

- In read-only mode, CHKDSK quits before it completes all three phases if it encounters inconsistencies, and CHKDSK is prone to falsely reporting errors when run in read-only mode. For example, CHKDSK may report disk corruption if NTFS happens to modify areas of a disk while CHKDSK is examining the disk. For correct verification, a volume must be static, and the only way to guarantee a static state is to lock the volume. CHKDSK locks the volume only if you specify the /F switch (or the /R switch, which implies /F). You may need to run CHKDSK more than once to get CHKDSK to complete all its passes in read-only mode.
- CHKDSK is both CPU- and disk-intensive. The time that it takes to run CHKDSK is affected by how much load is on the system and whether CHKDSK runs online or during the Windows 2000 startup sequence. Which factor becomes the bottleneck depends on the hardware configuration, but high CPU usage or heavy disk I/O while CHKDSK is running in read-only mode will inflate the CHKDSK running time. In addition, Autochk.exe runs in a different environment from that of Chkdsk.exe. Running CHKDSK through Autochk.exe gives CHKDSK exclusive use of CPU and I/O resources, but it also prevents CHKDSK from using virtual memory. Although you might expect Autochk.exe to run faster than Chkdsk.exe, Autochk.exe may actually take longer if the computer has relatively little available RAM.
- Fixing corruption adds to the time required. In read-only mode, CHKDSK runs to completion only if CHKDSK does not find any significant corruption. If a disk shows only minor corruption, you can predict that fixing the problems will not add much to the time that is required just to run CHKDSK. But if CHKDSK finds major damage, for example from a serious hardware failure, you can predict that the time that is required to run CHKDSK will increase in proportion to the number of damaged files that CHKDSK must repair. In extreme cases, this can more than double the time that it takes for CHKDSK to run.

What are My Options

When disk corruption is detected on a volume, you have three basic choices:

- **Run a Full CHKDSK**
This option repairs all file system data, restoring all user data that can be recovered by means of an automated process. The drawback to this option is that a full CHKDSK can require several hours of downtime for a mission critical server at an inopportune time. This is the recommended course of action.
- **Run an Abbreviated CHKDSK** using some combination of the /C and /I switches
This option repairs the severe kinds of corruption that can "snowball" into bigger problems in much less time than a full CHKDSK would require, but does not repair all the corruption that might exist. A full CHKDSK will still be required at some future time to guarantee that all the data that can be recovered has been recovered.
- **Do Nothing**
For a mission critical server that is expected to be online 24 hours a day, this is often the choice of necessity. The drawback to this option is that relatively minor corruption can "snowball" into major corruption if it is not repaired as soon as possible after it is detected. Therefore, this option should only be considered when keeping a system up is more important than the integrity of the data stored on the corrupted volume because all data on the corrupted volume should be considered "at risk" until CHKDSK is run.
- **Format the Partition and Restore from Tape**
When CHKDSK is run against a volume, there is a chance that it will not be able to correctly recover 100% of the data if there is extreme corruption. If you have a high-speed tape backup solution and a known last good backup, it may be just as fast or faster to perform a clean format the partition and restore the data from tape. This is a rare scenario and this option should only be considered under extreme situations with careful consideration.

Note: Running an abbreviated CHKDSK does not repair all of the corruption that might exist. You still need to run a full CHKDSK at some future time to guarantee that all recoverable data has in fact been recovered.

NTFS does not guarantee the integrity of user data after an instance of disk corruption, even if you immediately run a full CHKDSK operation. There might be files that CHKDSK cannot recover, and files that CHKDSK does recover might still be internally corrupted. It remains vitally important that you protect mission-critical data by performing periodic backups or by using some other robust method of data recovery.

Running CHKDSK

There are several ways that you can execute CHKDSK to run on a volume. You must be logged on with an account that is a member of the local Administrators group.

1. Go Start, Run, then type **CMD.EXE**
2. Once you have a Command Prompt open, type "**CHKDSK C:**" (*or whatever the volume letter is*) then press **Enter**.

This will run CHKDSK in read-only mode. Running CHKDSK in read-only mode will not correct any corruption, CHKDSK will be run against the volume and the results will be displayed to the screen. This can assist in determining if the volume is potentially corrupt, and a maintenance window needs to be scheduled so that a CHKDSK /F can be run against the volume. Care must be taken when reviewing the results of CHKDSK in read-only mode, because the volume is online and CHKDSK cannot lock the volume for exclusive use, any file CHKDSK runs across that is currently in-use, with an open handle, will be incorrectly reported by CHKDSK to be corrupt. If CHKDSK in read-only mode reports corruption one method to determine if there really is corruption is to run CHKDSK against the volume 3 – 4 times, then compare the results to see if the same files are being reported as corrupt. In-use files should have their handles released after a period of time, and in-use files should not appear across all instances. To set CHKDSK to repair the corruption, use one of the switches listed below.

Or

1. Open up My Computer or Windows Explorer
2. Right-click on the drive and select **Properties**
3. Select the **Tools** tab, under "**Error-checking**", click on **Check Now...**
4. Check the box for any options you wish to run
5. *Automatically fix the file system errors* will run a CHKDSK /F on the volume
6. *Scan for and attempt to recovery of bad sectors* will run a CHKDSK /R on the volume

Note: The /R switch in chkdsk implies /F. However, in the GUI, "*Scan for and attempt recovery of bad sectors*" does not imply "*Automatically fix file system errors*" be used. So just selecting "*Scan for and attempt recovery of bad sectors*" is not equivalent to a CHKDSK /F /R from the command line. Ideally both "*Scan for and attempt recovery of bad sectors*" and the "*Automatically fix file system errors*" checkbox should be both checked and used together if you want to run a CHKDSK /F /R.

7. Click **Start**

Initiating CHKDSK from the command line and the graphical user interface (GUI), has exactly the same results. Although, running chkdsk from the command line will provide more verbose output than the GUI.

CHKDSK Switches

The syntax used to initiate CHKDSK is the following:

CHKDSK [volume[[path]filename]] [/F] [/V] [/R] [/X] [/I] [/C] [/L[:size]]

Creates and displays a status report for a disk, based on the file system used. CHKDSK also lists and corrects errors on the disk. If CHKDSK cannot lock the drive, it will offer to check it the next time the computer restarts.

CHKDSK [drive:][[path] filename] [/f] [/v] [/r] [/l[:size]] [/x]

| Switch | Description |
|-----------------|---|
| none | Used without parameters, CHKDSK displays the status of the disk in the current drive. |
| drive: | Specifies the drive that contains the disk that you want CHKDSK to check. |
| [path] filename | Specifies the location and name of a file or set of files that you want CHKDSK to check for fragmentation. You can use wildcard characters (* and ?) to specify multiple files. |
| /f | <p>Fixes errors on the disk. The disk must be locked. If CHKDSK cannot lock the volume to run at the time of command execution, it presents the following dialog box before the prompt to schedule the command the next time the system restarts:</p> <p style="text-align: center;">CHKDSK cannot run because the volume is in use by another process. CHKDSK may run if this volume is dismounted first. ALL OPENED HANDLES TO THIS VOLUME WOULD THEN BE INVALID. Would you like to force a dismount on this volume? (Y/N)</p> <p>If the administrator schedules the command to run the next time the system restarts, CHKDSK does not set the "Dirty Bit" on an in-use volume in order to check the volume at the next boot. Instead, it sets a registry entry to tell AUTOCHK to run against that volume. The "Dirty Bit" is set by the file system itself only if it detects a problem. Versions of CHKDSK /F in Windows NT 4.0 and previous versions of Windows used to set the actual Dirty Bit.</p> |
| /v | FAT/FAT32 displays the name of each file in every directory as the disk is checked. For NTFS volumes, /v displays cleanup messages if any. |
| /r | Locates bad sectors and recovers readable information. The disk must be locked. |
| /l[:size] | NTFS only. Changes the log file size to the size you enter. Displays the current size if you do not enter a new one. |
| /x | Runs CHKDSK /F and forces a volume dismount to close open file handles on non-system volumes so it can be checked immediately. This eliminates the need of a potential reboot in order to perform the CHKDSK and repair the volume. |
| /i | NTFS only. Performs a less vigorous check of index entries, reducing the amount of time needed to run CHKDSK. |
| /c | NTFS only. Skips the checking of cycles within the folder structure, reducing the amount of time needed to run CHKDSK. |

Using the /C and /I Switches to Run an Abbreviated CHKDSK

The /C Switch:

The /C switch directs CHKDSK to skip the checks that detect cycles in the directory structure. Cycles are a very rare form of corruption in which a subdirectory has itself for an "ancestor."

Using the /C switch can speed up CHKDSK by about 1 to 2 percent, but using this switch can also leave directory "loops" on an NTFS volume. Such loops might be inaccessible from the rest of the directory tree, and some files might be orphaned in the sense that Win32 programs, including backup programs, cannot see the files.

The /I Switch:

The /I switch directs CHKDSK to skip the checks that compare directory entries to their corresponding file record segments. With this switch in effect, directory entries are still checked for internal consistency, but the directory entries are not necessarily consistent with the data that is stored in the corresponding file record segments.

How much time you will save by using the /I switch is difficult to predict.

Typically, the /I switch lowers CHKDSK times by 50 to 70 percent, depending on factors such as the ratio of files to directories and the speed of disk I/O relative to CPU speed.

Using the /I switch has these limitations:

- You may have directory entries that refer to incorrect file record segments. In this case, any program that tries to use such an entry will encounter errors.
- You may have file record segments that no directory entry references (another way that orphaned files occur). A file that is actually intact, as represented by the file record segment, may be invisible to all Win32 programs, including backup programs.

Note: For more information on chkdsk, see the following Microsoft Knowledge Base article:

- Q187941 An Explanation of CHKDSK and the New /C and /I Switches
 - Q314835 An Explanation of the New /C and /I Switches That Are Available
-

AUTOCHK

Every time Windows 2000 boots, AUTOCHK.exe is called by the Kernel to scan all volumes to check if the volume dirty bit is set. If the dirty bit is set, AUTOCHK performs an immediate CHKDSK /f on that volume verifying file system integrity and attempting to fix problems with the volume.

The "Dirty Bit" is a byte on the physical drive stored in the \$volume in the Master File Table (MFT), the only way to manually clear or set the dirty bit is by using the FSUtil.exe utility, once it has been set from an unclean shutdown / event.

AUTOCHK.exe in Windows 2000 distinguishes between a volume check that has been manually scheduled and one that is automatically scheduled because the file system found the volume to be in a "dirty" state, and then writes an associated message in the Application Event log. It does this by looking both at the dirty bit on each volume and for registry settings set by CHKDSK /F and CHKNTFS /X or /C to determine if a volume will be checked or skipped.

NOTE: CHKDSK /F or CHKNTFS /C to schedule a CHKDSK against a volume and CHKNTFS /X to exclude a volume from being checked override each other. This gives the administrator complete control over whether or not CHKDSK is run against a given volume unconditionally, conditionally, or not at all during boot. The last command issued sets/resets the registry entries accordingly.

If the administrator schedules CHKDSK to run the next time the system restarts, Chkdsk does not set the "Dirty Bit" on an in-use volume in order to check the volume at the next boot. Instead, it sets a registry entry to tell Autochk to run against that volume. The "Dirty Bit" is only set by the file system if it detects a problem.

If you schedule a CHKDSK to run at boot time against a volume, and at boot time decide to bypass AUTOCHK by pressing 'any key', AUTOCHK does not run against that volume and removes the registry entry so AUTOCHK is no longer scheduled to run against that volume in the future. In certain situations, it may be helpful to bypass running this CHKDSK. By modifying the following registry key, the amount of time Windows waits before running Autochk.exe can be changed, allowing the user time to bypass running it by pressing any key.

`HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\Session Manager \AutoChkTimeOut`

If the entry is not found, the default count down of 10 seconds will be used. If it is set to 0 second, there will not be any count down. If it is set to more than 3 days (259,200 seconds), the default value will be used.

When AUTOCHK runs against a volume at boot time it records its output to a file called Bootex.log in the root of the volume being checked. The Winlogon service then moves the contents of each Bootex.log file to the Application Event log.

One event log message for each volume checked is recorded as follows:

```
Event-ID: 1001
Source: Winlogon
Description: This includes file system type; drive letter or GUID, and volume name or serial number to help determine what volume CHKDSK ran against. Also included is whether CHKDSK ran because a user scheduled it or because the dirty bit was set.
```

Additional entries that can be found in BootExecute are:

| Registry value | Function |
|---------------------|--|
| /P \??\Volume: | Schedules an unconditional CHKDSK against the volume. |
| /p \??\VOLUME{GUID} | Schedules an unconditional CHKDSK against a volume mount point. |
| /k:Volume * | Excludes CHKDSK from running against the volume. |
| /m \??\Volume: | Tells AUTOCHK to look only at the dirty bit on the volume, and if set, run CHKDSK. |

| Sample command | Resulting registry entry |
|----------------|-----------------------------|
| CHKDSK C: /F | Autocheck AUTOCHK /p \??\C: |

| | |
|-------------------------|---------------------------------------|
| CHKDSK C:\mountpoint /F | Autocheck AUTOCHK /p \??\VOLUME{GUID} |
| CHKNTFS D: E: /X | Autocheck AUTOCHK /k:D /k:E * |
| CHKNTFS G: /C | Autocheck AUTOCHK /m \??\G: |

NOTE: Volumes that are shared between Server Cluster nodes running the Cluster service do not allow access to the volume at boot time to AUTOCHK.exe. When the volumes are brought online the Cluster service performs the similar functionality as running CHKNTFS.EXE to check if the dirty-bit is set and to see if the volume is dirty. If it is, CHKDSK /F runs against the volume before being brought online. This does not check the BootExecute registry value so it does not take these values into account.

Preventing Autocheck from running:

When you run the "CHKDSK /F /R" command, Windows asks if you want to schedule CHKDSK to run the next time the system is restarted. To stop the execution of CHKDSK /F /R take the following steps:

1. Start Registry Editor (REGEDT32.EXE)
2. From the HKEY_LOCAL_MACHINE hive, go to the following subkey:

\SYSTEM\CurrentControlSet\Control\Session Manager

3. Change the BootExecute entry from:
 - a. autocheck autochk * /r\DosDevices\C:

To:

- b. autocheck autochk *

If you have scheduled CHKDSK for multiple volumes, there will be an autocheck entry for each volume. Delete the string from the BootExecute registry value for each volume that you do not want checked. You can also use this key to determine which volumes will be checked at boot time.

NOTE: This is the default setting for AUTOCHK and also the result of using CHKNTFS /d to have all volumes checked at boot time.

CHKNTFS

The Chkntfs.exe utility is designed to view that status of and disable the automatic running of chkdsk on specific volumes, when Windows restarts from an improper shutdown. Chkntfs can also be used to unschedule a chkdsk if chkdsk /f was used to schedule a chkdsk on an active volume on the next system restart and see if the dirty bit is set

Chkntfs is a utility that enables a system administrator to exclude volumes from being checked by the autochk program. The utility is run from a command prompt and has the following command line options:

Displays or specifies whether automatic system checking is scheduled to be run on a FAT, FAT32, or NTFS volume when the computer is started.

You must be a member of the local Administrators group to use the CHKNTFS command.

CHKNTFS [/t[:time]] [/x] [/c] volume: [...]

CHKNTFS /d

| Switch | Description |
|-------------------------|--|
| CHKNTFS /X | Excludes a drive from the default boot-time check. Excluded drives are not accumulated between command invocations. |
| CHKNTFS <i>volume</i> | Displays the file system type, the status of the file system dirty bit, and whether or not CHKDSK is manually scheduled to run against the volume at boot time. |
| CHKNTFS /C | Schedules a volume to be checked at boot |
| CHKNTFS /D | Restores the computer to default behavior and removes registry settings invoked by CHKDSK /F or CHKNTFS /X. This means all drives are checked at boot time and CHKDSK is run against those found to be dirty. |
| CHKNTFS /t: <i>time</i> | Changes the AUTOCHK time-out value to the specified amount of time in seconds. If the time is not specified, displays the current settings. Though you can set countdown time to zero, doing so prevents the user/operator from canceling a potentially time-consuming automatic file check. |

If no switches are specified, CHKNTFS will display dirty or scheduled to be checked on next reboot.

Examples:

chkntfs /x c: This disables chkdsk from running on drive C:

chkntfs /x d: e: This disables chkdsk from running on drives D: and E:.

The *chkntfs /x* commands are not cumulative; the command overwrites any previous drive exclusions that have been established. In the above example, *chkntfs* only disables the *chkdsk* checking on drives D and E, drive C is still checked for the presence of a dirty bit.

The *chkntfs* utility also works by modifying the *BootExecute* value in the system registry. The *BootExecute* value is located in the following registry key:

HKEY_LOCAL_MACHINE\SYSTEM\CURRENTCONTROLSET\CONTROL\Session Manager

The default value is:

*BootExecute:REG_MULTI_SZ:autocheck autochk **

Chkntfs /X will also add a */K* parameter prior to the asterisk. The */k* parameter excludes volumes from being checked for the presence of a dirty bit.

For example, the command "*chkntfs /x D:*" would modify this registry entry to *autocheck autochk /k:d **

Chkdsk /f schedules *autochk.exe* to run at the next reboot by setting a key in the registry. *Chkntfs /X* disables the checking for this registry key. *Autochk* will not run during boot on volumes that are excluded from dirty bit checking by *chkntfs*.

In order to run a *chkdsk /f* on a drive that has been excluded by the *chkntfs* utility, you must run the *chkntfs /d* option to return the system to its normal state or edit the *BootExecute* value in the registry and remove the applicable drive letter from the */k* parameter.

The Chkntfs.exe utility treats all drives as local to the node. This includes Server Cluster (MSCS) physical disk resources on a shared drive array.

Note: For more information on CHKNTFS, see the following Microsoft Knowledge Base articles:

- Q191603 Modifying the Autochk.exe Time-out Value
 - Q160963 CHKNTFS.EXE: What You Can Use It For
-

CHKDSK and Server Clusters

Running CHKDSK on a Server cluster is the same as a stand-alone server, with a few considerations with how to initiate CHKDSK.

- A filter driver called ClusDisk.sys blocks all access to the shared disks, until the Physical Disk resource has come online, this means that not all CHKDSK functionality during boot (Autochk.exe) is able to see the disks and run.
 - When the cluster service brings a disk online, it only checks to see if the dirty bit is set on the physical disk
 - There is no way to initiate CHKDSK via a registry setting (Chkntfs /c or CHKDSK /F) on a Server Cluster. Chkdsk can only be initiated by setting the dirty bit or bringing down the entire cluster.
 - Use FSUtil.exe to set the dirty bit, instead of Chkntfs /C to set the registry key on a Cluster
- Note:** See the appendix for more information on FSUtil.exe
- Before Running CHKDSK /F against a volume, bring all other resources in the group that the disk is in, offline.

The Cluster Service also has some enhanced volume corruption checking and functions that are more granular added to give administrators more control over CHKDSK running.

When attempting to bring a physical disk resource online in Cluster Administrator, you may see one or more of the following symptoms:

- The resource may fail, or it may come online.
- The disk resource may appear 'hung' in an Online Pending state
- Chkdsk /F may run on the drive.
- The following event may be logged in the System Event log:

Event ID 1066

Cluster resource Disk Y:: is corrupt. Running ChkDsk /F to repair problems.

When the Cluster Service brings a physical disk resource online it checks the physical disk to see if the dirty bit is set on the disk. If the dirty bit is set, the default behavior is that chkdsk /F will be run on the disk before mounting the volume and making it available to the system.

As an additional check to detect a potential corruption problem, the following steps are taken by the Cluster service when it brings a physical disk resource online:

1. Opens a temporary file, at the root of the drive: zClusterOnlineChk.tmp.
2. Writes a test pattern, and saves the file.
3. Reopens the file, reads the data and verifies the data.
4. Deletes the file.

If any part of this fails, the cluster service assumes that the disk is corrupt and Chkdsk /F is run.

If for any reason the cluster service can physically attach to the disk, but cannot mount the volume. The cluster service will attempt to run CHKDSK /F on the volume in the event that it is corrupt. This can sometimes be misleading when troubleshooting, because the problem with the volume may not always be corruption but another volume mounting issue.

Here is an example of a Cluster log capture of the online process and chkdsk being run:

```
00000584.000005dc::YYYY/MM/DD-HH:MM:SS.SSS Physical Disk <Disk G:>: DriveLetterIsAlive called for Online check
00000584.00000430:: YYYY/MM/DD-HH:MM:SS.SSS Physical Disk: PnP Event GUID_IO_VOLUME_MOUNT for G
(Partition1) received.
000009d4.000009f4:: YYYY/MM/DD-HH:MM:SS.SSS [EVT] s_ApiEvPropEvents: Calling into EvpPropPendingEvents,
size=648...
000009d4.000009f4:: YYYY/MM/DD-HH:MM:SS.SSS [EVT] s_ApiEvPropEvents: Called EvpPropPendingEvents...
00000584.000005dc:: YYYY/MM/DD-HH:MM:SS.SSS Physical Disk <Disk G:>: DisksOpenChkdskLogFile: Chkdsk output is in
file: E:\WINNT\CLUSTER\Chkdsk_Disk3_Sig3252EAA8.log
00000584.00000430:: YYYY/MM/DD-HH:MM:SS.SSS Physical Disk: PnP Event GUID_IO_VOLUME_LOCK for 587488
received
00000584.00000430:: YYYY/MM/DD-HH:MM:SS.SSS Physical Disk: PnP Event GUID_IO_VOLUME_DISMOUNT for 587488
received
00000584.00000430:: YYYY/MM/DD-HH:MM:SS.SSS Physical Disk: PnP Event GUID_IO_VOLUME_UNLOCK for 587488
received
00000584.000005dc:: YYYY/MM/DD-HH:MM:SS.SSS Physical Disk <Disk G:>: FixCorruption: CHKDSK returned status of 2.
00000584.000005dc:: YYYY/MM/DD-HH:MM:SS.SSS Physical Disk <Disk G:>: DisksMountDrives: letter mask is 00000040.
00000584.00000430:: YYYY/MM/DD-HH:MM:SS.SSS Physical Disk: PnP Event GUID_IO_VOLUME_MOUNT for G
(Partition1) received.
```

How to Run CHKDSK on a Cluster

Running CHKDSK on a Cluster is exactly the same as a stand-alone server. When you try to run the "*chkdsk /f*" or "*chkdsk /f /r*" command on a cluster shared drive, Chkdsk may not run and may state that the drive could not be locked for exclusive use. If you schedule Chkdsk to run after the computer restarts, Chkdsk may generate the following error message during the start up process:

Cannot determine file system on drive \??\<drive letter>.

Under most circumstances, running Chkdsk with the /F or /R switch requires the computer to be restarted because of open handles on the disk. Typically, there are no services or drivers running that prevent AUTOCHK from checking the disk when the computer restarts. However, if you are using Server Clustering, the file system does not mount the shared disk until the Cluster service starts and brings the Physical Disk resources online. The ClusDisk.sys driver blocks all access to the disks until they are brought online, this causes Chkdsk to report that it cannot determine the file system on a shared cluster disk. Running Chkdsk in Read-Only mode may seem to work, but Chkdsk does not fix any problems. FSUtil.exe is a utility from Windows XP / .NET that can be used on a Windows 2000 Server Cluster to set the dirty bit on the physical drive, then when the disk is failed over or brought online the next time, Chkdsk /F will be run against the drive.

If you suspect that there is file corruption on the shared disk, use the following steps to close all open handles to the shared disk and run Chkdsk on the drive:

1. Quit all programs, stop all non-cluster-aware services and log on with an account that has Administrative rights.
2. Start the **Cluster Administrator** tool, right-click the cluster name, and then click **Properties**.
3. On the **Quorum** tab, note which hard disk is the quorum disk. If the hard disk on which you want to run Chkdsk contains the quorum log, temporarily move the quorum to another shared disk.

Note: For more information on chkdsk, see the following Microsoft Knowledge Base article:

- Q280353 How to Change Quorum Disk Designation

-
4. Use the Cluster Administrator tool to find the group that contains the shared hard disk on which you want to run Chkdsk.
 5. After you find the physical disk resource on which you want to run Chkdsk, take the entire group offline, including the shared disk. This closes all of the handles to the physical disk. To take the group offline, right-click the group name, and then click "**Take offline**".
 6. In the Cluster Administrator, click just the shared disk on which you want to run Chkdsk on, and then bring it online. To do this, right-click the disk resource, and then click "**Bring online**".

NOTE: If the dirty bit was previously set, Chkdsk may automatically run and the Physical Disk resource may stay in an "*Online Pending*" state and take awhile to come online. In Windows 2000 you will see a Command Prompt window with Chkdsk running and if you open Task Manager you will see Chkdsk running as a process.

7. At a command prompt, change to a drive other than the drive on which you are attempting to run Chkdsk, and then type the following command, where <X> is the shared disk:

"CHKDSK <x>: /F" (without the quotation marks)

If you receive a "Disk cannot be locked" error message when you try to run Chkdsk, verify that all services and tools that have access to the drive are stopped, and then try to run Chkdsk again. Any running service or program that has an open handle to the drive can prevent Chkdsk from running. Windows 2000 and later versions of Windows can attempt to close open handles to the shared disk by using the /X switch. If you are prompted to close open handles, press the Y key.

If Handles Remain Open or the Cluster Contains a Single Shared Disk

If programs or drivers maintain an open handle to the shared disk, or if there is only a single shared disk (on which the quorum log is stored), you may need to take the entire cluster down to manually run CHKDSK (one exception to this would be under Windows 2000 you could set the dirty bit using FSUtil.exe). Doing this requires that you disable the clustering components temporarily so that the file system can mount the shared disk when you restart the node. You must also shut down the other nodes in the cluster so that they do not take ownership of the shared disk when the node restarts. To do this, use the steps in the appropriate section.

Windows NT 4.0:

1. Shutdown node B.

2. Log on to node A as an Administrator.
3. Run the "**chkdsk /F**" command on the shared disk. When you are prompted to schedule Chkdsk to run when the computer next restarts, Press Y.
4. In the Devices tool in Control Panel, click **Cluster Disk**, and then click Startup.
5. Change the Startup type to **Disabled**.
6. In the Services tool in Control Panel, click the **Cluster Server** service, and then click Startup.
7. Change the Startup type to **Disabled**.
8. Quit Control Panel, and then restart node A. Chkdsk will run during boot without interference from the Cluster Disk driver or any other application / driver.
9. After Chkdsk is finished, change the Startup type back to its original setting, and then restart node A.
10. Turn on node B.

Windows 2000:

1. Quit all programs, stop all non-cluster-aware services and log on with an account that has Administrative rights.
2. Start the **Cluster Administrator** tool, right-click the cluster name, and then click **Properties**.
3. On the **Quorum** tab, note which hard disk is the quorum disk. If the hard disk on which you want to run Chkdsk contains the quorum log, temporarily move the quorum to another shared disk.
4. Copy FSUtil.exe from the %SystemRoot%\System32 directory from a Windows XP or Windows .NET Server to the local hard drive on the Windows 2000 Server Cluster
5. At a command prompt, change to change to that directory, and then type the following command, where <X> is the shared disk:
6. "*FSUTIL DIRTY SET <x>*:" (without the quotation marks)
7. Use the Cluster Administrator tool to find the group that contains the shared hard disk on which you want to run Chkdsk.
8. After you find the physical disk resource on which you want to run Chkdsk, take the entire group offline, including the shared disk. This closes all of the handles to the physical disk. To take the group offline, right-click the group name, and then click "**Take offline**".
9. After CHKDSK completes running on the volume, bring all other resources in the group Online

Cluster resource

Windows 2000 includes enhanced Physical Disk Resource private properties when you are using a Server Cluster. Some of these enhancements provide administrators better control over when CHKDSK is run against a cluster Disk Resource.

Differences Between Microsoft Windows NT 4.0 and Windows 2000

When you run a command to display the private properties of the resource named Disk Q: as displayed in Cluster Administrator, note the difference in output between Windows NT 4.0 and Windows 2000:

Command: `"cluster <clustername> res "Disk Q:" /priv"` (without the quotation marks)

Windows NT 4.0 Cluster:

Private properties for Physical Disk Resource 'Disk Q:':

```
R Name                                     Value
-----
Signature                                 1415371731 (0x545cdbd3)
DiskInfo                                  <Unknown Type>
```

Windows 2000 Cluster:

Private properties for Physical Disk Resource 'Disk Q:':

```
T Resource      Name                                     Value
-----
D Disk Q:      Signature                                 1415371731 (0x545cdbd3)
D Disk Q:      SkipCHKDSK                                  0 (0x0)
D Disk Q:      ConditionalMount                            1 (0x1)
B Disk Q:      DiskInfo                                    03 00 00 00 ... (264 bytes)
B Disk Q:      MountVolumeInfo                             D3 DB 5C 54 ... (104 bytes)
```

New Private Properties in Windows 2000

When comparing the Private Properties, the new ones introduced in Windows 2000 are:

- SkipCHKDSK
- ConditionalMount
- MountVolumeInfo

The behaviour of CHKDSK depends on the values assigned to the SkipCHKDSK and ConditionalMount properties. If the disk is found to be corrupt by checking the dirty bit, or in the case of the quorum drive the MSCS folder is inaccessible,

CHKDSK behaves as follows on cluster disk resources:

| Registry Key | Description |
|---|--|
| SkipCHKDSK = 1 | The Cluster service does not run CHKDSK against the dirty drive and mounts the disk for immediate use. <i>NOTE: SkipCHKDSK = 1 overrides the ConditionalMount setting and the Cluster service behaves the same no matter what the ConditionalMount property is set to.</i> |
| SkipCHKDSK = 0 and ConditionalMount = 0 | The Cluster service fails the disk resource and leaves it offline. |
| SkipCHKDSK = 0 and ConditionalMount = 1 | The Cluster service runs "CHKDSK /f" against the volume found to be dirty and then mounts it, after it has been repaired. This is the current default behaviour for Windows 2000 clusters and the only behaviour for Windows NT 4.0 clusters. |

These resource private properties can be modified using the following commands:

- `cluster <clustername> res "Disk Q:" /priv SkipCHKDSK=0[1]`
- `cluster <clustername> res "Disk Q:" /priv ConditionalMount=0[1]`

Chkdsk Status Codes in Cluster Log

When analysing the Cluster.log during a Physical Disk's Online process, you may see the following error message:

FixCorruption: CHKDSK returned status of <error code>.

Here is an example of what the cluster log will display:

00000584.000005dc::YYYY/MM/DD-HH:MM:SS.SSS Physical Disk <Disk Q:>: FixCorruption: CHKDSK returned status of 2.

The following error codes may be displayed:

| Code | Meaning |
|------------|--|
| 0 | Success, no errors found. |
| 1 | Success, CHKDSK has detected and fixed major errors. |
| 2 | Success, CHKDSK has detected and fixed minor inconsistencies |
| 3 | Error, CHKDSK could not complete |
| 3221225786 | CHKDSK was aborted. 3221225786 is C000013A(hex) or STATUS_CONTROL_C_EXIT |

Capturing Chkdsk Results for Server Clusters

When you bring a cluster server online and it has the dirty bit set chkdsk will run to verify the integrity of the cluster server. The version of the Windows operating system that your computer is running determines the degree of chkdsk logging that is available:

- Windows NT 4.0: A status code is logged in the Cluster log.
- Windows 2000: Results are logged in the Application and status codes in the Cluster log.
- Windows 2000 Service Pack 1 (SP1) and Windows Datacenter Server: Results are logged in the Application log and status codes in the Cluster log. In addition, the Cluster log also references a log file in which detailed chkdsk output is recorded.

Windows NT 4.0

On a Windows NT 4.0 Server Cluster there is no detailed information saved from CHKDSK running. The physical disk resource will appear to be in a hung in an Online Pending state, with no notification or logging as CHKDSK is being run in the background.

To capture detailed information about chkdsk, you need to run Autochk during boot. To do this you need to disable the cluster service and schedule chkdsk to run the next time that you restart one of the nodes, and have the results logged in the Application log as part of the autochk process:

1. Shut down all but one node in the cluster.
2. On the remaining node, use the following settings:
 - a. Cluster Disk – Manual
 - b. Cluster Service – Manual
3. Run chkdsk by typing the following line, and then click Yes to schedule it for the next time that you restart the cluster server:
 - a. `chkdsk <drive>: /f`

Note: If no open handles are present on the disk and chkdsk runs immediately, the chkdsk results appear on the screen instead of being logged in the Application log. To ensure that chkdsk is scheduled to run the next time that you restart the cluster server, change the directory to the chosen drive before you run chkdsk.

4. Restart the node and Autochk.exe will run on boot-up .The autochk results are logged in the Application log with the source "AUTOCHK."

Note: Not all of the data may fit in the single Application log entry; some data may be truncated for post analysis.

5. Reset Cluster Disk and Cluster Server back to the original settings:
 - a. Cluster Disk – System
 - b. Cluster Service – Automatic
6. Restart the node. Verify that the Cluster Service starts, and then bring all other nodes online.

You can also pipe the results of chkdsk run while the server is up to a text file. Make sure that no programs or services have handles open on the drive. Start chkdsk manually, and then redirect the output to a text file. For example:

```
chkdsk x: /f >output.txt 2>error.txt
```

Windows 2000

Here are some examples of chkdsk output for a cluster server that is running a version of Windows 2000 earlier than SP1:

- Cluster log
 - Physical Disk <Disk S:>: FixCorruption: chkdsk returned status of 1
- System log

Event Source: ClusSvc

Event ID: 1066

Description: Cluster disk resource <Disk S>: is corrupt. Running Chkdsk /F to repair problems.

- Application log

Event Source: Chkdsk

Event ID: 26180

Description:

Checking file system on S:

The type of the file system is NTFS.

Volume label is S_Drive.

The MFT mirror is different from the MFT.

Correcting errors in the Master File Table (MFT) mirror.

Windows has made corrections to the file system.

528367 KB total disk space.

15547 KB in 26 files.

20 KB in 24 indexes.

0 KB in bad sectors.

4643 KB in use by the system.

4096 KB occupied by the log file.

508157 KB available on disk.

1024 bytes in each allocation unit.

528367 total allocation units on disk.

508157 allocation units available on disk.

Note: For more information on CHKDSK information in Cluster logs, see the following Microsoft Knowledge Base article:

- Q265533 Explanation of Chkdsk Status Codes in Cluster Log
- Q272244 Location of the Chkdsk Results for Windows Clustering Resources
- Q176970 How to Run the CHKDSK /F Command on a Shared Cluster Disk

Windows 2000 Advanced Server SP1 and Datacenter

In Windows 2000 Advanced Server SP1 and Datacenter, additional logging is placed in the Cluster.log that references an additional log file. The additional log file has a name in the following format:

Chkdsk_Disk#_SigXXXXXXXXX.log

Note: This chkdsk log file represents the last chkdsk results for that specific hard disk. For example, if Disk 2 has two partitions, and you ran chkdsk against both of them, only the results of the last partition will be recorded in the log file.

*This functionality is only provided by the Physical Disk Resource type provided by Microsoft. Other disk resource types provided by 3rd party vendors may **not** support this feature.*

Here is an example of a Cluster.log capture:

00000584.000005dc::YYYY/MM/DD-HH:MM:SS.SSS Physical Disk <Disk Q:>:
DisksOpenChkdskLogFile: Chkdsk output is in file:
E:\WINNT\CLUSTER\Chkdsk_Disk3_Sig3252EAA8.log

Followed by:

00000584.000005dc::YYYY/MM/DD-HH:MM:SS.SSS Physical Disk <Disk Q:>: FixCorruption:
CHKDSK returned status of 2.

Example Chkdsk_Disk#_SigXXXXXXXXX.log

Note: The type of the file system is the NTFS file system and the volume label is S_Drive.

```
chkdsk is verifying files (stage 1 of 3)...
0 percent completed.
1 percent completed.
.....
98 percent completed.
100 percent completed.
File verification completed.
chkdsk is verifying indexes (stage 2 of 3)...
0 percent completed.
1 percent completed.
.....
90 percent completed.
100 percent completed.
Index verification completed.
CHKDSK is verifying security descriptors (stage 3 of 3)...
0 percent completed.
1 percent completed.
.....
98 percent completed.
100 percent completed.
Security descriptor verification completed.
Correcting errors in the Master File Table (MFT) mirror.
Windows has made corrections to the file system.

528367 KB total disk space.
15547 KB in 26 files.
20 KB in 24 indexes.
0 KB in bad sectors.
4643 KB in use by the system.
4096 KB occupied by the log file.
508157 KB available on disk.

1024 bytes in each allocation unit.
528367 total allocation units on disk.
508157 allocation units available on disk.
```

Conclusion

Running CHKDSK and correcting file system corruption after it has occurred is only a reactive measure. The key aspect is to understand what CHKDSK is, what it does, and what your options are to correct the corruption. After the corruption has been addressed, corrective steps must be taken to identify the source of the corruption and correct it.

For More Information

For more information about Windows 2000 Cluster service, see the following resources:

- [Q187941 An Explanation of CHKDSK and the New /C and /I Switches](#)
- [Q191603 Modifying the Autochk.exe Time-out Value](#)
- [Q160963 CHKNTFS.EXE: What You Can Use It For](#)
- [Q280353 How to Change Quorum Disk Designation](#)
- [Q265533 Explanation of Chkdsk Status Codes in Cluster Log](#)
- [Q272244 Location of the Chkdsk Results for Windows Clustering Resources](#)
- [Q176970 How to Run the CHKDSK /F Command on a Shared Cluster Disk](#)
- [Q218461 Enhanced Chkdsk, Autochk, and Chkntfs Tools in Windows 2000](#)
- [Inside Windows 2000, Third Edition by Microsoft Press](#)

Appendix

How to Automate CHKDSK

If Chkdsk is to run in /F (fix) mode to attempt to remedy corruption, in most cases, there will be locked files and it requires restarting the machine to have exclusive access to the partition or partitions. This process can be automated and requires no user intervention. Usually, a user needs to press "y" to schedule Chkdsk to run in write-mode on the next restart, however, this process can be automated as follows:

Create a batch file and then disseminate it to the preferred system or systems:

```
@echo off
echo y|chkdsk [target drive, i.e. D:] /f/r
rem sleep 3600
rem c:\utils\shutdown.exe /l /r /y /t:6
```

The last two lines are optional. Sleep.exe and Shutdown.exe are from the Windows Resource Kit. "Sleep 3600" causes the system to wait for 60 minutes before proceeding to the next line in the batch file. Shutdown.exe is then called to shut down and restart the target system. If Shutdown.exe is not called, and the drive could not be locked for exclusive use, Chkdsk runs the next time you manually restart the target system.

In those cases where you would like Chkdsk, or the above batch file, to be scheduled to run on specific days or times use the Task Scheduler or AT Scheduler (command-line interface)

FSUtil.exe

Included in Windows XP and Windows .NET is a new command line utility called FSUtil.exe. FSUtil has several uses beyond the scope of this CHKDSK paper that will not be discussed. The key aspect that you can do with FSUtil that was not previously possible is that the "Dirty Bit" byte in the MFT on the physical disk can be modified. It can be used to set the dirty bit on a Windows 2000, Windows XP, or Windows .NET computer.

The command to be used with FSUtil is 'dirty' to manage the volumes dirty bit. The 'set' or 'query' commands can then be used to either change or check the status of the dirty bit. CHKNTFS.EXE can be used to check if the Dirty Bit is set. However, it cannot be used to set the Dirty Bit.

FSUtil Syntax:

To query the current status of the dirty bit on a volume, use the following command:

```
fsutil dirty query <drive>:
```

To manually set the dirty bit on a volume, use the following command:

```
fsutil dirty set <drive>:
```

For both commands, replace <drive> with the drive letter of the volume that you want to manage.

For example: To query the dirty bit on drive C, type:

fsutil dirty query C:

Sample output:

Volume C: is dirty

or

Volume C: is not dirty

To set the dirty bit on drive C, type:

fsutil dirty set C: