

CHAPTER 1

Disk Concepts and Troubleshooting



Understanding the organizational structure of information on hard disks, as well as being familiar with disk terminology, is key to diagnosing and troubleshooting disk problems. When troubleshooting system problems, you can refer to this chapter for detailed descriptions of the master boot record (MBR) and boot sector, two disk sectors that are critical to the startup process.

In This Chapter

Basic and Dynamic Disks	5
Disk Sectors Critical to Startup	10
Troubleshooting Disk Problems	34

Related Information in the Resource Kit

- For more information about startup, see “Startup Process” in this book.
- For more information about file systems, see “File Systems” in this book.
- For more information about preparing for and performing recovery, see “Repair, Recovery, and Restore” in this book.

Basic and Dynamic Disks

Microsoft® Windows® 2000 offers two types of disk storage configurations: basic disk and dynamic disk. Basic disk is similar to the disk structures used in Microsoft® Windows NT®. Dynamic disk is new to Windows 2000. By default, Windows 2000 initializes hard disks as basic disk.

The Disk Administrator tool found in Microsoft® Windows NT® version 4.0 and earlier has been replaced in Windows 2000 with the Disk Management snap-in for Microsoft Management Console (MMC). Disk Management supports both basic and dynamic disks. You can use the upgrade wizard in Disk Management to convert hard disks to dynamic disks.

You can use both basic and dynamic disks on the same computer system, and with any combination of file systems (file allocation table [FAT], including FAT16 and FAT32, and NTFS file system). However, all volumes on a physical disk must be either basic or dynamic.

You can upgrade from basic to dynamic storage at any time. Any changes made to your disk are immediately available in Windows 2000—you do not need to quit Disk Management to save them or restart your computer to implement them. However, if you upgrade the startup disk to dynamic, or if a volume or partition is in use on the disk that you are upgrading, the computer must be restarted for the upgrade to succeed.

Terms

To help you understand the differences between basic disk and dynamic disk, a set of definitions are provided.

Basic Disk

A basic disk is a physical disk that contains primary partitions and/or extended partitions with logical drives used by Windows 2000 and all versions of Windows NT. Basic disks can also contain volume, striped, mirror, or redundant array of independent disks (RAID) Level 5 (also known as striped with parity) sets that were created using Windows NT 4.0 or earlier. As long as a compatible file format is used, basic disks can be accessed by Microsoft® MS-DOS®, Microsoft® Windows® 95, Microsoft® Windows® 98, and all versions of Windows NT.

Since Windows 2000 automatically initializes disks as basic, you can troubleshoot partitions and volumes using the same methods as in Windows NT.

Note FAT32 is new in Windows 2000. Disk troubleshooting tools from Windows NT will likely not recognize FAT32 boot sectors and may cause problems with FAT32-formatted volumes. If FAT32 is used on your computer, be sure to use a disk troubleshooting tool designed for Windows 2000 that recognizes this file format.

New or empty disks can be initialized as either basic or dynamic after the hardware installation is complete. However, to set up a new fault-tolerant (FT) disk system, you must upgrade to dynamic disk.

Basic Volume

A basic volume is a volume on a basic disk. Basic volumes include primary partitions, logical drives within extended partitions, as well as volume, striped, mirror, or RAID-5 sets created using Windows NT 4.0 or earlier. You cannot create basic volumes on dynamic disks.

Note Creating new FT sets, such as mirrored and RAID-5 volumes, is only available on computers running Windows 2000 Server. The disk must be upgraded to dynamic disk before these volumes can be created. You can, however, use a computer running Windows 2000 Professional to create mirrored and RAID-5 volumes on a remote computer running Windows 2000 Server.

Dynamic Disk

A dynamic disk is a physical disk that has been upgraded by and is managed with Disk Management. Dynamic disks do not use partitions or logical drives. They can contain only dynamic volumes created by Disk Management. Only computers running Windows 2000 can access dynamic volumes.

Note Disks that have been upgraded from basic to dynamic disk still contain references to partitions in the partition table of the MBR. However, the MBR's reference to these partitions identifies the partition types as dynamic, indicating to Windows 2000 that the disk configuration data is now maintained in the disk management database at the end of the disk. Furthermore, any new changes made to the disk, such as deleting existing or creating additional volumes, are not recorded in the partition table.

Dynamic disks use dynamic volumes to subdivide physical disks into one or more drives enumerated by letters of the alphabet. Disk configuration data is contained in a disk management database stored in the last 1 megabyte (MB) of space at the end of the disk. Since dynamic disks do not use the traditional disk organization scheme of partitions and logical volumes, they cannot be directly accessed by MS-DOS, Windows 95, Windows 98, or any versions of Windows NT. Disk shares on dynamic disks, however, are available to computers running all of these operating systems.

Dynamic Volume

A dynamic volume is a logical volume that is created on a dynamic disk using Disk Management. Dynamic volume types include simple, spanned, striped, mirrored, and RAID-5, although only Windows 2000 Server supports the FT volume types (mirrored and RAID-5). You cannot create dynamic volumes on basic disks. Dynamic volumes are not supported on portable computers or removable media.

Note Dynamic volumes that were upgraded from basic disk partitions cannot be extended. This specifically includes the system volume, which contains hardware-specific files needed to start Windows 2000, and the boot volume, which contains the Windows 2000 system files required for startup. Only volumes created after the disk was upgraded to dynamic can be extended.

Partitions and Volumes

When you upgrade to dynamic disk, existing partitions and logical volumes are converted into dynamic volumes. Table 1.1 illustrates the translation of terms between basic and dynamic disk structures.

Table 1.1 Translation of Terms Between Basic and Dynamic Disk

Basic Disk Organization	Dynamic Disk Organization
Primary partition	Simple volume
System and boot partitions	System and boot volumes
Active partition	Active volume
Extended partition	Volume and unallocated space
Logical drive	Simple volume
Volume set	Spanned volume
Stripe set	Striped volume
Stripe set with parity	RAID-5 volume
Mirror set	Mirrored volume

Features of Basic Disk

You can use partitions on a basic disk just as you did with Microsoft® Windows NT® Server version 4.0, but you do not need to commit changes to save them or to restart your computer to make the changes effective. Changes made by Disk Management are implemented immediately. Unless you are making a change that affects existing files on the disk, the system executes your change without confirmation.

You can create up to four partitions in the free space on a physical hard disk; one of these can be an extended partition. You can use the free space in the extended partition to create one or more logical drives. You cannot use basic disk to create any kind of multiple volume sets or FT volumes.

You can perform the following tasks only on a basic disk:

- Create and delete primary and extended partitions.
- Create and delete logical drives within an extended partition.
- Format a partition and mark it as active.
- Delete volume, striped, mirror, or stripe sets with parity.
- Break a mirror from a mirror set.
- Repair failed legacy FT volumes such as mirror sets or stripe sets with parity.

Certain legacy functions are no longer available on basic disks because multiple-disk storage systems need to use dynamic disks. Disk Management supports legacy volumes that exceed a single partition on more than one hard disk, but it does not allow you to create new ones. For example, you cannot create volume, striped, mirror, or RAID-5 sets or extend volumes and volume sets on a basic disk.

While you cannot create new multiple disk sets on basic disks, you can delete them.

Be sure to back up all the information on the set before you delete it.

To establish a new spanned, striped, mirror, or RAID-5 set, first upgrade the disk to dynamic disk. To convert an existing volume, striped, mirror, or RAID-5 set, upgrade the physical disks on which the set resides to dynamic disk.

Features of Dynamic Disk

Disk Management is very flexible. The number of volumes that you can create on a physical hard disk is limited only by the amount of available free space on the disk. You can also create volumes that span two or more disks and that, if you are running Windows 2000 Server, are fault tolerant.

You can perform the following tasks only on a dynamic disk:

- Create and delete simple, spanned, striped, mirrored, and RAID-5 volumes.
- Extend a simple or spanned volume.
- Remove a mirror from a mirrored volume or break the mirrored volume into two volumes.
- Repair mirrored or RAID-5 volumes.
- Reactivate a missing or offline disk.

Dynamic disks are not supported on portable computers. If you are using a portable computer and right-click a disk in the graphical or list view in Disk Management, you will not see the option to upgrade the disk to dynamic.

Note On some older and non-Advanced Configuration and Power Interface (ACPI)-compliant portable computers, you might be able to upgrade to dynamic disk, but it is neither recommended nor supported.

The limitations of dynamic volumes occur in the following situations:

When installing Windows 2000 If a dynamic volume is created from unallocated space on a dynamic disk, you cannot install Windows 2000 on that volume.

The setup limitation occurs because Windows 2000 Setup uses BIOS calls that only recognize volumes listed in the partition table. Only basic disk partitions, as well as simple and mirrored volumes of dynamic disks that were upgraded from basic disk partitions, appear in the partition table. Dynamic disk does not use the partition table to manage its volumes, so new dynamic volumes are not registered in the partition table as they are created. Windows 2000 must be installed on a volume that is correctly represented in the partition table.

When extending a volume You can install Windows 2000 on a dynamic volume that was upgraded from a basic disk partition, but you cannot extend either the system volume or the boot volume. Neither can be part of a spanned volume, since Windows 2000 considers extended volumes to be the same as spanned volumes.

Windows 2000 cannot extend a dynamic volume that was a basic volume before the dynamic disk upgrade, and the system and boot volumes (which might be one and the same) are likely the same volumes that existed under basic disk. The upgraded simple volume must match the listing found in the partition table. Extending the dynamic volume changes its size, but the partition table's registration of that volume is not updated to reflect the change. The only dynamic volumes on which you can install Windows 2000 are simple and mirrored volumes, and since these volumes must be registered in the partition table, they must be upgraded from basic to dynamic.

Features Common to Both Basic and Dynamic Disks

You can perform the following tasks on both basic and dynamic disks:

- Check disk properties such as capacity, available free space, and current status.
- View volume and partition properties such as size, drive-letter assignment, label, type, and file system.
- Establish drive-letter assignments for disk volumes or partitions, and for CD-ROM devices.
- Establish disk sharing and security arrangements for a volume or partition.
- Upgrade a basic disk to dynamic or revert a dynamic disk to basic.

Disk Sectors Critical to Startup

The two sectors critical to starting your computer are the master boot record (MBR), which is always located at sector 1 of cylinder 0, head 0, the first sector of a hard disk, and 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.

Note The use of basic or dynamic disk does not affect where the MBR is located on disk and only minor differences exist between the two for how the partition table is configured. However, as the Disk Management database contains the information where dynamic volumes begin and end, the method of walking, or navigating, partition tables to find the start and end of partitions and logical volumes, as well as finding volume boot sectors, does not work on dynamic disks. Disk editing tools, such as DiskProbe and third-party tools, can walk the partitions as expected with basic disks. Also, many disk editor tools that work with Windows NT and NTFS are not currently compatible with FAT32 boot sectors and volumes.

Master Boot Record

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 2000 uses the disk signature as an index to store and retrieve information about the disk in the registry subkey:

```
HKEY_LOCAL_MACHINE\SYSTEM\MountedDevices
```

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 one of the following error messages:

- Invalid partition table
- Error loading operating system
- Missing operating system

Note There is no MBR on a floppy disk. The first sector on a floppy disk 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 “Damaged MBRs and Boot Sectors” later in this chapter.

Partition Table

The partition table, a 64-byte data structure 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)

Note Only basic disk makes use of the partition table in Windows 2000. Dynamic disk uses the Disk Management database located at the end of the disk for disk configuration information. The partition table is not updated when volumes are deleted or extended after the dynamic disk upgrade, or when new dynamic volumes are created.

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 00  .....
    
```

Table 1.2 describes the fields in each entry in the partition table. The sample values correspond to the first partition table entry shown in the preceding 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).

Table 1.2 Partition Table Fields

Byte Offset	Field Length	Sample Value	Field Name and Definition
0x01BE	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	BYTE	0x01	Starting Head.
0x01C0	6 bits	0x01 *	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, with a maximum value of 1023.
0x01C2	BYTE	0x07	System ID. Defines the volume type. See Table 1.3 for sample values.
0x01C3	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	DWORD	0x3F000000	Relative Sectors. The offset from the beginning of the disk to the beginning of the volume, counting by sectors.
0x01CA	DWORD	0x4BF57F00	Total Sectors. The total number of sectors in the volume.

A BYTE is 8 bits, a WORD is 16 bits, a DWORD is 32 bits, and a LONGLONG is 64 bits. 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.

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.

Boot Indicator Field

The first element of the partition table, the **Boot Indicator** field, indicates whether or not the volume is the active partition. Only one primary partition on the disk can have this field set.

It is possible to have different operating systems and different file systems on different volumes. By using disk configuration tools such as the Windows 2000-based Disk Management 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 and the FT characteristics of the volume. The **System ID** field also identifies an extended partition, if one is defined. Windows 2000 uses the **System ID** field to determine which file system device drivers to load during startup. Table 1.3 identifies the values for the **System ID** field.

Table 1.3 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
0x42	Dynamic disk volume
0x86	Legacy FT FAT16 disk *
0x87	Legacy FT NTFS disk *
0x8B	Legacy FT volume formatted with FAT32 *
0x8C	Legacy FT volume using BIOS INT 13h extensions formatted with FAT32 *

Partition types denoted with an asterisk (*) indicate that they are also used to designate non-FT configurations such as striped and spanned volumes.

When a mirrored or RAID-5 volume is created in Windows NT 4.0 or earlier, the high bit of the **System ID** field byte is set for each primary partition or logical drive that is a member of the volume. For example, a FAT16 primary partition or logical drive that is a member of a mirrored or RAID-5 volume, has a **System ID** value of 0x86. A FAT32 primary partition or logical drive has a **System ID** value of 0x8B, and an NTFS primary partition or logical drive has a **System ID** value of 0x87. Volumes that have the high bit set can only be directly accessed by Windows 2000 and Windows NT. Disk shares on FT disks, however, are also available to computers running MS-DOS, Windows 95, and Windows 98.

Note 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 1.3 using MS-DOS tools such as Fdisk. If you use a low-level disk editor, such as DiskProbe, you can read and write to any sector, including ones that are in NTFS volumes.

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 EBR of the first logical drive in the extended partition as shown in Figure 1.1.

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.

Note To have a written record of the starting and ending sectors of the partitions on your hard disk, as well as other useful disk configuration data, use the DiskMap tool. For more information about DiskMap, see the documentation provided on the *Windows 2000 Resource Kit* companion CD.

Figure 1.1 shows the MBR, partition table, and boot sectors on a disk with four partitions. The definitions of the fields in the partition table and the extended partition tables are the same.

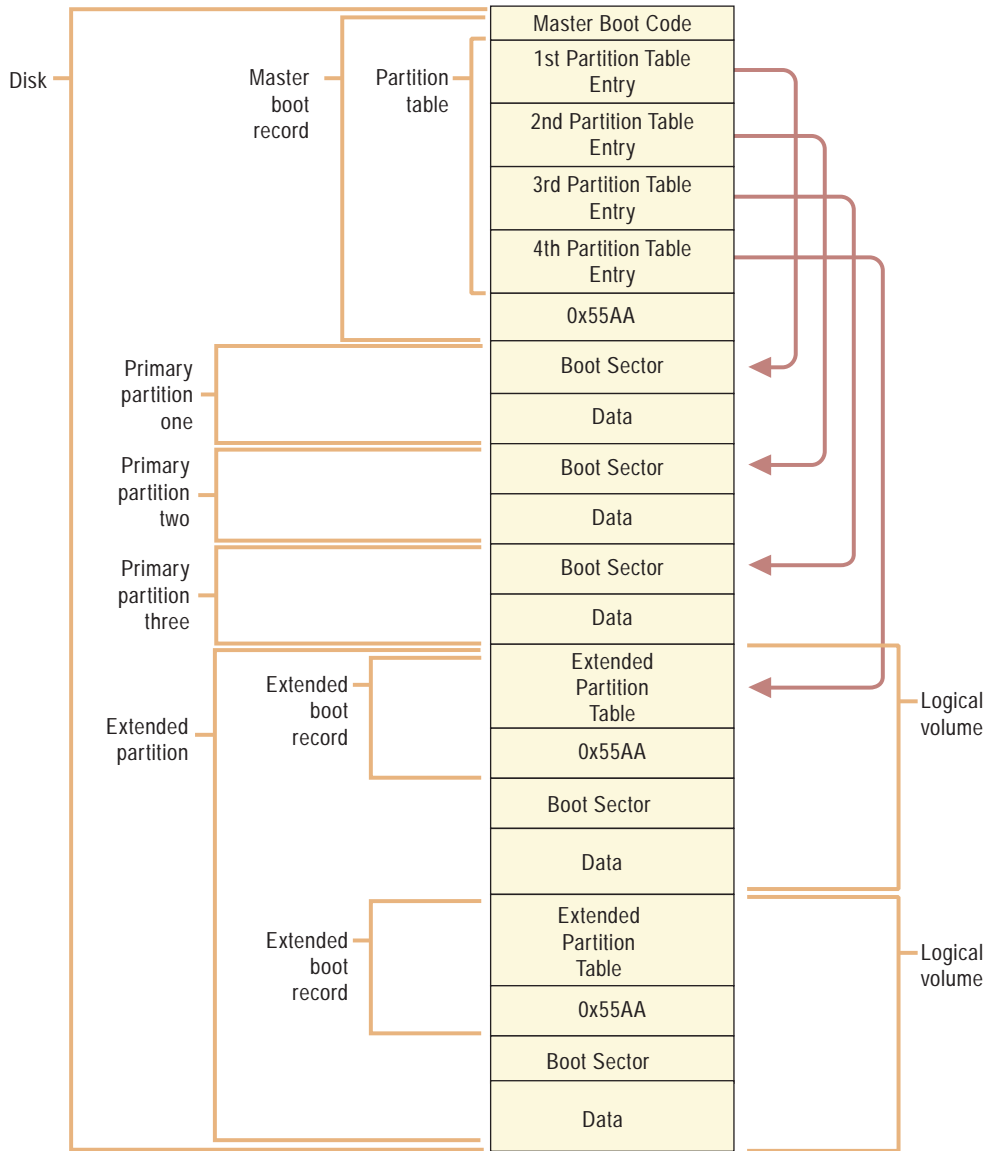


Figure 1.1 Detail of a Basic Disk with Four Partitions

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 0–1,023. The **Starting Head** and **Ending Head** fields are each one byte long, which limits the field range to 0–255. The **Starting Sector** and **Ending Sector** fields are each six bits long, which limits the range of these fields to 0–63. 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:

$512 \times 1024 \times 256 \times 63$ (or 512×2^{24}) = 8,455,716,864 bytes or 7.8 GB

The calculation results in a maximum capacity of slightly less than 8 gigabytes (GB). Before BIOS INT 13h extensions drive geometry translation (also known as logical block addressing, or LBA) were introduced, the active, primary partition could not exceed 7.8 GB, regardless of the file system used.

Important When using the standard 512-byte sector, the maximum cluster size that you can use for FAT16 volumes while running Windows 2000 is 64 kilobytes (KB). Therefore, the maximum size for a FAT16 volume is 4 GB.

If you use a multiple-boot configuration with Windows 95, Windows 98, or MS-DOS, FAT16 volumes must be limited to 2 GB to be accessed from these operating systems. In addition, a Macintosh computer that accesses volumes on a computer running Windows 2000 cannot access a FAT16 volume that is larger than 2 GB. If you try to use a FAT16 volume larger than 2 GB when running MS-DOS, Windows 95, or Windows 98, or try to access such a volume from a Macintosh computer, you might get a message that zero bytes are available.

The maximum FAT16 volume size that you can use on a computer depends on the disk geometry and the maximum values that fit in the partition table entry fields. Table 1.4 shows the typical FAT16 volume size when LBA is enabled or disabled. The number of cylinders in both cases is 1,024 (0–1,023). When a primary partition or logical drive extends beyond the 1,023rd cylinder, all fields described in this section contain the maximum values.

Table 1.4 FAT16 Volume Size When LBA Is Enabled or Disabled

Translation Mode	Number of Heads	Sectors per Track	Maximum Size for System or Boot Partition
Disabled	64	32	1 GB
Enabled	255	63	4 GB

Warning Do not change the LBA setting on any hard disk containing data. You can adversely affect the process in which the system translates the disk attributes for storing data and corrupt all the files and partitions on the physical disk. Refer to your computer owner's manual before modifying this BIOS setting.

To accommodate sizes larger than 7.8 GB, Windows 2000 ignores the values in the **Starting** and **Ending Sector** fields of the partition table in favor of the **Relative Sectors** and **Total Sectors** fields.

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 partitions containing up to 2^{32} sectors to be defined. 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).

This addressing scheme is only used in Windows 2000 with NTFS and FAT32.

Note In addition, the Format tool of Windows 2000 limits the maximum size of FAT32 volumes it can create to 32 GB. However, Windows 2000 can directly access larger FAT32 volumes created by Windows 95 OSR2 or Windows 98.

Windows 2000 uses the fields in the partition table entries to access all partitions. A partition that is formatted while Windows 2000 is running puts data into the **Starting** and **Ending CHS** fields to have compatibility with MS-DOS, Windows 95, and Windows 98, and to maintain compatibility with the BIOS INT 13h for startup.

Extended Boot Record

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.

Note This information applies only to disks configured with 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 zeroes. 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 1.2, 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.

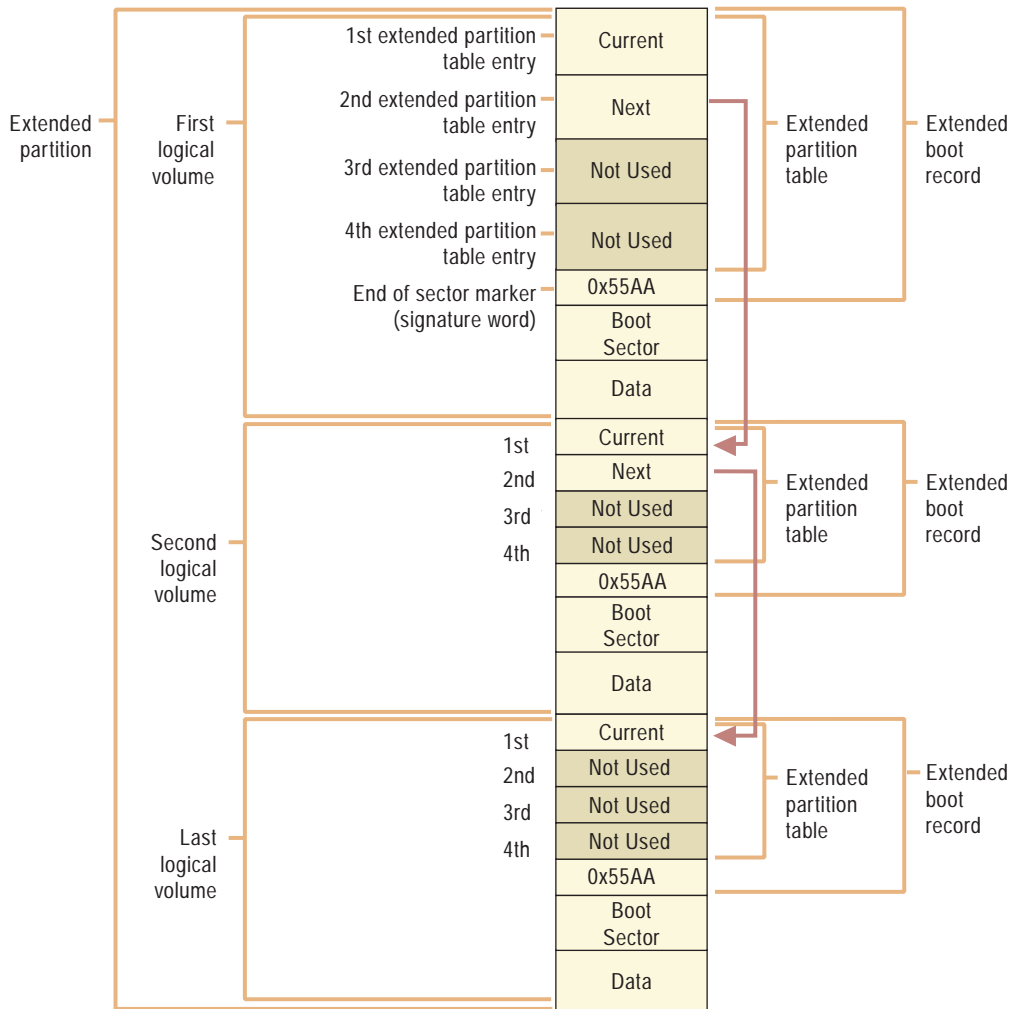


Figure 1.2 Detail of an Extended Partition

With the exception of the last logical drive on the extended partition, the format of the extended partition table, described in Table 1.5, 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 1.5 Contents of Extended Partition Table Entries

Extended Partition Table 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 further 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 1.2 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 as well as regularly back up all your data files to protect against losing access to a volume or an entire disk.

Boot Sector

The boot sector, located at sector 1 of each volume, is a critical disk structure for starting your computer. It contains executable code and 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 two-byte structure called a signature word or end of sector marker, which is always set to 0x55AA. On computers running Windows 2000, the boot sector on the active partition loads into memory and starts Ntldr, which loads the operating system.

The Windows 2000 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.

Note All Windows 2000 boot sectors contain these elements. However, the NTFS boot sector, the FAT16, and the FAT32 boot sectors are all formatted differently.

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.

The information in the BPB and the extended BPB is used by disk device drivers 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 searches for a boot device (typically a 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 2000, the executable boot code in the boot sector finds Ntldr, loads it into memory, and transfers execution to that file.

Note Windows 2000 cannot start up from a spanned, striped, or RAID-5 volume that is running dynamic disk. These disk structures cannot be registered into the MBR's partition table, so a system partition using these structures is not startable. Windows 2000 must be fully loaded into memory before they can be used.

If there is a floppy disk in drive A, the system BIOS loads the first sector (the boot sector) of the disk into memory. If the disk is startable—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 bootable, the executable boot code displays an error message such as:

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

Note This error will 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 for 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 “Damaged MBRs and Boot Sectors” later 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 2000 records “MSDOS5.0” in this field on FAT16 and FAT32 disks. On NTFS disks, Windows 2000 records “NTFS.”

Note You may also see the OEM ID “MSWIN4.0” on disks formatted by Windows 95 and “MSWIN4.1” on disks formatted by Windows 95 OSR2 and Windows 98. Windows 2000 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 1.6 describes the boot sector of a volume formatted with the FAT16 file system.

Table 1.6 Boot Sector Sections on a FAT16 Volume

Byte Offset	Field Length	Field Name
0x00	3 bytes	Jump Instruction
0x03	LONGLONG	OEM ID
0x0B	25 bytes	BPB
0x24	26 bytes	Extended BPB
0x3E	448 bytes	Bootstrap Code
0x01FE	WORD	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 00 - 00 00 00 00 00 00 00 R .....
000001F0: 00 00 00 00 00 00 00 - 00 00 00 00 00 00 55 AA .....U.
    
```

Tables 1.7 and 1.8 illustrate the layout of the BPB and the extended BPB for FAT16 volumes. The sample values correspond to the data in the preceding example.

Table 1.7 BPB Fields for FAT16 Volumes

Byte Offset	Field Length	Value	Field Name and Definition
0x0B	WORD	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	BYTE	0x40	Sectors Per Cluster. The number of sectors in a cluster. Because FAT16 can track only a limited number of clusters (up to 65,536), large volumes are supported 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. Values that lead to clusters larger than 32 KB (Bytes Per Sector * Sectors Per Cluster) can cause disk and software errors.
0x0E	WORD	0x0100	Reserved Sectors. The number of sectors preceding the start of the first FAT, including the boot sector. The value of this field is always 1.

(continued)

Table 1.7 BPB Fields for FAT16 Volumes (continued)

Byte Offset	Field Length	Value	Field Name and Definition
0x10	BYTE	0x02	Number of FATs. The number of copies of the FAT on the volume. The value of this field is always 2.
0x11	WORD	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 is always 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, but entries run out before you reach that number if long file names are used.
0x13	WORD	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	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 2000.
0x16	WORD	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 hidden 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	WORD	0x3F00	Sectors Per Track. Part of the apparent disk geometry used on a low-level formatted disk.
0x1A	WORD	0x4000	Number of Heads. Part of the apparent disk geometry used on a low-level formatted disk.
0x1C	DWORD	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	DWORD	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 1.8 Extended BPB Fields for FAT16 Volumes

Byte Offset	Field Length	Value	Field Name and Definition
0x24	BYTE	0x80	Physical Drive Number. Related to the BIOS physical drive number. Floppy disk 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. The value is only relevant if the device is a boot device.
0x25	BYTE	0x00	Reserved. FAT16 volumes are always set to zero.
0x26	BYTE	0x29	Extended Boot Signature. A field that must have the value 0x28 or 0x29 to be recognized by Windows 2000.
0x27	DWORD	0xA88B3652	Volume Serial Number. A random serial number created when formatting a disk, which helps to distinguish between disks.
0x2B	11 bytes	NO NAME	Volume Label. A field once used to store the volume label. The volume label is now stored as a special file in the root directory.
0x36	LONGLONG	FAT16	File System Type. A field with a value of either FAT, FAT12 or FAT16, depending on the disk format.

FAT32 Boot Sector

Table 1.9 describes the boot sector of a volume formatted with the FAT32 file system.

Note 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. Drives formatted in FAT32 are not readable by operating systems that are incompatible with FAT32.

Table 1.9 Boot Sector Sections on a FAT32 Volume

Byte Offset	Field Length	Field Name
0x00	3 bytes	Jump Instruction
0x03	LONGLONG	OEM ID
0x0B	53 bytes	BPB
0x40	26 bytes	Extended BPB
0x5A	420 bytes	Bootstrap Code
0x01FE	WORD	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 08 20 00 .X.MSDOS5.0...
00000010: 02 00 00 00 00 00 F8 00 00 - 3F 00 FF 00 EE 39 D7 00 .....?....9..
00000020: 7F 32 4E 00 83 13 00 00 - 00 00 00 00 02 00 00 00 □2N.....
00000030: 01 00 06 00 00 00 00 00 - 00 00 00 00 00 00 00 00 .....
00000040: 80 00 29 8B 93 6D 54 4E - 4F 20 4E 41 4D 45 20 20 ..).mTNO 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 0D 0A 4E 54 4C NTLDR ..NTL
00000180: 44 52 20 69 73 20 6D 69 - 73 73 69 6E 67 FF 0D 0A DR is missing...
00000190: 44 69 73 6B 20 65 72 72 - 6F 72 FF 0D 0A 50 72 65 Disk error...Pre
000001A0: 73 73 20 61 6E 79 20 6B - 65 79 20 74 6F 20 72 65 ss any key to re
000001B0: 73 74 61 72 74 0D 0A 00 - 00 00 00 00 00 00 00 00 start.....
000001C0: 00 00 00 00 00 00 00 00 - 00 00 00 00 00 00 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 7B 8E 9B 00 00 55 AA .....{.....U.
```

Tables 1.10 and 1.11 illustrate the layout of the BPB and the extended BPB for FAT32 volumes. The sample values correspond to the data in the preceding example.

Table 1.10 BPB Fields for FAT32 Volumes

Byte Offset	Field Length	Value	Field Name and Definition
0x0B	WORD	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	BYTE	0x08	Sectors Per Cluster. The number of sectors in a cluster. Because FAT32 can only track a finite number of clusters (up to 4,294,967,296), extremely large volumes are supported 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. The Windows 2000 implementation of FAT32 allows for the creation of volumes only up to a maximum of 32 GB. However, larger volumes created by other operating systems (Windows 95 OSR2 and later) are accessible in Windows 2000.
0x0E	WORD	0x0200	Reserved Sectors. The number of sectors preceding the start of the first FAT, including the boot sector. The decimal value of this field is typically 32.
0x10	BYTE	0x02	Number of FATs. The number of copies of the FAT on the volume. The value of this field is always 2.
0x11	WORD	0x0000	Root Entries (FAT12/FAT16 only). For FAT32 volumes, this field must be set to zero.
0x13	WORD	0x0000	Small Sectors (FAT12/FAT16 only). For FAT32 volumes, this field must be set to zero.
0x15	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 2000.
0x16	WORD	0x0000	Sectors Per FAT (FAT12/FAT16 only). For FAT32 volumes, this field must be set to zero.
0x18	WORD	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	WORD	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.

(continued)

Table 1.10 BPB Fields for FAT32 Volumes (continued)

Byte Offset	Field Length	Value	Field Name and Definition
0x1C	DWORD	0xEE39D700	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	DWORD	0x7F324E00	Large Sectors. Contains the total number of sectors in the FAT32 volume.
0x24	DWORD	0x83130000	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 hidden 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	WORD	0x0000	Extended Flags (FAT32 only). The value of the bits in this two-byte structure are: Bits 0–3: Number of the active FAT (starting count at 0, not 1). It is only valid if mirroring is disabled. Bits 4–6: Reserved. Bit 7: A value of 0 means the FAT is mirrored at run time into all FATs. A value of 1 means only one FAT is active (referenced in bits 0-3). Bits 8–15: Reserved.
0x2A	WORD	0x0000	File System Version (FAT32 only). The high byte is the major revision number, whereas 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. If the field is non-zero, back-level Windows versions will not mount the volume.
0x2C	DWORD	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	WORD	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.
0x34	WORD	0x0600	Backup Boot Sector (FAT32 only). A non-zero value indicates the sector number in the reserved area of the volume in which a copy of the boot sector is stored. The value of this field is typically 6. No other value is recommended.
0x36	12 bytes	0x000000000 00000000000 0000	Reserved (FAT32 only). Reserved space for future expansion. The value of this field should always be zero.

Table 1.11 Extended BPB Fields for FAT32 Volumes

Byte Offset	Field Length	Value	Field Name and Definition
0x40	BYTE	0x80	Physical Drive Number. Related to the BIOS physical drive number. Floppy disk 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. It is only relevant if the device is a boot device.
0x41	BYTE	0x00	Reserved. FAT32 volumes are always set to zero.
0x42	BYTE	0x29	Extended Boot Signature. A field that must have the value 0x28 or 0x29 to be recognized by Windows 2000.
0x43	DWORD	0xA88B3652	Volume Serial Number. A random serial number created when formatting a disk, which helps to distinguish between disks.
0x47	11 bytes	NO NAME	Volume Label. A field once used to store the volume label. The volume label is now stored as a special file in the root directory.
0x52	LONGLONG	FAT32	System ID. A text field with a value of FAT32.

NTFS Boot Sector

Table 1.12 describes the boot sector of a volume formatted with NTFS. The bootstrap code for an NTFS volume is longer than the 426 bytes, as shown in Table 1.12. When you format an NTFS volume, the format program allocates the first 16 sectors for the boot sector and the bootstrap code.

Table 1.12 Boot Sector Sections on an NTFS Volume

Byte Offset	Field Length	Field Name
0x00	3 bytes	Jump Instruction
0x03	LONGLONG	OEM ID
0x0B	25 bytes	BPB
0x24	48 bytes	Extended BPB
0x54	426 bytes	Bootstrap Code
0x01FE	WORD	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, the MFT can be moved if there is a bad sector in its normal location. However, if the data is corrupted, the MFT cannot be located, and Windows 2000 assumes that the volume has not been formatted.

The following example illustrates the boot sector of an NTFS volume formatted while running Windows 2000. 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 - 4A F5 7F 00 00 00 00 00 .....J.□.....
00000030: 04 00 00 00 00 00 00 00 - 54 FF 07 00 00 00 00 00 .....T.....
00000040: F6 00 00 00 01 00 00 00 - 14 A5 1B 74 C9 1B 74 1C .....t..t.
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 1.13 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 the preceding example.

Table 1.13 BPB and Extended BPB Fields on NTFS Volumes

Byte Offset	Field Length	Sample Value	Field Name
0x0B	WORD	0x0002	Bytes Per Sector
0x0D	BYTE	0x08	Sectors Per Cluster
0x0E	WORD	0x0000	Reserved Sectors
0x10	3 BYTES	0x000000	<i>always 0</i>
0x13	WORD	0x0000	<i>not used by NTFS</i>
0x15	BYTE	0xF8	Media Descriptor
0x16	WORD	0x0000	<i>always 0</i>
0x18	WORD	0x3F00	Sectors Per Track
0x1A	WORD	0xFF00	Number Of Heads
0x1C	DWORD	0x3F000000	Hidden Sectors
0x20	DWORD	0x00000000	<i>not used by NTFS</i>
0x24	DWORD	0x80008000	<i>not used by NTFS</i>
0x28	LONGLONG	0x4AF57F0000000000	Total Sectors
0x30	LONGLONG	0x0400000000000000	Logical Cluster Number for the file \$MFT
0x38	LONGLONG	0x54FF070000000000	Logical Cluster Number for the file \$MFTMirr
0x40	DWORD	0xF6000000	Clusters Per File Record Segment
0x44	DWORD	0x01000000	Clusters Per Index Block
0x48	LONGLONG	0x14A51B74C91B741C	Volume Serial Number
0x50	DWORD	0x00000000	Checksum

Protecting the Boot Sector

Because a normally functioning system relies on the boot sector to access a volume, it is highly recommended that you run disk scanning tools such as Chkdsk regularly, as well as back up all of your data files to protect against data loss if you lose access to a volume.

Troubleshooting Disk Problems

There are various causes of disk problems and means of recovering from them. The following are tools that you can use to troubleshoot disk problems:

- DiskProbe can be used to examine and change information on individual disk sectors.
- DiskMap can be used to display the layout of partitions and logical volumes on your disk.

Neither of these tools is designed for use with dynamic disks because they cannot read the dynamic Disk Management database. 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, so it might be impossible to find the sector you want to view or edit. Therefore it is generally recommended that these tools only be used on basic disks.

DiskProbe is part of the Support Tools collection and can be installed from the Support\Tools folder of the Windows 2000 product CD. For more information about using DiskProbe, see the document Dskprtrb.doc in the folder Program Files\Support Tools.

DiskMap is a tool included on the *Windows 2000 Resource Kit* companion CD and is installed with the rest of the Resource Kit tools. For more information about DiskMap, see the document Diskmap.doc in the folder C:\Program Files\Resource Kit.

Warning Be extremely cautious about making any changes to the structures of your hard disk! 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.

You can easily make changes that have serious consequences, resulting in the following error messages:

```
You cannot start any operating system.  
A volume is no longer accessible.  
You have to recreate and reformat all of the partitions and logical  
volumes.
```

DiskProbe displays a messages asking you to verify any change that you want recorded to disk. Please carefully consider any changes before accepting them.

With careful use of the disk tools, you can solve problems whether they occur through human error, hardware problems, power outages, or other events. It is a good idea to familiarize yourself with these tools in a test situation. Testing is especially important if your configuration has spanned, striped, mirror, or RAID-5 sets.

Note Using DiskProbe, you can save, restore, find, examine, and change the bytes of any sector on the disk, including the MBR and the boot sector. The MBR of disk 0 is used to start Windows 2000–based computers, and the system and boot volumes of disk 0 must be defined in the partition table, making the boot sectors easily located, regardless of the disk configuration used. As a result, DiskProbe can be used to back up and restore these disk structures on computers using dynamic disk.

Viruses

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.

MBR Viruses

MBR viruses exploit the master boot code that runs automatically when the 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 disk. Once 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 active, primary partition's listing in the partition table is destroyed, the computer cannot start. Other viruses relocate the MBR to the last sector of the disk; if that sector is not protected by the virus, it might be overwritten during normal use of the computer, preventing the system from being restarted.

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 disk. Once 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. If that sector is not protected by the virus, it might be overwritten by normal use of the computer, rendering the volume inaccessible or preventing the system from restarting, depending upon which volume was affected.

How MBR and Boot Sector Viruses Affect Windows 2000

A computer can contract an MBR or boot sector virus by one of two common methods: by starting up from an infected floppy disk; or by running an infected program, causing the virus to drop an altered MBR or boot sector onto the hard disk. The function of an MBR or boot sector virus is typically contained once Windows 2000 has started. If a payload is not run during system startup and the virus preserved the original MBR or boot sector, Windows 2000 prevents the virus from self-replicating to other disks.

Windows 2000 is immune to viruses infecting these disk structures during normal operation, because it only accesses physical disks through protected mode disk drivers. Viruses typically subvert the BIOS INT 13h disk access routines, which are ignored once Windows 2000 has started. However, Windows 2000 computers that are multiple-booted with MS-DOS, Windows 95, or Windows 98 can become infected when Windows 2000 is not running the computer.

If a multiple-boot computer on which Windows 2000 has been installed becomes infected by an MBR or boot sector virus while running another operating system, Windows 2000 is vulnerable to damage.

Once the protected mode disk drivers have been activated, 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. However, viruses that have a payload trigger that executes during startup are a threat to computers that are running Windows 2000 because the trigger process is initiated before the control during the computer startup process passes to Windows 2000.

Treating an MBR or Boot Sector Virus Infection

To remove a virus from your computer, use a current, well-known, commercial antivirus program designed for Windows 2000, and update it regularly. In addition to scanning the hard disks in 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 computer. Scan them even if you believe they are not infected. Many infections recur because one or more copies of the virus were not detected.

If the computer is already infected with a boot sector virus when Windows 2000 is installed, standard antivirus programs might not completely eliminate the infection because Windows 2000 copies the original MS-DOS boot sector to a file called `Bootsect.dos` and replaces it with its own boot sector. The Windows 2000 installation is not infected, but if the user chooses to start MS-DOS, Windows 95, or Windows 98, the infected boot sector is reapplied to the system, reinfesting the computer. Antivirus tools that are not specifically designed for Windows 2000 do not know to check `Bootsect.dos` for viruses.

AVBoot

Microsoft provides a customized antivirus tool that can be used for these types of viruses. AVBoot is located in the `\Valueadd\3rdparty\Ca_antiv` folder of the Windows 2000 Setup CD. Insert an empty, high-density, 3.5-inch floppy disk, and use Windows 2000 Explorer to locate and double-click `Makedisk.bat` to create a startup floppy disk that automatically runs AVBoot.

AVBoot scans the memory as well as the MBR and all boot sectors of every locally installed disk. If a virus is found, it offers to remove the virus.

Important Whether you use a third-party antivirus program or AVBoot, be sure to regularly update the virus signature files. Once you install an antivirus program, immediately update the signature files, usually through an Internet connection. Check with the software manufacturer's documentation for specific instructions. AVBoot includes update instructions in the installation folder and on the AVBoot floppy disk.

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. False negative reports can result when using an out-of-date virus scanner. Most commercial antivirus software manufacturers offer monthly updates. Take advantage of the latest download to ensure that your system is protected with the latest virus defenses.

Fdisk /mbr

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, and restoring the MBR does not solve the problem if the virus immediately reinfects the system. In addition, running **Fdisk /mbr** on a system infected by MBR viruses that do 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 disk management program, running this command eliminates the program overlay control and you cannot start up from the disk.

Important Running **Fdisk /mbr** 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 zeroes. If an MBR virus overwrites the signature word, access to all partitions and logical volumes is lost.

Fixmbr command

The Recovery Console, a new troubleshooting tool in Windows 2000, 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 “Repair, Recovery, and Restore” in this book.

Damaged MBRs and Boot Sectors

When you start a computer from the hard disk, the system BIOS code identifies the startup disk and reads the MBR. The master boot code in the MBR searches for the active, primary 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 system partition’s boot sector to start the operating system, the MBR displays one of the following error messages:

Invalid partition table.

Error loading operating system.

Missing operating system.

There might not be an active partition on the hard disk that you want to use to start the computer, or the wrong partition might be identified as the active partition. In this case, use an MS-DOS startup floppy disk to start the computer and use the MS-DOS tool Fdisk to set or change the active partition.

Note Fdisk can only set primary partitions as the active partition. If MBR corruption prevents Fdisk from setting or changing the active partition, you might need to use a third-party, low-level disk editor that can work under MS-DOS to make this change manually. The partition table field that needs to be changed is the **System ID** field. For more information about the fields in the partition table, see “Master Boot Record” earlier in this chapter.

Using an Emergency Repair Disk

If the boot sector cannot find Ntldr, Windows 2000 cannot start. This can be caused by moving, renaming, or deleting Ntldr, corruption of Ntldr, or corruption of the boot sector. Under these circumstances, the computer might not respond to input or might display one of the following error messages:

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

If Ntldr is damaged or missing, or if the boot sector is corrupted, you can resolve either problem by starting the Emergency Repair Disk (ERD) and following the prompts for repairing the installation. For more information about the ERD, see Windows 2000 Server Help.

Using the Recovery Console

If the system cannot start due to a corrupted MBR or boot sector, you can recreate either of these disk structures by using the Recovery Console.

To start the Recovery Console, start the computer from the Windows 2000 Setup CD or the Windows 2000 Setup floppy disks. If you do not have Windows 2000 Setup floppy disks and your computer cannot start from the CD, use another Windows 2000–based computer to create the setup disks. For information about creating the Windows 2000 Setup floppy disks, see Windows 2000 Server Help. Start the computer and enter Windows 2000 Setup. Press ENTER at the **Setup Notification** screen to go to the **Welcome to Setup** screen. Press R to repair a Windows 2000 installation, and then press C to use the Recovery Console.

The Recovery Console displays all valid installations of Windows 2000 on the computer. To access the hard disk, press the number key representing the Windows 2000 installation you want to repair (typically represented as 1: C:\WINNT), and then press ENTER.

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

The Recovery Console may also show valid installations of Windows NT. However, as the Recovery Console was not specifically designed to work with Windows NT, the results of attempting to access a Windows NT installation can be unpredictable.

The Recovery Console then prompts you for the Administrator password.

Note To access the hard disks with Recovery Console, you must know the password for the local Administrator account. If you do not have the correct password, or if the security database for the installation of Windows 2000 you are attempting to access is corrupted, Recovery Console does not allow access to the local disks.

To replace the MBR, at the Recovery Console command prompt, type:

fixmbr

Verify if you want to proceed because 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 Y to proceed, or N to cancel.

Important 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** may not resolve the problem.

To have the Recovery Console replace the boot sector, at the Recovery Console command prompt, type:

fixboot

If you do not specify a particular drive, the Recovery Console replaces the boot sector of the boot partition. If another volume's boot sector is corrupted, enter the **Fixboot** command, followed by a space, and then specify the drive letter with a colon immediately afterward.

You can also use DiskProbe to edit these disk structures. Because DiskProbe only runs under Windows 2000 and Windows NT, you can only use it to fix errors on a boot sector, a non-startup partition, or an MBR on a non-startup disk. For more information about using DiskProbe to edit boot sectors and MBRs, install the Support Tools from the Windows 2000 Setup CD and see the document Dskprtrb.doc. For more information about the Recovery Console, see “Repair, Recovery, and Restore” in this book.

Other Disk Problems

Disk problems can occur that do not involve the MBR, partition table, extended partition table, or boot sector. Typically, the Windows 2000 disk tools cannot be used to troubleshoot these disk problems.

Stop 0x0000007B — Inaccessible Boot Device

This Stop message, also known as Stop 0x7B, indicates that Windows 2000 lost access to the system partition during the startup process.

This error can be caused by a number of factors, including the failure of the boot device driver to initialize, the installation of an incompatible disk or disk controller, an incompatible device driver, disk cabling problems, disk corruption, viruses, or incompatible logical block addressing (LBA).

The system BIOS allows access to fixed disks that use fewer than 1024 cylinders. Many later disks, however, exceed 1024 cylinders. LBA is used to provide support for these disks. Such support is often built into the system BIOS. However, there are potential problems with LBA, such as:

- If partitions are created and formatted with LBA disabled, and LBA is subsequently enabled, a STOP 0x7B can result. The partitions must be created and formatted while LBA is enabled.
- Some LBA schemes are not compatible with Windows 2000. Check with your vendor.

Warning Changing LBA modes from one scheme to another can force you to recreate and reformat the partitions.

For more information about Stop message 0x7B, see “Windows 2000 Stop Messages” in this book.

Volume Displays as Unknown

If you create and format a volume with NTFS, FAT16, or FAT32, but you cannot access files on it, and Disk Management displays the volume as Unknown, the boot sector for the volume might be corrupted. For NTFS volumes, there are two other possible causes for a volume to display as Unknown:

- Permissions for the volume have been changed.
- The master file table (MFT) is corrupted.

The boot sector can be corrupted by viruses. For more information about cleaning an infected computer, see “Viruses” earlier in this chapter.

Permission problems can occur when you perform the following tasks:

- Create a second volume.
- Remove the group Everyone from the access control list (ACL).
- Grant access to a specific user.

The single user has normal access, but if other users log on, or if Windows 2000 is reinstalled, Disk Management shows the drive as Unknown. To correct this problem, log on as an administrator and take ownership of all folders, or return full control to the group Everyone.

If the MFT file is corrupted, there is no general solution, and you need to contact Microsoft Product Support Services.

CMOS Problems

The CMOS typically stores configuration information about the basic elements of the computer, including RAM, video, and storage devices. If the 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 on the CMOS, and what the standard configuration is. You can access the CMOS by using either a keyboard sequence at startup or 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 problems:

- 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 go bad, 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 may 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 2000 drivers. Try disabling **sync negotiation** in the SCSI BIOS, checking the SCSI identifiers of each device, and confirming proper termination. For enhanced integrated drive electronics (EIDE) devices, define the onboard EIDE port as **Primary only**. Also, check each EIDE device for the proper master, slave, or stand-alone setting. Try removing all EIDE devices except for hard disks.

To make sure that any new disks and disk controllers are supported, see the Microsoft Windows 2000 Hardware Compatibility List (HCL) link on the Web Resources page at <http://windows.microsoft.com/windows2000/reskit/webresources>.

Additional Resources

- For more information about basic and dynamic disks, see Windows 2000 Server Help.
- For more information about troubleshooting disk problems, see Windows 2000 Server Help.

