

Closes the selected device or volume handle. Once the handle is closed, it is not possible to read from or write to the device or volume.

[More information](#)

If this box is checked, the selected handle is in Read Only mode. If an attempt to write to the device is made, a dialog box will prompt for confirmation to change the access mode to Read Write.

[More information](#)

Sets the selected handle status to Active. Two handles may be open at one time, but only one may be Active. A handle must be Active before data can be read from or written to it.

[More information](#)

Displays the currently available physical drives. Drives which are already open or unavailable will be grayed out. To open a drive, double-click the selection.

[More information](#)

Displays the currently available logical volumes.. To open a Volume, double click the selection.

[More information](#)

Displays the currently selected handle to read from.

[More information](#)

Displays the access mode of the currently selected handle

[More information](#)

Displays the number of sectors to read into the application buffer.

[More information](#)

Displays the first sector to read into the application buffer.

[More information](#)

Select the device or volume to which to write the current run of sectors.

[More information](#)

Enter the starting sector to which to write the current run of sectors.

[More information](#)

Displays the number of sectors that will be written to the device or volume.

[More information](#)

Writes the current run of sectors in the application buffer to the selected device or volume. Sector data writes are permanent: once a sector is overwritten, the previous contents of the sector cannot be recovered.

[More information](#)

Check this box to search for ASCII characters, which are typically used in text documents. An ASCII character is one byte long.

[More information](#)

Check this box to search for a string only at the the specific offset within the sector specified in the **Offset in hex** edit control for the **Search string**. A hexadecimal value must be specified.

This type of search is much faster because only one byte per sector is examined for a match to the first byte in the search string.

[More information](#)

Check this box to search every byte in each sector for **Search string**. Using the default settings, every byte on the device will be searched. This type of search is much slower.

[More information](#)

Check this box to make a case-insensitive search. Check this control if you are unsure if the **Search string** contains both upper-case and lower-case characters.

If this box is not selected, the string match will only succeed if **Search string** finds a perfect match of both upper-case and lower-case characters.

[More information](#)

Check this box to search for a Unicode string. No special formatting is required when typing the string into the **Search string** edit control.

A Unicode character is two bytes long. US English Unicode characters have the high byte set to NULL.

Unicode is the default for the Windows 2000, which uses Unicode internally. For example, on NTFS partitions, Windows 2000 stores filenames in the File Record Segment in Unicode characters.

[More information](#)

Enter the string to search for. The string may be up to 64 characters long and can consist of any printable characters. A non printable character will terminate the string. Spaces are legal characters.

[More information](#)

Enter the starting sector for the search. Any legal sector value can be entered.

[More information](#)

Enter the last sector for the search. Any legal sector value can be entered.

[More information](#)

Enter the sector offset to search as a hexadecimal value. If nothing is entered, sector 0 is assumed. If the first character of the search string does not match the character at this offset, the search jumps unconditionally to following sector.

[More information](#)

Select this button to start the search. The search always begins in the sector displayed in the **First sector to search** edit control. A Progress dialog box will be displayed.

[More information](#)

Displays the total number of cylinders on the physical drive. Windows 2000 is not constrained by the x86 BIOS limit of 1024 cylinders. This number can be up to 64 bits long.

[More information](#)

Displays the type of device the current active handle is mapped to. The Current Volume Information dialog displays data about the underlying physical device.

The legal values for this field are:

- DRIVE_PHYSICAL_DEVICE: A physical drive
- DRIVE_LOGICAL_VOLUME: A logical volume

[More information](#)

Displays the logical Sector per Track value currently being used by the operating system.

[More information](#)

Displays the logical Sector Size value currently being used by the operating system. With Windows 2000, this is the same as the physical device sector size.

[More information](#)

Displays the total number of sectors on the physical device to which the current active handle is mapped.

[More information](#)

Displays the logical tracks per cylinder for the physical device the current active handle is mapped to. This is the value that Windows 2000 uses internally. It corresponds to the BIOS value for number of sides or number of heads.

[More information](#)

Displays the name for current active handle. It is usually a logical drive letter, or a string identifying the current physical drive.

[More information](#)

Displays the number of bytes per sector for this volume. This field is used on both FAT and NTFS volumes.

[More information](#)

Displays the number of sectors in the File Allocation Table. This number should never need to be edited. On NTFS volumes this number should always be 0.

[More information](#)

Displays the number of "hidden" sectors. Normally the number of sectors from the partition table to the bootsector. For primary partitions this number will correspond with the **Sectors per track** field. For volumes on extended partitions this will be the number of sectors on the physical drive prior to this sector. This field is used on both FAT and NTFS volumes.

[More information](#)

Displays a hexadecimal value which describe the media type for the volume. For a fixed disk this will normally be F8. Only hexadecimal numbers can be entered in this field.

[More information](#)

Displays the number of File Allocation Tables. FAT file systems prior to FAT32 will have 2 FATs. FAT32 volumes may have either 1 or 2. On NTFS volumes this number should always be 0.

[More information](#)

Displays the number of logical sides or heads on the volume. This should correspond to the **Tracks per Cylinder** value displayed in the **Current Volume Information** dialog box. This field is used on both FAT and NTFS volumes.

[More information](#)

On NTFS volumes, this value will typically be zero. On FAT volumes it will typically be 1.

[More information](#)

Displays the number of permitted files or sub directories permitted in the root directory. On most FAT volumes this number should be 512. NTFS volumes do not use this field. On NTFS volumes the value will be zero.

[More information](#)

Displays the number of sectors per cluster for this volume. A cluster is the allocation unit size, or the minimum amount of space which can be allocated for a file. The value is a variable that typically varies with the size of the volume. This field is used by both FAT and NTFS volumes.

[More information](#)

Used only by MS-DOS version 3 or lower. The field is restricted to a value of 65,535 and was used for the total number of sectors on the volume. Both FAT and NTFS volumes should have this field set to 0.

[More information](#)

On FAT volumes prior to FAT32, this is the total number of sectors in the volume. With NTFS this field is not used and should be 0. It is replaced by the 64-bit value **Total Sectors**.

[More information](#)

Displays the number of sectors per track. This number will normally be either 32 or 63 depending on the drive geometry translation method used. This field is used by both FAT and NTFS volumes.

[More information](#)

Displays the hexadecimal value for the current drive number. This field is almost always filled in with the value 80. This would correspond to the first physical drive, however, this value is used for all drives in the system. This field should not normally need to be edited.

[More information](#)

Displays eight characters used to identify the operating system used to format the partition. This is not a null terminated string. Typical values include:

- MSDOS5.0 MS-DOS from 5.0 to 6.22 or Windows 2000 FAT
- MSWIN40 Windows 95/98
- NTFS Any version of Windows 2000

[More information](#)

Used by the file system driver to mark the volume as dirty while the disk is being written. On boot, if this value is set, Chkdsk will run. MS-DOS used this field to store the current side.

[More information](#)

Displays an 8-byte character field used to identify the file system type. Typical values include:

- FAT12
- FAT16

If the field appears empty, it contains spaces which must be deleted in order to enter new characters.

[More information](#)

Displays a hexadecimal value used as an extended boot record signature. The value should be hex 29.

[More information](#)

Displays a Double-Word field which contains the volume serial number.

[More information](#)

Displays the volume label. Only FAT volume labels are stored in the boot sector. The field contains 11 characters. If the volume does not have a label this field will contain spaces which must be deleted in order to enter new characters.

[More information](#)

Jumps to the last sector in this volume.

[More information](#)

Displays the checksum for the boot sector. This value will normally be zero.

[More information](#)

Displays a value currently unknown, but believed to be used by NTFS tools. The default value for this field is 128 (0x80)

[More information](#)

Displays the number of clusters per NTFS File Record Segment.

[More information](#)

Displays the logical number of clusters to the Master File Table. This is a 64-bit value. If the **Sectors per Cluster** field is 2 and Clusters to MFT is 16, then the Master File Table will begin in sector 32. If this value is incorrect, the volume will be marked as "Unknown".

[More information](#)

Displays the logical number of clusters to the Master File Table Mirror. The mirror contains duplicates of the first 3 records in the Master File Table.

[More information](#)

Displays the number of clusters for each Index allocation buffer.

[More information](#)

Displays the 64-bit volume serial number

[More information](#)

Displays the total number of sectors in the volume. This is a 64-bit value.

[More information](#)

Jumps to the starting sector of the Master File Table. This is a calculated value.

[More information](#)

Jumps to the starting sector of the Master File Table Mirror. This is a calculated value.

[More information](#)

Jumps to the sector defined in the **Total Sectors** field.

[More information](#)

Jumps to the logical center of the volume.

[More information](#)

Displays the starting cylinder address for the partition table entry.

[More information](#)

Displays the starting sector for the partition table entry. In BIOS nomenclature, sector numbering starts from 1. Windows 2000 uses Logical Sector addressing and starts sector numbering from 0.

[More information](#)

Displays the starting side for the partition table entry. Starting head and starting side refer to the same value.

[More information](#)

Displays the value that determines if a partition is the system partition, sometimes called the 'active' partition. Double-click on a value to select it. If this byte is set to 0x80, the BIOS or Firmware will assume this is the partition that loads the operating system boot loader or kernel. More than one partition can be active. The BIOS or firmware attempts to start the boot loader or kernel from the first system partition it finds.

[More information](#)

Displays the ending cylinder for the partition table entry.

[More information](#)

Displays the ending sector for the partition table entry.

[More information](#)

Displays the ending side for the partition table entry. Ending head and ending side refer to the same value.

[More information](#)

Displays the relative sector for the start of the partition table entry. This is the first sector of the partition relative to the first sector on the physical drive or the first sector of the previous partition.

[More information](#)

Displays a byte that defines the partition type. Double-click on a value to select it. For primary partitions, the byte defines the file system type such as FAT or NTFS. Extended partitions have a type of 0x05, so it is not possible to determine the file system type without examining the boot sector.

[More information](#)

Displays the total number of sectors for the partition.

[More information](#)

Double-click on the selection to display the values for the selected partition table entry. A partition table can contain a maximum of 4 partition table entries. In Windows 2000, this can be a maximum of 3 primary partitions and 1 extended partition. An extended partition can contain any number of logical drives because the entries for each logical drive are located in a separate partition table at the beginning of each logical volume.

[More information](#)

Jumps to the next partition. If the calculated values are invalid, an error is returned.

[More information](#)

Jumps to the sector defined in the Relative Sector field.

[More information](#)

Select a sector or a run of sectors to read into the application data buffer. Once the sector is read into the buffer, different views of the data may be selected and the data can be edited.

[More information](#)

Allows the sectors in the application buffer to be written back to disk.

[More information](#)

Displays information about the currently active handle.

[More information](#)

Jumps to the first sector in the current run of sectors.

[More information](#)

Jumps to the previous sector in the current run of sectors.

[More information](#)

Jumps to the next sector in the current run of sectors.

[More information](#)

Jumps to the last sector in the current run of sectors.

[More information](#)

Displays the ordinal value of the current physical drive or logical volume.

[More information](#)

Displays the current sector number.

[More information](#)

Displays the current offset in the sector as a hexadecimal value. The caret is located at the current offset.

[More information](#)

Opens the dialog box to open a disk file.

[More information](#)

Opens the dialog box to save the application buffer to a disk file.

[More information](#)



DiskProbe



Overview



Command Prompt Usage



Accessing a Disk or Partition



To open a physical drive or logical volume



Physical Drive Handles



To select a physical drive



Logical Volume Handles



To select a logical volume



Reading Sectors



Writing Sectors



To write the current buffer to disk



To write to another handle



To save sectors as disk files



Editing Disk Files



Searching Sectors



UsingViews



DiskProbe

Overview

What is DiskProbe?

DiskProbe is a sector editor. It allows a user with Administrator privileges to directly modify data on the local computer's hard drive. Data structures such as the Master Boot Record, partition tables, and partition boot sectors can be edited directly. Because these structures are not considered a part of a file system, they are not accessible through any other applications currently available.

When Should I Use DiskProbe?

DiskProbe is designed to aid in the recovery of corrupt data structures which cannot be repaired with the disk tools included with the operating system.

If data structures such as Master Boot Records or partition boot sectors become corrupted, they can be repaired or replaced following the procedures outlined in this documentation.

DiskProbe can also be used as a preventative maintenance tool. Master Boot Records and partition boot sectors can be saved as disk files and kept with the Emergency Repair Disk. In the event of corruption caused by viruses or faulty hardware, these critical data structures can be easily replaced.

Caution

- **Before you make changes with a low-level tool such as DiskProbe, make sure that a reliable, complete backup of the drive is available. Misuse of DiskProbe may make all data on the drive or volume permanently inaccessible.**

DiskProbe and other sector editors function at a level "below" the file system, so the normal checks for maintaining disk consistency are not in force. This tool gives the user direct access to every byte on the physical disk without regard to access privilege, which makes it possible to damage or permanently overwrite critical on-disk data structures.

What Are DiskProbe's Limitations?

Administrators Only

Only users with local Administrator rights can use DiskProbe to access the physical disk. Other users can run DiskProbe, but in the **Open Physical Drive** dialog box, no physical drives will be listed.

DiskProbe on Windows 2000

The version of DiskProbe included with the Windows 2000 Resource Kit does not support all the disk-based features found in Windows 2000.

No FAT32 Support

Support for FAT32 boot sectors has not been included in this release of DiskProbe. While DiskProbe can display, edit, and search for sectors on a FAT32-formatted volume, it does not offer a FAT32 Boot Sector view to specifically show all of the BIOS parameter block (BPB) and extended BPB fields within the FAT32 boot sector. In order to edit the BPB and extended BPB in a FAT32 boot sector, you must use the Bytes view and carefully count the byte addresses to ensure you are editing the bytes in the correct BPB field. For more information on the fields in the FAT32 BPB and extended BPB, see the chapter "Disk Concepts and Troubleshooting" in the *Windows 2000 Professional Resource Kit* or the Server Operations Guide of the *Windows 2000 Server Resource Kit*.

No Dynamic Disk Support

Windows 2000 offers two types of disk configurations: basic disk and dynamic disk. The basic disk configuration uses the same disk structures found on all earlier versions of Windows NT, using primary partitions and logical volumes within extended partitions as containers of data. All disks are initially configured as basic disks. DiskProbe fully supports basic disks.

Dynamic disk is new to Windows 2000. It uses a disk management database located at the end of the disk to manage the structure of disk volumes. Each physical disk has to be upgraded to use dynamic disk.

DiskProbe cannot read the disk management database. That means users who upgrade their disks to dynamic disk will not be able to use all of the functionality of DiskProbe on those disks. While the upgrade to dynamic disk does not change the sectors themselves, nor the ability of DiskProbe to successfully read and write changes to sectors on disks, the ability of DiskProbe to navigate partitions to find boot sectors of particular volumes is impaired. Dynamic disk allows users to have as many volumes as they wish without the limits imposed by the partition table, and disk volumes can be easily extended and spanned. None of these types of additions or changes are recorded in the partition table on dynamic disks, which inhibits DiskProbe's ability to navigate between volumes, locate boot sectors, and identify the start and end sectors of individual volumes. For more information on basic and dynamic disks, the chapter "Disk Concepts and Troubleshooting" in the *Windows 2000 Professional Resource Kit* or the Server Operations Guide of the *Windows 2000 Server Resource Kit*.

DiskProbe on Windows 9x

DiskProbe runs under Windows 95/98, but no physical drives are available. Windows 95 and Windows 98 use BIOS Interrupt 13 calls for disk access, and do not support the kernel mode calls necessary to access large physical drives. But with Windows 95/98 you can still open, edit, and save files as raw hex data.



DiskProbe

Command Prompt Usage

Although it is a GUI tool, DiskProbe can also be run from the command prompt. Only the path and file name are supported as arguments, for example:

dskprobe c:\mydir\sector00.dsk

This example runs DiskProbe and opens Sector00.dsk in the c:\mydir folder.

After the program has been run, double-clicking a file with the .dsk extension will start DiskProbe and load the file.



DiskProbe

Accessing a Disk or Partition

In order to access a [physical disk](#) or [logical volume](#) on the hard disk, or a partition within them, you must open a handle to the device. DiskProbe considers a logical drive or a physical disk to be a device.

To open a device handle to a physical disk or logical volume, you must use either the **Open Physical Drive** or **Open Logical Volume** menu options, accessible from the **Drives** menu.

Drives Menu Options

| Menu Option | Function |
|---------------------------|---|
| Physical Drive | Opens the entire physical disk and gives you access to any sector on the disk. The sector number is the absolute sector from the beginning of the disk. Using the Physical Drive option requires you to read the partition table to find the boot sector address. Only locally-installed hard disks are available from this option. |
| Logical Volume | Opens the volume associated with a drive letter, such as a primary partition, a logical volume within an extended partition, or a floppy disk. Sector numbers are relative to the start of the volume, and you can only access sectors within the range of the selected volume. Using the Logical Volume option requires you to read sector 0 of the logical volume to view the boot sector. While letters mapped to network drives will appear in this option, network drives are not accessible. |
| Volume Information | Allows you access to information about the characteristics of the disk or volume that you open using either Physical Drive or Logical Volume . In either case, you see the same type of information in the Current Volume Information dialog box. Volume Information is the same data shown when you run the tool DiskMap. |

Notes

- Windows 2000 only offers one programmatic handle per device, including disk drives. If another program is using the handle of a physical disk drive, DiskProbe will be unable to access that disk. Similarly, if DiskProbe has a disk's handle open, other programs, such as Disk Management, will not be able to access that disk.
- You can read the MBR and partition table when you use **Physical Drive**, but not when you use **Logical Volume**. However, you can use either **Physical Drive** or **Logical Volume** on the **Drives** menu to read the boot sector.

[Physical device handles](#)

[Logical device handles](#)


How To

[To open a physical drive or logical volume](#)



DiskProbe

To open a physical drive or logical volume

- 1 If DiskProbe is not already running, click here  to run it.
- 2 Select **Drives** from the main menu.
- 3 Select either **Open Physical Drive** or **Open Logical Volume**.

{button ,AL(` 1')} [Related Topics](#)



DiskProbe

Physical Drive Handles

A physical drive handle gives the application access to an entire physical disk. This includes all partitions and free space on the disk, regardless of file systems installed. A physical drive does not provide access to volumes such as Stripe sets, mirror sets, stripe sets with parity, or Volume sets created with Disk Administrator. Each disk that contains members of these volumes must be accessed separately.

Only one handle to a physical device can be open at a time. For example, if DiskProbe has a handle open for Physical Drive 0, and Disk Administrator is opened, the drive will be displayed as Off Line. Administrator privilege is required to open device handles.

Before you can read or write to the disk, you must set its status to Active. Although two device handles can be open at one time, only one can be active. The active handle is the one you are currently reading from or writing to. It is possible to have a Physical Drive handle and a Logical Volume handle for the physical area on the disk. Since you can only read to or write to one active handle at a time, this should not cause a problem.

How To


To select a physical drive

{button ,AL(`1')} [Related Topics](#)



DiskProbe

To select a physical drive

- 1 If DiskProbe is not already running, click here  to run it.
- 2 From the **Drives** menu, select **Physical Drive**.
- 3 In the **Open Physical Drive** dialog box, select the drive to open in the **Available Physical Drives** list box.
- 4 Double-click the drive you wish to open.
The drive will appear next to the first available open handle.
- 5 To make the drive handle active, click the **Set Active** button.

Note

- Before you can read or write to the disk, you must set its status to Active. Although two device handles can be open at one time, only one can be active. The active handle is the one you are currently reading from or writing to. It is possible to have a physical drive handle and a logical volume handle for the physical area on the disk. Since, you can only read to or write to one active handle at a time, this should not cause a problem.

{button ,AL(`2')} [Related Topics](#)



DiskProbe

Logical Volume Handles

Logical volume handles allow you to open a primary partition or a logical drive as a logical device. For example you can open drive C, drive A, or a logical drive in an extended partition.

Once a logical volume handle is opened, it is possible to read all sectors on that volume. Sector numbering starts at 0 for the first sector on the volume. On a typical hard drive with a C: as the first partition on the disk, the partition boot sector might be on sector 32 or 63 if you are viewing it from a physical drive handle, the same sector would be sector 0 if you opened drive C as a logical device.

Before you can read or write to a logical volume, you must set its status to Active. Although two device handles can be open at one time, only one can be active. The active handle is the one you are currently reading from or writing to. It is possible to have a physical drive handle and a logical volume handle for the physical area on the disk. Since you can only read to or write to one active handle at a time, this should not cause a problem.

How To

To select a logical drive

{button ,AL(`1')} [Related Topics](#)

- **DiskProbe**

- To select a logical volume**

- 1 If DiskProbe is not already running, click here [▪](#) to run it.
- 2 From the **Drives** menu, select **Logical Volume**.
- 3 In the **Open Logical Volume** dialog box, select the drive to open in the **Logical Volumes** list box.
- 4 Double-click the drive you wish to open.
The drive will appear next to the first available open handle.
- 5 To make the drive handle active, click on the **Set Active** button.

- Note**

- Before you can read or write to a logical volume, you must set its status to Active. Although two device handles can be open at one time, only one can be active. The active handle is the one you are currently reading from or writing to. It is possible to have a physical drive handle and a logical volume handle for the physical area on the disk. Since, you can only read to or write to one active handle at a time, this should not cause a problem.

{button ,AL(` 3')} [Related Topics](#)

▪ **DiskProbe**

Reading Sectors

You can read sectors from a disk or volume by making choices in the **Read Sectors** dialog box.

When you use DiskProbe to read a sector, it displays the number of the first sector it reads in the title bar. DiskProbe also displays the number of the sector that you are currently viewing. The status bar's sector number display is useful when you want to read more than one sector at a time.

It is recommended that you record the sector number for the start of each partition and logical volume, as well as the sector number of the boot sector for each partition or logical volume. You can use another Windows 2000 Resource Kit tool, DiskMap, to print a hardcopy of this information and other disk configuration data.

Note

- Like DiskProbe, DiskMap is not fully compatible with dynamic disks. DiskMap works correctly with basic disks. Reports generated by DiskMap for dynamic disks may not show all of the dynamic volumes installed nor correctly interpret the size of all dynamic volumes installed.

Remembering which partition or logical volume you are viewing can become a challenge when a disk contains spanned, striped, mirrored, or redundant array of independent disks (RAID) level 5 volumes. If necessary, start over by using DiskProbe to display data in sector 1, which contains the partition table for the disk.

Important

- DiskProbe lists sectors as starting at zero. If the active handle is a physical drive (as selected from the **Open Physical Drive** dialog box), sector 0 in DiskProbe displays the MBR, even though the MBR is commonly identified as sector 1 of cylinder 0, head 0, on a hard disk. If the active handle is a logical volume, sector 0 displays the boot sector.

When DiskProbe displays information about spanned, striped, mirrored, or RAID-5 volumes, it reads the System ID byte stored in the partition table to determine which disks contain members of the volumes. You can also use DiskProbe to determine the starting and ending location of primary partitions, logical volumes, and spanned, striped, mirrored, and RAID-5 volumes when you view the partition table.

However, DiskProbe displays only the information that it finds in each partition table. It is up to you to determine how entries in the partition table relate to the configurations on your disks. For an example of information that DiskProbe displays for an extended partition, see the section "Walking an Extended Partition" in "Troubleshooting With Diskprobe" (Dskprtrb.doc).

Caution

- Be cautious when running other applications when you run DiskProbe. DiskProbe writes changes directly to the disk without using the file cache, the NTFS file system log file, or any file system device drivers. Any changes written to disk with DiskProbe may alter or corrupt data used by open applications, causing them to fail or perform unpredictably.

Selecting a Starting Sector

The Starting sector is the first sector to read. The default will be the current sector, or sector zero if there is no current sector.

If a Physical Device handle is currently the active handle, sector 0 is the first sector on the physical drive and will contain the Master Boot Record and partition tables.

If a Logical Volume handle is the current active handle, sector 0 will be the Bootsector on the volume.

Selecting the Number of Sectors

The maximum value for the number of sectors to read is 4096. If there is insufficient memory to load the run of sectors into memory, a dialog box with a warning will open.

If more than one sector is loaded into memory, the toolbar buttons for moving through the run of sectors will be enabled.

▪ **DiskProbe**

Writing Sectors

You can write the buffer containing the current run of sectors to disk by making choices in the **Write Sector** dialog box.

[To write the current buffer to disk](#)

Selecting the Destination Handle

The destination handle is the device handle to which you want to write. It is not required that you write to the currently active handle. If more than one device handle is open, the secondary handle will have its **Set Active** button enabled.

[To write to another handle](#)

Selecting the Destination Sector

The destination sector is the sector on the currently active device where the data in the application buffer will be written. The data currently in the destination sector will be permanently overwritten. It will not be possible to recover data overwritten at the sector level.

Caution Be sure to [save any sectors you wish to modify as a disk file](#) first and carefully check the destination information.

Certain sectors on the disk contain data that the BIOS or operating system needs to access the disk. If this data is destroyed, it may not be possible to access any data on the drive.

It is critical to insure that the destination-sector information is correct before writing the data to disk.

Once the buffer has been written to disk, the information in the overwritten sectors is permanently lost.

How To

[To write the current buffer to disk](#)

[To write to another handle](#)

{button ,AL(`4;5')} [Related Topics](#)

- **DiskProbe**

- To write the current buffer to disk**

To write the buffer containing the current run of sectors to disk:

- 1 If DiskProbe is not already running, click here
 - to run it.
- 2 [Save the sectors you wish to modify as a disk file.](#)
- 3 Carefully check the destination information.
- 4 From the **Sectors** menu, select **Write**.
 - Or, on the **Toolbar**: click the **Write to disk** button (the fourth from the left).
 - The **Write Sector** dialog box opens.
- 5 Select a destination handle and sector.

- Note**

- When opening the **Write Sector** dialog box, the following message may be displayed:
The current handle is in READONLY mode. Do you wish to change the current mode to allow writing the current buffer to disk?
- Clicking the **Yes** button will change to mode to OPEN_READWRITE. This will allow you to write directly to sectors on the volume.
- Clicking the **No** button will exit the dialog. The **No Changes were made** message box will be displayed.

{button ,AL(` 5')} [Related Topics](#)

- **DiskProbe**

To write to another handle

You must open the handle to the volume to which you want to write prior to opening the **Write Sector** dialog box.

- 1 If DiskProbe is not already running, click here [▪](#) to run it.
- 2 Select either **Open a Physical Drive** or **Open a Logical Volume** to open the handle.
- 3 Click the handle to which you wish to write.
- 4 Click the **Set Active** button for the handle.

{button ,AL(`5')} [Related Topics](#)

▪ **DiskProbe**

To save sectors as disk files

- 1 If DiskProbe is not already running, click here [▪](#) to run it.
- 2 From the **File** menu, select **File Save As...**
- 3 Select a location and file name.

The Disk Probe default file extension of .DSK will be used unless another extension is specified.

Note

- When you save the contents of the application buffer as a disk file, the data is written to disk without any formatting. It is a raw image of the contents of the application buffer. You can [edit this disk file](#) with DiskProbe.

{button ,AL(` 5;6')} [Related Topics](#)

- **DiskProbe**

Editing Disk Files

You can open disk files with the **Open Disk File** toolbar button or the **File** menu. Either selection will open the **File Open** dialog box. When a file is selected, it is loaded into the application buffer.

Once a file has been loaded, it can be edited as raw byte data or viewed and edited with any of the available views. The application buffer can then be either saved as a disk file or written directly to sectors on the disk.

{button ,AL(`6')} [Related Topics](#)

- **DiskProbe**

- Searching Sectors**

- Selecting the Sector Search button from the Toolbar, or Search Sectors from the Tools menu displays the **Search Sector** dialog box. In this dialog box, you can select criteria for searching the currently selected volume for a string.

- Search at offset

- Exhaustive search

- Ignore case

- ASCII characters

- Unicode characters

- **DiskProbe**

Using Views

Disk Probe has several different ways of viewing the data in the application buffer. Each view displays the application's buffer in a different format. You can select views from the View menu.

Byte View

Partition Table View

NTFS Bootsector View

FAT Bootsector View

Byte View

This view displays the data in the application buffer as a hexadecimal dump, in 3 columns:

- The left column is the offset address in hexadecimal format.
- The central column displays the raw data.
- The right column displays the same data in ASCII format. In the ASCII column non-printable characters are displayed as dots.

To edit data in Byte view, move the caret with the arrow keys or the mouse to select the byte to edit. Only valid hexadecimal characters may be entered. Only the data in the application buffer is changed.

To flush the changes to disk, select **Write sector** from either the **Toolbar** or the **Sectors** menu

Partition Table View

This view displays the application buffer as a partition table. All fields in the partition table can be edited, but no checks are made to insure that the data being entered is valid.

Changing the Boot Indicator

The boot indicator determines if the system BIOS or firmware sees the partition table entry as a system partition. The system BIOS or firmware will attempt to start the operating system boot loader from the first system partition it finds. To change the boot indicator, double-click the selected entry.

NTFS Bootsector View

This view displays the current sector as a NTFS boot sector. All fields can be edited.

FAT Bootsector View

This view displays the current sector as a FAT16 boot sector. All fields can be edited.

physical disk

The actual hard or floppy disk. Each physical disk (or drive) can be partitioned into one or more logical volumes or drives to which the operating system assigns a letter.

From the point of view of an operating system, the physical disk is the hardware-determined structure of a hard disk, with its particular geometry of tracks and cylinders, heads and sides, sectors and clusters. The Master Boot Record is always located on the first sector of the physical disk.

logical volume

A kind of device handle that you can use to open a logical drive. A logical drive is a continuous area in an extended partition, encompassing all or part of a hard drive, that a user creates by using the **Create** button in **Disk Administrator**. The operating system assigns each logical drive a drive letter and treats it like a physical disk.

Partitioning a hard disk creates a set of partitions on the physical disk that can be formatted and used by a file system, which treats them as logical volumes.

Master Boot Record

The first sector on every hard disk, which contains the partition table for the disk and a small amount of executable code. The Master Boot Record (MBR) is always located at track (cylinder) 0, side (head) 0, and sector 1 on the hard disk.

On x86-based computers, the executable code examines the partition table at bootup and identifies the system (or bootable) partition. The MBR then finds the system partition's starting location on the disk, loads an image of its Partition Boot Sector into memory, and transfers execution to the Partition Boot Sector image.

partition table

A 64-byte data structure, contained in the Master Boot Record for a hard disk, that the computer uses to determine how to access the disk. A partition table can contain up to four entries, called *partitions*, that make it easier to organize information. Each partition must be completely contained on one physical disk.

Partitioning a hard disk creates a set of logical volumes or disks on a physical disk. Each of these logical disks looks like a physical disk to the operating system, which assigns drive letters to each partition.

partition boot sector

The sector in each partition that contains the basic information used by the file system to access the partition.

In Windows 2000, the Master Boot Record uses the Partition Boot Sector on the system partition to load the multi-boot loader. The Partition Boot Sector consists of a jump instruction, the OEM name and version, the BIOS Parameter Block, the extended BIOS Parameter Block, and the loader routine.

