Windows XP Professional

Chapter 27 - Troubleshooting Disks and File Systems

Hard disk and file system errors can result from a variety of problems, such as hardware failures, power outages, poor system maintenance, viruses, and human error. When you are troubleshooting problems related to disks and file systems, you can refer to this chapter for information about troubleshooting tools, volume and disk error conditions, viruses, and stop messages. You can also use this chapter to obtain detailed descriptions of the master boot record (MBR), the GUID partition table (GPT), and the boot sectors.

In This Chapter

Related Information

- For more information about the file allocation table (FAT) and NTFS file systems, see "File Systems" in this book.
- For more information about managing disks and volumes, see "Disk Management" in this book.
- For more information about the tools that Microsoft® Windows® XP Professional provides for troubleshooting, see "Tools for Troubleshooting" in this book.
- For more information about how to troubleshoot, see "Troubleshooting Concepts and Strategies" in this book.

New in Troubleshooting Disks and File Systems

Microsoft® Windows® XP Professional provides improved troubleshooting tools for disks and file systems. Table 27.1 summarizes the enhancements made from Microsoft® Windows® 2000 to Windows XP Professional.

Table 27.1 Enhancements Since Windows 2000

New Feature	Feature Description
Automated System Recovery (ASR)	Automated System Recovery (ASR) is a two-part recovery system that allows you to restore the operating system state by using files saved to tape media and hard disk configuration information saved to a floppy disk.
Disk Defragmenter has new capabilities, including a command-line option.	Windows XP Professional offers two choices for defragmenting disks: the Disk Defragmenter snap-in and a new command-line version of the tool (Defrag.exe). Both tools can defragment NTFS volumes that have cluster sizes larger than 4 KB and files smaller than 16 clusters. Both tools can also defragment the master file table (MFT) on NTFS volumes.
Troubleshoot disks and volumes at the command line by using DiskPart.	Use the new command-line tool DiskPart to troubleshoot disks and volumes at the command line as an alternative to using the Disk Management snap-in.
Create GUID partition table disks on Itanium-based computers.	Windows XP 64-Bit Edition supports a new partition style called GUID partition table (GPT). GPT disks contain redundant partition tables for improved partition structure integrity.
Use the Fsutil.exe tool to determine whether a volume is marked as dirty.	The Fsutil.exe command-line tool offers many commands that you can use to manage file system behavior. For example, you can use the **fsutil dirty** command to determine if a volume is dirty. If a volume is dirty, it has experienced file system errors, and you must run Chkdsk on the volume to repair the problem. You can also use the **fsutil dirty** command to mark a volume as dirty.

If you are migrating from Microsoft® Windows NT® version 4.0, the enhancements in Table 27.2 apply in

addition to those outlined in Table 27.1.

Table 27.2 Enhancements Since Windows NT 4.0

New Feature	Feature Description
Chkntfs.exe	Chkntfs.exe is a command-line tool that you can use to disable the automatic running (at restart) of Chkdsk on dirty volumes. You can also use Chkntfs to cancel a scheduled session of Chkdsk so that it does not run when the computer restarts.
New Chkdsk parameters	Chkdsk offers two new parameters, /**c** and /**i**, to reduce the length of time required to run Chkdsk on NTFS volumes.
Recovery Console	Recovery Console is a command-line startup environment that allows you to gain access to the hard disk for basic troubleshooting and system maintenance.
Dmdiag.exe	Dmdiag.exe is a command-line tool that displays the location and layout of dynamic disks and volumes. Dynamic disks were new in Windows 2000 and do not use the traditional partition layout that was used by disks in Windows NT 4.0 and earlier.

Maintenance and Troubleshooting Tools

Windows XP Professional provides many tools that you can use to maintain and troubleshoot disks and file systems. The tools described in this section are:

- Chkdsk
- Disk Defragmenter
- Recovery Console
- Automated System Recovery
- DiskProbe
- Dmdiag

For more information about troubleshooting problems with Windows XP Professional, see "Tools for Troubleshooting" and "Troubleshooting Concepts and Strategies" in this book.

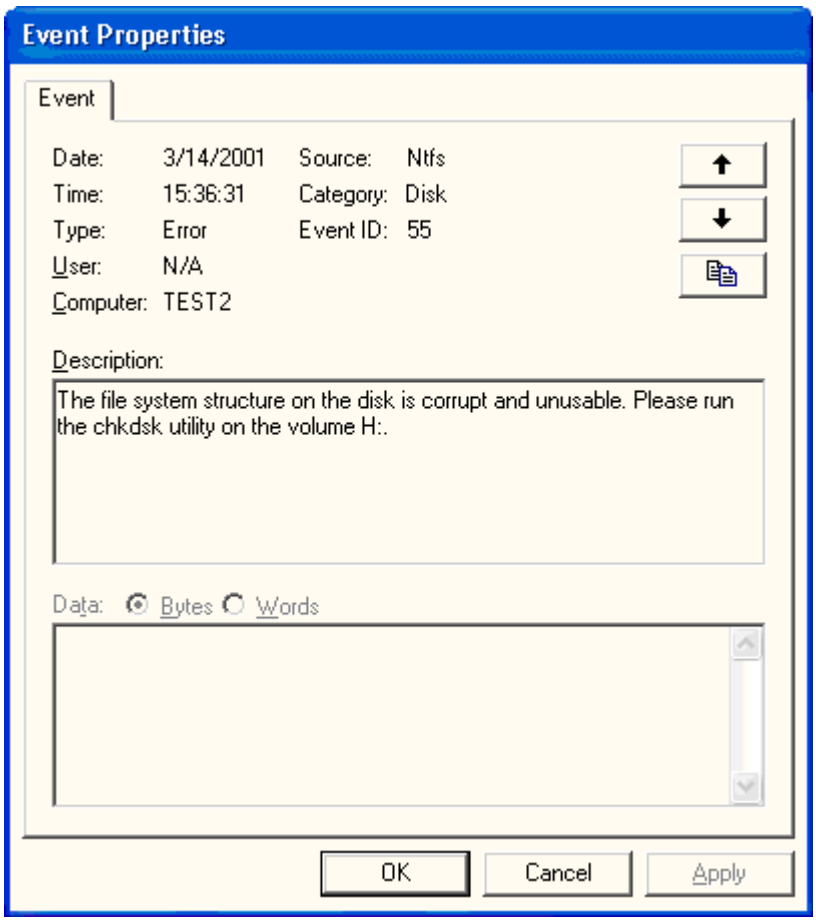
Chkdsk

Chkdsk.exe is a command-line tool that verifies the logical integrity of a file system on a Windows XP Professional volume. If file system structures become damaged, Windows XP Professional automatically schedules Chkdsk to run the next time the computer is restarted. At any time, you can manually run Chkdsk at the command prompt or from Windows Explorer or My Computer. For more information about running the graphical version of Chkdsk, see "Running Chkdsk from My Computer or Windows Explorer" later in this chapter.

Volumes that have file system errors are known as dirty volumes. To indicate that a file system problem has occurred and that the volume is dirty, Windows XP Professional displays a message similar to the following when you attempt to open, delete, or rename a file or folder by using Microsoft® Windows® Explorer or the command prompt:

```
The file or directory filename is corrupt and unreadable. Please run the Chkdsk
utility.
```

You might also see messages in the system log in the Event Viewer snap-in. Figure 27.1 illustrates a Chkdsk entry in the system log.



If your browser does not support inline frames, click here to view on a separate page.

Figure 27.1 Chkdsk message in the system log in Event Viewer

You can also determine whether a volume is dirty by using the **fsutil dirty query** command or the **chkntfs** command.

For example, to determine whether volume C is dirty, you can type:

```
fsutil dirty query c:
```

–or–

```
chkntfs c:
```

Running Chkdsk to Repair File Systems

You can run Chkdsk in two modes:

- **Chkdsk without parameters**. When you run Chkdsk without parameters, it runs in read-only mode. In this mode, Chkdsk examines the disk and then reports whether it found any file system errors but does not repair the errors.
- **Chkdsk with parameters**. When you run Chkdsk with parameters, such as **/f** or **/r**, Chkdsk repairs errors related to file system structures.

Before running Chkdsk to repair a volume

Before running Chkdsk to repair a volume, you must do the following:

- Back up key data files or make sure that you have a known good backup.
- Be prepared to let the Chkdsk process complete.

If you use the **/f** or **/r** parameter on a large volume (for example, 70 GB) or on a volume with a very large number of files (in the millions), Chkdsk can take a long time to complete. The volume is not available during this time because Chkdsk does not relinquish control until it is done. If a volume is being checked during the startup process, the computer is not available until the Chkdsk process is complete.

Chkdsk does not include parameters that let you cancel the Chkdsk process; however, when you run Chkdsk you can specify parameters that shorten the process. For more information about minimizing downtime during Chkdsk, see "Reducing the Time Required to Run Chkdsk on NTFS Volumes" later in this chapter.

Running Chkdsk on the boot volume

When you use the **/f** or **/r** parameters to run Chkdsk on the boot volume, Chkdsk displays the following message:

```
Chkdsk cannot run because the volume is in use by another process. Would you like to
schedule this volume to be checked the next time the system restarts? (Y/N)
```

Chkdsk cannot gain exclusive use of the boot volume because it contains the Windows XP operating system files. Therefore, Chkdsk must always restart the computer to check the boot volume. If you press the **Y** key, a version of Chkdsk known as Autochk runs the next time the computer restarts. After Autochk checks the boot volume, the computer automatically restarts.

Running Chkdsk on a volume other than the boot volume

When you use the **/f** or **/r** parameters on a volume other than the boot volume, Chkdsk must lock the volume for exclusive use before it can repair errors. If the volume has open files or programs, Chkdsk displays the following message:

```
Chkdsk cannot run because the volume is in use by another process. Chkdsk may run if
this volume is dismounted first. ALL OPEN HANDLES TO THIS VOLUME WOULD THEN BE
INVALID. Would you like to force a dismount on this volume? (Y/N)
```

If you press the **Y** key, Chkdsk attempts to close all handles and lock the volume. If Chkdsk is successful in locking the volume, the repair process begins. The duration of the repair process is determined by the number of files and folders on the volume and the level of damage, if any.

If Chkdsk cannot lock the volume, or if you press the **N** key, you can specify that you want to check the volume by running Autochk the next time the computer restarts. For more information about Autochk, see "Running Autochk When the Computer Restarts" later in this chapter.

Chkdsk Syntax

The command-line syntax for Chkdsk is as follows:

chkdsk [*volume*[[*path*] *filename*]] [**/f**] [**/v**] [**/r**] [**/x**] [**/i**] [**/c**] [**/l**[:*size*]]

Chkdsk Parameters

Table 27.3 lists all Chkdsk command-line parameters.

Table 27.3 Chkdsk Parameters

Parameter	Description
volume	Specifies the volume that you want Chkdsk to check. You can specify the volume by using any of the formats in the following examples: • To run Chkdsk on the C volume, specify: **c:** • To run Chkdsk on a mounted volume called **data** that is mounted on the C volume, specify: **c:\data** • To run Chkdsk on a volume, you can specify the symbolic link name for a volume, such as: **\\?\Volume{2d9bd2a8-5df8-11d2-bdaa-000000000000}** You can determine a symbolic link name for a volume by using the **mountvol** command. For more information about mountvol, see Windows XP Professional Help.
path	FAT/FAT32 only. Specifies the location of a file or set of files within the folder structure of the volume.
filename	FAT/FAT32 only. Specifies the file or set of files to check for fragmentation. Wildcard characters (* and ?) are allowed.
/f	Fixes errors on the disk. The volume must be locked. If Chkdsk cannot lock the volume, Chkdsk offers to check it the next time the computer restarts.
/v	On FAT/FAT32: Displays the full path and name of every file on the disk. On NTFS: Displays additional information or cleanup messages, if any.
/r	Locates bad sectors and recovers readable information (implies **/f**). If Chkdsk cannot lock the volume, it offers to check it the next time the computer starts.

	Because NTFS also identifies and remaps bad sectors during the course of normal operations, it is usually not necessary to use the **/r** parameter unless you suspect that a disk has bad sectors.
/x	Forces the volume to dismount first, if necessary. All opened handles to the volume are then invalid (implies **/f**). This parameter does not work on the boot volume. You must restart the computer to dismount the boot volume.
/i	NTFS only. Performs a less detailed 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.
/l:*size*	NTFS only. Changes the size of the log file to the specified number of kilobytes. Displays the current size if you do not enter a new size. If the system loses power, stops responding, or is restarted unexpectedly, NTFS runs a recovery procedure when Windows XP Professional restarts that accesses information stored in this log file. The size of the log file depends on the size of the volume. In most conditions, you do not need to change the size of the log file. However, if the number of changes to the volume is so great that NTFS fills the log before all metadata is written to disk, then NTFS must force the metadata to disk and free the log space. When this condition occurs, you might notice that Windows XP Professional stops responding for 5 or more seconds. You can eliminate the performance impact of forcing the metadata to disk by increasing the size of the log file. For more information about NTFS recoverability, see "File Systems" in this book.
/?	Displays this list of Chkdsk parameters.

For more information about the Chkdsk parameters, see Windows XP Professional Help. For more information about running the graphical version of Chkdsk, see "Running Chkdsk from My Computer or Windows Explorer" later in this chapter.

Chkdsk Examples

To run Chkdsk to repair errors on the D volume, type:

```
chkdsk d: /f
```

If you need to run Chkdsk on a large D volume and you want Chkdsk to complete as quickly as possible, type:

```
chkdsk d: /f /c /i
```

You can script Chkdsk and Autochk by using the Windows Management Instrumentation (WMI) classes Win32_LogicalDisk, Win32_AutoChkSetting, and Win32_OperatingSystemAutochkSetting. For more information about WMI, see the Microsoft Windows Management Instrumentation (WMI) SDK link on the Web Resources Page at http://www.microsoft.com/windows/reskits/webresources .

The Chkdsk Process on NTFS Volumes

When you run Chkdsk on NTFS volumes, the Chkdsk process consists of three major stages, and optional fourth and fifth stages. Chkdsk displays its progress for each stage with the following messages:

```
CHKDSK is verifying files (stage 1 of 3)...
File verification completed.
CHKDSK is verifying indexes (stage 2 of 3)...
Index verification completed.
CHKDSK is verifying security descriptors (stage 3 of 3)...
Security descriptor verification completed.
```

The following describes each of the Chkdsk stages.

Stage 1: Chkdsk verifies each file record segment in the master file table

During stage 1, 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 complete that Chkdsk displays during this phase is the percent of the MFT that has been verified.

Stage 2: Chkdsk checks the directories in the volume

During stage 2, Chkdsk examines each of the indexes (directories) on the volume for internal consistency and verifies that every file and directory represented by a file record segment in the MFT is referenced by at least one directory. Chkdsk also confirms that every file or subdirectory referenced in each directory actually exists as a valid file record segment in the MFT and checks for circular directory references. Chkdsk then confirms that the time stamps and the file size information associated with files are up-to-date in the directory listings

for those files.

The percent complete that Chkdsk displays during this phase is the percent of the total number of files on the volume that are checked. For volumes with many thousands of files and folders, the time required to complete this stage can be significant.

Stage 3: Chkdsk verifies the security descriptors for each volume

During stage 3, Chkdsk examines each of the security descriptors associated with each file and directory on the volume by verifying that each security descriptor structure is well formed and internally consistent. The percent complete that Chkdsk displays during this phase is the percent of the number of files and directories on the volume that are checked.

Stages 4 and 5 (optional stages): Chkdsk reads every sector on the volume to confirm stability

Chkdsk performs stages 4 and 5 if you specify the /**r** parameter when you run Chkdsk. The **/r** parameter confirms that the sectors in each cluster are usable. Specifying the **/r** parameter is usually not necessary because NTFS identifies and remaps bad sectors during the course of normal operations, but you can use the **/r** parameter if you suspect the disk has bad sectors.

During stage 4, Chkdsk verifies all clusters in use; during stage 5, Chkdsk verifies unused clusters.

The percent complete that Chkdsk displays during stage 4 is based on the percent of used clusters that are checked. The percent complete that Chkdsk displays during stage 5 is the percent of unused clusters that are checked. Used clusters typically take longer to check than unused clusters, so stage 4 lasts longer than stage 5 on a volume with equal amounts of used and unused clusters. For a volume with mostly unused clusters, stage 5 takes longer than stage 4.

During stages 1 and 3, the percent complete indicator advances relatively smoothly, although some unevenness might occur in the rate at which these phases progress. For example, file record segments that are not in use require less time to process than do those that are in use, and larger security descriptors take more time to process than do smaller ones. Overall the percent complete is a fairly accurate representation of the actual time required for that phase.

The duration of stage 2 varies because the amount of time required to process a directory is closely tied to the number of files or subdirectories listed in that directory. Because of this dependency, the percent complete indicator might not advance smoothly during stage 2, though the indicator continues to advance even for large directories. Therefore, do not use the percent complete as a reliable representation of the actual time remaining for this phase.

For more information, see "Determining How Long Chkdsk Will Run" later in this chapter.

Running Autochk When the Computer Restarts

Autochk.exe is a version of Chkdsk that runs only before Windows XP Professional starts. Autochk runs in the following situations:

- **Autochk runs if you try to run Chkdsk on the boot volume**. Chkdsk cannot dismount the boot volume, so Chkdsk offers to run the repair process by using Autochk when the computer is restarted. If you press the **Y** key to schedule Autochk, you have 10 seconds after the computer restarts to press any key and cancel the repair process. If you cancel Autochk before the 10-second delay lapses, Autochk does not run the next time you restart the computer. If you want to run Chkdsk again, you can do so from the command line.
- **Autochk runs if Chkdsk cannot gain exclusive use of the volume**. If Chkdsk cannot gain exclusive use of a volume when you run Chkdsk from the command line, Chkdsk offers to dismount the volume. If you press the **Y** key and Chkdsk still cannot dismount the volume, or if you press the **N** key, then Chkdsk offers to run the repair process by using Autochk when the computer is restarted. If you press the **Y** key to schedule Autochk, you have 10 seconds after the computer restarts to press any key and cancel the repair process. If you cancel Autochk before the 10-second delay lapses, Autochk does not run the next time you restart the computer. If you want to run Chkdsk again, you can do so from the command line.
- **Autochk runs if the volume is flagged as dirty**. If the file system has flagged the volume as dirty, Autochk runs the repair process at startup. Volumes are flagged as dirty when the file system detects an error on the volume. If Autochk detects a dirty volume, it provides a 10-second delay and then begins the repair process. If you cancel Autochk when a volume is dirty, Autochk attempts to run again after a 10-second delay each time the computer is restarted.

You can use the Chkntfs.exe command-line tool to change the Autochk delay from 0 seconds to up to 3 days (259,200 seconds). However, a long delay means that the computer does not start until the time elapses or until you press a key to cancel Autochk.

If you choose to let Autochk run, you can review the Autochk report in the application log of the Event Viewer snap-in. Autochk information is logged by the Winlogon service, so look for entries with **Winlogon** listed as the source of the entry.

Note You can use the **fsutil dirty** command to query and set the volume as dirty, but you must use the **chkntfs** command to exclude a dirty volume from being repaired by Autochk. For more information about using the **fsutil dirty** command, see Windows XP Professional Help.

Using Chkntfs to Prevent Autochk from Running

For heavily used computers that cannot be offline for the length of time required to complete the repair process, you can use the Chkntfs.exe command-line tool to exclude dirty volumes from being checked by Autochk. You can also use Chkntfs to cancel previously scheduled sessions of Autochk and to check the status of a volume.

For example, by typing **chkntfs c:** at the command prompt, you can find out:

- Whether you manually scheduled Autochk to run on volume C the next time the computer is restarted.
- Whether volume C is dirty, in which case Autochk runs automatically the next time the computer is restarted unless you manually run Chkdsk on volume C, cancel Autochk during the delay at startup, or exclude volume C by using the **/x** parameter.

Caution If a volume is flagged as dirty, do not postpone running Chkdsk indefinitely. File system damage can become worse over time, so you must consider dirty volumes at risk until you run Chkdsk. Use Chkntfs only if you need to control when Chkdsk is run.

Chkntfs Syntax

The command-line syntax for Chkntfs is as follows:

```
chkntfs volume […]
chkntfs [/d]
chkntfs [/t[:time]]
chkntfs [/x volume […]]
chkntfs [/c volume […]]
```

Chkntfs Parameters

Table 27.4 lists all Chkntfs command-line parameters. When using Chkntfs, you can specify only one parameter at a time.

For more information about the Chkntfs parameters, see Windows XP Professional Help. For more information about the registry changes that occur when you use Chkntfs, see article Q218461, "Enhanced Chkdsk, Autochk, and Chkntfs Tools in Windows 2000." To find this article, see the Microsoft Knowledge Base link on the Web Resources page at http://www.microsoft.com/windows/reskits/webresources .

Table 27.4 Chkntfs Parameters

Parameter	Description
volume […]	Specifies the volume that you want to check. You can specify the volume by using any of the formats in the following examples: • To run Chkntfs on the C volume, specify: **c:** • To run Chkdsk on a mounted volume called **data** that is mounted on the C volume, specify: **c:\data** • To run Chkntfs on a volume you can specify the symbolic link name for a volume, such as: **\\?\Volume{2d9bd2a8-5df8-11d2-bdaa-000000000000}** You can determine a symbolic link name for a volume by using the **mountvol** command. For more information about mountvol, see Windows XP Professional Help.
[**/d**]	Restores all Chkntfs default settings except the countdown time for Autochk. The **/d** parameter clears the list of volumes you excluded by using the **/x** parameter and also cancels any sessions of Autochk previously scheduled to take place when the computer restarts. After you use the **/d** parameter, Autochk runs on volumes that are flagged as dirty when the computer restarts.
[**/t**[**:***time*]]	Changes the Autochk initiation countdown time to the specified amount of time entered in seconds. The default countdown time is 10 seconds. If you use the **/t** parameter without using **:***time*, the countdown time that you last used is displayed.
[**/x** *volume* […]]	Excludes the specified volume from being checked when the computer starts. This parameter excludes only dirty volumes; Autochk still checks the volumes that you

	manually schedule.
[**/c** *volume* […]]	Schedules Autochk to run on the specified volumes if they are dirty when the computer starts, overriding any volumes excluded by the **/x** parameter.

Using the /x Parameter to Exclude Volumes

Use the **/x** parameter to prevent Autochk from running at startup on dirty volumes. Although it is not recommended that you use this parameter to postpone running Autochk indefinitely, you can use this parameter to prevent Autochk from running. For example, when you know the volume is dirty, you can use the **/x** parameter to postpone running Autochk until a period of low computer activity, such as overnight or during the weekend.

The **/x** parameter is not cumulative. Each time you use the **/x** parameter, you override the previous entry. For example, typing **chkntfs e: /x**, followed by **chkntfs f: /x**, excludes only the F volume from being checked.

To exclude multiple volumes, list them all in one command. For example, you can exclude both the E and F volumes by typing:

```
chkntfs e: f: /x
```

Note You can configure physical disk resources in cluster disks so that Chkdsk is skipped when the system mounts the disk. You can also configure the system to mount the disk even if Chkdsk encounters errors. For more information about configuring Chkdsk to run on a cluster disk, see article Q223023, "Enhanced Disk Resource Private Properties Using Cluster Server." To find this article, see the Microsoft Knowledge Base link on the Web Resource page at http://www.microsoft.com/windows/reskits/webresources .

Using the /c Parameter to Run Autochk on Excluded Volumes

If you use Chkntfs to determine that an excluded volume is marked as dirty and if you want Autochk to run on the volume when the computer starts, use the **/c** parameter. The **/c** parameter overrides the **/x** parameter so that you can run Autochk on volumes that you previously excluded. After Autochk runs on the volume, the volume remains on the excluded list.

Note You can also use the **chkdsk /f** or **chkdsk /r** command to check volumes that you previously excluded by using the **chkntfs /x** command. The **chkntfs /c** command runs Autochk only at startup and only if the volume is marked as dirty.

The **/c** parameter is cumulative. For example, you can specify multiple volumes by typing:

```
chkntfs c: /c
chkntfs d: /c
chkntfs e: /c
```

You can specify multiple volumes at the same time by typing:

```
chkntfs c: d: e: /c
```

Reducing the Time Required to Run Chkdsk on NTFS Volumes

NTFS is a journaling file system because it guarantees the consistency of the volume by using standard transaction logging and recovery techniques. If a disk becomes corrupted, NTFS runs a recovery procedure that accesses information stored in a transaction log file. The NTFS recovery procedure guarantees that the volume is restored to a consistent state. For this reason, it is unlikely that NTFS volumes might become corrupted.

Caution NTFS does not guarantee the integrity of user data following an instance of disk corruption, even when a full Chkdsk is run immediately after corruption is detected. Chkdsk might not recover all files, and files that are recovered might be internally corrupted. Therefore, you must protect important data by performing periodic backups.

If file system errors do occur on an NTFS volume, you must run Chkdsk to repair the damage. The recommended procedure is to run **chkdsk /f** as soon as possible, but you can also run a shorter version of Chkdsk by using the **/c** and **/i** parameters. These parameters were designed for administrators who manage exceptionally large NTFS volumes and who require flexibility in managing the downtime that is incurred when Chkdsk is running.

Caution Using the **/c** and **/i** parameters can result in a volume remaining corrupted after Chkdsk completes. Therefore, you must use these parameters only in situations where you need to keep system downtime to a minimum.

Table 27.5 provides a brief overview of each parameter and the potential reduction in Chkdsk duration. The actual reduction depends on a combination of factors, such as the ratio of files to directories and the relative speed of disk input/output (I/O) versus CPU speed, making the completion time difficult to predict.

Table 27.5 Overview of the /c and /i Chkdsk Parameters

Parameter	What It Does	Potential Reduction in Chkdsk Duration
/c	Skips the process that detects cycles in the directory structure.	1 to 2 percent
/i	Skips the process that compares directory entries to the file record segments that correspond to those entries.	50 to 70 percent

Using the /c Parameter

Use the **/c** parameter to skip the process that detects cycles in the directory structure. Cycles are a rare form of corruption in which a subdirectory has itself as a parent. Although you can speed up the Chkdsk process by using the **/c** parameter, using the **/c** parameter can also leave directory loops on an NTFS volume. Such loops might be inaccessible from the rest of the directory tree and could result in orphaned files. Files can become orphaned when file record segments remain but are not referenced by any directory entry. The file represented by the file record segment might be intact in all ways except that the file is invisible to all programs, including backup programs.

Using the /i Parameter

Use the **/i** parameter to skip the process that compares directory entries to the file record segments that correspond to those entries. A file record segment in the master file table (MFT) uniquely identifies every file and directory in an NTFS volume. When you use the **/i** parameter, the directory entries are checked to verify that they are self-consistent, but the directory entries are not necessarily consistent with the data stored in their corresponding file record segments.

When you use the **/i** parameter, files can become orphaned if directory entries remain, but the directory entries refer to incorrect file record segments. In this case, the files exist, but programs encounter errors when attempting to access them.

For more information about using the **/i** and **/c** parameters, see article Q187941, "An Explanation of CHKDSK and the New /C and /I Switches." To find this article, see the Microsoft Knowledge Base link on the Web Resource page at http://www.microsoft.com/windows/reskits/webresources .

Running Chkdsk on Mission-Critical Computers

If you use the Chkntfs or Fsutil command-line tool and discover that a volume in a mission-critical computer is flagged as dirty, you must choose among the following three choices:

Do nothing. For a mission-critical computer that is expected to be online 24 hours a day, doing nothing might be a necessary choice. The drawback to this option is that relatively minor corruption can become major corruption if you do not repair the volume as soon as possible after you detect the corruption. Therefore, consider this option only if keeping a system online is more important than the integrity of the data stored on the corrupted volume. You must consider all data on the corrupted volume at risk until you run Chkdsk.

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 is that a full Chkdsk might require several hours of downtime for a mission-critical computer at an inopportune time.

Run an abbreviated Chkdsk by using a combination of the /c and /i parameters. This option repairs minor corruption that can become major corruption in much less time than a full Chkdsk requires, but might not repair all corruption. A full Chkdsk is required to guarantee that all the data that can be recovered has been recovered.

Determining How Long Chkdsk Will Run

The best way to predict how long Chkdsk will take to run on a given volume is to perform a trial run in read-only mode during a period of low system usage. However, you must use caution when using read-only mode to estimate run time because of the following reasons:

Chkdsk might fail in read-only mode or might report false errors. The read-only Chkdsk process involves three phases. If Chkdsk encounters errors in the early phases, Chkdsk might abort before it completes all three phases. In addition, Chkdsk is prone to falsely reporting errors when in read-only mode and might report that a volume is corrupted even when no corruption is present. For example, Chkdsk might report corruption if NTFS modifies an area of the disk on behalf of a program at the same time Chkdsk is examining the same area. To verify a volume correctly, the volume must be in a static state, and the only way to guarantee that state is to lock the volume. Chkdsk locks the volume only when you specify the **/f**, **/r**, or **/x** parameters. Thus, you might need to run Chkdsk more than once for Chkdsk to complete all stages in read-only mode.

System load can influence the time required to run Chkdsk. Chkdsk is both CPU intensive and disk

intensive. If heavy disk I/O or high CPU usage is occurring when you run Chkdsk in read-only mode, the time required to complete the process increases.

Chkdsk and Autochk do not take the same time to complete. Chkdsk runs while Windows XP Professional is running, and Autochk runs before Windows XP Professional loads. Although running Autochk at startup gives exclusive use of CPU and disk I/O resources to Chkdsk, it also deprives Autochk of the benefit of virtual memory. Thus, while Autochk usually runs faster than Chkdsk, systems with relatively low amounts of RAM might see longer times for Autochk than for Chkdsk.

Repairing corruption lengthens the Chkdsk process. The read-only Chkdsk process can complete only if no significant corruption is found. If a disk suffers only minor corruption, the time to fix the problems is only slightly longer than the time required for read-only Chkdsk. However, if the volume has major corruption, the time required to run Chkdsk can increase in proportion to the number of files damaged.

Recovering Lost Clusters on FAT Volumes

Because some repairs on FAT volumes, such as correcting lost clusters (also known as allocation units) or cross-linked files, change the volume's file allocation table and can cause data loss, Chkdsk first prompts you with a confirmation message similar to the following:

```
10 lost allocation units found in 3 chains.
Convert lost chains to files? (Y/N)
```

If you press the N key, Windows XP Professional fixes the errors on the volume but does not save the contents of the lost clusters.

If you press the Y key, Windows XP Professional attempts to identify the folder to which they belong. If the folder is identified, the lost cluster chains are saved as files.

If Windows XP Professional cannot identify the folder or if the folder does not exist, it saves each chain of lost clusters in a folder called Found.*xxx*, where *xxx* is a sequential number starting with 000. If no folder Found.000 exists, one is created at the root. If one or more sequential folders called Found.*xxx* (starting at 000) exist, a folder that uses the next number in the sequence is created.

Windows XP Professional creates Found.xxx folders as hidden system folders. To see a list of Found.xxx folders, at the root folder in the command prompt, type **dir /a**. For information about viewing hidden system folders in My Computer or Windows Explorer, see Windows XP Professional Help.

After the storage folder has been identified or created, one or more files with a name in the format File*nnnn*.chk are saved. (The first saved file is named File0000.chk, the second is named File0001.chk, and so on in sequence.) When Chkdsk finishes, you can examine the contents of these files with a text editor such as Notepad to see whether they contain any needed data (if the converted chains came from corrupted binary files, they are of no value). You can delete the .chk files after you save any useful data.

Caution Because other programs might create and use files with the .chk extension, you must be careful to delete only the .chk files that are in the Found.xxx folders.

Running Chkdsk from My Computer or Windows Explorer

In addition to using the command-line version of Chkdsk, you can run Chkdsk from My Computer or Windows Explorer. The graphical version of Chkdsk offers the equivalent of read-only mode, the **/f** parameter, and the **/r** parameter.

If Chkdsk cannot lock the volume, you can schedule Autochk to run the next time you restart the computer. You cannot choose to dismount the volume like you can when you use the command-line version of Chkdsk, nor can you use other Chkdsk parameters, such as **/c** or **/i**. To take advantage of all the Chkdsk parameters, use the command-line version of Chkdsk.

To run Chkdsk from My Computer or Windows Explorer

1. In My Computer or Windows Explorer, right-click the volume you want to check, and then click **Properties**.
2. On the **Tools** tab, click **Check Now**.
3. Do one of the following:
   - To run Chkdsk in read-only mode, click **Start**.
   - To run Chkdsk by using the **/f** parameter, select the **Automatically fix file system errors** check box, and then click **Start**.
   - To run Chkdsk by using the **/r** parameter, select the **Scan for and attempt recovery of bad sectors** check box, and then click **Start**.

Disk Defragmenter

Fragmentation causes your disk subsystem to perform more seeks, which slows the transfer rate and results in

sluggish disk performance. Defragmenting is occasionally necessary because of the way files are stored on disk. Fragmentation can occur when:

- You create a file, but the volume does not have a group of contiguous, free clusters that is large enough to contain the entire file. Therefore, the file is broken into fragments rather than residing in contiguous clusters on the disk.
- You edit a file so that it outgrows its existing space on the disk. When a file uses all the clusters in a group of contiguous, free clusters, the file is then broken into fragments that are stored in free clusters elsewhere on the disk.

Although FAT and NTFS are designed to make storage faster and more efficient when you save files, these file systems take longer to read and write fragmented files than unfragmented files. When the files on a disk become badly fragmented, performance noticeably suffers because the disk heads must move to different tracks on the disk to locate all the clusters of the file.

Defragmentation tools fix this problem by moving the files into contiguous clusters on the disk. Reducing fragmentation reduces the amount of mechanical movement required to locate all clusters of a file, which improves hard disk performance.

Windows XP Professional provides two methods for defragmenting FAT and NTFS volumes:

- The Disk Defragmenter snap-in (Dfrg.msc).
- The new Disk Defragmenter command-line tool (Defrag.exe).

Both tools rearrange files, folders, programs, and unused space on your computer's hard disk to optimize disk performance. In addition, the defragmentation tools are improved in Windows XP Professional so that you can:

- Defragment volumes that use any cluster size.
- Defragment files that are smaller than 16 clusters.
- Defragment the master file table (MFT).

The amount of time that the defragmentation process takes depends on several factors, including the size of the volume, the number and size of files on the volume, the amount of fragmentation, and how busy the system is during defragmentation.

Before Using the Disk Defragmentation Tools

When you use the disk defragmentation tools, keep the following restrictions in mind:

- You can defragment only local volumes, and you can defragment only one volume at a time.
- You must be logged on as an administrator or as a member of the Administrators group to defragment volumes.
- You cannot use the Disk Defragmenter command-line tool (Defrag.exe) while the Disk Defragmenter snap-in is open.
- You cannot defragment volumes that are marked as dirty by the file system. You must run Chkdsk on the dirty volume before you can defragment it. To determine if a volume is dirty, use the **fsutil dirty query** command. For more information about running Chkdsk, see "Chkdsk" earlier in this chapter.

In addition, to obtain best results when you use the disk defragmentation tools, follow these guidelines:

- Ensure you have at least 15 percent free disk space when you defragment a volume. Windows XP Professional uses the free disk space as a sorting area for file fragments.

  Although the defragmentation tools can partially defragment volumes that have less than 15 percent free space, for best results delete unneeded files or move them to another volume to increase the free space to at least 15 percent. You can also use the Disk Cleanup tool to delete unnecessary files. For more information about Disk Cleanup, see Windows XP Professional Help.
- Do not run Backup (either a manual or a scheduled start) at the same time that you run the defragmentation tools because using Backup causes the defragmentation process to pause.

  The Backup program included with Windows XP Professional uses volume snapshots to allow users or applications to continue working while a backup occurs. The defragmentation process resumes after Backup removes the volume snapshot. For more information about volume snapshots, see "Backup and Restore" in this book.

Running the Disk Defragmenter Snap-in

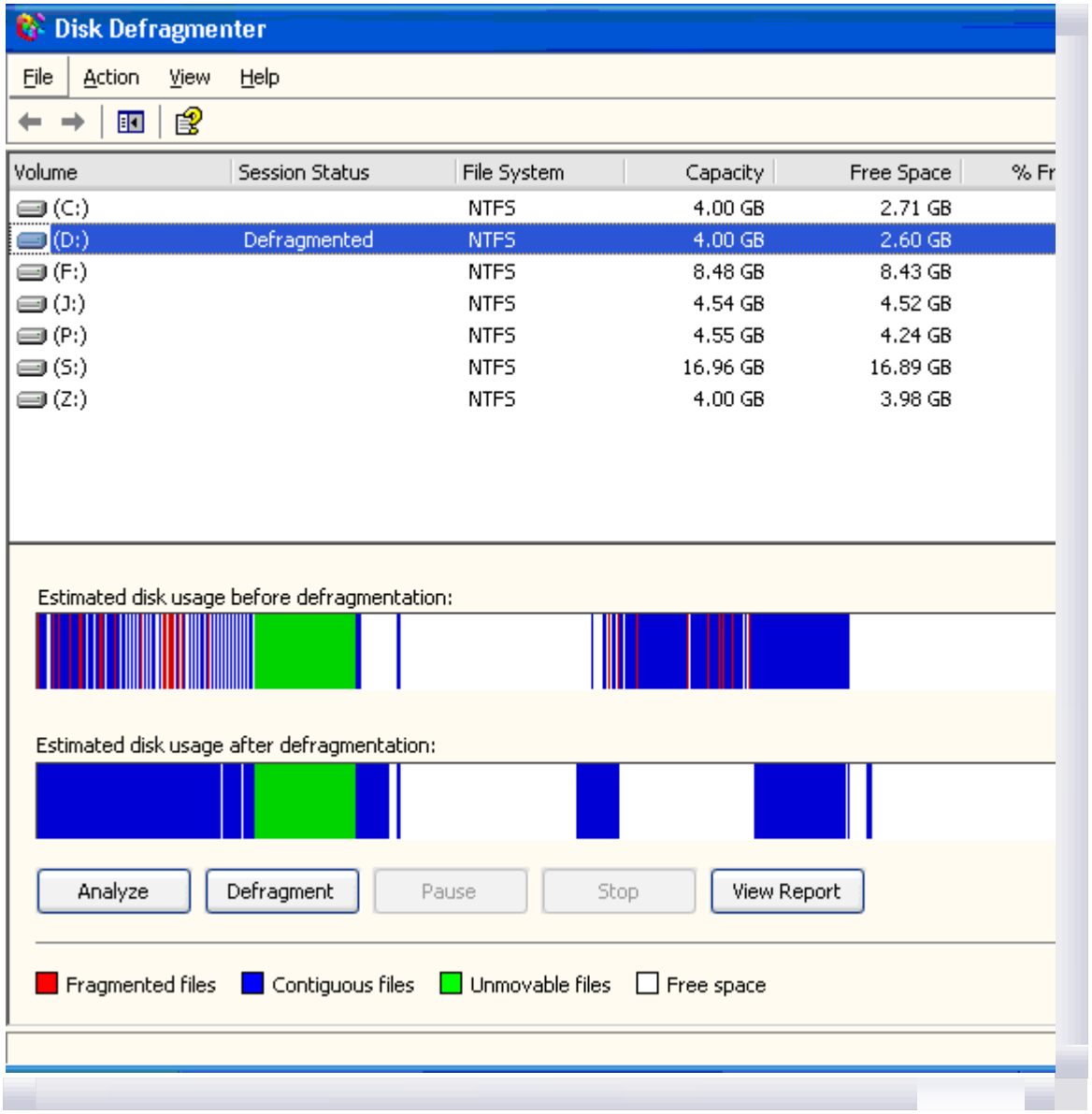
By using the Disk Defragmenter snap-in, you can analyze the volume before you defragment to see how many fragmented files and folders exist. If 10 percent or more of the files and folders are fragmented, Disk Defragmenter recommends that you defragment the volume. If the volume is less than 10 percent fragmented, you can still defragment the volume or you can simply view the fragmentation report.

To open the Disk Defragmenter snap-in

- Click **Start**, click **Run**, type **dfrg.msc**, and then click **OK**.

Figure 27.2 shows that the Disk Defragmenter snap-in is divided into two main areas. The upper part lists the volumes on the local computer and allows you to select a volume to analyze and defragment. The lower part displays a graphical representation of how fragmented the volume is. The colors indicate the condition of the volume:

- Red areas show fragmented files.
- Blue areas show contiguous (unfragmented) files.
- White areas show free space on the volume. White areas on an NTFS volume might also represent the MFT zone. For more information about the MFT zone, see "File Systems" in this book.
- Green areas show files that cannot be moved. The green areas usually represent the paging file, but on NTFS volumes, green areas might also represent space used by the NTFS change journal and the NTFS log file.



If your browser does not support inline frames, click here to view on a separate page.

Figure 27.2 The Disk Defragmenter snap-in

By comparing the **Estimated disk usage before defragmentation** band to the **Estimated disk usage after defragmentation** band, you can see the improvement in your volume after defragmenting. The defragmentation report provides further details on the fragmentation state of the volume. Analyze volumes regularly and defragment them when Disk Defragmenter recommends it.

Defragmenting Volumes by Using the Defrag Command

The Disk Defragmenter command-line tool and the Disk Defragmenter snap-in both defragment volumes by using the same method. However, the command-line tool differs from the snap-in in the following ways:

- The command does not provide a graphical analysis of a volume's fragmentation status, but it does provide a summary.
- The command does not provide a status indicator.

To see a list of parameters for Defrag.exe, at the command prompt, type:

```
defrag /?
```

To defragment a volume, specify the drive letter. For example, to defragment the C volume, type:

```
defrag c:
```

To defragment the C volume and view a detailed report that is similar to the report shown in the Disk Management snap-in, type:

```
defrag c -v
```

To analyze the C volume and view a detailed analysis report, type:

```
defrag c: -a -v
```

While the command-line Disk Defragmenter is analyzing and defragmenting a volume, it displays a blinking cursor. When Disk Defragmenter finishes analyzing a volume, it displays the analysis report. When Disk Defragmenter finishes defragmenting a volume, it displays the defragmentation report. After the process completes, Disk Defragmenter exits to the command prompt.

To interrupt the defragmentation process, at the command line, press CTRL+C.

For more information about using Defrag.exe, see Windows XP Professional Help.

Tips for Using the Disk Defragmentation Tools

For best results when defragmenting volumes, follow these tips:

- Before defragmenting a volume, delete any unnecessary files, such as temporary files. You can delete unnecessary files by using Disk Cleanup. For more information about Disk Cleanup, see Windows XP Professional Help.
- Defragment a volume before you add a large number of files to the volume, such as before you install programs. This ensures that the files occupy contiguous space and do not become fragmented after you add them.
- Defragment a volume after you delete a large number of files from the volume.
- Defragment a volume after you install programs on it.
- Defragment the system and boot volumes after installing Windows XP Professional.
- Defragment volumes during periods of low system activity.

Optimizing Startup Times by Using Defragmentation Tools

Windows XP Professional monitors the files that are used when the computer starts and when you start applications. By monitoring these files, Windows XP Professional can prefetch them. Prefetching data is the process whereby data that is expected to be requested is read ahead into the cache. Prefetching boot files and applications decreases the time needed to start Windows XP Professional and start applications.

Prefetching is further improved if the files are located next to each other on the outer edge of the disk. Windows XP Professional optimizes the location of boot files and applications when the computer is idle. The optimization occurs in the background and lasts only a minute or two; you might hear the hard disk being accessed when optimization occurs. After the initial optimization takes place, subsequent optimization occurs, at most, every three days.

When you run the Disk Defragmenter tools that are included with Windows XP Professional, they can perform any optimization updates that are scheduled to take place during the next idle period. The Disk Defragmenter tools do not disturb the existing layout of optimized boot files and applications.

Note Computers running Windows XP Home Edition also prefetch and optimize boot files and applications.

Files That You Cannot Defragment

After you defragment a volume, you can view the defragmentation report to see the results. The report includes a list of files that remain fragmented (having two or more fragments). Some reasons that a file might

remain fragmented include:

- The volume lacks adequate contiguous free space to defragment all files. Disk Defragmenter requires at least 15 percent free disk space to completely defragment a volume.
- During defragmentation, a new file is created on the volume in disk space that was previously free space. In this case, if Disk Defragmenter tries to move a file to that space to defragment the file, the move fails and the file remains fragmented.
- The file is the master file table (MFT) on an NTFS volume. Because the first fragment of the MFT cannot be moved, the MFT is typically contained within two fragments when sufficient space is available on the volume. If the MFT is contained within three or more fragments, Disk Defragmenter looks for free space where the MFT might fit. If sufficient free space exists, the MFT is moved as a whole (minus the first fragment). If space is not available, the MFT is not defragmented.
- The file is permanently excluded, in which case it might appear in the defragmentation report as still being fragmented no matter how many times you defragment the volume. The following files are permanently excluded from being defragmented.
  - Bootsect.dos
  - Safeboot.fs
  - Safeboot.csv
  - Safeboot.rsv
  - Hiberfil.sys
  - Memory.dmp
  - Paging file

The paging file is a hidden file on the hard disk that Windows XP Professional uses to hold parts of programs and data files that do not fit in memory. (The paging file and physical memory make up virtual memory.) In Windows 2000, the size of the paging file was conservative and often needed to be increased, which caused the paging file to become fragmented. Because Windows XP Professional creates a larger paging file than the default size used in Windows 2000, it is unlikely that your paging file will become fragmented.

You can determine whether the paging file is fragmented by analyzing the volume that contains the paging file and then viewing the analysis report. The report shows the size of the paging file and the number of fragments. You cannot use Disk Defragmenter to defragment the paging file because Windows XP Professional holds the paging file open for exclusive use. However, you can reduce the degree of fragmentation by deleting and then re-creating the paging file. You must have at least two volumes to perform this procedure.

To defragment the paging file

1. From the **Start** menu, click **Control Panel**, click **Performance and Maintenance**, and then click **System**.
2. On the **Advanced** tab, under **Performance**, click **Settings**.
3. On the **Advanced** tab, click **Change** to open the **Virtual Memory** dialog box.
4. In the list of drives, select a volume to store a temporary paging file.
5. Click **Custom size**, type an initial and maximum size to match the current paging file, and then click **Set**.
6. Select the original paging file in the drive list, reduce the minimum and maximum size of the original paging file to 0 MB, and then click **Set**.
7. Restart your computer to have the system use the new paging file.
8. Run Disk Defragmenter on the original volume to consolidate the free space segments created by moving the paging file.
9. Re-create the paging file on the original volume.
10. Reduce the minimum and maximum size of the temporary paging file to 0 MB.
11. Restart your computer.

Recovery Console

The Recovery Console is a text-mode command-line interpreter that you can use for basic troubleshooting and system maintenance. You can run the Recovery Console directly from the Windows XP operating system CD, or for x86-based systems install it as a startup option. The Recovery Console is separate from the Windows XP Professional command prompt and grants limited local hard disk access for both NTFS and FAT volumes.

Because starting the Windows XP Professional graphical user interface (GUI) is not a prerequisite for using the Recovery Console, it can help you recover a Windows XP Professional-based computer that cannot start in safe

mode or normal mode. For example, if the computer does not start because the master boot record (MBR) or boot sector is corrupted, you can use the Recovery Console to repair the MBR or boot sector.

Note Certain Recovery Console commands are not fully functional on dynamic disks or GPT disks.

For more information about using the Recovery Console to repair MBR and boot sector errors, see "Repairing Damaged MBRs and Boot Sectors in x86-based Computers" later in this chapter.

For more information about installing and using Recovery Console, see "Tools for Troubleshooting" in this book.

Automated System Recovery

If changes to the operating system cause instability or startup problems, you can use the Automated System Recovery (ASR) tool to restore the system state and all files stored on the system volume. The term *system state* refers to all the components that determine the current state of the operating system, including:

- User account information, hard disk configuration, and registry information that includes application, hardware, network, video, and software settings.
- Operating system files that are required to start the system, including those in the systemroot directory and boot files such as Ntldr or IA64ldr.efi.

ASR is a last resort option to use after you have unsuccessfully tried other recovery methods, such as rolling back drivers, restoring from backups, performing parallel installations, and using System Restore. ASR restores system state files and settings, and restores your ability to start your system. For example, hard disk corruption might prevent you from starting Windows XP Professional, and the damage might be serious enough to prevent you from using safe mode, Recovery Console, or the Last Known Good Configuration. ASR automates the process of backing up and restoring system state information and files that are needed on the system volume to start Windows XP Professional.

ASR is accessible through the Windows XP Professional Backup application NTBackup.exe and through other programs created by independent software vendors (ISVs). ASR replaces the Emergency Repair Disk option found in Windows 2000 and Windows NT 4.0. For more information about using ASR and other recovery tools, see "Backup and Restore" in this book.

DiskProbe

DiskProbe is a sector editor tool for Windows XP Professional that allows users who are members of the Administrators group to directly edit, save, and copy data on a physical hard disk. With careful use of DiskProbe, you can replace the master boot record (MBR), repair damaged partition table information, and repair or replace damaged boot sectors or other file system data. You can also use DiskProbe to save MBRs and boot sectors as backup binary files in case the original sectors become damaged by viruses, human error, hardware problems, power outages, or similar events. Unless you are familiar with using DiskProbe, try other troubleshooting tools, such as Recovery Console, before using DiskProbe.

Caution Be cautious when making any changes to the structures of your hard disk. Because DiskProbe does not validate the proposed changes to records, incorrect values in key data structures can render the hard disk inaccessible or prevent the operating system from starting. If you cannot correct the changes you entered, you must re-create and reformat all volumes on the disk.

DiskProbe can change the values of individual bytes in any sector on a dynamic disk, but it cannot navigate the structure of a dynamic disk. Therefore, it is recommended that you use DiskProbe only on basic disks. You can, however, use DiskProbe to back up and restore the boot sector and MBR of dynamic disks.

DiskProbe is part of Windows Support Tools. For more information about DiskProbe, click **Tools** in Help and Support Center, and then click **Windows Support Tools**.

Caution Do not use DiskProbe on GPT disks in Itanium-based computers. Structures on GPT disks are self-repairing. Making direct changes to GPT structures could cause the partition table checksums to become invalid, rendering the disk inaccessible. For more information about GPT disks, see "Disk Sectors on GPT Disks" later in this chapter.

Dmdiag

Dmdiag.exe is a command-line tool that displays the location and layout of dynamic disks (MBR and GPT) and dynamic volumes. This information is primarily useful if you are working with Microsoft Product Support Services to troubleshoot problems with dynamic disks and volumes.

Dmdiag.exe is part of Windows Support Tools and can be run from the command line by using the following syntax:

```
dmdiag [-f filename] [-v] [/?]
```

When used without parameters, Dmdiag.exe displays information about the dynamic disks and volumes installed on the computer. Table 27.6 describes the Dmdiag.exe parameters.

Table 27.6 Dmdiag.exe Parameters

Parameter	Description
-f *filename*	Specifies the name of the file that stores the output. If you do not specify an output file, the file is saved as Dmdiag.txt in the same folder where you run Dmdiag.exe. If you do not specify the **-f** parameter, the output is displayed at the command prompt.
-v	Runs Dmdiag in verbose mode, which contains additional information about dynamic disks and volumes. Use this mode to obtain a report that product support can use to help you troubleshoot dynamic disks and volumes.
/?	Displays a Help screen with usage syntax.

For more information about Dmdiag.exe, click **Tools** in Help and Support Center, and then click **Windows Support Tools**.

Disk and Volume Status Descriptions

Members of the Administrators group can use the Disk Management snap-in to view the status of disks and volumes.

To open Disk Management

1. From the **Start** menu, click **Run**.
2. In the **Open** box, type **diskmgmt.msc**, and then click **OK**.

As Figure 27.3 shows, if no errors are present on the disk, Disk Management displays an Online status for disks and a Healthy status for volumes.

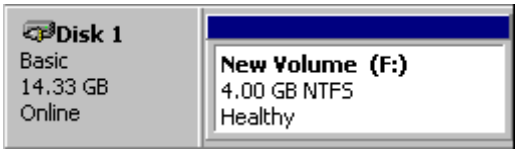


Figure 27.3 Online status and Healthy status

You can use the information in this section to diagnose and resolve problems identified by Disk Management, which uses a number of predefined status descriptions to indicate a problem has occurred. In many cases, a problem with the underlying disk also results in a volume error condition. For example, Figure 27.4 shows a disk with the Online (Errors) error condition and a volume with the Healthy (At Risk) error condition.

Figure 27.4 Online (Errors) and Healthy (At Risk) error conditions

You can also use the DiskPart command-line tool to view the status of disks and volumes. DiskPart is a text-mode command interpreter that is separate from the Windows XP Professional command prompt. DiskPart allows you to manage fixed (non-removable) disks and volumes by using scripts or direct input.

To run DiskPart, at the command prompt, type:

```
diskpart
```

To view the status of disks, at the DiskPart command prompt, type:

```
list disk
```

To view the status of volumes, at the DiskPart command prompt, type:

```
list volume
```

To view a list of DiskPart commands, at the DiskPart command prompt, type:

```
commands
```

For more information about using DiskPart, see Windows XP Professional Help.

Disk Status Descriptions

The following status descriptions appear if Disk Management detects a problem with a disk or if Disk

Management does not recognize the disk.

Foreign

The **Foreign** status occurs when you install a dynamic disk in the local computer. You must right-click the disk and then click **Import Foreign Disks** before you can access data on the disk. If you do not want to import the disk, you can right-click the disk and click **Convert to Basic Disk**. Disk Management displays a warning message before erasing all data from the disk and converting the disk to basic. For more information about importing foreign disks, see "Disk Management" in this book.

Missing

The **Missing** status occurs when a dynamic disk is corrupted, turned off, or disconnected. After you reconnect or turn on the missing disk, open Disk Management, right-click the missing disk, and then click **Reactivate Disk**.

Not Initialized

The **Not Initialized** status indicates that the disk does not contain a valid disk signature in the master boot record (MBR) or a valid disk GUID in the GUID partition table. After you install a new disk, Windows XP Professional must write the MBR or GUID partition table before you can create partitions on the disk.

When you first start Disk Management after installing a new disk, a wizard appears that provides a list of the new disks that Windows XP Professional detects. If you cancel the wizard before the MBR or GUID partition table is written, the disk status remains **Not Initialized** until you right-click the disk and then click **Initialize Disk**.

Offline

The **Offline** status occurs when a dynamic disk is not accessible. The disk might be corrupted or intermittently unavailable. **Offline** also appears if you attempt to import a foreign (dynamic) disk, but the import fails. An error icon appears on the offline disk. Only dynamic disks display the **Offline** status.

If the disk status is **Offline** and the disk name changes to **Missing**, the disk was recently available on the system but can no longer be located or identified. The missing disk might be damaged, turned off, or disconnected.

To bring a disk that is Offline and Missing back online

1. Repair any disk, controller, or cable problems and make sure that the physical disk is turned on, plugged in, and attached to the computer.
2. In Disk Management, right-click the disk, and then click **Reactivate Disk** to bring the disk back online.

If the disk status remains **Offline** and the disk name remains **Missing**, and you determine that the disk has a problem that cannot be repaired, you can remove the disk from the computer.

After you remove a dynamic disk from a computer, the remaining online dynamic disks retain information about the removed disk and its volumes in the dynamic disk database. You can delete all references to the removed disk by updating the dynamic disk database. To do this, use Disk Management to remove all volumes on the missing disk. After you remove all the volumes, right-click the missing disk and then click **Remove Disk**. The missing disk no longer appears in Disk Management.

Caution Deleting a volume destroys the data on the volume, so you should remove a disk only if you are absolutely certain that the disk is permanently damaged and unusable.

To bring a disk that is Offline but not Missing back online

1. In Disk Management, use the **Reactivate Disk** command to bring the disk back online.
2. If the disk status remains **Offline**, check the cables and disk controller, and make sure that the physical disk is healthy. Correct any problems and try to reactivate the disk again.

If the disk reactivation succeeds, any volumes on the disk automatically return to the **Healthy** status.

Online (Errors)

The **Online (Errors)** status indicates that I/O errors have been detected on a region of the disk. A warning icon appears on the disk with errors. Only dynamic disks display the **Online (Errors)** status.

If the I/O errors are transient, reactivate the disk by right-clicking the disk and then clicking **Reactivate Disk** to return the disk to the **Online** status.

Unreadable

The **Unreadable** status occurs when the disk is not accessible for the following reasons:

- The disk is spinning up.

- The disk might have experienced hardware failure, corruption, or I/O errors.
- The disk's copy of the dynamic disk database might be corrupted.

An error icon appears on disks that display the **Unreadable** status. Both dynamic and basic disks display the **Unreadable** status.

Disks might display the **Unreadable** status while they are spinning up or when Disk Management is rescanning all the disks on the system. In some cases, an unreadable disk has failed and is not recoverable. For dynamic disks, the **Unreadable** status usually results from corruption or I/O errors on part of the disk, rather than failure of the entire disk. You can rescan the disks (by using the **Rescan Disks** command on the **Action** menu in Disk Management) or restart the computer to see if the disk status changes.

Volume Status Descriptions

The following status descriptions appear if Disk Management detects a problem with dynamic volumes or if Disk Management does not recognize volumes, such as those created by non-Windows-brand operating systems.

Failed

The **Failed** status occurs when the dynamic disk is damaged or the file system is corrupted. Unless you can repair the disk or file system, the **Failed** status might indicate data loss.

To troubleshoot a volume with the Failed status, make sure that the underlying physical disk is turned on, plugged in, and attached to the computer. Try returning the disk to the **Online** status by using the **Reactivate Disk** command. If this procedure succeeds, the volume automatically restarts and returns to the **Healthy** status.

If the disk returns to the **Online** status but the volume does not return to the **Healthy** status, you can reactivate the volume manually by using the **Reactivate Volume** command.

In some situations, the **Failed** status does not indicate data loss even though the **Reactivate Disk** and **Reactivate Volume** commands fail. These situations occur when:

- You import a mirrored or RAID-5 volume into a computer running Windows XP Professional or Windows XP 64-Bit Edition. These operating systems do not support mirrored or RAID-5 volumes. You must move the disks back to the original computer to access data on the mirrored or RAID-5 volume.
- You install Windows XP Professional to upgrade a computer that is running Windows NT 4.0 Workstation and that contains multidisk volumes. Because Windows XP Professional cannot access multidisk volumes created by using Windows NT 4.0, you must use Ftonline.exe to return the volumes to Healthy status so that you can access data on them. For more information about managing multidisk volumes during Windows XP Professional Setup, see "Disk Management" in this book. For more information about using Ftonline.exe, click **Tools** in Help and Support Center, and then click **Windows Support Tools**.

Healthy (At Risk)

The **Healthy (At Risk)** status occurs when a dynamic volume is experiencing I/O errors caused by bad sectors on the physical disk. The disk remaps the bad sectors by using sectors reserved exclusively for remapping. If the errors are transient, you can use the **Reactivate Disk** command in Disk Management to return the volume to the **Healthy** status. If the **At Risk** status persists, your disk might be failing. Back up the data and replace the disk as soon as possible.

Healthy (Unknown Partition)

The **Healthy (Unknown Partition)** status occurs when Windows XP Professional or Windows XP 64-Bit Edition does not recognize the System ID of a partition on an MBR disk. Partitions with the **Healthy (Unknown Partition)** status might be unknown original equipment manufacturer (OEM) partitions or partitions created by operating systems other than Windows or third-party utilities. You cannot format, assign drive letters or drive paths to, or access data on partitions with **Healthy (Unknown Partition)** status. You can, however, delete these partitions by using Disk Management or the DiskPart command.

Caution If Windows XP Professional recognizes an OEM partition, Disk Management displays the partition as **Healthy (EISA Configuration)**. You cannot use Disk Management to format, delete, assign drive letters or drive paths to, or access data on **Healthy (EISA Configuration)** partitions. However, you can use DiskPart to delete OEM partitions by using the **delete partition** command and specifying the **override** parameter. Deleting an OEM partition can prevent Windows XP Professional from starting, so it is recommended that you do not delete OEM partitions.

Windows XP 64-Bit Edition recognizes partitions on GPT disks that use known partition type GUIDs. If Windows XP 64-Bit Edition does not recognize the partition type GUID of a partition, then it displays the partition as **Healthy (Unknown Partition)**.

Windows XP 64-Bit Edition recognizes the following partitions on GPT disks and displays them in the Disk

Management interface:

- Extensible Firmware Interface (EFI) System partition
- Primary partition on a basic disk

Windows XP 64-Bit Edition also recognizes and displays primary partitions, extended partitions, and logical drives on MBR disks.

Windows XP 64-Bit Edition recognizes the following partitions on GPT disks but does not display them in Disk Management:

- Microsoft Reserved partition (MSR)
- Logical Disk Manager (LDM) Metadata partition on a dynamic disk
- LDM Data partition on a dynamic disk

If an OEM created a primary partition on a GPT disk or if you use an operating system other than Windows XP 64-Bit Edition to create a primary partition on a GPT disk, then Windows XP 64-Bit Edition might not recognize the partition type GUID of the partition. If the partition type GUID is unrecognized, Windows XP 64-Bit Edition displays the partition but does not allow you to assign a drive letter or drive path or to access data on the partition. You can, however, delete these partitions by using Disk Management or the DiskPart command.

For more information about the types of partitions that Windows XP Professional recognizes, see "Master Boot Record on Basic Disks." For more information about the types of partitions that Windows XP 64-Bit Edition recognizes, see "GPT Partition Table Header" later in this chapter. For more information about partitions on GPT disks, see "Disk Management" in this book.

Unknown

The **Unknown** status occurs when the boot sector for the volume is corrupted and you can no longer access data on the volume. The boot sector might be infected by a virus. For more information about cleaning an infected computer, see "Viruses That Affect the MBR and Boot Sectors" later in this chapter. For more information about repairing boot sectors, see "Repairing Damaged MBRs and Boot Sectors in x86-based Computers" later in this chapter.

Viruses That Affect the MBR and Boot Sectors

It is always important to take precautions to protect your computer and the data on it from viruses. Many computer viruses exploit the disk structures that your computer uses to start up by replacing, redirecting, or corrupting the code and data that start the operating system.

For more information about the master boot record (MBR) and boot sector on x86-based computers, see "Disk Sectors on MBR Disks" later in this chapter.

MBR Viruses

MBR viruses exploit the master boot code within the master boot record (MBR) that runs automatically when an x86-based computer starts up. MBR viruses are activated when the BIOS activates the master boot code, before the operating system is loaded.

Many viruses replace the MBR sector with their own code and move the original MBR to another location on the disk. After the virus is activated, it stays in memory and passes the execution to the original MBR so that startup appears to function normally.

Some viruses do not relocate the original MBR, causing all volumes on the disk to become inaccessible. If the listing in the partition table for the active primary partition is destroyed, the computer cannot start. Other viruses relocate the MBR to the last sector of the disk or to an unused sector on the first track of the disk. If the virus does not protect the sector that contains the MBR, normal use of the computer might overwrite the MBR, and the system might not restart.

For more information about the master boot code, see "Disk Sectors on MBR Disks" later in this chapter.

Boot Sector Viruses

As with the master boot code, the boot sector's executable code also runs automatically at startup, creating another vulnerable spot exploited by viruses. Boot sector viruses are activated before the operating system is loaded and run when the master boot code in the MBR identifies the active primary partition and activates the executable boot code for that volume.

Many viruses update the boot sector with their own code and move the original boot sector to another location on the disk. After the virus is activated, it stays in memory and passes the execution to the original boot sector so that startup appears normal.

Some viruses do not relocate the original boot sector, making the volume inaccessible. If the affected volume is the active primary partition, the system cannot start. Other viruses relocate the boot sector to the last sector

of the disk or to an unused sector on the first track of the disk. If the virus does not protect the altered boot sector, normal use of the computer might overwrite it, rendering the volume inaccessible or preventing the system from restarting.

How MBR and Boot Sector Viruses Affect Windows XP Professional

Two common ways that a computer can contract an MBR or boot sector virus are: by starting up from an infected floppy disk; or by running an infected program, which causes the virus to drop an altered MBR or boot sector onto the hard disk.

The malicious activity of an MBR or boot sector virus is typically contained after Windows XP Professional starts. If the virus payload (the malicious activity of the virus) does not run during system startup and if the virus does not alter the original MBR or boot sector, Windows XP Professional prevents the virus from self-replicating to other disks.

During normal operation, Windows XP Professional is immune to viruses infecting these disk structures because it accesses physical disks only through protected-mode disk drivers. Viruses typically subvert the BIOS INT 13h disk access routines, which are ignored after Windows XP Professional starts. However, on computers with multiple-boot configurations, such as Windows XP Professional with Microsoft® MS-DOS®, Microsoft® Windows® 95, Microsoft® Windows® 98, or Microsoft® Windows® Millennium Edition (Me), an MBR or boot sector virus might infect the computer when you are running another operating system. If this happens, Windows XP Professional is vulnerable to damage.

Viruses that execute their payload during startup are a threat to computers that are running Windows XP Professional because the virus executes before Windows XP Professional takes control of the computer. After Windows XP Professional activates the protected-mode disk drivers, the virus cannot copy itself to other hard disks or floppy disks because the BIOS mechanism on which the virus depends is not used for disk access.

Guidelines for Avoiding Viruses

Follow these guidelines to avoid infecting computers with viruses:

- Install on your system at least one commercial virus-detection program and use it regularly to check your computers for viruses. Be sure to regularly update the virus signature files. After you install an antivirus program, immediately update the virus signature files from the software manufacturer's Internet site. Check with the software manufacturer's documentation for specific instructions.

  Important It is extremely important that you regularly update your antivirus program. In most cases, antivirus programs are unable to reliably detect and clean viruses of which they are unaware. Most commercial antivirus software manufacturers offer frequent updates. Take advantage of the latest download to ensure that your system is protected with the latest virus defenses.
- Before you install Windows XP Professional in a multiple-boot configuration, scan the other operating systems for viruses.
- Back up files nightly or as needed so that damage is minimized if a virus attack does occur.
- Before opening a file from a floppy disk or before starting a computer from a floppy disk, scan the floppy disk for viruses.
- Do not open e-mail attachments from unknown senders. Delete the e-mail and attachments immediately.
- When you receive an unexpected e-mail attachment from someone you know, verify that the sender intended to send you the attachment. Simply scanning the attachment for viruses is not sufficient because a new virus can propagate without the sender's knowledge. A virus scanner that does not know about the new virus might not catch the virus.

  If the sender did not intend to send you the attachment, permanently delete the e-mail without opening it.
- Never run a file that has a .vbs or .js file name extension unless you know exactly what it is going to do before you run it.
- Regularly check the Microsoft Windows Update Web site and the Microsoft Office Update Web site for patches that fix vulnerabilities and provide security enhancements. In addition, independent software vendors (ISVs) might also provide security-related patches for other programs installed on the computer. For more information, see the Windows Update and Microsoft Office Update links on the Web Resources Page at http://www.microsoft.com/windows/reskits/webresources .
- Configure the security settings in Microsoft Internet Explorer to protect against downloading infected files or malicious scripts. For more information about protecting computers from unsafe software, see Internet Explorer Help.
- Do not allow users to log on as members of the Administrators group on their own computers because viruses can do more damage if activated from an account with Administrator permissions. Allow users to log on as members of the Users group so that they have only the permissions that are necessary to

perform their tasks.

- Configure Windows Explorer and My Computer to show extensions for known file types, show hidden files and folders, and show protected operating system files. For example, a malicious file with the name Report.doc.vbs appears in Windows Explorer and My Computer as Report.doc unless you deselect the option to hide extensions for known file types. To change these settings, in My Computer, click the **Tools** menu, click **Folder Options**, and then click the **View** tab.

Treating an MBR or Boot Sector Virus Infection

To remove a virus from your computer, use a current, well-known commercial antivirus program that is compatible with Windows XP Professional. In addition to scanning the hard disks on your computer, be sure to scan all floppy disks that have been used in the infected computer, in any other computers, or with other operating systems in an infected multiple-boot configuration. Scan floppy disks even if you believe they are not infected. Many infections recur because one or more copies of the virus were not detected and eliminated.

If the computer is already infected with a boot sector virus and you install Windows XP Professional into a multiple-boot configuration, standard antivirus programs might not completely eliminate the infection because Windows XP Professional copies the original MS-DOS boot sector to a file called Bootsect.dos and replaces it with its own boot sector. The Windows XP Professional installation is not initially infected, but if the user chooses to start MS-DOS, Windows 95, Windows 98, or Windows Me, the infected boot sector is reapplied to the system, reinfecting the computer.

Avoid Using the Fdisk /mbr Command to Treat Viruses

Do not depend on the MS-DOS command **Fdisk /mbr**, which rewrites the MBR on the hard disk, to resolve MBR infections. Many newer viruses have the properties of both file infector and MBR viruses, so restoring the MBR does not solve the problem if the virus immediately reinfects the system. In addition, running **Fdisk /mbr** in MS-DOS on a system infected by an MBR virus that does not preserve or encrypt the original MBR partition table permanently prevents access to the lost partitions. If the disk was configured with a third-party drive overlay program to enable support for large disks, running this command eliminates the drive overlay program and you cannot start up from the disk.

Caution Before you use the **Fdisk /mbr** command, note the following:

- Running **Fdisk /mbr** is not supported on dynamic disks or GPT disks.
- Running **Fdisk /mbr** in MS-DOS overwrites only the first 446 bytes of the MBR, the portion known as the master boot code, leaving the existing partition table intact. However, if the signature word (the last two bytes of the MBR) has been deleted, the partition table entries are overwritten with zeros. If an MBR virus overwrites the signature word, access to all partitions and logical volumes is lost.

Avoid Using the Fixmbr Command to Treat Viruses

The Recovery Console, a troubleshooting tool in Windows XP Professional, offers a feature called **Fixmbr**. However, it functions identically to the **Fdisk /mbr** command, replacing only the master boot code and not affecting the partition table. For this reason, it is also unlikely to help resolve an infected MBR.

For more information about the Recovery Console, see "Tools for Troubleshooting" in this book.

Repairing Damaged MBRs and Boot Sectors in x86-based Computers

When you start a computer from the hard disk, the BIOS identifies the startup disk and reads the master boot record (MBR). The master boot code in the MBR searches for the active partition on the hard disk. If the first hard disk on the system does not contain an active partition, or if the master boot code cannot locate the boot sector of the system volume so that it can start the operating system, the MBR displays messages similar to the following:

```
Invalid partition table.
Error loading operating system.
Missing operating system.
```

If the active partition exists and the master boot record locates the boot sector of the system volume, the master boot code loads the boot sector of the active partition and transfers CPU execution to that memory address. On computers that are running Windows XP Professional, the executable boot code in the boot sector finds Ntldr, loads it into memory, and transfers execution to that file. However, if the boot sector cannot find Ntldr, which is the file that loads the operating system files from the boot volume, Windows XP Professional cannot start. Windows XP Professional might be unable to find Ntldr in these circumstances:

- If Ntldr is moved, renamed, or deleted.
- If Ntldr is corrupted.
- If the boot sector is corrupted.
- If you install Windows XP Professional and then later install any of the following on the same computer:

MS-DOS, Windows 95, Windows 98, or Windows NT 4.0. For more information about configuring a multiple-boot system, see "Planning Deployments" in this book.

Under the preceding circumstances, the computer might not respond to input or might display one of the following messages:

```
A disk read error occurred.
NTLDR is missing.
NTLDR is compressed.
```

Restoring the MBR

You must repair the MBR if it becomes corrupted and you can no longer access any volumes on that disk. You can use several tools to repair the MBR. Which tool you choose depends on whether the partition table is also damaged and whether you can start Windows XP Professional.

- **Use the Recovery Console.** You can use the **fixmbr** command in Recovery Console to repair the MBR. You can start Recovery Console by booting from the Windows XP Professional operating system CD; so this troubleshooting method is available even if Windows XP Professional does not start in normal or safe mode. However, you cannot use Recovery Console to repair partition tables that were damaged by viruses or other corruption.
- **Use DiskProbe.** You can use DiskProbe to restore both the MBR and the partition table, but you must have previously backed up this information by using DiskProbe, and you must be able to start Windows XP Professional.
- **Use a third-party disk editor.** You can use a third-party MS-DOS-based, low-level disk editor to repair the partition table if Windows XP Professional does not start. This method is for experienced users only and involves manually editing the partition table.

Using the Recovery Console to Replace the MBR

You can use the **fixmbr** command in Recovery Console to rewrite the MBR to resolve a corrupted MBR on a startup disk. However, running **fixmbr** overwrites only the master boot code, leaving the existing partition table intact. If the corruption in the MBR affects the partition table, running **fixmbr** might not resolve the problem.

Caution Use this command with care because it can damage your partition table if any of the following apply:

- A virus is present and a third-party operating system is installed on the same computer.
- A nonstandard MBR is installed by a third-party disk utility.
- A hardware problem exists.

It is recommended that you run antivirus software before you use the **fixbr** command.

To start the computer and use the Recovery Console to replace the MBR

1. Insert the Windows XP Professional Setup CD-ROM into the CD-ROM drive.
2. Restart the computer. If prompted to press a key to start the computer from the CD-ROM, press the appropriate key.
3. When the text-based part of Setup begins, follow the prompts. Press the **R** key to repair a Windows XP Professional installation.
4. If you are repairing a system that has more than one operating system installed, from the Recovery Console choose the Windows XP Professional installation that you need to repair.

   Note If you press ENTER without typing a number, the Recovery Console quits and restarts the computer.

   The Recovery Console might also show valid installations of Windows NT 4.0. However, the results of attempting to access a Windows NT 4.0 installation can be unpredictable.
5. When prompted, type the Administrator password. If you do not have the correct password, or if the security database for the installation of Windows XP Professional you are attempting to access is corrupted, Recovery Console does not allow access to the local disks and you cannot repair the MBR.
6. To replace the MBR, at the Recovery Console command prompt, type:

   ```
   fixmbr
   ```

Verify if you want to proceed. Depending upon the location and the cause of the corruption within the damaged MBR, this operation can cause the data on the hard disk to become inaccessible. Press the **Y** key to proceed, or press the **N** key to cancel.

Using DiskProbe to Replace the MBR and Partition Table

If you have backed up the MBR by using DiskProbe, you can use it to restore the MBR on any disk that is not

used to start the computer. Restoring the backup MBR rewrites the entire sector, including the partition table. However, DiskProbe only runs under Windows XP Professional, Windows 2000 and Windows NT 4.0. It does not run under MS-DOS, Windows 95, Windows 98, or Windows Me.

If the disk that starts Windows XP Professional has a corrupted MBR, Windows XP Professional does not start. Therefore, you cannot use DiskProbe and must use the Recovery Console to replace the MBR.

For more information about restoring backed up MBRs by using DiskProbe, click **Tools** in Help and Support Center, and then click **Windows Support Tools**.

Using a Third-Party Disk Editor to Replace the Partition Table

Before you can repair the partition table, you must know the exact values to use to recreate the partition table. If you backed up your MBR and partition table by using DiskProbe, and you have the backup available on a floppy disk or on another computer, you can use DiskProbe on a different computer to see the correct values so that you can manually recreate the partition table.

Replacing the Boot Sector

If Ntldr is damaged or missing, or if the boot sector is corrupted, you can resolve either problem by using the Recovery Console.

To start the computer and use the Recovery Console to replace the boot sector

1. Insert the Windows XP Professional Setup CD-ROM into the CD-ROM drive.
2. Restart the computer. If prompted to press a key to start the computer from the CD-ROM, press the appropriate key.
3. When the text-based part of Setup begins, follow the prompts. Press the **R** key to repair a Windows XP Professional installation.
4. If you are repairing a system that has more than one operating system installed, from the Recovery Console choose the Windows XP Professional installation that you need to repair.

   Note If you press ENTER without typing a number, the Recovery Console quits and restarts the computer.

   The Recovery Console might also show valid installations of Windows NT 4.0. However, the results of attempting to access a Windows NT 4.0 installation can be unpredictable.
5. When prompted, type the Administrator password. If you do not have the correct password, or if the security database for the installation of Windows XP Professional that you are attempting to access is corrupted, Recovery Console does not allow access to the local disks and you cannot replace the boot sector.
6. To replace the boot sector, at the Recovery Console command prompt, type:

   ```
   fixboot [drive:]
   ```

If you do not specify a drive letter, the Recovery Console replaces the boot sector of the system volume. If you need to replace the boot sector of a volume that is not the system volume, then you must specify the appropriate drive letter.

Using a Disk Editor to Replace the Boot Sector

If the boot sector is not from the boot volume on the hard disk, you can use several methods to replace it. If you backed up the boot sector by using DiskProbe, then restoring it by using DiskProbe is the fastest method.

If you want to replace the boot sector on an NTFS volume, you have another alternative. When you create or reformat an existing volume as an NTFS volume, NTFS writes a duplicate of the boot sector in the following location:

- **At the end of the volume** on volumes formatted with Windows XP Professional, Windows 2000, and Windows NT 4.0.
- **At the logical center of the volume** on disks formatted with Windows NT 3.51 and earlier.

You can use DiskProbe to locate and copy a duplicate boot sector to the beginning of the volume. There are also third-party MS-DOS-based disk tools that you can use to locate and copy this backup boot sector to the primary boot sector on the volume.

For specifically replacing corrupted boot sectors from boot volumes, DiskProbe is not always an option. Unless you created a Windows XP Professional startup floppy disk, you cannot start Windows XP Professional, which is required by DiskProbe. You can use a third-party MS-DOS-based, low-level disk editor to restore the backup boot sector.

For more information about creating a startup floppy disk, see article Q119467, "How to Create a Bootable Disk for an NTFS or FAT Partition." To find this article, see the Microsoft Knowledge Base link on the Web Resources page at http://www.microsoft.com/windows/reskits/webresources .

Stop Messages for Disks and File Systems

When Windows XP Professional detects an error from which it cannot recover, it reports error information in full screen, non-windowed, text mode. These Stop messages, which are also referred to as stop errors or blue screens, provide information that is specific to the problem detected by the Windows XP Professional kernel.

File system errors, viruses, hard disk corruption, or controller problems can cause the following Stop messages.

Stop 0x00000024 or NTFS_FILE_SYSTEM

This Stop message, also known as Stop 0x24, indicates that a problem occurred within Ntfs.sys, which is the driver file that allows the system to read and write to NTFS volumes.

Stop 0x00000050 or PAGE_FAULT_IN_NONPAGED_AREA

This Stop message, also known as Stop 0x50, occurs when requested data is not found in memory. The system generates a fault, which indicates that invalid system memory has been referenced. This fault can occur due to a variety of error conditions, such as bugs in antivirus software, a corrupted NTFS volume, or faulty hardware (typically related to defective RAM, be it main memory, L2 RAM cache, or video RAM).

Stop 0x00000077 or KERNEL_STACK_INPAGE_ERROR

This Stop message, also known as Stop 0x77, indicates that the requested page of kernel data from the paging file could not be read into memory. Stop 0x77 can be caused by a number of problems, such as:

- Bad sectors on the hard disk.
- Defective or loose cabling, improper SCSI termination, or the controller not seeing the hard disk.
- Another device is causing a resource conflict with the storage controller.
- Failing RAM.

Stop 0x0000007A or KERNEL_DATA_INPAGE_ERROR

This Stop message, also known as Stop 0x7A, indicates that the requested page of kernel data from the paging file could not be read into memory.

One of the following conditions usually causes a Stop 0x7A: a bad sector in a paging file, a virus, a disk controller error, defective hardware, or failing RAM. In rare cases, a Stop 0x7A occurs when nonpaged pool resources run out.

Stop 0x0000007B or INACCESSIBLE_BOOT_DEVICE

This Stop message, also known as Stop 0x7B, indicates that Windows XP Professional lost access to the system volume or boot volume during the startup process. This error always occurs while the system is starting and is often caused by one of the following:

- Hardware problems
- Corrupted or incompatible storage drivers
- File system problems
- Boot sector viruses
- Outdated firmware

During I/O system initialization, this error can occur when:

- The controller or driver for the startup device (typically the hard disk) failed to initialize the necessary hardware.
- File system initialization failed because the system did not recognize the data on the boot device.

For more information about these and other Stop messages, see "Common Stop Messages for Troubleshooting" in this book.

Other Disk Problems

Disk problems can occur that do not involve the MBR, partition table, or boot sector. Typically, you cannot use the Windows XP Professional disk tools to troubleshoot these disk problems.

CMOS Problems

The CMOS setup utility is accessible during startup. CMOS typically stores configuration information about the basic elements of the computer, including RAM, video, and storage devices. If CMOS is damaged or incapable of retaining its configuration data, the computer might be unable to start.

Each manufacturer and BIOS vendor can decide what a user can configure in the CMOS utility, and what the

standard configuration is. You can access the CMOS utility by using the keyboard sequence that is displayed during startup or by using a software tool, depending on the manufacturer's specifications. It is recommended that you record or print all CMOS information.

The computer uses the CMOS checksum to determine if any CMOS values have been changed other than by using the CMOS setup program. If the checksum is not correct, the computer cannot start.

After the CMOS is correctly configured, any CMOS problem is usually caused by one of the following situations:

- A weak battery, which can happen when the computer has been turned off for a long time.
- A loose or faulty connection between the CMOS and the battery.
- A damaged CMOS caused by static electric discharge.

Cables and Connectors

Another source of disk problems can be cabling and connectors. Cables can become defective, but if the cable works initially, it is likely to work for a long time. When new disks are added to the computer, check for cabling problems. New problems might stem from a previously unused connector on an existing cable or from a faulty, longer cable used to connect all the disks that might have replaced the working original. Also check the connections to the disk themselves. If the cables are tightly stretched, one or more connectors might work themselves loose over time, resulting in intermittent problems with the disks.

If your system has *small computer system interface* (SCSI) adapters, contact the manufacturer for updated Windows XP Professional drivers. Try disabling **sync negotiation** in the SCSI BIOS, checking the SCSI identifiers of each device, and confirming proper termination. For *AT Attachment* devices, define the onboard IDE port as **Primary only**. Also, check each ATA device for the proper master, slave, or stand-alone setting. Try removing all ATA devices except for hard disks. For universal serial bus and IEEE 1394 disks, verify that the cables are correctly connected and that the adapter card, if used, is securely seated.

To make sure that any new disks and disk controllers are supported, see the Microsoft Windows XP Professional Hardware Compatibility List (HCL) link on the Web Resources page at http://www.microsoft.com/windows/reskits/webresources .

Disk Sectors Critical to Startup

A sector is a unit of storage on a hard disk and is typically 512 bytes. Computers access certain sectors on a hard disk during startup to determine which operating system to start and where the partitions are located. The data stored on these sectors varies depending on the computer platform.

Computers that are x86-based begin the startup process from disks that contain a master boot record and are referred to as MBR disks. Itanium-based computers start up from disks that contain a GUID partition table and are referred to as GPT disks. MBR and GPT disks each have disk sectors critical to startup, but the differences in the sectors are not visible in the graphical user interface. Instead, you must use a disk-editing tool, such as DiskProbe, to see how the data on these sectors is structured. For more information about using DiskProbe, see "DiskProbe" earlier in this chapter.

Caution The ability to edit and repair these sectors on a byte-by-byte basis after corruption has occurred is invaluable, but using a disk-editing tool requires a thorough understanding of how data is organized on the sectors. If you make a mistake, you can damage or permanently overwrite critical on-disk data structures that might make all data on a disk or volume permanently inaccessible. Therefore, it is a good idea to use the information in this chapter to learn about the sectors used for startup, and then to examine the sectors on your own disks to further your learning.

Disk Sectors on MBR Disks

The two sectors critical to starting x86-based computers are the following:

- The master boot record (MBR), which is always located at sector 1 of cylinder 0, head 0, the first sector of a hard disk.
- The boot sector, which resides at sector 1 of each volume.

These sectors contain both executable code and the data required to run the code.

Master Boot Record on Basic Disks

The MBR, the most important data structure on the disk, is created when the disk is partitioned. The MBR contains a small amount of executable code called the master boot code, the disk signature, and the partition table for the disk. At the end of the MBR is a 2-byte structure called a signature word or end of sector marker, which is always set to 0x55AA. A signature word also marks the end of an *extended boot record* (EBR) and the boot sector.

The disk signature, a unique number at offset 0x01B8, identifies the disk to the operating system. Windows XP Professional uses the disk signature as an index to store and retrieve disk information, such as drive letters, in

the registry.

Master Boot Code

The *master boot code* performs the following activities:

1. Scans the partition table for the active partition.
2. Finds the starting sector of the active partition.
3. Loads a copy of the boot sector from the active partition into memory.
4. Transfers control to the executable code in the boot sector.

If the master boot code cannot complete these functions, the system displays a message similar to one of the following:

```
Invalid partition table.
Error loading operating system.
Missing operating system.
```

Note Floppy disks and removable disks, such as Iomega Zip disks, do not contain an MBR. The first sector on these disks is the boot sector. Although every hard disk contains an MBR, the master boot code is used only if the disk contains the active primary partition.

For more information about troubleshooting MBR problems, see "Repairing Damaged MBRs and Boot Sectors in x86-based Computers" earlier in this chapter.

Partition Table on Basic Disks

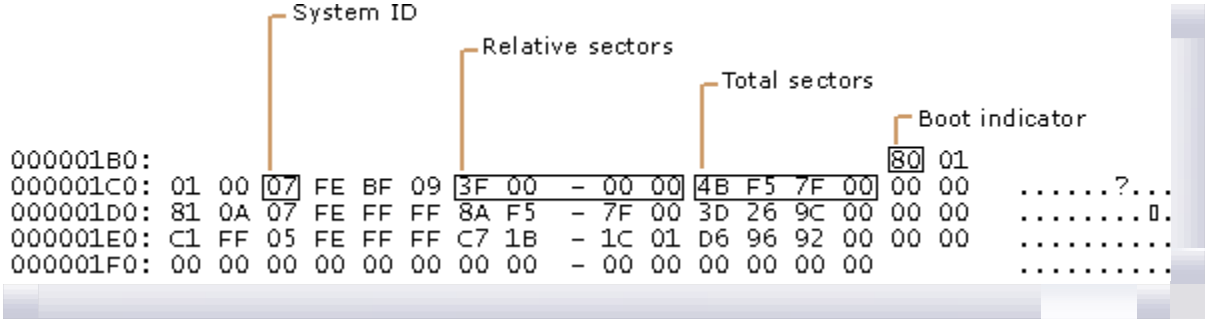
The partition table, which is a 64-byte data structure that is used to identify the type and location of partitions on a hard disk, conforms to a standard layout independent of the operating system. Each partition table entry is 16 bytes long, with a maximum of four entries. Each entry starts at a predetermined offset from the beginning of the sector, as follows:

- Partition 1 0x01BE (446)
- Partition 2 0x01CE (462)
- Partition 3 0x01DE (478)
- Partition 4 0x01EE (494)

The following example shows a partial printout of an MBR revealing the partition table from a computer with three partitions. When there are fewer than four partitions on a disk, the remaining partition table fields are set to the value 0.

```
000001B0: 80 01 ..
000001C0: 01 00 07 FE BF 09 3F 00 - 00 00 4B F5 7F 00 00 00 ......?...K.....
000001D0: 81 0A 07 FE FF FF 8A F5 - 7F 00 3D 26 9C 00 00 00 ..........=&....
000001E0: C1 FF 05 FE FF FF C7 1B - 1C 01 D6 96 92 00 00 00 ................
000001F0: 00 00 00 00 00 00 00 00 - 00 00 00 00 00 00 ..............
```

Figure 27.5 provides an example of how to interpret the sector printout of the partition table by using Table 27.7. The Boot indicator, System ID, Relative sectors, and Total sectors values correspond to Table 27.7.



If your browser does not support inline frames, click here to view on a separate page.

Figure 27.5 Interpreting data in the partition table

Table 27.7 describes the fields in each entry in the partition table. The sample values correspond to the first partition table entry shown in this example. The Byte Offset values correspond to the addresses of the first partition table entry. There are three additional entries whose values can be calculated by adding 10h to the byte offset value specific for each additional partition table entry (for example, add 20h for partition table entry 3 and 30h for partition table entry 4).

The subsections that follow Table 27.7 provide additional detail about these fields.

Table 27.7 Partition Table Fields

Byte Offset	Field Length	Sample Value (1)	Field Name and Definition
0x01BE	1 byte	0x80	**Boot Indicator**. Indicates whether the volume is the active partition. Legal values include: • 00. Do not use for booting. • 80. Active partition.
0x01BF	1 byte	0x01	**Starting Head**.
0x01C0	6 bits	0x01 ***(2)**	**Starting Sector**. Only bits 0–5 are used. The upper two bits, 6 and 7, are used by the Starting Cylinder field.
0x01C1	10 bits	0x00 *	**Starting Cylinder**. Uses 1 byte in addition to the upper 2 bits from the Starting Sector field to make up the cylinder value. The Starting Cylinder is a 10-bit number that has a maximum value of 1023.
0x01C2	1 byte	0x07	**System ID**. Defines the volume type. See Table 27.8 for sample values.
0x01C3	1 byte	0xFE	**Ending Head**.
0x01C4	6 bits	0xBF *	**Ending Sector**. Only bits 0–5 are used. The upper two bits, 6 and 7, are used by the Ending Cylinder field.
0x01C5	10 bits	0x09 *	**Ending Cylinder**. Uses 1 byte in addition to the upper 2 bits from the Ending Sector field to make up the cylinder value. The Ending Cylinder is a 10-bit number, with a maximum value of 1023.
0x01C6	4 bytes	0x3F000000	**Relative Sectors**. The offset from the beginning of the disk to the beginning of the volume, counting by sectors.
0x01CA	4 bytes	0x4BF57F00	**Total Sectors**. The total number of sectors in the volume.

(1) Numbers larger than one byte are stored in little endian format, or reverse-byte ordering. Little endian format is a method of storing a number so that the least significant byte appears first in the hexadecimal number notation. For example, the sample value for the Relative Sectors field in the previous table, 0x3F000000, is a little endian representation of 0x0000003F. The decimal equivalent of this little endian number is 63.

(2) Sample values marked with an asterisk (*) do not accurately represent the value of the fields, because the fields are either 6 bits or 10 bits and the data is recorded in bytes.

Boot Indicator Field

The first element of the partition table, the **Boot Indicator** field, indicates whether the volume is the active partition. Only one primary partition on the disk can have this field set. See Table 27.7 for the legal values.

It is possible to have different operating systems and different file systems on different volumes. By using disk configuration tools, such as the Windows XP Professional-based Disk Management and DiskPart, or the MS-DOS-based Fdisk to designate a primary partition as active, the **Boot Indicator** field for that partition is set in the partition table.

System ID Field

Another element of the partition table is the **System ID** field. It defines which file system, such as FAT16, FAT32, or NTFS, was used to format the volume. The **System ID** field also identifies an extended partition, if one is defined. Windows XP Professional uses the **System ID** field to determine which file system device drivers to load during startup. Table 27.8 identifies the values for the **System ID** field.

Table 27.8 System ID Values

Partition Type	ID Value
0x01	FAT12 primary partition or logical drive (fewer than 32,680 sectors in the volume)
0x04	FAT16 partition or logical drive (32,680–65,535 sectors or 16 MB–33 MB)
0x05	Extended partition

0x06	BIGDOS FAT16 partition or logical drive (33 MB–4 GB)
0x07	Installable File System (NTFS partition or logical drive)
0x0B	FAT32 partition or logical drive
0x0C	FAT32 partition or logical drive using BIOS INT 13h extensions
0x0E	BIGDOS FAT16 partition or logical drive using BIOS INT 13h extensions
0x0F	Extended partition using BIOS INT 13h extensions
0x12	EISA partition or OEM partition
0x42	Dynamic volume
0x84	Power management hibernation partition
0x86	Multidisk FAT16 volume created by using Windows NT 4.0
0x87	Multidisk NTFS volume created by using Windows NT 4.0
0xA0	Laptop hibernation partition
0xDE	Dell OEM partition
0xFE	IBM OEM partition
0xEE	GPT partition
0xEF	EFI System partition on an MBR disk

Windows XP Professional does not support multidisk volumes that are created by using Windows NT 4.0 and earlier, and that use System ID values 0x86, 0x87, 0x8B, or 0x8C.

If you are upgrading from Windows NT Workstation 4.0 to Windows XP Professional, you must first back up and then delete all multidisk volumes before you upgrade. After you complete the upgrade, create dynamic volumes and restore the data . If you do not delete the multidisk volumes before beginning Setup, you must use the Ftonline tool, which is part of Windows Support Tools, to access the volume after Setup completes. For more information about using Ftonline.exe, click **Tools** in Help and Support Center, and then click **Windows Support Tools**.

If you are upgrading from Windows 2000 to Windows XP Professional, you must convert the multidisk volumes to dynamic before you begin Setup, or Setup does not continue. For more information about multidisk volumes and Setup, see "Disk Management" in this book.

MS-DOS can only access volumes that have a System ID value of 0x01, 0x04, 0x05, or 0x06. However, you can delete volumes that have the other values listed in Table 27.8 by using Disk Management, DiskPart, or the MS-DOS tool Fdisk.

Starting and Ending Cylinder, Head, and Sector Fields

The **Starting** and **Ending Cylinder**, **Head**, and **Sector** fields (collectively known as the **CHS** fields) are additional elements of the partition table. These fields are essential for starting the computer. The master boot code uses these fields to find and load the boot sector of the active partition. The **Starting CHS** fields for non-active partitions point to the boot sectors of the remaining primary partitions and the extended boot record (EBR) of the first logical drive in the extended partition, as shown in Figure 27.6.

Knowing the starting sector of an extended partition is very important for low-level disk troubleshooting. If your disk fails, you need to work with the partition starting point (among other factors) to retrieve stored data.

The **Ending Cylinder** field in the partition table is 10 bits long, which limits the number of cylinders that can be described in the partition table to a range of 0 through 1,023. The **Starting Head** and **Ending Head** fields are each one byte long, which limits the field range from 0 through 255. The **Starting Sector** and **Ending Sector** fields are each six bits long, which limits the range of these fields from 0 through 63. However, the enumeration of sectors starts at 1 (not 0, as for other fields), so the maximum number of sectors per track is 63.

Because all hard disks are low-level formatted with a standard 512-byte sector, the maximum disk capacity described by the partition table is calculated as follows:

Maximum capacity = sector size x cylinders (10 bits) x heads (8 bits) x sectors per track (6 bits)

Using the maximum possible values yields:

512 x 1024 x 256 x 63 (or 512 x 224) = 8,455,716,864 bytes or 7.8 GB

Windows XP Professional and other Windows-based operating systems that support BIOS INT 13h extensions

can access partitions that exceed the first 7.8 GB of the disk by ignoring the **Starting** and **Ending CHS** fields in favor of the **Relative Sectors** and **Total Sectors** fields.

Windows 2000 and Windows XP Professional ignore the **Starting** and **Ending CHS** fields regardless of whether the partition exceeds the first 7.8 GB of the disk. However, Windows XP Professional must place the appropriate values in the **Starting** and **Ending CHS** fields because Windows 95, Windows 98, and Windows Me (which all support BIOS INT 13h extensions) use the **Starting** and **Ending CHS** fields if the partition does not exceed the first 7.8 GB of the disk. These fields are also required to maintain compatibility with the BIOS INT 13h for startup.

MS-DOS and other Windows operating systems that do not support BIOS INT 13h extensions ignore partitions that exceed the 7.8 GB boundary because these partitions use a System ID that is recognized only by operating systems that support BIOS INT 13h extensions.

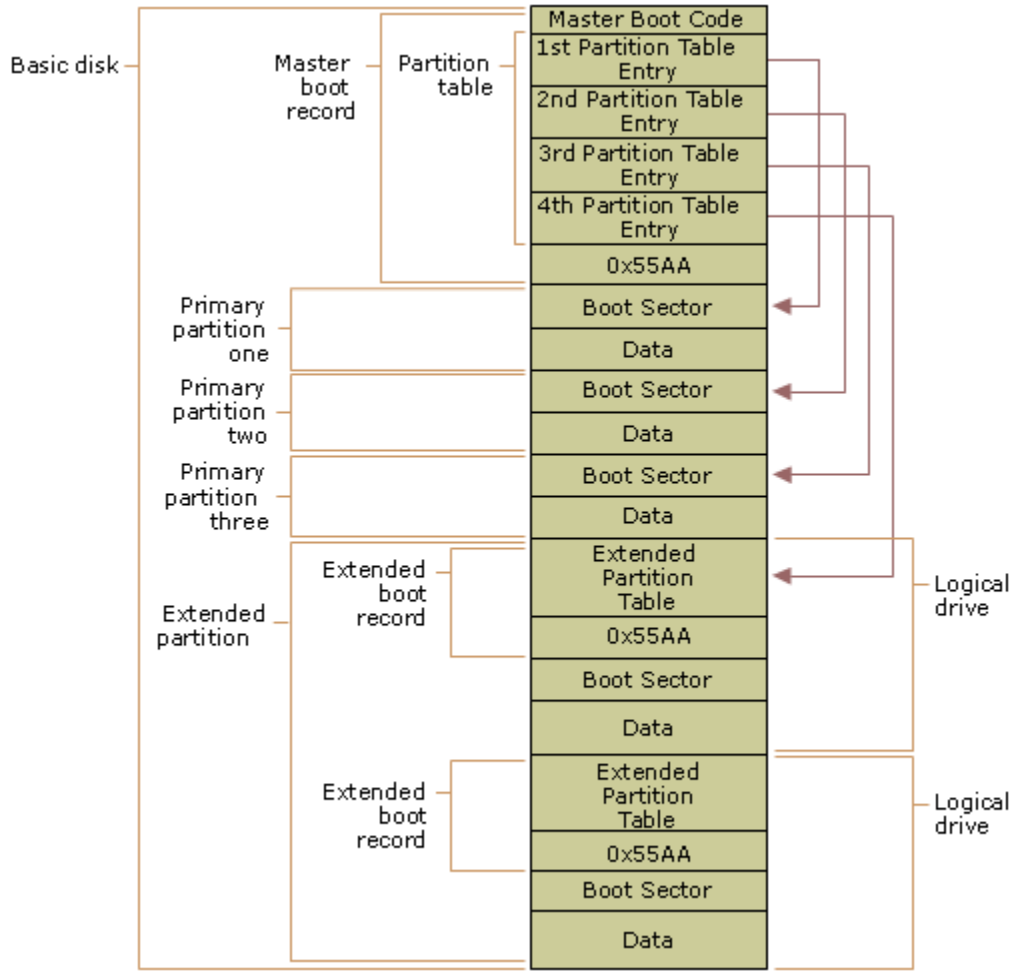
Both the operating system and the computer must support BIOS INT 13h extensions if you want to create partitions that exceed the first 7.8 GB of the disk.

Relative Sectors and Total Sectors Fields

The **Relative Sectors** field represents the offset from the beginning of the disk to the beginning of the volume, counting by sectors, for the volume described by the partition table entry. The **Total Sectors** field represents the total number of sectors in the volume.

Using the **Relative Sectors** and **Total Sectors** fields (resulting in a 32-bit number) provides eight more bits than the CHS scheme to represent the total number of sectors. This allows you to create partitions that contain up to 2^{32} sectors. With a standard sector size of 512 bytes, the 32 bits used to represent the **Relative Sectors** and **Total Sectors** fields translates into a maximum partition size of 2 terabytes (or 2,199,023,255,552 bytes).

Figure 27.6 shows the MBR, partition table, and boot sectors on a basic disk with four partitions. The definitions of the fields in the partition table and the extended partition tables are the same.



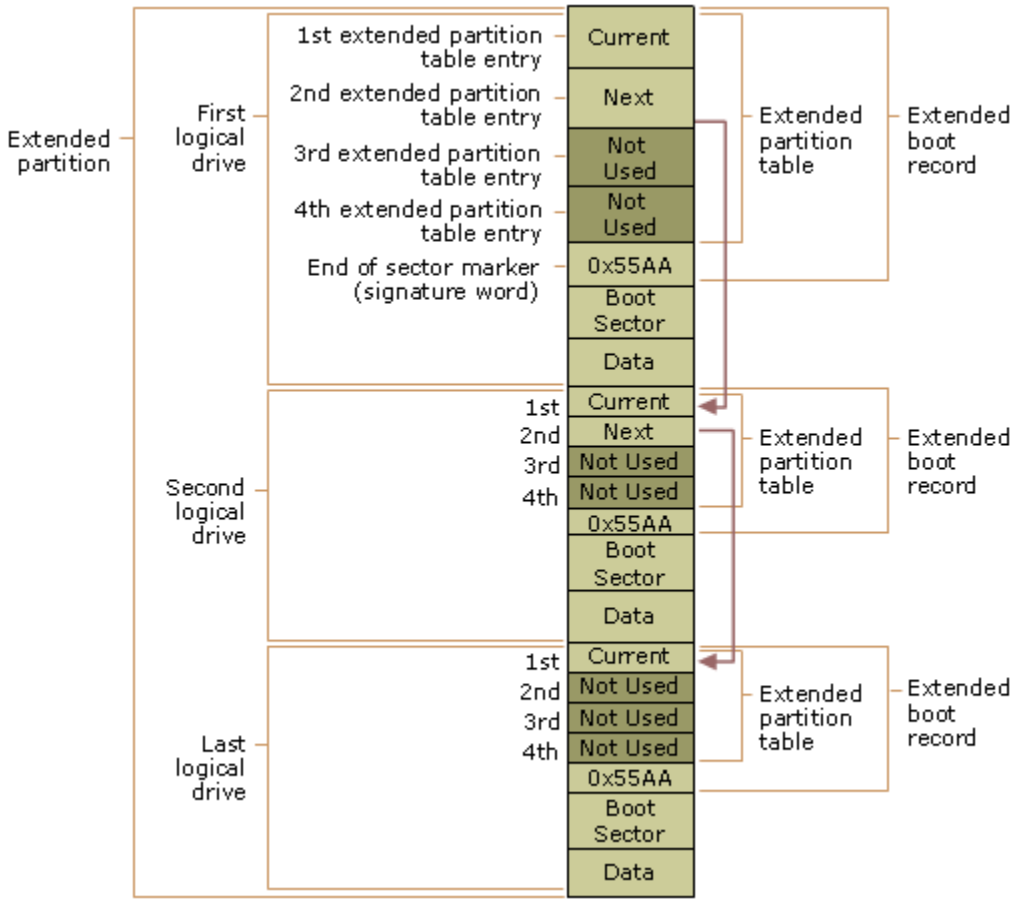
If your browser does not support inline frames, click here to view on a separate page.

Figure 27.6 Detail of a basic disk with four partitions

Note For more information about the maximum partition size that each file system supports, see "File Systems" in this book.

Extended Boot Record on Basic Disks

An EBR, which consists of an extended partition table and the signature word for the sector, exists for each logical drive in the extended partition. It contains the only information on the first side of the first cylinder of each logical drive in the extended partition. The boot sector in a logical drive is usually located at either Relative Sector 32 or 63. However, if there is no extended partition on a disk, there are no EBRs and no logical drives.



If your browser does not support inline frames, click here to view on a separate page.

Figure 27.7 Detail of an extended partition on a basic disk

The first entry in an extended partition table for the first logical drive points to its own boot sector. The second entry points to the EBR of the next logical drive. If no further logical drives exist, the second entry is not used and is recorded as a series of zeros. If there are additional logical drives, the first entry of the extended partition table for the second logical drive points to its own boot sector. The second entry of the extended partition table for the second logical drive points to the EBR of the next logical drive. The third and fourth entries of an extended partition table are never used.

As shown in Figure 27.7, the EBRs of the logical drives in the extended partition are a linked list. The figure shows three logical drives on an extended partition, illustrating the difference in extended partition tables between preceding logical drives and the last logical drive.

With the exception of the last logical drive on the extended partition, the format of the extended partition table, which is described in Table 27.9, is repeated for each logical drive: the first entry identifies the logical drive's own boot sector and the second entry identifies the next logical drive's EBR. The extended partition table for the last logical drive has only its own partition entry listed. The second through fourth entries of the last extended partition table are not used.

Table 27.9 Contents of Extended Partition Table Entries

Entry	Entry Contents
First	Information about the current logical drive in the extended partition, including the starting address for data.
Second	Information about the next logical drive in the extended partition, including the address of the sector that contains the EBR for the next logical drive. If no additional logical drives exist, this field is not used.
Third	Not used.
Fourth	Not used.

The fields in each entry of the extended partition table are identical to the MBR partition table entries. See Table 27.7 for more information about partition table fields.

The **Relative Sectors** field in an extended partition table entry shows the number of bytes that are offset from the beginning of the extended partition to the first sector in the logical drive. The number in the **Total Sectors** field refers to the number of sectors that make up the logical drive. The value of the **Total Sectors** field equals the number of sectors from the boot sector defined by the extended partition table entry to the end of the logical drive.

Because of the importance of the MBR and EBR sectors, it is recommended that you run disk-scanning tools regularly and that you regularly back up all your data files to protect against losing access to a volume or an entire disk.

Master Boot Record on Dynamic Disks

Like basic disks, dynamic disks contain an MBR that includes the master boot code, the disk signature, and the partition table for the disk. However, the partition table on a dynamic disk does not contain an entry for each volume on the disk because volume information is stored in the dynamic disk database. Instead, the partition table contains entries for the system volume, boot volume (if it is not the same as the system volume), and one or more additional partitions that cover all the remaining unallocated space on the disk. All these partitions use System ID 0x42, which indicates that these partitions are on a dynamic disk. Placing these partitions in the partition table prevents MBR-based disk utilities from interpreting the space as available for new partitions.

Note In Windows 2000, the partition entries for existing basic volumes were preserved in the partition table when the disk was converted to dynamic. These entries prevented the converted dynamic volumes from being extended. This limitation has been removed from Windows XP Professional for all converted volumes except the boot and system volumes. Partition entries for all other converted volumes are removed from the partition table, and therefore these volumes can be extended.

The following example shows a partial printout of an MBR on a dynamic disk that contains four simple volumes: the system volume, the boot volume, and two data volumes. The first entry is the system volume, which is marked as active. The second entry is the boot volume, and the third entry is the container volume for all other simple volumes on the disk. All entries are type 0x42, which specifies dynamic volumes.

```
000001B0: 80 01 .....,Dc!.!.....
000001C0: 01 00 42 FE 7F 04 3F 00 - 00 00 86 FA 3F 00 00 00 ..B...?.....?...
000001D0: 41 05 42 FE FF 02 C5 FA - 3F 00 7E 04 7D 00 00 00 A.B.....?.~.}...
000001E0: C1 03 42 FE FF FF 43 FF - BC 00 58 53 54 00 00 00 ..B...C...XST...
000001F0: 00 00 00 00 00 00 00 00 - 00 00 00 00 00 00 55 AA ..............U.
```

Boot Sectors on MBR Disks

The boot sector, which is located at sector 1 of each volume, is a critical disk structure for starting your computer. It contains executable code and the data required by the code, including information that the file system uses to access the volume. The boot sector is created when you format a volume. At the end of the boot sector is a 2-byte structure called a signature word or end of sector marker, which is always set to 0x55AA. On computers running Windows XP Professional, the boot sector on the active partition loads into memory and starts Ntldr, which loads the boot menu if multiple versions of Windows are installed or loads the operating system if only one operating system is installed.

A boot sector consists of the following elements:

- An x86-based CPU jump instruction.
- The original equipment manufacturer identification (OEM ID).
- The BIOS parameter block (BPB), a data structure.
- The extended BPB.
- The executable boot code (or bootstrap code) that starts the operating system.

All Windows XP Professional boot sectors contain the preceding elements regardless of the type of disk (basic disk or dynamic disk).

Note Disk editing tools such as DiskProbe and third-party tools that work with Windows NT 4.0 and NTFS might not support FAT32 boot sectors and volumes.

The BPB describes the physical parameters of the volume: the extended BPB begins immediately after the BPB. Due to differing types of fields and the amount of data they contain, the length of the BPB is different for FAT16, FAT32, and NTFS boot sectors.

Disk device drivers use the information in the BPB and the extended BPB to read and configure volumes. The area following the extended BPB typically contains executable boot code, which performs the actions necessary to continue the startup process.

Boot Sector Startup Processes

Computers use the boot sector to run instructions during startup. The initial startup process is summarized in the following steps:

1. The system BIOS and the CPU initiate the power-on self test (POST).
2. The BIOS finds the boot device, which is typically the first disk the BIOS finds, unless the controller is configured to boot from a different disk.
3. The BIOS loads the first physical sector of the boot device into memory and transfers CPU execution to that memory address.

If the boot device is on a hard disk, the BIOS loads the MBR. The master boot code in the MBR loads the boot sector of the active partition, and transfers CPU execution to that memory address. On computers that are running Windows XP Professional, the executable boot code in the boot sector finds Ntldr, loads it into memory, and transfers execution to that file.

Note Windows XP Professional cannot start up from a spanned or striped volume on dynamic disks. These disk structures cannot be registered into the MBR partition table; therefore, a system volume that uses these structures cannot start.

If drive A contains a floppy disk, the system BIOS loads the first sector (the boot sector) of the disk into memory. If the disk is a startup disk (formatted by MS-DOS with core operating system files applied), the boot sector loads into memory and uses the executable boot code to transfer CPU execution to Io.sys, a core MS-DOS operating system file. If the floppy disk is not a startup disk, the executable boot code displays a message such as the following:

```
Non-system disk or disk error.
Replace and press any key when ready.
```

Note These messages do not appear on normally functioning systems that are configured to look for the startup files on drive C first. On many computers, an option in the CMOS setup program allows the user to set the sequence of installed disks that the system searches to find the startup files.

If you get similar errors when trying to start the computer from the hard disk, the boot sector might be corrupted. For more information about troubleshooting boot sector problems, see "Repairing Damaged MBRs and Boot Sectors in x86-based Computers" earlier in this chapter.

Initially, the startup process is independent of disk format and operating system. The unique characteristics of operating and file systems become important when the boot sector's executable boot code starts.

Components of a Boot Sector

The MBR transfers CPU execution to the boot sector, so the first three bytes of the boot sector must be valid, executable x86-based CPU instructions. This includes a jump instruction that skips the next several nonexecutable bytes.

Following the jump instruction is the 8-byte OEM ID, a string of characters that identifies the name and version number of the operating system that formatted the volume. To preserve compatibility with MS-DOS, Windows XP Professional records "MSDOS5.0" in this field on FAT16 and FAT32 disks. On NTFS disks, Windows XP Professional records "NTFS."

Note You might also see the OEM ID "MSWIN4.0" on disks formatted by Windows 95 and "MSWIN4.1" on disks formatted by Windows 95 OEM Service Release 2 (OSR2), Windows 98, and Windows Me. Windows XP Professional does not use the OEM ID field in the boot sector except for verifying NTFS volumes.

Following the OEM ID is the BPB, which provides information that enables the executable boot code to locate Ntldr. The BPB always starts at the same offset, so standard parameters are in a known location. Disk size and geometry variables are encapsulated in the BPB. Because the first part of the boot sector is an x86 jump instruction, the BPB can be extended in the future by appending new information at the end. The jump instruction needs only a minor adjustment to accommodate this change. The BPB is stored in a packed (unaligned) format.

FAT16 Boot Sector

Table 27.10 describes the boot sector of a volume formatted with the FAT16 file system.

Table 27.10 Boot Sector Sections on a FAT16 Volume

Byte Offset	Field Length	Field Name
0x00	3 bytes	Jump instruction
0x03	8 bytes	OEM ID
0x0B	25 bytes	BPB
0x24	26 bytes	Extended BPB
0x3E	448 bytes	Bootstrap code
0x01FE	2 bytes	End of sector marker

The following example illustrates a hexadecimal printout of the boot sector on a FAT16 volume. The printout is formatted in three sections:

- Bytes 0x00– 0x0A are the jump instruction and the OEM ID (shown in bold print).
- Bytes 0x0B– 0x3D are the BPB and the extended BPB.
- The remaining section is the bootstrap code and the end of sector marker (shown in bold print).

```
Physical Sector: Cyl 0, Side 1, Sector 1
00000000: EB 3C 90 4D 53 44 4F 53 - 35 2E 30 00 02 40 01 00 .<.MSDOS5.0..@..
00000010: 02 00 02 00 00 F8 FC 00 - 3F 00 40 00 3F 00 00 00 ........?.@.?...
00000020: 01 F0 3E 00 80 00 29 A8 - 8B 36 52 4E 4F 20 4E 41 ..>...)..6RNO NA
00000030: 4D 45 20 20 20 20 46 41 - 54 31 36 20 20 20 33 C0 ME FAT16 3.
00000040: 8E D0 BC 00 7C 68 C0 07 - 1F A0 10 00 F7 26 16 00 ....|h......&..
00000050: 03 06 0E 00 50 91 B8 20 - 00 F7 26 11 00 8B 1E 0B ....P.. ..&.....
00000060: 00 03 C3 48 F7 F3 03 C8 - 89 0E 08 02 68 00 10 07 ...H........h...
00000070: 33 DB 8F 06 13 02 89 1E - 15 02 0E E8 90 00 72 57 3.............rW
00000080: 33 DB 8B 0E 11 00 8B FB - 51 B9 0B 00 BE DC 01 F3 3.......Q.......
00000090: A6 59 74 05 83 C3 20 E2 - ED E3 37 26 8B 57 1A 52 .Yt... ...7&.W.R
000000A0: B8 01 00 68 00 20 07 33 - DB 0E E8 48 00 72 28 5B ...h. .3...H.r([
000000B0: 8D 36 0B 00 8D 3E 0B 02 - 1E 8F 45 02 C7 05 F5 00 .6...>....E.....
000000C0: 1E 8F 45 06 C7 45 04 0E - 01 8A 16 24 00 EA 03 00 ..E..E.....$....
000000D0: 00 20 BE 86 01 EB 03 BE - A2 01 E8 09 00 BE C1 01 . ..............
000000E0: E8 03 00 FB EB FE AC 0A - C0 74 09 B4 0E BB 07 00 .........t......
000000F0: CD 10 EB F2 C3 50 4A 4A - A0 0D 00 32 E4 F7 E2 03 .....PJJ...2....
00000100: 06 08 02 83 D2 00 A3 13 - 02 89 16 15 02 58 A2 07 .............X..
00000110: 02 A1 13 02 8B 16 15 02 - 03 06 1C 00 13 16 1E 00 ................
00000120: F7 36 18 00 FE C2 88 16 - 06 02 33 D2 F7 36 1A 00 .6........3..6..
00000130: 88 16 25 00 A3 04 02 A1 - 18 00 2A 06 06 02 40 3A ..%.......*...@:
00000140: 06 07 02 76 05 A0 07 02 - 32 E4 50 B4 02 8B 0E 04 ...v....2.P.....
00000150: 02 C0 E5 06 0A 2E 06 02 - 86 E9 8B 16 24 00 CD 13 ............$...
00000160: 0F 83 05 00 83 C4 02 F9 - CB 58 28 06 07 02 76 11 .........X(...v.
00000170: 01 06 13 02 83 16 15 02 - 00 F7 26 0B 00 03 D8 EB ..........&.....
00000180: 90 A2 07 02 F8 CB 42 4F - 4F 54 3A 20 43 6F 75 6C ......BOOT: Coul
00000190: 64 6E 27 74 20 66 69 6E - 64 20 4E 54 4C 44 52 0D dn't find NTLDR.
000001A0: 0A 00 42 4F 4F 54 3A 20 - 49 2F 4F 20 65 72 72 6F ..BOOT: I/O erro
000001B0: 72 20 72 65 61 64 69 6E - 67 20 64 69 73 6B 0D 0A r reading disk..
000001C0: 00 50 6C 65 61 73 65 20 - 69 6E 73 65 72 74 20 61 .Please insert a
000001D0: 6E 6F 74 68 65 72 20 64 - 69 73 6B 00 4E 54 4C 44 nother disk.NTLD
000001E0: 52 20 20 20 20 20 20 20 - 00 00 00 00 00 00 00 00 R ........
000001F0: 00 00 00 00 00 00 00 00 - 00 00 00 00 00 00 55 AA ..............U.
```

Tables 27.11 and 27.12 illustrate the layout of the BPB and the extended BPB for FAT16 volumes. The sample values correspond to the data in this example.

Table 27.11 BPB Fields for FAT16 Volumes

Byte Offset	Field Length	Sample Value	Field Name and Definition
0x0B	2 bytes	0x0002	**Bytes Per Sector**. The size of a hardware sector. Valid decimal values for this field are 512, 1024, 2048, and 4096. For most disks used in the United States, the value of this field is 512.

0x0D	1 byte	0x40	**Sectors Per Cluster**. The number of sectors in a cluster. Because FAT16 can track only a limited number of clusters (up to 65,524), FAT16 supports large volumes by increasing the number of sectors per cluster. The default cluster size for a volume depends on the volume size. Valid decimal values for this field are 1, 2, 4, 8, 16, 32, 64, and 128.
0x0E	2 bytes	0x0100	**Reserved Sectors**. The number of sectors that precede the start of the first FAT, including the boot sector. The value of this field is typically 1.
0x10	1 byte	0x02	**Number of FATs**. The number of copies of the FAT on the volume. The value of this field is typically 2.
0x11	2 bytes	0x0002	**Root Entries**. The total number of 32-byte file and folder name entries that can be stored in the root folder of the volume. On a typical hard disk, the value of this field is 512. One entry can be used as a volume label, and files and folders with long names use multiple entries per file. The largest number of file and folder entries is typically 511; however, if long file names are used, entries usually run out before you reach that number.
0x13	2 bytes	0x0000	**Small Sectors**. The number of sectors on the volume represented in 16 bits (< 65,536). For volumes larger than 65,536 sectors, this field has a value of zero and the **Large Sectors** field is used instead.
0x15	1 byte	0xF8	**Media Descriptor**. Provides information about the media being used. A value of 0xF8 indicates a hard disk and 0xF0 indicates a high-density 3.5-inch floppy disk. Media descriptor entries are a legacy of MS-DOS FAT16 and FAT12 disks and are not used in Windows XP Professional.
0x16	2 bytes	0xFC00	**Sectors Per FAT**. The number of sectors occupied by each FAT on the volume. The computer uses this number and the number of FATs and reserved sectors to determine where the root directory begins. The computer can also determine where the user data area of the volume begins based on the number of entries in the root directory (512).
0x18	2 bytes	0x3F00	**Sectors Per Track**. Part of the apparent disk geometry used on a low-level formatted disk.
0x1A	2 bytes	0x4000	**Number of Heads**. Part of the apparent disk geometry used on a low-level formatted disk.
0x1C	4 bytes	0x3F000000	**Hidden Sectors**. The number of sectors on the volume before the boot sector. This value is used during the boot sequence to calculate the absolute offset to the root directory and data areas.
0x20	4 bytes	0x01F03E00	**Large Sectors**. If the value of the **Small Sectors** field is zero, this field contains the total number of sectors in the FAT16 volume. If the value of the **Small Sectors** field is not zero, the value of this field is zero.

Table 27.12 Extended BPB Fields for FAT16 Volumes

Byte Offset	Field Length	Sample Value	Field Name and Definition
0x24	1 byte	0x80	**Physical Drive Number**. Related to the BIOS physical drive number. Floppy drives are identified as 0x00 and physical hard disks are identified as 0x80, regardless of the number of physical disk drives. Typically, this value is set prior to issuing an INT 13h BIOS call to specify the device to access. This value is only relevant if the device is a boot device.
0x25	1 byte	0x00	**Reserved**. FAT16 volumes are always set to zero.

0x26	1 byte	0x29	**Extended Boot Signature**. A field that must have the value 0x28 or 0x29 to be recognized by Windows XP Professional.
0x27	4 bytes	0xA88B3652	**Volume Serial Number**. A random serial number that is created when a volume is formatted and that helps to distinguish between disks.
0x2B	11 bytes	NO NAME	**Volume Label**. A field that was once used to store the volume label. The volume label is now stored as a special file in the root directory.
0x36	8 bytes	FAT16	Not used by Windows XP Professional.

FAT32 Boot Sector

The FAT32 boot sector is structurally very similar to the FAT16 boot sector, but the FAT32 BPB contains additional fields. The FAT32 extended BPB uses the same fields as FAT16, but the offset addresses of these fields within the boot sector are different than those found in FAT16 boot sectors. Volumes formatted in FAT32 are not readable by operating systems that are incompatible with FAT32.

Table 27.13 describes the boot sector of a volume formatted with the FAT32 file system.

Table 27.13 Boot Sector Sections on a FAT32 Volume

Byte Offset	Field Length	Field Name
0x00	3 bytes	Jump instruction
0x03	8 bytes	OEM ID
0x0B	53 bytes	BPB
0x40	26 bytes	Extended BPB
0x5A	420 bytes	Bootstrap code
0x01FE	2 bytes	End of sector marker

The following example illustrates a hexadecimal printout of the boot sector on a FAT32 volume. The printout is formatted in three sections:

- Bytes 0x00– 0x0A are the jump instruction and the OEM ID (shown in bold print).
- Bytes 0x0B– 0x59 are the BPB and the extended BPB.
- The remaining section is the bootstrap code and the end of sector marker (shown in bold print).

```
Physical Sector: Cyl 878, Side 0, Sector 1
00000000: EB 58 90 4D 53 44 4F 53 - 35 2E 30 00 02 10 24 00 .X.MSDOS5.0...$.
00000010: 02 00 00 00 00 F8 00 00 - 3F 00 FF 00 3F 00 00 00 ........?...?...
00000020: 1D 91 11 01 2A 22 00 00 - 00 00 00 00 02 00 00 00 ....*"..........
00000030: 01 00 06 00 00 00 00 00 - 00 00 00 00 00 00 00 00 ................
00000040: 80 00 29 F1 9E 5E 5E 4E - 4F 20 4E 41 4D 45 20 20 ..)..^^NO NAME
00000050: 20 20 46 41 54 33 32 20 - 20 20 33 C9 8E D1 BC F4 FAT32 3.....
00000060: 7B 8E C1 8E D9 BD 00 7C - 88 4E 02 8A 56 40 B4 08 {......|.N..V@..
00000070: CD 13 73 05 B9 FF FF 8A - F1 66 0F B6 C6 40 66 0F ..s......f...@f.
00000080: B6 D1 80 E2 3F F7 E2 86 - CD C0 ED 06 41 66 0F B7 ....?.......Af..
00000090: C9 66 F7 E1 66 89 46 F8 - 83 7E 16 00 75 38 83 7E .f..f.F..~..u8.~
000000A0: 2A 00 77 32 66 8B 46 1C - 66 83 C0 0C BB 00 80 B9 *.w2f.F.f.......
000000B0: 01 00 E8 2B 00 E9 48 03 - A0 FA 7D B4 7D 8B F0 AC ...+..H...}.}...
000000C0: 84 C0 74 17 3C FF 74 09 - B4 0E BB 07 00 CD 10 EB ..t.<.t.........
000000D0: EE A0 FB 7D EB E5 A0 F9 - 7D EB E0 98 CD 16 CD 19 ...}....}.......
000000E0: 66 60 66 3B 46 F8 0F 82 - 4A 00 66 6A 00 66 50 06 f`f;F...J.fj.fP.
000000F0: 53 66 68 10 00 01 00 80 - 7E 02 00 0F 85 20 00 B4 Sfh.....~.... ..
00000100: 41 BB AA 55 8A 56 40 CD - 13 0F 82 1C 00 81 FB 55 A..U.V@........U
00000110: AA 0F 85 14 00 F6 C1 01 - 0F 84 0D 00 FE 46 02 B4 .............F..
00000120: 42 8A 56 40 8B F4 CD 13 - B0 F9 66 58 66 58 66 58 B.V@......fXfXfX
00000130: 66 58 EB 2A 66 33 D2 66 - 0F B7 4E 18 66 F7 F1 FE fX.*f3.f..N.f...
00000140: C2 8A CA 66 8B D0 66 C1 - EA 10 F7 76 1A 86 D6 8A ...f..f....v....
00000150: 56 40 8A E8 C0 E4 06 0A - CC B8 01 02 CD 13 66 61 V@............fa
00000160: 0F 82 54 FF 81 C3 00 02 - 66 40 49 0F 85 71 FF C3 ..T.....f@I..q..
00000170: 4E 54 4C 44 52 20 20 20 - 20 20 20 00 00 00 00 00 NTLDR .....
00000180: 00 00 00 00 00 00 00 00 - 00 00 00 00 00 00 00 00 ................
00000190: 00 00 00 00 00 00 00 00 - 00 00 00 00 00 00 00 00 ................
000001A0: 00 00 00 00 00 00 00 00 - 00 00 00 00 0D 0A 43 61 ..............Ca
```

```
000001B0: 6E 6E 6F 74 20 73 74 61 - 72 74 2E 20 20 52 65 6D nnot start. Rem
000001C0: 6F 76 65 20 6D 65 64 69 - 61 2E FF 0D 0A 44 69 73 ove media....Dis
000001D0: 6B 20 65 72 72 6F 72 FF - 0D 0A 50 72 65 73 73 20 k error...Press
000001E0: 61 6E 79 20 6B 65 79 20 - 74 6F 20 72 65 73 74 61 any key to resta
000001F0: 72 74 0D 0A 00 00 00 00 - 00 AC CB D8 00 00 55 AA rt.............U.
```

Tables 27.14 and 27.15 illustrate the layout of the BPB and the extended BPB for FAT32 volumes. The sample values correspond to the data in this example.

Table 27.14 BPB Fields for FAT32 Volumes

Byte Offset	Field Length	Sample Value	Field Name and Definition
0x0B	2 bytes	0x0002	**Bytes Per Sector**. The size of a hardware sector. Valid decimal values for this field are 512, 1024, 2048, and 4096. For most disks used in the United States, the value of this field is 512.
0x0D	1 byte	0x10	**Sectors Per Cluster**. The number of sectors in a cluster. The default cluster size for a volume depends on the volume size. Valid decimal values for this field are 1, 2, 4, 8, 16, 32, 64, and 128. The Windows XP Professional implementation of FAT32 allows for creation of volumes up to a maximum of 32 GB. However, larger volumes created by other operating systems (Windows 95 OSR2 and later) are accessible in Windows XP Professional.
0x0E	2 bytes	0x2400	**Reserved Sectors**. The number of sectors that precede the start of the first FAT, including the boot sector.
0x10	1 byte	0x02	**Number of FATs**. The number of copies of the FAT on the volume. The value of this field is always 2.
0x11	2 bytes	0x0000	**Root Entries (FAT12/FAT16 only)**. For FAT32 volumes, this field must be set to zero.
0x13	2 bytes	0x0000	**Small Sectors (FAT12/FAT16 only)**. For FAT32 volumes, this field must be set to zero.
0x15	1 byte	0xF8	**Media Descriptor**. Provides information about the media being used. A value of 0xF8 indicates a hard disk and 0xF0 indicates a high-density 3.5-inch floppy disk. Media descriptor entries are a legacy of MS-DOS FAT16 disks and are not used in Windows XP Professional.
0x16	2 bytes	0x0000	**Sectors Per FAT (FAT12/FAT16 only)**. For FAT32 volumes, this field must be set to zero.
0x18	2 bytes	0x3F00	**Sectors Per Track**. Contains the "sectors per track" geometry value for disks that use INT 13h. The volume is broken down into tracks by multiple heads and cylinders.
0x1A	2 bytes	0xFF00	**Number of Heads**. Contains the "count of heads" geometry value for disks that use INT 13h. For example, on a 1.44-MB, 3.5-inch floppy disk this value is 2.
0x1C	4 bytes	0x3F000000	**Hidden Sectors**. The number of sectors on the volume before the boot sector. This value is used during the boot sequence to calculate the absolute offset to the root directory and data areas. This field is generally only relevant for media that are visible on interrupt 13h. It must always be zero on media that are not partitioned.
0x20	4 bytes	0x1D911101	**Large Sectors**. Contains the total number of sectors in the FAT32 volume.

0x24	4 bytes	0x2A220000	**Sectors Per FAT (FAT32 only)**. The number of sectors occupied by each FAT on the volume. The computer uses this number and the number of FATs and reserved sectors (described in this table) to determine where the root directory begins. The computer can also determine where the user data area of the volume begins based on the number of entries in the root directory.
0x28	2 bytes	0x0000	Not used by Windows XP Professional.
0x2A	2 bytes	0x0000	**File System Version (FAT32 only)**. The high byte is the major revision number; the low byte is the minor revision number. This field supports the ability to extend the FAT32 media type in the future with concern for old FAT32 drivers mounting the volume. Both bytes are zero in Windows XP Professional, Windows 2000, and Windows Me and earlier.
0x2C	4 bytes	0x02000000	**Root Cluster Number (FAT32 only)**. The cluster number of the first cluster of the root directory. This value is typically, but not always, 2.
0x30	2 bytes	0x0100	**File System Information Sector Number (FAT32 only)**. The sector number of the File System Information (FSINFO) structure in the reserved area of the FAT32 volume. The value is typically 1. A copy of the FSINFO structure is kept in the Backup Boot Sector, but it is not kept up-to-date.
0x32	2 bytes	0x0600	**Backup Boot Sector (FAT32 only)**. A value other than zero specifies the sector number in the reserved area of the volume where a copy of the boot sector is stored. The value of this field is typically 6. No other value is recommended.
0x34	12 bytes	0x000000000000000000000000	**Reserved (FAT32 only)**. Reserved space for future expansion. The value of this field must always be zero.

Table 27.15 Extended BPB Fields for FAT32 Volumes

Byte Offset	Field Length	Sample Value	Field Name and Definition
0x40	1 byte	0x80	**Physical Drive Number**. Related to the BIOS physical drive number. Floppy drives are identified as 0x00 and physical hard disks are identified as 0x80, regardless of the number of physical disk drives. Typically, this value is set prior to issuing an INT 13h BIOS call to specify the device to access. This value is only relevant if the device is a boot device.
0x41	1 byte	0x00	**Reserved**. FAT32 volumes are always set to zero.
0x42	1 byte	0x29	**Extended Boot Signature**. A field that must have the value 0x28 or 0x29 to be recognized by Windows XP Professional.
0x43	4 bytes	0xF19E5E5E	**Volume Serial Number**. A random serial number that is created when a volume is formatted and that helps to distinguish between disks.
0x47	11 bytes	NO NAME	**Volume Label**. A field that was once used to store the volume label. The volume label is now stored as a special file in the root directory.
0x52	8 bytes	FAT32	**System ID**. A text field with a value of FAT32.

NTFS Boot Sector

Table 27.16 describes the boot sector of a volume that is formatted with NTFS. When you format an NTFS volume, the format program allocates the first 16 sectors for the boot sector and the bootstrap code.

Table 27.16 Boot Sector Sections on an NTFS Volume

Byte Offset	Field Length	Field Name
0x00	3 bytes	Jump instruction
0x03	8 bytes	OEM ID
0x0B	25 bytes	BPB
0x24	48 bytes	Extended BPB
0x54	426 bytes	Bootstrap code
0x01FE	2 bytes	End of sector marker

On NTFS volumes, the data fields that follow the BPB form an extended BPB. The data in these fields enables Ntldr to find the master file table (MFT) during startup. On NTFS volumes, the MFT is not located in a predefined sector, as on FAT16 and FAT32 volumes. For this reason, NTFS can move the MFT if there is a bad sector in the current location of the MFT. However, if the data is corrupted, the MFT cannot be located, and Windows XP Professional assumes that the volume has not been formatted.

The following example illustrates the boot sector of an NTFS volume that is formatted by using Windows XP Professional. The printout is formatted in three sections:

- Bytes 0x00– 0x0A are the jump instruction and the OEM ID (shown in bold print).
- Bytes 0x0B–0x53 are the BPB and the extended BPB.
- The remaining code is the bootstrap code and the end of sector marker (shown in bold print).

```
Physical Sector: Cyl 0, Side 1, Sector 1
00000000: EB 52 90 4E 54 46 53 20 - 20 20 20 00 02 08 00 00 .R.NTFS .....
00000010: 00 00 00 00 00 F8 00 00 - 3F 00 FF 00 3F 00 00 00 ........?...?...
00000020: 00 00 00 00 80 00 80 00 - 1C 91 11 01 00 00 00 00 ................
00000030: 00 00 04 00 00 00 00 00 - 11 19 11 00 00 00 00 00 ................
00000040: F6 00 00 00 01 00 00 00 - 3A B2 7B 82 CD 7B 82 14 ........:.{..{..
00000050: 00 00 00 00 FA 33 C0 8E - D0 BC 00 7C FB B8 C0 07 .....3.....|....
00000060: 8E D8 E8 16 00 B8 00 0D - 8E C0 33 DB C6 06 0E 00 ..........3.....
00000070: 10 E8 53 00 68 00 0D 68 - 6A 02 CB 8A 16 24 00 B4 ..S.h..hj....$..
00000080: 08 CD 13 73 05 B9 FF FF - 8A F1 66 0F B6 C6 40 66 ...s......f...@f
00000090: 0F B6 D1 80 E2 3F F7 E2 - 86 CD C0 ED 06 41 66 0F .....?.......Af.
000000A0: B7 C9 66 F7 E1 66 A3 20 - 00 C3 B4 41 BB AA 55 8A ..f..f. ...A..U.
000000B0: 16 24 00 CD 13 72 0F 81 - FB 55 AA 75 09 F6 C1 01 .$...r...U.u....
000000C0: 74 04 FE 06 14 00 C3 66 - 60 1E 06 66 A1 10 00 66 t......f`..f...f
000000D0: 03 06 1C 00 66 3B 06 20 - 00 0F 82 3A 00 1E 66 6A ....f;. ...:..fj
000000E0: 00 66 50 06 53 66 68 10 - 00 01 00 80 3E 14 00 00 .fP.Sfh.....>...
000000F0: 0F 85 0C 00 E8 B3 FF 80 - 3E 14 00 00 0F 84 61 00 ........>.....a.
00000100: B4 42 8A 16 24 00 16 1F - 8B F4 CD 13 66 58 5B 07 .B..$.......fX[.
00000110: 66 58 66 58 1F EB 2D 66 - 33 D2 66 0F B7 0E 18 00 fXfX.-f3.f.....
00000120: 66 F7 F1 FE C2 8A CA 66 - 8B D0 66 C1 EA 10 F7 36 f......f..f....6
00000130: 1A 00 86 D6 8A 16 24 00 - 8A E8 C0 E4 06 0A CC B8 ......$.........
00000140: 01 02 CD 13 0F 82 19 00 - 8C C0 05 20 00 8E C0 66 ........... ...f
00000150: FF 06 10 00 FF 0E 0E 00 - 0F 85 6F FF 07 1F 66 61 ..........o..fa
00000160: C3 A0 F8 01 E8 09 00 A0 - FB 01 E8 03 00 FB EB FE ................
00000170: B4 01 8B F0 AC 3C 00 74 - 09 B4 0E BB 07 00 CD 10 .....<.t........
00000180: EB F2 C3 0D 0A 41 20 64 - 69 73 6B 20 72 65 61 64 .....A disk read
00000190: 20 65 72 72 6F 72 20 6F - 63 63 75 72 72 65 64 00 error occurred.
000001A0: 0D 0A 4E 54 4C 44 52 20 - 69 73 20 6D 69 73 73 69 ..NTLDR is missi
000001B0: 6E 67 00 0D 0A 4E 54 4C - 44 52 20 69 73 20 63 6F ng...NTLDR is co
000001C0: 6D 70 72 65 73 73 65 64 - 00 0D 0A 50 72 65 73 73 mpressed...Press
000001D0: 20 43 74 72 6C 2B 41 6C - 74 2B 44 65 6C 20 74 6F Ctrl+Alt+Del to
000001E0: 20 72 65 73 74 61 72 74 - 0D 0A 00 00 00 00 00 00 restart........
000001F0: 00 00 00 00 00 00 00 00 - 83 A0 B3 C9 00 00 55 AA ..............U.
```

Table 27.17 describes the fields in the BPB and the extended BPB on NTFS volumes. The fields starting at 0x0B, 0x0D, 0x15, 0x18, 0x1A, and 0x1C match those on FAT16 and FAT32 volumes. The sample values correspond to the data in this example.

Table 27.17 BPB and Extended BPB Fields on NTFS Volumes

Byte Offset	Field Length	Sample Value	Field Name and Definition
0x0B	2 bytes	0x0002	**Bytes Per Sector**. The size of a hardware sector. For most disks used in the United States, the value of this field is 512.
0x0D	1 byte	0x08	**Sectors Per Cluster**. The number of sectors in a cluster.
0x0E	2 bytes	0x0000	**Reserved Sectors**. Always 0 because NTFS places the boot sector at the beginning of the partition. If the value is not 0, NTFS fails to mount the volume.
0x10	3 bytes	0x000000	Value must be 0 or NTFS fails to mount the volume.
0x13	2 bytes	0x0000	Value must be 0 or NTFS fails to mount the volume.
0x15	1 byte	0xF8	**Media Descriptor**. Provides information about the media being used. A value of 0xF8 indicates a hard disk and 0xF0 indicates a high-density 3.5-inch floppy disk. Media descriptor entries are a legacy of MS-DOS FAT16 disks and are not used in Windows XP Professional.
0x16	2 bytes	0x0000	Value must be 0 or NTFS fails to mount the volume.
0x18	2 bytes	0x3F00	Not used or checked by NTFS.
0x1A	2 bytes	0xFF00	Not used or checked by NTFS.
0x1C	4 bytes	0x3F000000	Not used or checked by NTFS.
0x20	4 bytes	0x00000000	The value must be 0 or NTFS fails to mount the volume.
0x24	4 bytes	0x80008000	Not used or checked by NTFS.
0x28	8 bytes	0x1C91110100000000	**Total Sectors**. The total number of sectors on the hard disk.
0x30	8 bytes	0x0000040000000000	**Logical Cluster Number for the File $MFT**. Identifies the location of the MFT by using its logical cluster number.
0x38	8 bytes	0x1119110000000000	**Logical Cluster Number for the File $MFTMirr**. Identifies the location of the mirrored copy of the MFT by using its logical cluster number.
0x40	1 byte	0xF6	**Clusters Per MFT Record**. The size of each record. NTFS creates a file record for each file and a folder record for each folder that is created on an NTFS volume. Files and folders smaller than this size are contained within the MFT. If this number is positive (up to 0x7F), then it represents clusters per MFT record. If the number is negative (0x80 to 0xFF), then the size of the file record is 2 raised to the absolute value of this number.
0x41	3 bytes	0x000000	Not used by NTFS.
0x44	1 byte	0x01	**Clusters Per Index Buffer**. The size of each index buffer, which is used to allocate space for directories. If this number is positive (up to 0x7F), then it represents clusters per MFT record. If the number is negative (0x80 to 0xFF), then the size of the file record is 2 raised to the

			absolute value of this number.
0x45	3 bytes	0x000000	Not used by NTFS.
0x48	8 bytes	0x3AB27B82CD7B8214	**Volume Serial Number**. The volume's serial number.
0x50	4 bytes	0x00000000	Not used by NTFS.

Protecting the Boot Sector

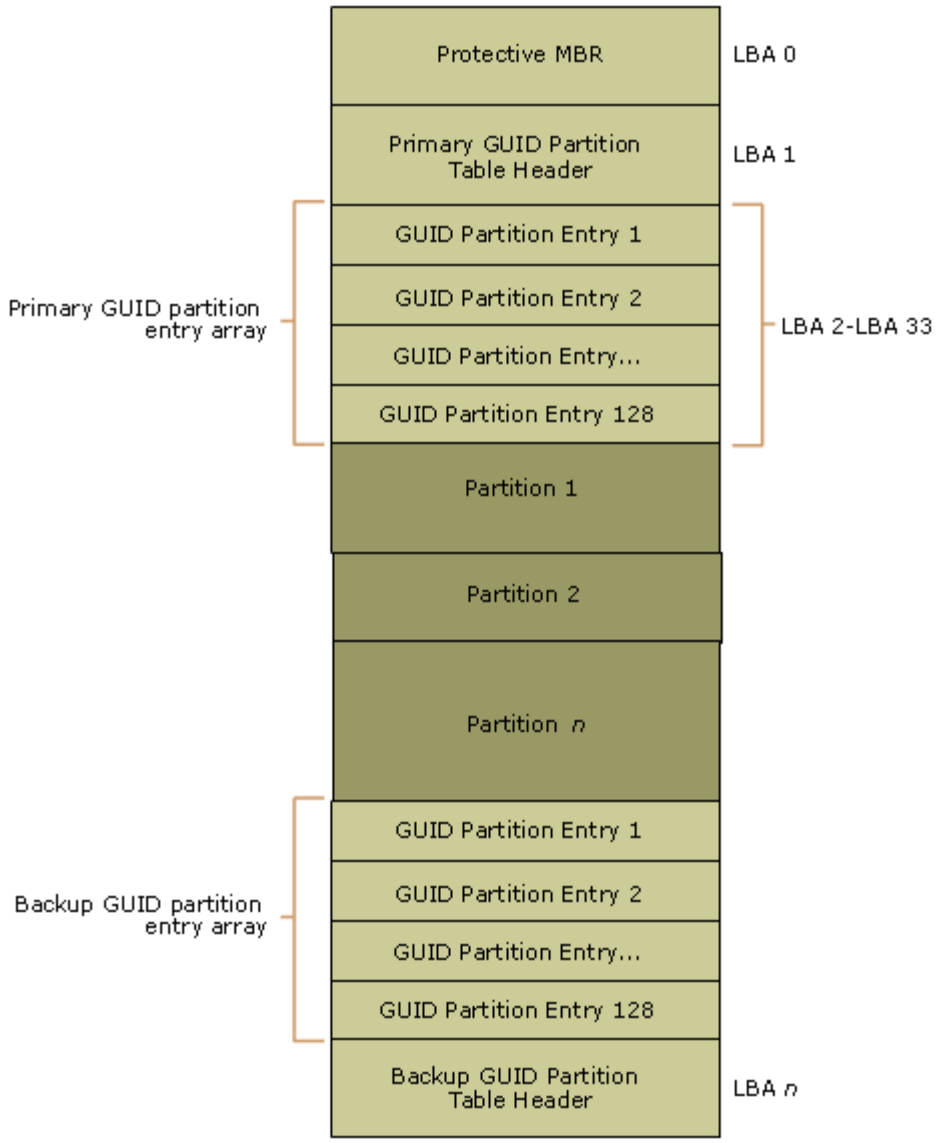
Because a functioning system relies on the boot sector to access a volume, it is highly recommended that you run Chkdsk when needed and back up all your data files regularly to protect against data loss if you lose access to a volume. For more information about Chkdsk, see "Chkdsk" earlier in this chapter. For more information about repairing boot sectors, see "Replacing the Boot Sector" earlier in this chapter.

Disk Sectors on GPT Disks

GPT uses primary and backup partition structures to provide redundancy. These structures are located at the beginning and the end of the disk. GPT identifies these structures by their logical block address (LBA) rather than by their relative sectors. Using this scheme, sectors on a disk are numbered from 0 to *n*-1, where *n* is the number of sectors on the disk.

As shown in Figure 27.8, the first structure on a GPT disk is the Protective MBR in LBA 0, followed by the primary GUID partition table (GPT) header in LBA 1. The GPT header is followed by the primary GUID partition entry array, which includes a partition entry for each partition on the disk.

Partitions on the disk are located between the primary and backup GUID partition entry arrays. The partitions must be placed within the first usable and last usable LBAs, as specified in the GPT partition header.



If your browser does not support inline frames, click here to view on a separate page.

Figure 27.8 Partition structures on a GPT disk

Protective MBR

The Extensible Firmware Interface (EFI) specification requires that LBA 0 be reserved for compatibility code and a Protective MBR. The Protective MBR has the same format as an existing MBR, and it contains one partition entry with a System ID value of 0xEE. This entry reserves the entire space of the disk, including the space used by the GPT header, as a single partition. The Protective MBR is included to prevent disk utilities that were designed for MBR disks from interpreting the disk as having available space and overwriting GPT partitions. The Protective MBR is ignored by EFI; no MBR code is executed.

The following example shows a partial printout of a Protective MBR.

```
000001B0: 00 00 00 00 00 00 00 00 - 04 06 04 06 00 00 00 00 ................
000001C0: 02 00 EE FF FF FF 01 00 - 00 00 FF FF FF FF 00 00 ................
000001D0: 00 00 00 00 00 00 00 00 - 00 00 00 00 00 00 00 00 ................
000001E0: 00 00 00 00 00 00 00 00 - 00 00 00 00 00 00 00 00 ................
000001F0: 00 00 00 00 00 00 00 00 - 00 00 00 00 00 00 55 AA ..............U.
```

Table 27.18 describes the fields in each entry in the Protective MBR.

Table 27.18 Protective MBR in GPT Disks

		Sample Value	

Byte Offset	Field Length	(1)	Field Name and Definition
0x01BE	1 byte	0x00	**Boot Indicator**. Must be set to 0x00 to indicate that this partition cannot be booted.
0x01BF	1 byte	0x00	**Starting Head**. Matches the Starting LBA of the single partition.
0x01C0	1 byte	0x02	**Starting Sector**. Matches the Starting LBA of the single partition.
0x01C1	1 byte	0x00	**Starting Cylinder**. Matches the Starting LBA of the GPT partition.
0x01C2	1 byte	0xEE	**System ID**. Must be 0xEE to specify that the single partition is a GPT partition. If you move a GPT disk to a computer running Windows 2000 with Service Pack 1 or greater or Windows XP Professional, the partition is displayed as a **GPT Protective Partition** and cannot be deleted.
0x01C3	1 byte	0xFF	**Ending Head**. Matches the Ending LBA of the single partition. If the Ending LBA is too large to be represented here, this field is set to 0xFF.
0x01C4	1 byte	0xFF	**Ending Sector**. Matches the Ending LBA of the single partition. If the Ending LBA is too large to be represented here, this field is set to 0xFF.
0x01C5	1 byte	0xFF	**Ending Cylinder**. Matches the Ending LBA of the single partition. If the Ending LBA is too large to be represented here, this field is set to 0xFF.
0x01C6	4 bytes	0x01000000	**Starting LBA**. Always set to 1. The Starting LBA begins at the GPT partition table header, which is located at LBA 1.
0x01CA	4 bytes	0xFFFFFFFF	**Size in LBA**. The size of the single partition. Must be set to 0xFFFFFFFF if this value is too large to be represented here.

(1) Numbers larger than one byte are stored in little endian format, or reverse-byte ordering. Little endian format is a method of storing a number so that the least significant byte appears first in the hexadecimal number notation.

GPT Partition Table Header

The GPT header defines the range of logical block addresses that are usable by partition entries. The GPT header also defines its location on the disk, its GUID, and a 32-bit cyclic redundancy check (CRC32) checksum that is used to verify the integrity of the GPT header.

GPT disks use a primary and a backup GUID partition table (GPT) header:

- The primary GPT header is located at LBA 1, directly after the Protective MBR.
- The backup GPT header is located in the last sector of the disk. No data follows the backup GPT header.

EFI verifies the integrity of the GPT headers by using a CRC32 checksum, which is a calculated value that is used to test data for the presence of errors. If the primary GPT header is corrupted, the system checks the backup GPT header checksum. If the backup checksum is valid, then the backup GPT header is used to restore the primary GPT header. This restoration process works in reverse if the primary GPT header is valid but the backup GPT header is corrupted. If both the primary and backup GPT headers are corrupted, then Windows XP 64-Bit Edition cannot access the disk.

Caution Do not use disk editing tools such as DiskProbe to make changes to GPT disks because any change that you make renders the checksums invalid, which might cause the disk to become inaccessible. To make changes to GPT disks, do either of the following:

- Use Diskpart.efi in the firmware environment.
- Use Diskpart.exe or Disk Management in Windows XP 64-Bit Edition.

The following example shows a partial printout of a GPT header.

```
00000000: 45 46 49 20 50 41 52 54 - 00 00 01 00 5C 00 00 00 EFI PART....\...
```

```
00000010: 27 6D 9F C9 00 00 00 00 - 01 00 00 00 00 00 00 00 'm..............
00000020: 37 C8 11 01 00 00 00 00 - 22 00 00 00 00 00 00 00 7.......".......
00000030: 17 C8 11 01 00 00 00 00 - 00 A2 DA 98 9F 79 C0 01 .............y..
00000040: A1 F4 04 62 2F D5 EC 6D - 02 00 00 00 00 00 00 00 ...b/..m........
00000050: 80 00 00 00 80 00 00 00 - 27 C3 F3 85 00 00 00 00 ........'.......
00000060: 00 00 00 00 00 00 00 00 - 00 00 00 00 00 00 00 00 ................
```

Table 27.19 describes the fields in the GPT header.

Table 27.19 GUID Partition Table Header

Byte Offset	Field Length	Sample Value	Field Name and Definition
0x00	8 bytes	0x4546492050415254	**Signature**. Used to identify all EFI-compatible GPT headers. The value must always be 0x4546492050415254.
0x08	4 bytes	0x00000100	**Revision**. The revision number of the EFI specification to which the GPT header complies. For version 1.0, the value is 0x00000100.
0x0C	4 bytes	0x5C000000	**Header Size**. The size, in bytes, of the GPT header. The size is always 0x5C000000 or 92 bytes. The remaining bytes in LBA 1 are reserved.
0x10	4 bytes	0x276D9FC9	**CRC32 Checksum**. Used to verify the integrity of the GPT header. The 32-bit cyclic redundancy check (CRC) algorithm is used to perform this calculation.
0x14	4 bytes	0x00000000	**Reserved**. Must be 0.
0x18	8 bytes	0x0100000000000000	**Primary LBA**. The LBA that contains the primary GPT header. The value is always equal to LBA 1.
0x20	8 bytes	0x37C8110100000000	**Backup LBA**. The LBA address of the backup GPT header. This value is always equal to the last LBA on the disk.
0x28	8 bytes	0x2200000000000000	**First Usable LBA**. The first usable LBA that can be contained in a GUID partition entry. In other words, the first partition begins at this LBA. In Windows XP 64-Bit Edition, this number is always LBA 34.
0x30	8 bytes	0x17C8110100000000	**Last Usable LBA**. The last usable LBA that can be contained in a GUID partition entry.
0x38	16 bytes	0x00A2DA989F79C001A 1F404622FD5EC6D	**Disk GUID**. A unique number that identifies the partition table header and the disk itself.
0x48	8 bytes	0x0200000000000000	**Partition Entry LBA**. The starting LBA of the GUID partition entry array. This number is always LBA 2.
0x50	4 bytes	0x80000000	**Number of Partition Entries**. The maximum number of partition entries that can be contained in the GUID partition entry array. In Windows XP 64-Bit Edition, this number is equal to 128.
0x54	4 bytes	0x80000000	**Size of Partition Entry**. The size, in bytes, of each partition entry in the GUID partition entry array. Each partition entry is 128 bytes.
0x58	4 bytes	0x27C3F385	**Partition Entry Array CRC32**. Used to verify the integrity of the GUID partition entry array. The 32-bit CRC algorithm is used to perform this calculation.
0x5C	420 bytes		**Reserved**. Must be 0.

GUID Partition Entry Array

Similar to the partition table on MBR disks, the GUID partition entry array contains partition entries that represent each partition on the disk. Windows XP 64-Bit Edition creates an array that is 16,384 bytes, so the first usable block must start at an LBA greater than or equal to 34. (LBA 0 contains the protective MBR; LBA 1

contains the GPT header; and LBAs 2 through 33 are used by the GUID partition entry array.)

Each GPT disk contains two GUID partition entry arrays:

- The primary GUID partition entry array is located after the GUID partition table header and ends before the first usable LBA.
- The backup GUID partition entry array is located after the last usable LBA and ends before the backup GUID partition table header.

A CRC32 checksum of the GUID partition entry array is stored in the GPT header. When a new partition is added, this checksum is updated in the primary and backup GUID partition entries, and then the GPT header size checksum is updated.

GUID Partition Entry

A GUID partition entry defines a single partition and is 128 bytes long. Because Windows XP 64-Bit Edition creates a GUID partition entry array that has 16,384 bytes, you can have a maximum of 128 partitions on a basic GPT disk.

Each GUID partition entry begins with a partition type GUID. The 16-byte partition type GUID, which is similar to a System ID in the partition table of an MBR disk, identifies the type of data that the partition contains and identifies how the partition is used. Windows XP 64-Bit Edition recognizes only the partition type GUIDs described in Table 27.20 and does not mount any other type of partition. However, original equipment manufacturers (OEMs) and independent software vendors (ISVs), as well as other operating systems might define additional partition type GUIDs.

For more information about the required partitions on GPT disks, see "Disk Management" in this book.

Table 27.20 Partition Type GUIDs

Partition Type	GUID Value
Unused entry	0x00000000000000000000000000000000
EFI System partition	0x28732AC11FF8D211BA4B00A0C93EC93B
Microsoft Reserved partition	0x16E3C9E35C0BB84D817DF92DF00215AE
Primary partition on a basic disk	0xA2A0D0EBE5B9334487C068B6B72699C7
LDM Metadata partition on a dynamic disk	0xAAC808588F7EE04285D2E1E90434CFB3
LDM Data partition on a dynamic disk	0xA0609BAF3114624FBC683311714A69AD

The following example illustrates a partial hexadecimal printout of the GUID partition entry array on a basic GPT disk. This printout shows three partition entries: an EFI System partition, a Microsoft Reserved partition, and a primary partition. The partition type GUIDs are bold and match the entries in Table 27.20.

```
00000000: 28 73 2A C1 1F F8 D2 11 - BA 4B 00 A0 C9 3E C9 3B (s*......K...>.;
00000010: C0 94 77 FC 43 86 C0 01 - 92 E0 3C 77 2E 43 AC 40 ..w.C.....<w.C.@
00000020: 3F 00 00 00 00 00 00 00 - CC 2F 03 00 00 00 00 00 ?......../......
00000030: 00 00 00 00 00 00 00 00 - 45 00 46 00 49 00 20 00 ........E.F.I. .
00000040: 73 00 79 00 73 00 74 00 - 65 00 6D 00 20 00 70 00 s.y.s.t.e.m. .p.
00000050: 61 00 72 00 74 00 69 00 - 74 00 69 00 6F 00 6E 00 a.r.t.i.t.i.o.n.
00000060: 00 00 00 00 00 00 00 00 - 00 00 00 00 00 00 00 00 ................
00000070: 00 00 00 00 00 00 00 00 - 00 00 00 00 00 00 00 00 ................
00000080: 16 E3 C9 E3 5C 0B B8 4D - 81 7D F9 2D F0 02 15 AE ....\..M.}.-....
00000090: 80 BC 80 FC 43 86 C0 01 - 50 7B 9E 5F 80 78 F5 31 ....C...P{._.x.1
000000A0: CD 2F 03 00 00 00 00 00 - D0 2A 04 00 00 00 00 00 ./.......*......
000000B0: 00 00 00 00 00 00 00 00 - 4D 00 69 00 63 00 72 00 ........M.i.c.r.
000000C0: 6F 00 73 00 6F 00 66 00 - 74 00 20 00 72 00 65 00 o.s.o.f.t. .r.e.
000000D0: 73 00 65 00 72 00 76 00 - 65 00 64 00 20 00 70 00 s.e.r.v.e.d. .p.
000000E0: 61 00 72 00 74 00 69 00 - 74 00 69 00 6F 00 6E 00 a.r.t.i.t.i.o.n.
000000F0: 00 00 00 00 00 00 00 00 - 00 00 00 00 00 00 00 00 ................
00000100: A2 A0 D0 EB E5 B9 33 44 - 87 C0 68 B6 B7 26 99 C7 ......3D..h..&..
00000110: C0 1B 0B 00 44 86 C0 01 - F1 B3 12 71 4F 75 88 21 ....D......qOu.!
00000120: D1 2A 04 00 00 00 00 00 - 4E 2F 81 00 00 00 00 00 .*......N/......
00000130: 00 00 00 00 00 00 00 00 - 42 00 61 00 73 00 69 00 ........B.a.s.i.
00000140: 63 00 20 00 64 00 61 00 - 74 00 61 00 20 00 70 00 c. .d.a.t.a. .p.
00000150: 61 00 72 00 74 00 69 00 - 74 00 69 00 6F 00 6E 00 a.r.t.i.t.i.o.n.
00000160: 00 00 00 00 00 00 00 00 - 00 00 00 00 00 00 00 00 ................
00000170: 00 00 00 00 00 00 00 00 - 00 00 00 00 00 00 00 00 ................
```

Table 27.21 illustrates the layout of a GUID partition entry. The sample values correspond to the EFI System

partition entry in the preceding example.

Table 27.21 GUID Partition Entry

Byte Offset	Field Length	Sample Value	Field Name and Definition
0x00	16 bytes	0x28732AC11FF8D211BA4B0 0A0C93EC93B	**Partition Type GUID**. Identifies the type of partition. The partition type GUID in this example identifies Microsoft Reserved partitions. See Table 27.20 for a description of the partition type GUIDs.
0x 10	16 bytes	0xC09477FC4386C00192E03C772E43AC40	**Unique Partition GUID**. A unique ID created for each partition entry.
0x 20	8 bytes	0x3F00000000000000	**Starting LBA**. The starting LBA of the partition that is defined by this partition entry.
0x 28	8 bytes	0xCC2F030000000000	**Ending LBA**. The ending LBA of the partition that is defined by this partition entry.
0x 30	8 bytes	0x0000000000000000	**Attribute Bits**. Describe how the partition is used. See Table 27.22 for a description of the attribute used by Windows XP 64-Bit Edition.
0x 38	72 bytes	EFI system partition	**Partition Name**. A 36-character Unicode string that can be used to name the partition.

The following example illustrates a partial hexadecimal printout of a GUID partition entry array on a dynamic GPT disk. Notice the differences between this example and the previous example of the basic GPT disk. The GUID partition entry array shows the Microsoft Reserved partition plus additional entries that appear only on dynamic GPT disks:

- The LDM Metadata partition is a 1-MB hidden partition that stores the dynamic disk database, which contains information about all dynamic disks and volumes installed on the computer.
- The LDM Data partition acts as a container for dynamic volumes. Individual dynamic volumes do not contain entries in the GUID partition entry array.

The partition type GUIDs are bold and match the entries in Table 27.20. For more information about partitions on GPT disks, see "Disk Management" in this book.

```
00000000: 16 E3 C9 E3 5C 0B B8 4D - 81 7D F9 2D F0 02 15 AE ....\..M.}.-....
00000010: 31 C3 97 A6 A4 9F 1D 44 - 85 61 15 49 4A E9 7C 24 1......D.a.IJ.|$
00000020: 22 08 00 00 00 00 00 00 - 21 00 01 00 00 00 00 00 ".......!.......
00000030: 00 00 00 00 00 00 00 00 - 4D 00 69 00 63 00 72 00 ........M.i.c.r.
00000040: 6F 00 73 00 6F 00 66 00 - 74 00 20 00 72 00 65 00 o.s.o.f.t. .r.e.
00000050: 73 00 65 00 72 00 76 00 - 65 00 64 00 20 00 70 00 s.e.r.v.e.d. .p.
00000060: 61 00 72 00 74 00 69 00 - 74 00 69 00 6F 00 6E 00 a.r.t.i.t.i.o.n.
00000070: 00 00 00 00 00 00 00 00 - 00 00 00 00 00 00 00 00 ................
00000080: AA C8 08 58 8F 7E E0 42 - 85 D2 E1 E9 04 34 CF B3 ...X.~.B.....4..
00000090: 66 F2 3F 3A 09 D9 EA 49 - B1 32 75 D5 98 04 3C 34 f.?:...I.2u...<4
000000A0: 22 00 00 00 00 00 00 00 - 21 08 00 00 00 00 00 00 ".......!.......
000000B0: 00 00 00 00 00 00 00 00 - 4C 00 44 00 4D 00 20 00 ........L.D.M. .
000000C0: 6D 00 65 00 74 00 61 00 - 64 00 61 00 74 00 61 00 m.e.t.a.d.a.t.a.
000000D0: 20 00 70 00 61 00 72 00 - 74 00 69 00 74 00 69 00 .p.a.r.t.i.t.i.
000000E0: 6F 00 6E 00 00 00 00 00 - 00 00 00 00 00 00 00 00 o.n.............
000000F0: 00 00 00 00 00 00 00 00 - 00 00 00 00 00 00 00 00 ................
00000100: A0 60 9B AF 31 14 62 4F - BC 68 33 11 71 4A 69 AD .`..1.bO.h3.qJi.
00000110: E2 33 A2 82 3A 5E D5 4C - AE 8E 4B EC 6B 76 4D ED .3..:^.L..K.kvM.
00000120: 22 00 01 00 00 00 00 00 - 09 77 11 01 00 00 00 00 ".........w......
00000130: 00 00 00 00 00 00 00 00 - 4C 00 44 00 4D 00 20 00 ........L.D.M. .
00000140: 64 00 61 00 74 00 61 00 - 20 00 70 00 61 00 72 00 d.a.t.a. .p.a.r.
00000150: 74 00 69 00 74 00 69 00 - 6F 00 6E 00 00 00 00 00 t.i.t.i.o.n.....
00000160: 00 00 00 00 00 00 00 00 - 00 00 00 00 00 00 00 00 ................
00000170: 00 00 00 00 00 00 00 00 - 00 00 00 00 00 00 00 00 ................
```

GUID Partition Entry Attributes

GUID partition entry attributes are descriptors for how a partition is used. The attributes are specified within a 64-bit value, so EFI supports up to 64 different attributes. Windows XP 64-Bit Edition uses two attributes as described in Table 27.22.

Table 27.22 GUID Partition Entry Attributes Used by Windows XP 64-Bit Edition

Bits	Description
Bit 0	Specifies that this partition is required for the platform to function.
Bits 48 to 63	Reserved by Microsoft for the following partition types: EFI System partitions, Microsoft Reserved partitions, primary partitions, LDM Metadata partitions, and LDM Data partitions. These partitions use the partition type GUIDs described in Table 27.20.

Boot Sectors on GPT Disks

Boot sectors on GPT disks are similar to boot sectors on MBR disks, except that EFI ignores all x86 code in the boot sector. Instead, EFI uses its own file system driver to read the BPB and then mount the volume.

Additional Resources

- "File Systems" in this book.
- "Disk Management" in this book.
- "Tools for Troubleshooting" in this book.
- "Troubleshooting Concepts and Strategies" in this book.
- *Inside Windows 2000 Server* by William Boswell, 2000, Indianapolis: New Riders Publishing.
- *Inside Windows 2000 Third Edition* by David A. Solomon and Mark E. Russinovich, 2000, Redmond: Microsoft Press.
- The Extensible Firmware Interface link on the Web Resources page at http://www.microsoft.com/windows/reskits/webresources .
