

# How Basic Disks and Volumes Work (Microsoft Corporation)

Basic disks and basic volumes are the storage types most often used with Windows operating systems. The term basic disk refers to a disk that contains basic volumes, such as primary partitions and logical drives. The term basic volume refers to a partition on a basic disk. Basic disks, which are found in both x86-based and Itanium-based computers, provide a simple, elegant storage solution that can accommodate changing storage requirements. Basic disks support clustered disks, Institute of Electrical and Electronics Engineers (IEEE) 1394 disks, and universal serial bus (USB) removable drives. Before you can use a basic disk, it must have a disk signature and be formatted with either the File Allocation Table (FAT) or NT file system (NTFS) file systems.

An optimal environment for basic disks and volumes is defined as follows:

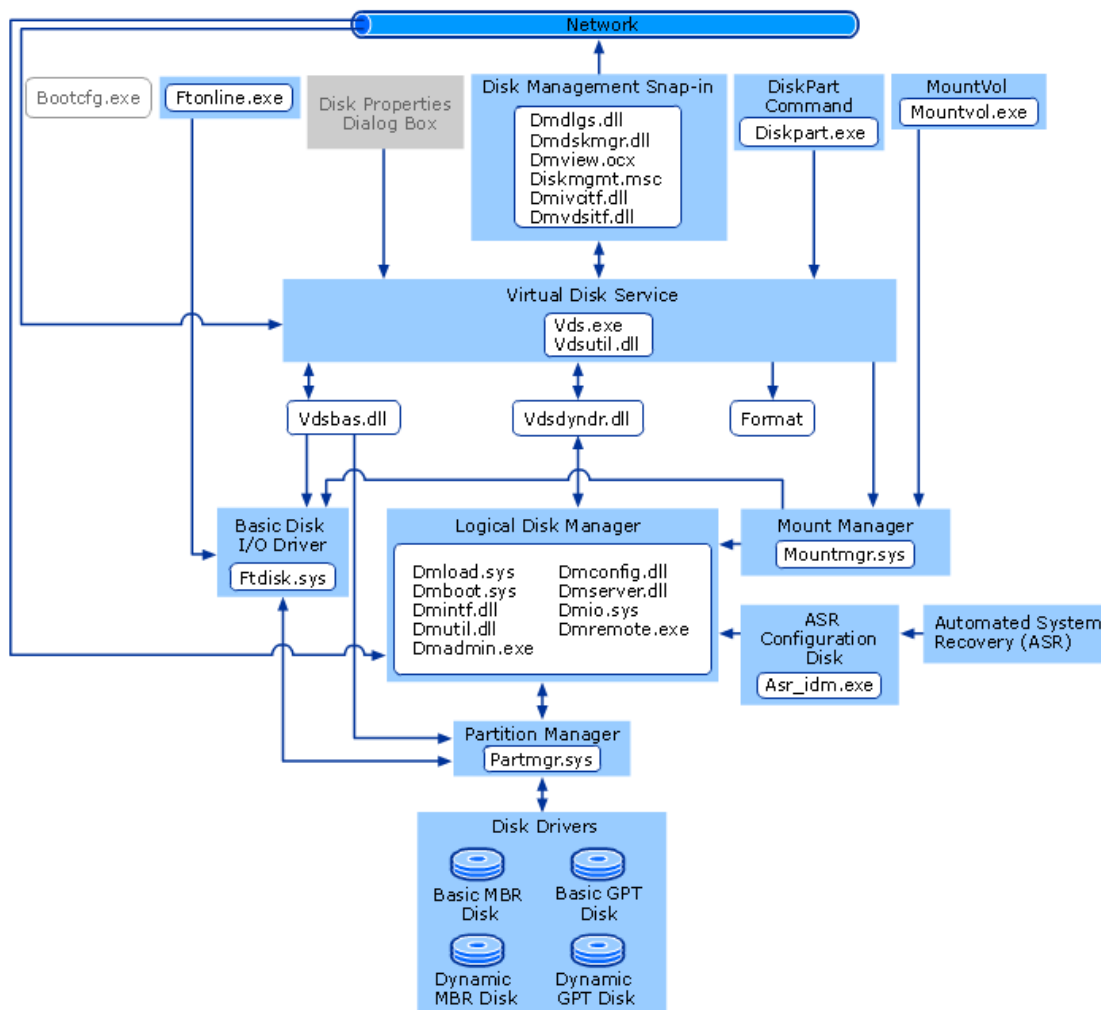
- Windows Server 2003 operating system is installed and functioning properly.
- The basic disk is functioning properly and displays the Healthy status in the Disk Management snap-in.

The following sections provide an in-depth view of how basic disks and volumes work in an optimal environment.

## Basic Disks and Volumes Architecture

Basic disks and volumes rely on the Logical Disk Manager (LDM) and Virtual Disk Service (VDS) and their associated components. These components enable you to perform tasks such as converting basic disks into dynamic disks, and creating fault-tolerant volumes. The following diagram shows the LDM and VDS components.

### Logical Disk Manager and Virtual Disk Service Components



## How Basic Disks and Volumes Work (Microsoft Corporation)

The following table lists the LDM and VDS components and provides a brief description of each.

### Logical Disk Manager and Virtual Disk Service Components

Component	Description
Disk Management snap-in: Dmdlgs.dll Dmdskmgr.dll Dmview.ocx Diskmgmt.msc	Binaries that comprise the Disk Management snap-in user interface.
DiskPart command line utility: Diskpart.exe	A scriptable alternative to the Disk Management snap-in.
Mount Manager command line: Mountvol.exe	A command line utility that can be used to create, delete, or list volume mount points
Virtual Disk Service: Vds.exe Vdsutil.dll	A program used to configure and maintain volume and disk storage.
Virtual Disk Service provider for basic disks and volumes: Vdsbas.dll	The Virtual Disk Service calls into the basic provider when configuring basic disks and volumes.
Virtual Disk Service provider for dynamic disks and volumes: Vdsdyndr.dll	The Virtual Disk Service calls into the dynamic provider when configuring dynamic disks and volumes.
Dmboot.sys Dmconfig.dll Dmintf.dll Dmio.sys Dmload.sys Dmremote.exe Dmutil.dll	Drivers and user mode components used to configure dynamic disks and volumes and perform I/O.
Logical Disk Administrator service: Dmadmin.exe	The VDS provider for dynamic disks and volumes uses the interfaces exposed by this service to configure dynamic disks.
Logical Disk Service: Dmserver.dll	A service that detects and monitors new hard disk drives and sends disk volume information to Logical Disk Manager Administrative Service for configuration. If this service is stopped, dynamic disk status and configuration information might become outdated. If this service is disabled, any services that explicitly depend on it will fail to start.
Basic disk I/O driver: Ftdisk.sys	A driver that manages all I/O for basic disks. Other system components, such as mount point manager, call into this driver to get information about basic disk volumes.
Mount point manager driver: Mountmgr.sys	A binary that tracks drive letters, folder mount paths and other mount points for volumes. Assigns a unique volume mount point of the form \\?\Volume<GUID> to each volume, in addition to any drive letters or folder paths that have been assigned by the user. Ensures that a volume will get the same drive letter each time the computer boots, and also tries to retain a volume's drive letter when the volume's disk is moved to a new computer.

## How Basic Disks and Volumes Work (Microsoft Corporation)

Partition manager: Partmgr.sys	A filter driver that sits on top of the disk driver. All disk driver requests pass through the partition manager driver. This driver creates partition devices and notifies the volume managers of partition arrivals and removals. Exposes IOCTLs that return information about partitions to other components, and allow partition configuration.
-----------------------------------	---

### Basic Disk and Volume Interfaces

Basic disks and volumes can be managed using the Disk Management snap-in. The following table lists the control codes used by the Disk Management snap-in when managing disks and volumes.

For more information about the interfaces used by Disk Management when managing disks and volumes, see *Disk Management Reference* on MSDN.

#### Disk Management Control Codes

Control Code	Operation
IOCTL_DISK_CREATE_DISK	Initializes the specified disk and disk partition table using the specified information.
IOCTL_DISK_DELETE_DRIVE_LAYOUT	Removes the boot signature from the master boot record.
IOCTL_DISK_GET_DRIVE_GEOMETRY_EX	Retrieves information about the physical disk's geometry.
IOCTL_DISK_GET_DRIVE_LAYOUT_EX	Retrieves information about the number of partitions on a disk and the features of each partition.
IOCTL_DISK_GET_LENGTH_INFO	Retrieves the length of the specified disk, volume, or partition.
IOCTL_DISK_GET_PARTITION_INFO_EX	Retrieves partition information for AT and EFI (Extensible Firmware Interface) partitions.
IOCTL_DISK_GROW_PARTITION	Enlarges the specified partition.
IOCTL_DISK_SET_DRIVE_LAYOUT_EX	Partitions a disk.
IOCTL_DISK_SET_PARTITION_INFO_EX	Sets the disk partition type.

For more information about the Disk Management control codes, see *Disk Management Control Codes* on MSDN.

### Basic Disks and Volumes Physical Structure

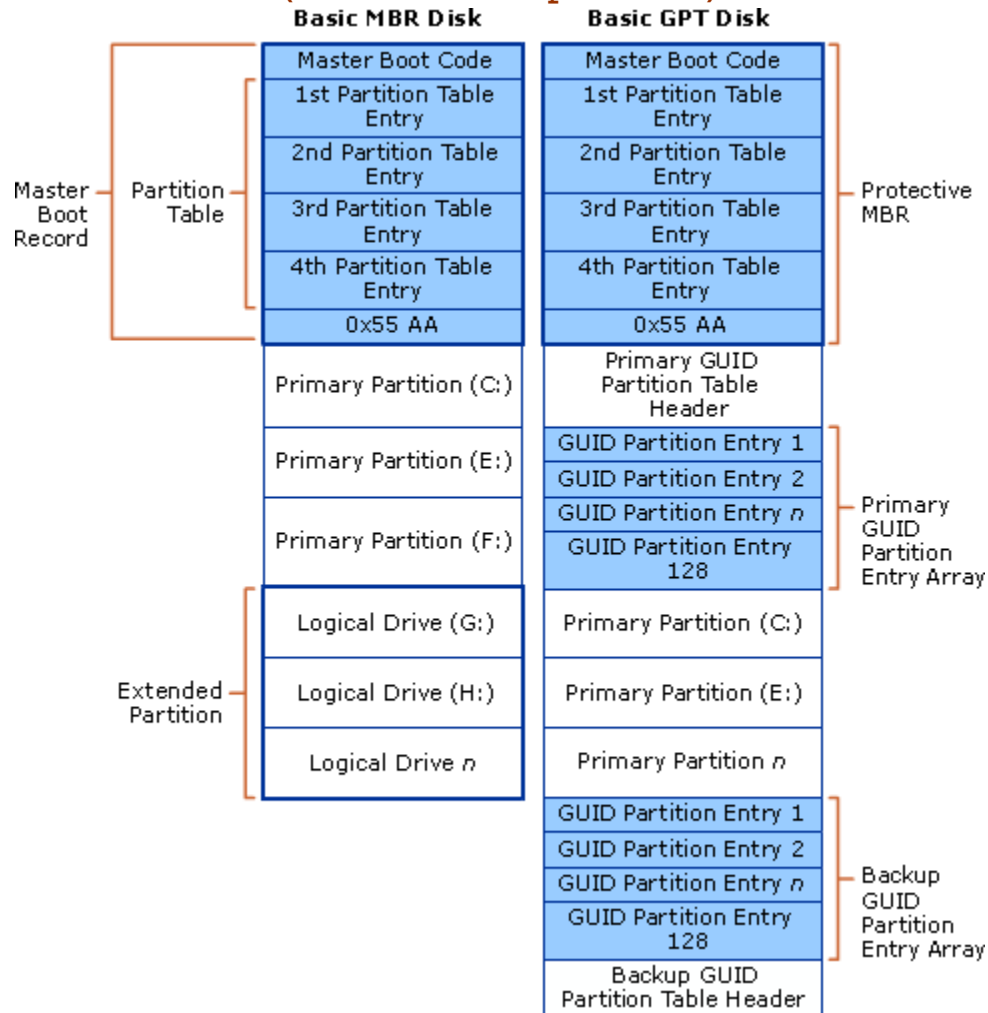
Basic disks can use either the master boot record (MBR) or GUID partition table (GPT) partitioning style. x86-based computers use disks with the MBR partitioning style and Itanium-based computers use disks with the GPT partitioning style.

The following figure compares a basic MBR disk to a basic GPT disk.

#### Comparison of MBR and GPT Disks

## How Basic Disks and Volumes Work

(Microsoft Corporation)



A comparison of MBR and GPT disks is listed in the following table.

### Comparison of MBR and GPT Disks

Characteristic	MBR Disk (x86-based Computer)	GPT Disk (Itanium-based Computer)
Number of partitions on basic disks	Supports up to: <ul style="list-style-type: none"> <li>▪ Four primary partitions per disk, or</li> <li>▪ Three primary partitions and an extended partition with up to 128 logical drives</li> </ul>	Supports up to 128 partitions
Compatible operating systems	Can be read by: <ul style="list-style-type: none"> <li>▪ Microsoft MS-DOS</li> <li>▪ Microsoft Windows 95</li> </ul>	Can be read by: <ul style="list-style-type: none"> <li>▪ Windows XP 64-Bit Edition</li> <li>▪ The 64-bit version of</li> </ul>

## How Basic Disks and Volumes Work (Microsoft Corporation)

	<ul style="list-style-type: none"> <li>▪ Microsoft Windows 98</li> <li>▪ Microsoft Windows Millennium Edition</li> <li>▪ Windows NT, all versions</li> <li>▪ Windows 2000, all versions</li> <li>▪ Windows XP</li> <li>▪ Windows Server 2003, all versions for x86-based computers and Itanium-based computers</li> </ul>	<p>Windows Server 2003, Enterprise Edition</p> <ul style="list-style-type: none"> <li>▪ The 64-bit version of Windows Server 2003, Datacenter Edition</li> </ul>
Maximum size of basic volumes	2 terabytes	2 terabytes
Partition tables (copies)	Contains one copy of the partition table.	Contains primary and backup partition tables for redundancy and checksum fields for improved partition structure integrity.
Locations for data storage	Stores data in partitions and in unpartitioned space. Although most data is stored within partitions, some data might be stored in hidden or unpartitioned sectors created by OEMs or other operating systems.	Stores user and program data in partitions that are visible to the user. Stores data that is critical to platform operation in partitions that the 64-bit versions of Windows Server 2003 recognize but do not make visible to the user. Does not store data in unpartitioned space.
Troubleshooting methods	Uses the same methods and tools used in Windows 2000.	Uses tools designed for GPT disks. (Do not use MBR troubleshooting tools on GPT disks.)

### Master Boot Record on Basic Disks

The master boot record (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 Server 2003 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:

```
Copy Code
Invalid partition table.
Error loading operating system.
```



## How Basic Disks and Volumes Work (Microsoft Corporation)

### Partition Table Fields

Byte Offset	Field Length	Sample Value1	Field Name and Definition
0x01BE	1 byte	80	<b>Boot Indicator.</b> Indicates whether the volume is the active partition. Legal values include: <ul style="list-style-type: none"> <li>▪ Do not use for booting.</li> <li>▪ 80. Active partition.</li> </ul>
0x01BF	1 byte	01	<b>Starting Head.</b>
0x01C0	6 bits	01 *2	<b>Starting Sector.</b> Only bits 0–5 are used. The upper two bits, 6 and 7, are used by the Starting Cylinder field.
0x01C1	10 bits	00 *	<b>Starting Cylinder.</b> 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	07	<b>System ID.</b> Defines the volume type. See the table titled “System ID Values” later in this section for sample values.
0x01C3	1 byte	FE	<b>Ending Head.</b>
0x01C4	6 bits	BF *	<b>Ending Sector.</b> Only bits 0–5 are used. The upper two bits, 6 and 7, are used by the Ending Cylinder field.
0x01C5	10 bits	09 *	<b>Ending Cylinder.</b> 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	3F 00 00 00	<b>Relative Sectors.</b> The offset from the beginning of the disk to the beginning of the volume, counting by sectors.
0x01CA	4 bytes	4B F5 7F 00	<b>Total Sectors.</b> 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, 3F 00 00 00, 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 the previous table for the acceptable 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 Server 2003-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.

## How Basic Disks and Volumes Work (Microsoft Corporation)

Windows Server 2003 uses the System ID field to determine which file system device drivers to load during startup. The following table identifies the values for the System ID field.

### 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 Server 2003 does not support multidisk volumes that are created by using Windows NT 4.0 or earlier, and that use System ID values 0x86, 0x87, 0x8B, or 0x8C.

Before you upgrade from Windows NT Server 4.0 to Windows Server 2003, you must first back up and then delete all multidisk volumes. 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.

If you are upgrading from Windows 2000 to Windows Server 2003, you must convert the multidisk volumes to dynamic before you begin Setup, or Setup does not continue.

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 the table titled "System ID Values" earlier in this section 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

## How Basic Disks and Volumes Work (Microsoft Corporation)

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 the figure titled "Interpreting Data in the Partition Table" earlier in this section.

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 Windows Server 2003 supports hard disks that 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 cylinders (10 bits) heads (8 bits) sectors per track (6 bits).

Using the maximum possible values yields:

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

Windows Server 2003 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 Server 2003 ignore the Starting and Ending CHS fields regardless of whether the partition exceeds the first 7.8 GB of the disk. However, Windows Server 2003 must place the appropriate values in the Starting and Ending CHS fields because Windows 95, Windows 98, and Windows Millennium Edition (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 232 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).

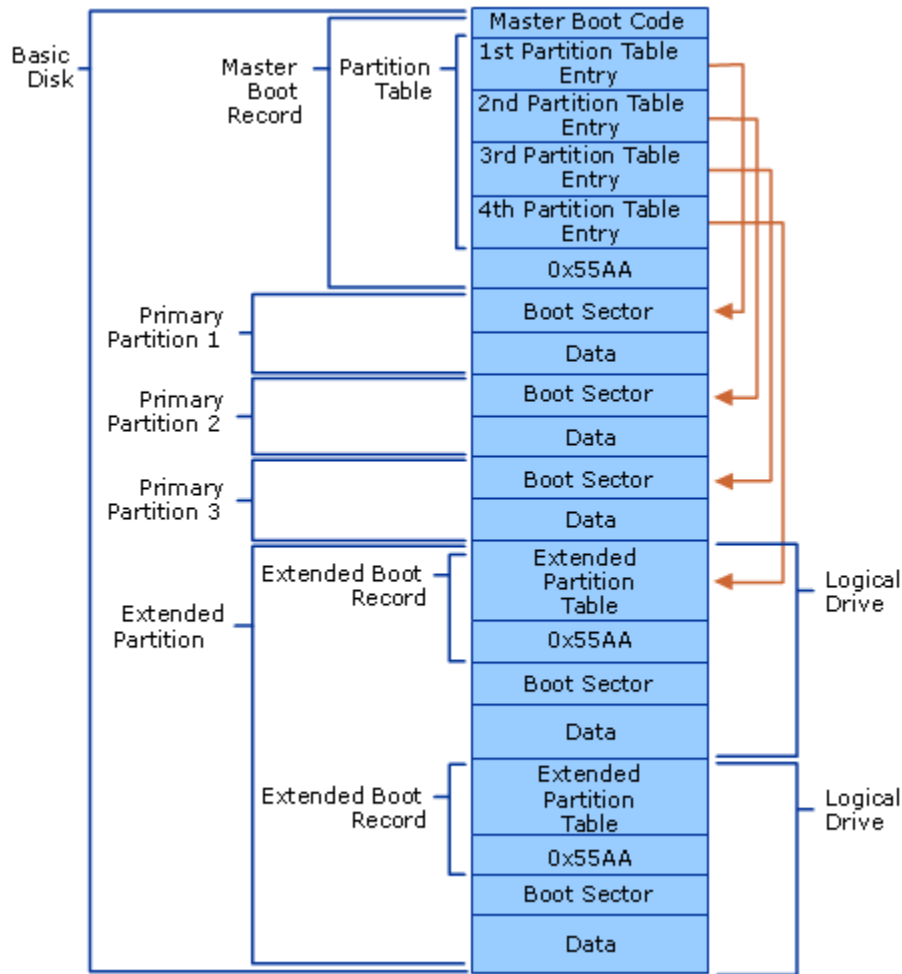
### Note

- For more information about the maximum partition size that each file system supports, see "NTFS Technical Reference [ [http://technet.microsoft.com/en-us/library/cc758691\(W.S.10\).aspx](http://technet.microsoft.com/en-us/library/cc758691(W.S.10).aspx) ] ."

The following figure 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.

### Detail of a Basic Disk with Four Partitions

## How Basic Disks and Volumes Work (Microsoft Corporation)



### Extended Boot Record on Basic Disks

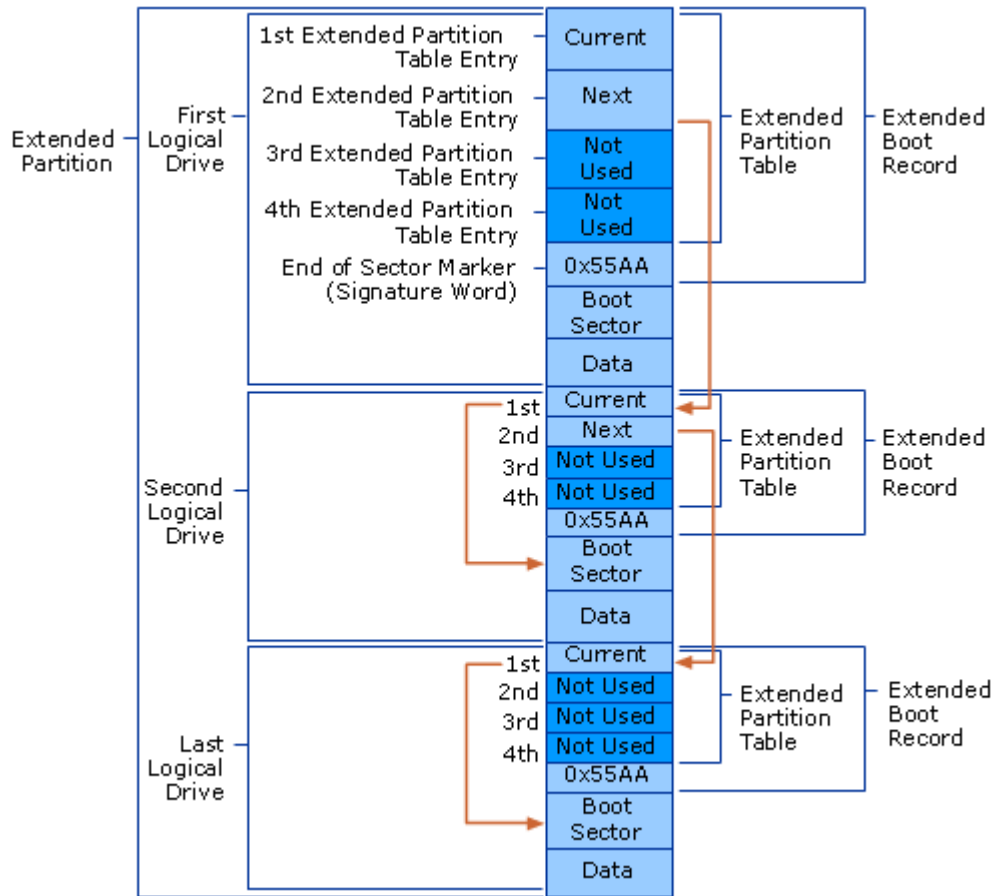
An extended boot record (EBR), which consists of an extended partition table and the signature word for the sector, exists for each logical drive in the extended partition. It contains the only information on the first side of the first cylinder of each logical drive in the extended partition. The boot 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.

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 the following figure, 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.

### Detail of an Extended Partition on a Basic Disk

## How Basic Disks and Volumes Work (Microsoft Corporation)



With the exception of the last logical drive on the extended partition, the format of the extended partition table, which is described in the following table, 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.

### Contents of Extended Partition Table Entries

Entry	Contents
First	Information about the current logical drive in the extended partition, including the starting address for the boot sector preceding the 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. For more information about partition table fields, see the table titled "Partition Table Fields" earlier in this section.

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.

# How Basic Disks and Volumes Work (Microsoft Corporation)

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.

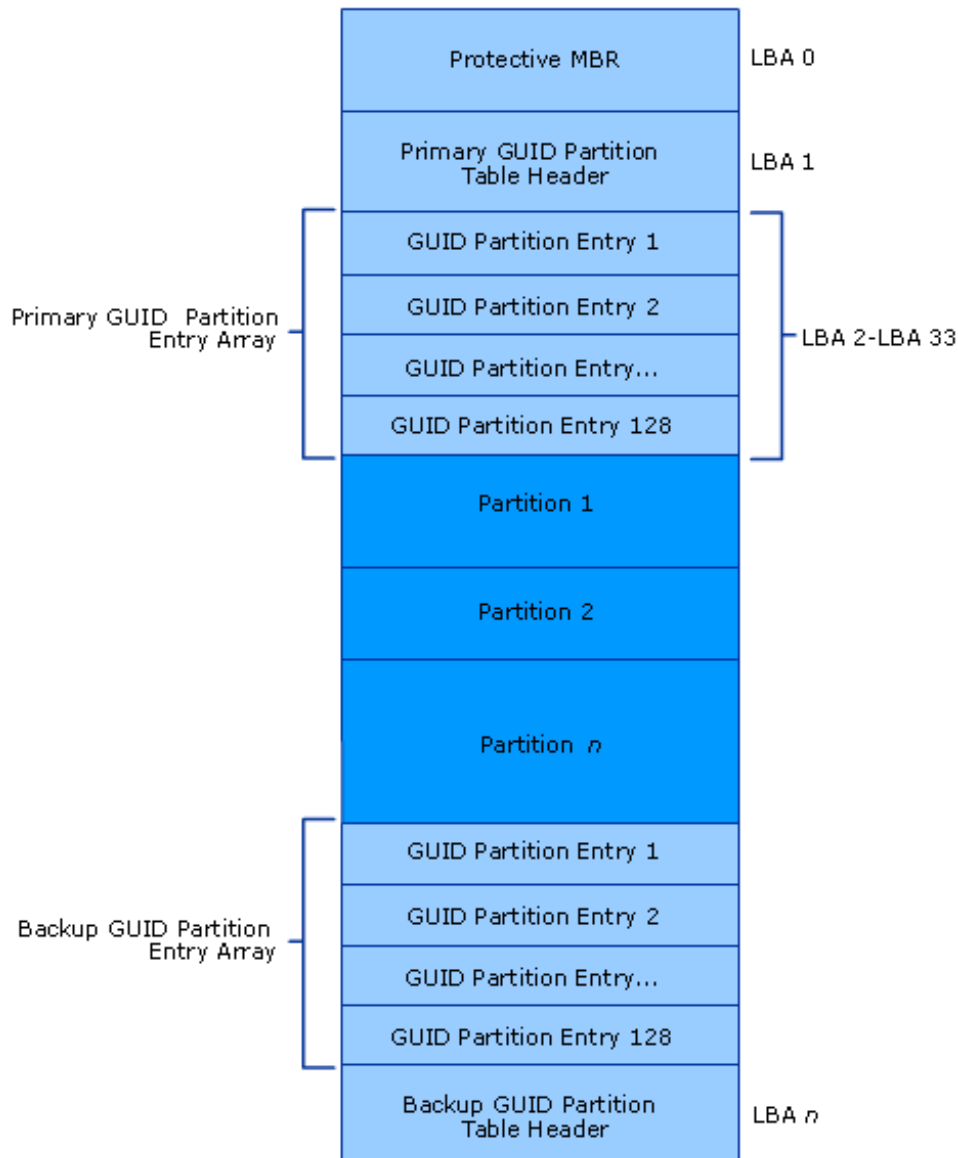
## Disk Sectors on GPT Disks

GUID partition table (GPT) disks use 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 the following figure, the first structure on a GPT disk is the Protective MBR in LBA 0, followed by the primary 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.

### Partition Structures on a GPT Disk



## How Basic Disks and Volumes Work (Microsoft Corporation)

### 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 run.

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

```
Copy Code
000001B0: 00 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 00 .....
000001E0: 00 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 00 55 AA .....U.
```

The following table describes the fields in each entry in the Protective MBR.

### Protective MBR in GPT Disks

Byte Offset	Field Length	Sample Value1	Field Name and Definition
0x01BE	1 byte	00	<b>Boot Indicator.</b> Must be set to 00 to indicate that this partition cannot be booted.
0x01BF	1 byte	00	<b>Starting Head.</b> Matches the Starting LBA of the single partition.
0x01C0	1 byte	02	<b>Starting Sector.</b> Matches the Starting LBA of the single partition.
0x01C1	1 byte	00	<b>Starting Cylinder.</b> Matches the Starting LBA of the GPT partition.
0x01C2	1 byte	EE	<b>System ID.</b> Must be EE 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 Server 2003, the partition is displayed as a <b>GPT Protective Partition</b> and cannot be deleted.
0x01C3	1 byte	FF	<b>Ending Head.</b> Matches the Ending LBA of the single partition. If the Ending LBA is too large to be represented here, this field is set to FF.
0x01C4	1 byte	FF	<b>Ending Sector.</b> Matches the Ending LBA of the single partition. If the Ending LBA is too large to be represented here, this field is set to FF.
0x01C5	1 byte	FF	<b>Ending Cylinder.</b> Matches the Ending LBA of the single partition. If the Ending LBA is too large to be represented here, this field is set to FF.
0x01C6	4 bytes	01 00 00 00	<b>Starting LBA.</b> Always set to 1. The Starting LBA begins at the GPT partition table header, which is located at LBA 1.
0x01CA	4 bytes	FF FF FF FF	<b>Size in LBA.</b> The size of the single partition. Must be set to FF FF FF FF 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

## How Basic Disks and Volumes Work (Microsoft Corporation)

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 the 64-bit versions of Windows Server 2003 cannot access the disk.

### Note

- 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 the 64-bit versions of Windows Server 2003.

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

```
Copy Code
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  .....
```

The following table describes the fields in the GPT header.

### GUID Partition Table Header

Byte Offset	Field Length	Sample Value1	Field Name and Definition
0x00	8 bytes	45 46 49 20 50 41 52 54	<b>Signature.</b> Used to identify all EFI-compatible GPT headers. The value must always be 45 46 49 20 50 41 52 54.
0x08	4 bytes	00 00 01 00	<b>Revision.</b> The revision number of the EFI specification to which the GPT header complies. For version 1.0, the value is 00 00 01 00.
0x0C	4 bytes	5C 00 00 00	<b>Header Size.</b> The size, in bytes, of the GPT header. The size is always 5C 00 00 00 or 92 bytes. The remaining bytes in LBA 1 are reserved.

## How Basic Disks and Volumes Work (Microsoft Corporation)

0x10	4 bytes	27 6D 9F C9	<b>CRC32 Checksum.</b> 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	00 00 00 00	<b>Reserved.</b> Must be 0.
0x18	8 bytes	01 00 00 00 00 00 00 00	<b>Primary LBA.</b> The LBA that contains the primary GPT header. The value is always equal to LBA 1.
0x20	8 bytes	37 C8 11 01 00 00 00 00	<b>Backup LBA.</b> The LBA address of the backup GPT header. This value is always equal to the last LBA on the disk.
0x28	8 bytes	22 00 00 00 00 00 00 00	<b>First Usable LBA.</b> The first usable LBA that can be contained in a GUID partition entry. In other words, the first partition begins at this LBA. In the 64-bit versions of Windows Server 2003, this number is always LBA 34.
0x30	8 bytes	17 C8 11 01 00 00 00 00	<b>Last Usable LBA.</b> The last usable LBA that can be contained in a GUID partition entry.
0x38	16 bytes	00 A2 DA 98 9F 79 C0 01 A1 F4 04 62 2F D5 EC 6D	<b>Disk GUID.</b> A unique number that identifies the partition table header and the disk itself.
0x48	8 bytes	02 00 00 00 00 00 00 00	<b>Partition Entry LBA.</b> The starting LBA of the GUID partition entry array. This number is always LBA 2.
0x50	4 bytes	80 00 00 00	<b>Number of Partition Entries.</b> The maximum number of partition entries that can be contained in the GUID partition entry array. In the 64-bit versions of Windows Server 2003, this number is equal to 128.
0x54	4 bytes	80 00 00 00	<b>Size of Partition Entry.</b> The size, in bytes, of each partition entry in the GUID partition entry array. Each partition entry is 128 bytes.
0x58	4 bytes	27 C3 F3 85	<b>Partition Entry Array CRC32.</b> 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		<b>Reserved.</b> Must be 0.

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.

### 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. The 64-bit versions of Windows Server 2003 create 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.

## How Basic Disks and Volumes Work (Microsoft Corporation)

### GUID Partition Entry

A GUID partition entry defines a single partition and is 128 bytes long. Because the 64-bit versions of Windows Server 2003 create 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. The 64-bit versions of Windows Server 2003 recognize only the partition type GUIDs described in the following table, and do 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.

### Partition Type GUIDs

Partition Type	GUID Value
Unused entry	{00000000-0000-0000-0000-000000000000}
EFI System partition	{28732AC1-1FF8-D211-BA4B-00A0C93EC93B}
Microsoft Reserved partition	{16E3C9E3-5C0B-B84D-817D-F92DF00215AE}
Primary partition on a basic disk	{A2A0D0EB-E5B9-3344-87C0-68B6B72699C7}
LDM Metadata partition on a dynamic disk	{AAC80858-8F7E-E042-85D2-E1E90434CFB3}
LDM Data partition on a dynamic disk	{A0609BAF-3114-624F-BC68-3311714A69AD}

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 the previous table.

```
Copy Code
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 .....
```

## How Basic Disks and Volumes Work (Microsoft Corporation)

```

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      .....

```

The following table illustrates the layout of a GUID partition entry. The sample values correspond to the EFI System partition entry in the preceding example.

### GUID Partition Entry

Byte Offset	Field Length	Sample Value1	Field Name and Definition
0x00	16 bytes	28 73 2A C1 1F F8 D2 11 BA 4B 00 A0 C9 3E C9 3B	<b>Partition Type GUID.</b> Identifies the type of partition. The partition type GUID in this example identifies Microsoft Reserved partitions. For a description of partition type GUIDs, see the table titled Partition Type GUIDs earlier in this section.
0x 10	16 bytes	C0 94 77 FC 43 86 C0 01 92 E0 3C 77 2E 43 AC 40	<b>Unique Partition GUID.</b> A unique ID created for each partition entry.
0x 20	8 bytes	3F 00 00 00 00 00 00 00	<b>Starting LBA.</b> The starting LBA of the partition that is defined by this partition entry.
0x 28	8 bytes	CC 2F 03 00 00 00 00 00	<b>Ending LBA.</b> The ending LBA of the partition that is defined by this partition entry.
0x 30	8 bytes	00 00 00 00 00 00 00 00	<b>Attribute Bits.</b> Describe how the partition is used. For a description of the attribute used by the 64-bit versions of Windows Server 2003, see the table titled "GUID Partition Entry Attributes Used by the 64-Bit Editions of Windows on Itanium-based Computers" later in this section.
0x 38	72 bytes	EFI system partition	<b>Partition Name.</b> A 36-character Unicode string that can be used to name the partition.

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.

### 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. The 64-bit versions of Windows Server 2003 use two attributes as described in the following table.

# How Basic Disks and Volumes Work (Microsoft Corporation)

## GUID Partition Entry Attributes Used by the 64-Bit Editions of Windows on Itanium-based Computers

Bits	Description
Bit 0	Specifies that this partition is required for the platform to function. All original equipment manufacturer (OEM) partitions must have this bit set to protect the OEM partition from being overwritten by the disk tools supplied with Windows Server 2003.
Bit 60	Marks the partition as read-only. Used only for primary basic partitions of type {EBD0A0A2-B9E5-4433-87C0-68B6B72699C7}.
Bit 62	Marks the partition as hidden. Used only for primary basic partitions of type {EBD0A0A2-B9E5-4433-87C0-68B6B72699C7}.
Bit 63	Prevents the system from assigning a default drive letter to the partition. Used only for primary basic partitions of type {EBD0A0A2-B9E5-4433-87C0-68B6B72699C7}.

### 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 BIOS parameter block (BPB) and then mount the volume.

## Basic Disks and Volumes Processes and Interactions

The following basic disk and volume processes and interactions assume that your computer has at least one basic disk and that the basic disk is functioning properly.

### Creating a Basic Volume

When you create a basic volume, the Virtual Disk Service (VDS) uses the IOCTL, IOCTL\_DISK\_SET\_DRIVE\_LAYOUT\_EX, to set the drive layout, and add a new partition.

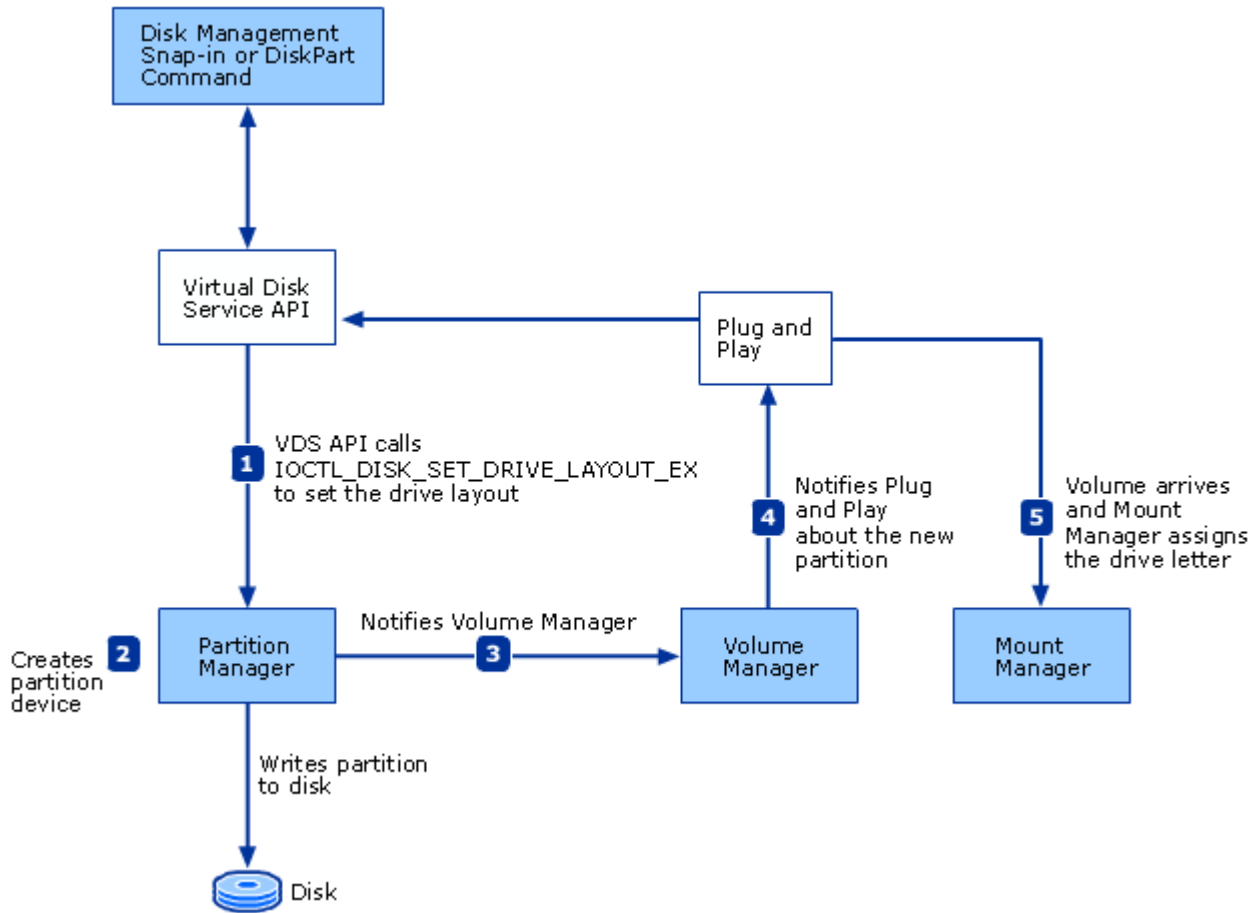
VDS calls the disk driver. Partition manager (Partmgr.sys), sits on top of the disk driver as a filter). Partition manager creates the partition device and notifies the volume manager that there is a new partition. The volume manager announces the volume device to Plug and Play and to the system. Plug and Play notifies the mount point manager (Mountmgr.sys) that a new volume has arrived and the mount point manager sets up a drive letter as long as AutoMount is enabled.

### Note

- AutoMount is disabled by default on Windows Server 2003, Enterprise Edition, and Windows Server 2003, Datacenter Edition.

### What Happens When a Basic Volume Is Created

## How Basic Disks and Volumes Work (Microsoft Corporation)



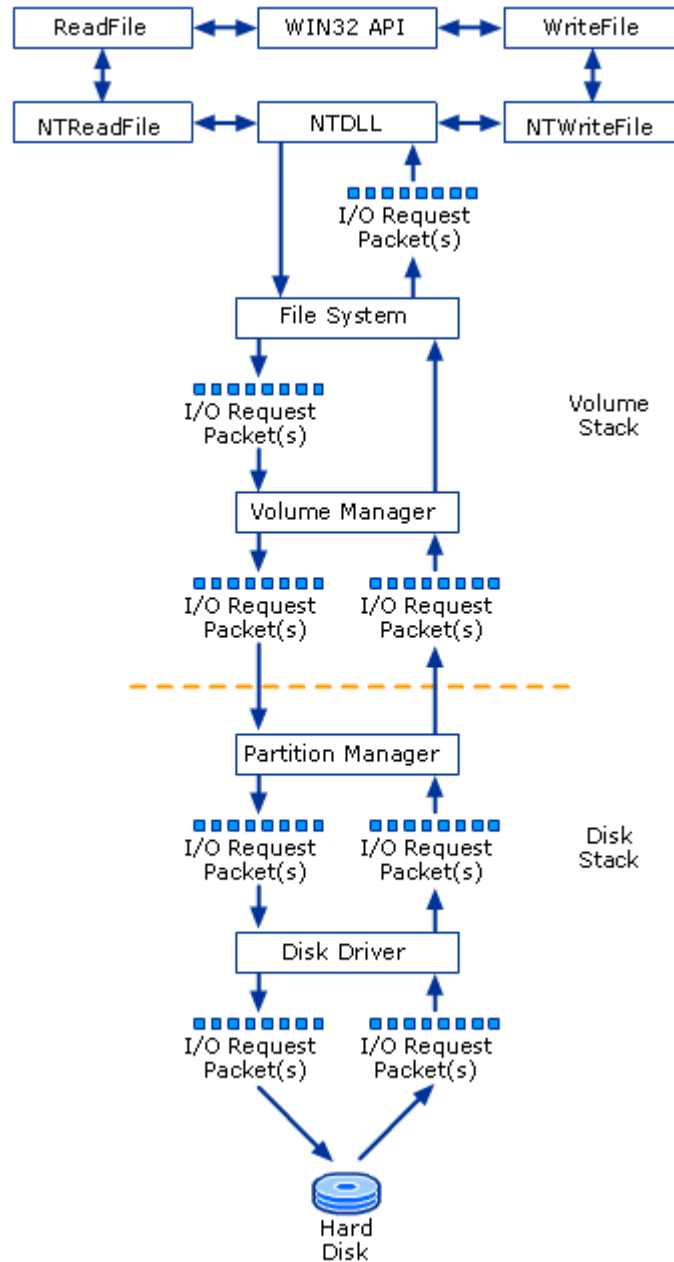
### Reading or Writing to a Basic Volume

When you read or write data to basic disk or volume, the input/output (I/O) is sent through the file system to the volume manager. The volume manager creates a new I/O request packet (IRP) and sends it to the partition manager (Partmgr.sys) in the disk stack, which relays it to the disk driver. The volume manager waits for the completed IRP to return from the disk stack. Then, the volume manager completes and returns the file system IRP.

The following figure illustrates the processes involved when reading or writing to a basic volume.

### Basic Disk and Volume Input/Output (I/O) Processes

## How Basic Disks and Volumes Work (Microsoft Corporation)



### Converting a Basic Disk into a Dynamic Disk

You can convert a basic disk to dynamic by using Disk Management or by using DiskPart, a command-line tool that provides the same functions as Disk Management.

When you convert a disk to dynamic, the following events occur:

- All existing primary partitions and logical drives become simple volumes.
- The disk joins the local disk group and receives a copy of the dynamic disk database.

### Note

- For certain disks, the menu command to convert the disk to dynamic is unavailable in Disk Management.

## How Basic Disks and Volumes Work (Microsoft Corporation)

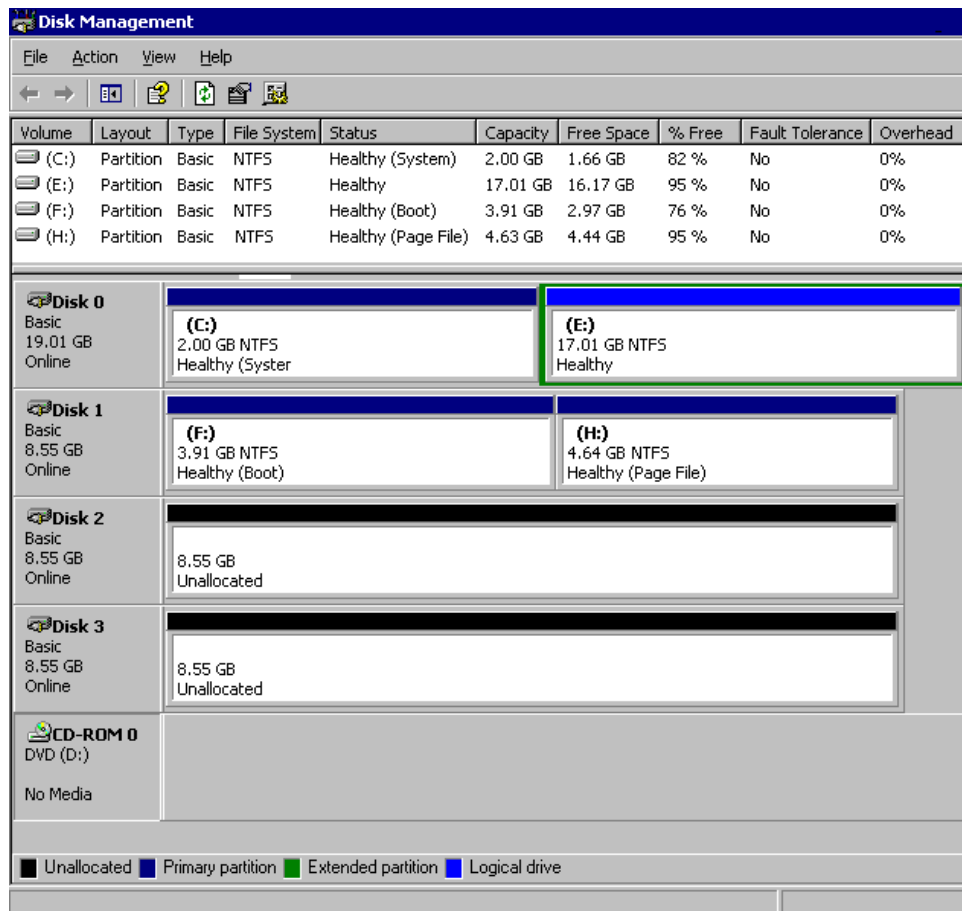
- You can convert basic disks to dynamic at any time. In most cases, you do not need to restart your computer to complete the conversion. However, you must restart the computer if the disks you are converting contain any of the following volumes:
  - **System volume (x86-based computers only).** The system volume contains hardware-specific files such as Ntldr and Boot.ini. These files are needed to load Windows Server 2003 in x86-based computers.
  - **Boot volume.** The boot volume contains the Windows Server 2003 operating system and its support files. In x86-based computers, the boot volume can be, but does not have to be, the same volume as the system volume. In Itanium-based computers, the boot volume is never the same volume as the EFI System partition.
  - **Volumes that contain the paging file.** The paging file is a hidden file on the hard disk that Windows Server 2003 uses to hold parts of programs and data files that do not fit in memory.

### Note

- When you convert MBR disks that contain the system, boot, or paging file volumes to dynamic, you are prompted to restart the computer two times. You must restart the computer both times to complete the conversion.

As shown in the following figure the Disk Management snap-in identifies the system and boot volumes, as well as those that contain the paging file in the graphical and disk list views. If you have a combined system and boot volume that also contains the paging file (the most common scenario), then only **System** is shown.

### How Disk Management Identifies Separate System, Boot, and Paging File Volumes for an x86-based Computer



## How Basic Disks and Volumes Work (Microsoft Corporation)

The list volume command in DiskPart shows the system, boot, and paging file volumes as follows:

Copy Code	Volume ###	Ltr	Label	Fs	Type	Size	Status	Info
	Volume 0	G			DVD-ROM	0 B	Healthy	
	Volume 1	C		NTFS	Partition	2048 MB	Healthy	System
	Volume 2	E		NTFS	Partition	17418 MB	Healthy	
	Volume 3	F		NTFS	Partition	4003 MB	Healthy	Boot
	Volume 4	H		NTFS	Partition	751 MB	Healthy	Pagefile

Even after you convert a disk to dynamic, some types of primary partitions do not become dynamic volumes. These partitions retain their partition entries in the partition table and are shown as primary partitions in Disk Management. These partitions are:

- Known OEM partitions (usually displayed in Disk Management as EISA Configuration partitions).
- EFI System partitions on GPT disks.

### Before Converting Disks to Dynamic

Converting a disk to dynamic changes the partition layout on the disk and creates the dynamic disk database. The result of these changes is increased flexibility for volume management in Windows Server 2003. However, these changes are not easily reversed, and the structure of dynamic disks is not compatible with some operating systems. Therefore, you must consider the following issues before you convert disks to dynamic.

#### If the disk contains partitions displayed as Healthy (Unknown) in Disk Management

Do not convert a disk to dynamic if it contains unknown partitions created by other operating systems. Windows Server 2003 converts unrecognized partitions to dynamic, making them unreadable to other operating systems.

#### If a disk contains shadow copies

If you use separate disks to store the source volume and the shadow copies, and you convert the disk that contains the source volume to dynamic, the shadow copies are lost. Shadow copies are retained only if the source files and the shadow copies are stored on the same volume.

#### If the disk contains an OEM partition that is not at the beginning of the disk

Do not convert a disk to dynamic if it contains an OEM partition that is not at the beginning of the disk. (In Disk Management, an OEM partition usually appears as an EISA Configuration partition.) When you convert a disk to dynamic, Windows Server 2003 preserves the OEM partition only if it is the first partition on the disk. Otherwise the partition is deleted during the conversion to dynamic.

#### If you want to extend a dynamic volume

You can extend dynamic volumes that do not retain their partition entries in the partition table. The following volumes retain their entries in the partition table and cannot be extended:

- The system volume and boot volume of the operating system that you used to convert the disk to dynamic.
- Any basic volume that was present on the disk when you converted the disk from basic to dynamic by using the version of Disk Management included with Windows 2000.
- Simple volumes on which you run the DiskPart command **retain**. This command adds a partition entry to the partition table. However, after you use this command, you can no longer extend the volume.

## How Basic Disks and Volumes Work (Microsoft Corporation)

### Note

- The **retain** command adds an entry to the partition table of an MBR disk only for simple volumes that are contiguous, start at cylinder-aligned offsets, and are an integral number of cylinders in size. If a volume does not meet these requirements, the **retain** command fails.
- The following examples describe volumes on which the **retain** command will succeed:
- The simple volume is contiguous and starts at the beginning of the disk.
- The simple volume was present on the disk when the disk was converted to dynamic.

The only way to add more space to the system or boot volume on a dynamic disk is to back up all data on the disk, repartition and reformat the disk, reinstall Windows Server 2003, convert the disks to dynamic, and then restore the data from backup.

The following volumes do not have partition entries and can be extended:

- Simple volumes and spanned volumes created from unallocated space on a dynamic disk.
- A basic volume that meets the following criteria:
- The basic volume is not the system or boot volume.
- The basic volume is on a disk that was converted from basic disk to dynamic disk by using Windows Server 2003.

Although striped, mirrored, or RAID-5 volumes do not have entries in the partition table, Windows Server 2003 does not support extending them. You can add more space to a striped, mirrored, or RAID-5 volume by backing up the data, deleting the volume, recreating the volume by using Windows Server 2003, and then restoring the data.

### If you want to install Windows Server 2003 on a dynamic volume

You can install Windows Server 2003 only on dynamic volumes that retain their partition entries in the partition table. The only dynamic volumes listed in the partition table are the following:

- The system volume and boot volume of the operating system (Windows Server 2003 or Windows 2000) that you used to convert the disk to dynamic. The system volume and boot volume can be simple or mirrored volumes.
- Any basic volume that was present on the disk when you used Windows 2000 to convert the disk from basic to dynamic.
- Simple volumes on which you run the DiskPart command **retain**. This command adds a partition entry to the partition table so that you can install Windows Server 2003 on the simple volume.
- A basic mirror set that was converted to a dynamic mirrored volume by using Windows 2000. If you break this mirrored volume into two simple volumes, you can also install Windows Server 2003 on either simple volume because they both retain their partition entries.

Because these dynamic volumes retain their partition entries, you can install Windows Server 2003 on them. However, you cannot extend any of these volumes because you can only extend volumes that do not have entries in the partition table.

### If you want to access the disk by using Windows Millennium Edition or earlier, or Windows NT 4.0

If you plan to move the disk after you convert it to dynamic, note that you can access dynamic disks only from computers that are running Windows 2000, Windows XP Professional, Windows XP 64-Bit Edition, or Windows Server 2003. You cannot access dynamic disks from computers running Windows NT 4.0 or earlier.

When moving disks, note that access to dynamic disks is further restricted by the partition style used on the dynamic disk:

## How Basic Disks and Volumes Work (Microsoft Corporation)

- **Dynamic MBR disks.** Only computers running Windows 2000, Windows XP Professional, Windows XP 64-Bit Edition, or Windows Server 2003 can access dynamic MBR disks.
- **Dynamic GPT disks.** Only Itanium-based computers running Windows XP 64-Bit Edition or the 64-bit versions of Windows Server 2003 can access dynamic GPT disks.

### Note

- Volumes on dynamic MBR and GPT disks are available across a network to computers running MS-DOS, Windows 95, Windows 98, Windows Millennium Edition, Windows NT 4.0 or earlier, Windows XP, and Windows Server 2003.

### If a disk or computer contains multiple copies of Windows XP Professional, Windows Server 2003 or Windows 2000

Do not convert a disk to dynamic if it contains multiple copies of Windows XP Professional, Windows Server 2003, or Windows 2000. Even though these operating systems support dynamic disks, they require certain registry entries that allow them to start from dynamic disks. If the operating systems are installed on the same disk and you use one of the operating systems to convert the disk to dynamic, the registry of the other operating system becomes out-of-date because the drivers required to start the operating system from a dynamic disk are not loaded. Therefore, you can no longer start the other operating system.

One way that you can use dynamic disks with Windows XP Professional, Windows Server 2003, and Windows 2000 in a multiple-boot configuration is to install each operating system to a different disk. However, startup problems can also occur if you boot from one of the operating systems and then convert the disks that contain the other operating systems to dynamic.

To ensure that each operating system can start, start each operating system and then convert only the disk that contains the current boot volume to dynamic. For example, install Windows XP Professional on disk 1 and Windows Server 2003 on disk 2. Use Windows XP Professional to convert disk 1 to dynamic, and then use Windows Server 2003 to convert disk 2 to dynamic. By using this method, you ensure that the registries are updated for each boot volume.

### Disks That Cannot Be Converted to Dynamic

Windows Server 2003 Setup and Disk Management ensure that disks initialized by Windows Server 2003 can be converted to dynamic. However, on some disks the conversion fails or the **Convert to Dynamic Disk** command is not available when you right-click a basic disk. The following conditions prevent you from converting a basic disk to dynamic.

#### Cluster disks

You cannot convert cluster disks to dynamic if they are connected to shared SCSI or Fibre Channel buses. Windows Cluster service cannot read disks that are dynamic and makes dynamic disks unavailable to programs or services that are dependent on these disk resources in the server cluster. For this reason, the option to convert these disks to dynamic is unavailable.

You must use Veritas Volume Manager to use dynamic disks with Cluster service.

#### Removable disks

You cannot use dynamic disks on the following:

- Removable media, such as Iomega Zip or Jaz disks, CDs, DVDs, or floppy disks.
- Disks that use universal serial bus (USB) or IEEE 1394 (also called FireWire) interfaces.

# How Basic Disks and Volumes Work

## (Microsoft Corporation)

### **Disks with sectors larger than 512 bytes**

A sector is a unit of storage on a hard disk. The majority of hard disks use 512-byte sectors. Windows Server 2003 supports converting basic disks to dynamic only if the sector size of the basic disk is 512 bytes.

### **GPT disks with non-contiguous partitions**

If an unknown partition lies between two known partitions on a GPT disk, you cannot convert the disk to dynamic. Unknown partitions are created by operating systems or utilities that use partition type GUIDs that the 64-bit versions of Windows Server 2003 do not recognize.

### **MBR disks that do not have space for the dynamic disk database**

An MBR disk requires 1 MB of free space at the end of the disk to be used for the dynamic disk database. Windows Server 2003 and Windows 2000 automatically reserve 1 MB or one cylinder, whichever is greater, when creating partitions on a disk, but in rare cases, disks with partitions created by other operating systems might not have this space available. If this space is not available, you cannot convert the disk to dynamic.

To convert the disk to dynamic, you must back up or move the data, delete the partitions, recreate the partitions, restore the data, and then convert the disk to dynamic. By using Windows Server 2003 to create the partitions, you ensure that the necessary space is available for the dynamic disk database.

This limitation does not affect GPT disks because the database is created in its own partition with space borrowed from the Microsoft Reserved partition.