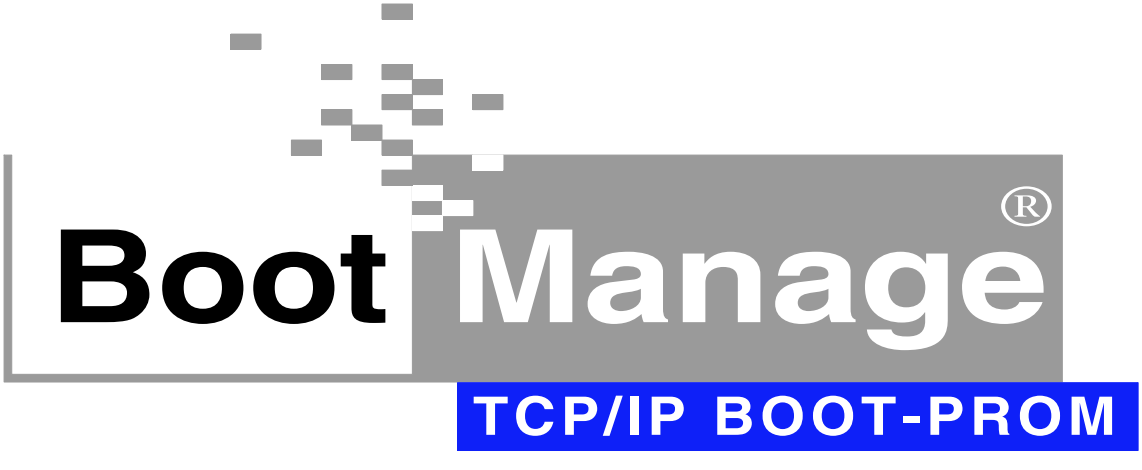
BootManage® TCP/IP BOOT-PROM

User and Reference Manual



BootManage® TCP/IP BOOT-PROM

User and Reference Manual

Colophon:

This document was produced using FrameMaker 6.0 on Windows 98 and NT workstations. Page proofs were printed and reviewed on a 1200 dpi PostScript Level II printer, and final output was run directly to film. The PDF file for online reading was created using Adobe Acrobat 4.05.

Revision history:

December 1993, covers TCP/IP BOOT-PROM software version 1.50.
November 1995, updated to cover TCP/IP BOOT-PROM software version 1.52.
April 1998, updated to cover TCP/IP BOOT-PROM software version 1.56
May 1998, revisited
April 1999, minor changes
May 2001, major revision and update

Printed in Germany.

Table Of Contents

Preface

This Manual explains how to:

- install the TCP/IP BOOT-PROM
- install and configure the server software
- create and maintain remote boot and remote installation configurations
- use the TCP/IP BOOT-PROM utility programs
- use the TCP/IP BOOT-PROM's Multicast TFTP feature
- setup advanced configurations
- troubleshoot remote boot and remote installation environments

How To Read This Manual

As the TCP/IP BOOT-PROM and its related utilities allow to implement a broad variety of operating system remote boot and remote installation environments, you do not need to read this manual as a whole.

"Overview" on page 23 describes in more detail how the TCP/IP BOOT-PROM works, what special features it provides, and what network protocols and server programs it uses. Also, you will find a short description of the TCP/IP BOOT-PROM utility programs.

"Installation" on page 29 shows you how to install, test and troubleshoot the TCP/IP BOOT-PROM. Before installing the TCP/IP BOOT-PROM, it is very important that you read this chapter!

"Remote Boot Concepts" on page 37 introduces you to the various concepts of operating system remote boot and remote installation. Several operating systems are discussed in detail. Before you try out one of the step-by-step examples, we recommend to read this chapter to "get the basic idea" of what you will do in the step-by-step examples.

Each of the step-by-step examples presents a remote boot or remote installation scenario for a specific operating system. Although fully operational, these examples should be understood as a starting environment that you can easily extend and modify to suit your specific needs. For this purpose, you will find several suggestions for extending and modification at the end of each example.

Following comes the "Reference Section", in which every TCP/IP BOOT-PROM utility program is described in detail with all its configuration options and small sample uses. If you want to get detailed information about one of the utility programs, here is the place to look for it.

If you are an experienced TCP/IP BOOT-PROM user and want to get more information about special configurations, you may want to read *"Advanced Techniques"* on page 251.

If you run into trouble of any kind, check with *"Troubleshooting"* on page 253 if your problem is listed there together with a solution. Especially do so before calling bootix technical support!

"Error Messages" on page 261 shows a list of all possible TCP/IP BOOT-PROM error messages together with an explanation of what the message means. If you receive a TCP/IP BOOT-PROM error message, you may want to look it up here in order to find out what might have caused the error.

Related Information

Further documentation and utility program updates can be obtained from the bootix Technology WWW site, http://www.bootix.com.

Before contacting bootix' technical support, be sure that you have checked the available documentation and release notes for possible solutions. In case you need technical support, the best and fastest way is to send an electronic mail to support@bootix.com, including a detailed description of the problem.

Typographic Conventions

Throughout this manual, the following typographic conventions are used:

Italic font is used for file and path names, as a command argument for which you must replace the argument name, and also for general emphasis.

`Constant width font` is used to highlight commands and command line options

In examples where user input should be distinguished from computer generated output, the **user input is set bold**.

```
Within this manual, you will find a lot of usage examples for the various
TCP/IP BOOT-PROM utility programs, representing user input and computer-
generated output. To increase readability, these examples are indented and
set in constant width.

The same format is also used to display the contents of a text file.
```

Special attention should be given to the following three note types:

The exclamation mark symbol indicates notes of special attention or tips. Please read these notes carefully, they often contain information that is important or even critical for understanding the TCP/IP BOOT-PROM and its related programs.

This note represents warnings for possible pitfalls and common mistakes. Especially when setting up complex TCP/IP BOOT-PROM environments, observe these notes to avoid running into trouble.

This note contains advanced (expert) information. When setting up simple TCP/IP BOOT-PROM environments, you do not need this information. However, expert notes may give you ideas about how to add value to existing configurations.

Overview

The TCP/IP BOOT-PROM

The BootManage® TCP/IP BOOT-PROM[1] product is a Boot ROM which implements the diskless boot or unattended installation of a PC by using the common protocols DHCP, BOOTP, and TFTP, which are defined in the "Request For Comments" documents RFC 1531, RFC 951 and RFC 783. RFC documents define protocol standards which are governed by the Internet Engineering Task Force (IETF).

The DHCP and BOOTP protocols allow network clients to determine their network configuration, IP address, boot host, and other vital client information. bootix' powerful extensions to the DHCP and BOOTP protocols allow the setting of user-defined variables which can be used to configure a client, i.e. load site-specific drivers or applications.

The network server runs a program known as the BOOTP Server, which implements the server part of the BOOTP protocol. The BOOTP Server configuration file (named *bootptab* by convention) holds a record for each client, specifying its configuration. Within this text-based file, configuration records for similar clients can be grouped to simplify administration.

The DHCP protocol is an extension to the BOOTP protocol and allows IP addresses to be assigned dynamically by the DHCP server.

The TFTP protocol is a simple protocol to transfer files between servers and clients. The TCP/IP BOOT-PROM uses the TFTP protocol to download the operating system and network software from the server. The client configuration and server name are obtained by the DHCP or BOOTP reply information from the initial inquiry as described above.

---

1 The full product name is "BootManage TCP/IP BOOT-PROM", pointing out that this product is part of the "BootManage" family. However, for convenience we will often simply use the abbreviated name "TCP/IP BOOT-PROM" throughout this manual. The terms "BootManage TCP/IP BOOT-PROM" and "TCP/IP BOOT-PROM" refer to the same product.

The DHCP, BOOTP, and TFTP protocols are operating system independent. Server implementations are available for many operating systems including Windows, UNIX, NetWare, OS/2 and VMS. The server system must be running the TCP/IP network protocol. If the server system does not implement the DHCP, BOOTP, or TFTP protocols, then public domain sources are available on the Internet.

Some older TCP/IP BOOT-PROM types only implement the BOOTP protocol, but not the DHCP protocol. To determine if your TCP/IP BOOT-PROM type supports DHCP, see *"Installation"* on page 29.

How The TCP/IP BOOT-PROM Works

After the ROM has been installed on the network controller, it sends out a combined DHCP and BOOTP request to determine its configuration. If a valid reply is received, the ROM's code starts a TFTP service to transfer the boot image containing the operating system (e.g. DOS) and network software from the server to the client. The programs are stored in a single file called the RAM disk image or container file on the server. After the RAM disk image has been successfully transferred, the ROM installs this file as a RAM disk and executes a System BIOS bootstrap from that RAM disk.

The operating system then starts from the RAM disk and installs the network software which then gives the PC a network drive. Further program loading is done from the network drive. The memory allocated by the RAM disk is freed by the ROM after a successful network logon. No resources are lost or allocated after a bootstrap using the TCP/IP BOOT-PROM. If the PC is rebooted, the ROM again gains control and starts the system from the network.

The Network Server Configuration

The administrator creates a bootable diskette holding the operating system, network software, and additional programs that are needed at system start. This diskette can be tested by booting a PC from that diskette. If the diskette is known to work, then an image is made from this diskette. This image (the boot image or RAM disk image) can be created using a DOS program which comes with the TCP/IP BOOT-PROM distribution.

The RAM disk image is transferred to the network server and a record in the DHCP or BOOTP configuration file is created for the client. This entry holds the network name, hardware address (MAC address), network address (IP address), name, and location of the RAM disk image as well as site-specific variables and

additional information like routers. Several clients can share one RAM disk image to ease administration. After this the network configuration is ready to support a diskless boot from the server.

Features Of The TCP/IP BOOT-PROM

The main features of the TCP/IP BOOT-PROM are listed below:

- The boot image file size can be anywhere between 160 KByte and 2.88 MByte.
- The TCP/IP BOOT-PROM supports the protocols DHCP, BOOTP, and TFTP and works with any standards-conforming BOOTP, DHCP and TFTP server.
- Custom configurations allow to specify the boot device order and/or to prevent booting from certain devices.
- Large block TFTP mode increases download performance.
- Multicast TFTP mode dramatically reduces network load.
- FLASH ROM updates can be scheduled remotely.
- Built-in controller and network diagnostic functions.
- Operation across routers is supported.

The TCP/IP BOOT-PROM Distribution Programs

The TCP/IP BOOT-PROM ships with a number of related programs. For every program, you will find a separate reference chapter later in this manual. The following list gives you a short overview.

BOOTPD32

A Windows 32-Bit BOOTP server that runs on all Windows 32-Bit operating systems (Windows 95/98/ME/NT/2000). `BOOTPD32` is a standards-conforming server implementation of the BOOTP protocol with extended functionality. For detailed information, see *"BOOTP Servers"* on page 81.

TFTPD32

A Windows 32-Bit TFTP server that runs on all Windows 32-Bit operating systems (Windows 95/98/ME/NT/2000). `TFTPD32` is a standards-conforming server implementation of the TFTP protocol with extended functionality, such as large block and multicast TFTP. For detailed information, see *"TFTP Servers"* on page 113.

BMIMAGE

A DOS command line program to create, restore, and modify DOS boot image files without requiring an actual diskette. By directly reading and modifying DOS files within the boot image, `BMIMAGE` can

- insert single files or complete directory trees into a DOS boot image.
- extract single files or complete directory trees from a DOS boot image.
- create optimized DOS boot images which only allocate the actual storage space occupied by the included files.
- create DOS boot images of all common diskette formats up to 2.88 MB without the need for an actual boot diskette.
- write DOS batch files that automate the process of creating or updating multiple DOS boot images.

For detailed information, see *"The BMIMAGE Program"* on page 133.

BPSHELL

A menu based DOS program to

- create, restore, and maintain RAM disk image files.
- create and maintain BOOTP configuration files.
- create passwords for diskette drive protection.
- create, maintain and display menu scripts used by the `BPMENU` boot loader.

For detailed information, see *"The BPSHELL Program"* on page 141.

BPUTIL

A DOS command line program and device driver to:

- View and patch DHCP/BOOTP information such as network name, network address, serial numbers, site-specific configurations, and routing information into ASCII, binary, and batch files.
- Protect the RAM disk image from being overwritten.
- Free the resources allocated by the TCP/IP BOOT-PROM.
- Transfer files using the TFTP protocol.

For detailed information, see *"The BPUTIL Program"* on page 157.

BMFDISK

A DOS program to perform several low-level hard disk operating such as

- create, delete, and modify partition entries in a hard disk's partition table
- retrieve information about partitions
- locate partitions according to their ID vaue
- write a master boot sector to a hard disk
- quick format a FAT16 partition
- read physical sectors from the hard disk and write them to a file
- write physical sectors to the hard disk by reading them from a file
- clear a hard disk (area) by writing 'zero' sectors to it
- return result codes using DOS ERRORLEVEL

For detailed information, see *"The BMFDISK Program"* on page 175.

BMPCSCAN

A DOS program to detect and identify PCI and PnP devices in order to

- create detailed hardware report lists
- store device information in environment variables
- display PCI and PnP information in terse or detailed form
- display information about selected device types only

For detailed information, see *"The BMPCSCAN Program"* on page 197.

BPMENU

A boot loader that is able to present a boot-time menu in order to

- give the user the ability to choose a RAM disk image to be downloaded.
- create a logon screen on the network client.
- set default configurations and help screens.
- handle several configurations through selection. For example, a user can choose whether to boot diskless NFS, NetWare, LanManager, or UNIX if the administrator provides RAM disk images for these applications.

For detailed information, see *"The BPMENU Boot Loader"* on page 205.

BPBOOT

A powerful multi-purpose bootstrap loader, driven by "magic keywords" that are embedded in the BOOTP/DHCP reply information. Based on these keywords and also on local hard disk status information, `BPBOOT` can perform commands before the operating system itself is loaded. Being able to dynamically decide whether to boot from a boot image file or from the local hard disk, `BPBOOT` takes an essential part in completely automated operating system installations.

For detailed information, see *"The BPBOOT Boot Loader"* on page 211.

BMDRV

A Windows NT/2000 device driver that allows one to access the TCP/IP BOOT-PROM's BOOTP/DHCP reply information from Windows NT/2000 applications and services (such as `BMUTIL32`). `BMDRV` can be configured to display a warning message and/or lock the PC if it has not been booted under control of the TCP/IP BOOT-PROM.

For detailed information, see *"The BMDRV Device Driver"* on page 229.

BMUTIL32

A multi-purpose Windows NT/2000 application program and service. `BMUTIL32` queries the `BMDRV` device driver for BOOTP/DHCP reply information and patches this information into both configuration files and the Windows NT/2000 Registry. In addition, `BMUTIL32` provides much of the functionality found in the real mode `BPUTIL` program described earlier.

For detailed information, see *"The BMUTIL32 Program"* on page 237.

Installation

Various brands of Ethernet and Token Ring network controllers are supported by the TCP/IP BOOT-PROM. Since each network controller differs in the way it is detected, initialized, and used, the driver which resides on each ROM is unique to the network controller for which it is designed. Therefore, a new TCP/IP BOOT-PROM must be purchased if the type of network controller hardware is changed.

The required driver type can be determined through its name, i.e. the driver for the Intel Pro/100 network adapter is named E100B. When the ROM initializes, it displays a message like the following:

```
TCP/IP BOOT-PROM 1.64, E100B 1.59MPDF
(c) 1989,2000 bootix Technology GmbH, D-41466 Neuss
Press <SPACE> to abort...
```

This shows that the version of the TCP/IP BOOT-PROM is 1.64 and that the type of network controller driver is E100B with the version 1.59. The letters following the driver version point out that the PROM provides special features as shown in the following table:

Letter	Implemented Feature
C	PCMCIA / CardBus
D	DHCP operation
F	API for low level adapter functions (open, close, ...)
L	32-Bit Large Block TFTP operation
M	Multicast TFTP operation
P	PCI local bus

Installing The TCP/IP BOOT-PROM

The TCP/IP BOOT-PROM code can be provided in several different ways:

- for network adapters that come with an empty BOOT ROM socket, bootix delivers physical EPROM or FLASH devices that fit into the network adapter's socket.
- for network adapters that come with an on-board FLASH chip, bootix delivers "FLASH disks" to program the TCP/IP BOOT-PROM code into the network adapter's FLASH device.
- for PC-Card network adapters without on-board FLASH, the TCP/IP BOOT-PROM code ships on a bootable diskette.
- for PCs with on-board network adapters that neither provide a BOOT ROM socket nor an on-board FLASH chip, the TCP/IP BOOT-PROM code can also be integrated in the PC's System BIOS.

Installing A Physical ROM

The ROM device of the TCP/IP BOOT-PROM usually fits into a socket on the network controller, but some motherboards have an extra socket into which an optional ROM can be placed. In addition, some motherboards allow the program code to be loaded into FLASH memory. Follow the instructions of your network controller for the correct insertion procedure for the ROM device.

When you insert the ROM device, make sure that the notches on ROM and socket point in the same direction. The device will be damaged immediately after the PC is powered on, if the device is inserted in the wrong direction.
If the socket has more pins than the ROM, align the ROM with the back side of the socket (at the opposite side of the notch).

For network adapter specific information related to the TCP/IP BOOT-PROM, check the bootix web site, www.bootix.com.

Using A FLASH Diskette

Some network adapters like 3Com 3C905C or Intel Pro/100 ship with a FLASH chip that is directly soldered to the adapter's board and therefore cannot be replaced. In this case, the TCP/IP BOOT-PROM code must be programmed into the on-board FLASH. For this purpose, bootix provides a FLASH disk that contains

the TCP/IP BOOT-PROM code together with a FLASH program and a license counter.

For detailed instructions about how to proceed for programming the FLASH device, please refer to the documentation that comes with the FLASH disk.

Using A Bootable TCP/IP BOOT-PROM Code Disk

As PC-Card network adapters for Notebook/Laptop computers do not provide FLASH chips, there is no device that can hold the TCP/IP BOOT-PROM code. For these adapters, the TCP/IP BOOT-PROM code ships on "bootable code disks". All you need to do is to boot your Notebook/Laptop computer using the bootable code disk. The loader program on this disk enables the PC-Card and executes the TCP/IP BOOT-PROM code in the same way it would be invoked from a FLASH.

System BIOS Integration

Some PCs mainboards ship with on-board network controllers, but do not provide a BOOT ROM socket or an on-board FLASH chip. In this case, it can be possible to integrate the TCP/IP BOOT-PROM code in the System BIOS. As special technical and licensing issues have to be addressed, please contact bootix if you want to integrate the TCP/IP BOOT-PROM code in the System BIOS.

Configuring The TCP/IP BOOT-PROM

The TCP/IP BOOT-PROM provides configuration parameters that are defined at manufacturing time. In standard configuration, the TCP/IP BOOT-PROM first tries to boot from the network. If this fails, then the PC's standard BIOS bootstrap (as defined in the PC's System Setup) is executed.

The TCP/IP BOOT-PROM configuration parameters control the initial startup device, the "fallback" devices in case of failure, as well as some other parameters such as timeout values.

For detailed information about the available configuration options, please refer to *"TCP/IP BOOT-PROM Parameters"* on page 147.

Determining The Network Controller

All network controllers on EISA, MCA, PCI, and PCMCIA systems can be automatically detected. This means that the TCP/IP BOOT-PROM searches all of the slots for the network controller's identification number and takes the first controller it finds. Slots are searched in ascending order, e.g. 0, 1, 2, 3.

On ISA systems, not all network controllers can be automatically detected. The TCP/IP BOOT-PROM then examines up to 5 addresses to look for the network controller. These addresses are hardwired into the ROM's program code.

Testing The TCP/IP BOOT-PROM

After the installation of the ROM, the Personal Computer comes up executing the System BIOS self-test and then displays the TCP/IP BOOT-PROM copyright message:

```
TCP/IP BOOT-PROM 1.64, E100B 1.59MPDF
(c) 1989,2000 bootix Technology GmbH, D-41466 Neuss
Press <SPACE> to abort...
```

If you do not get this message, the ROM is not installed correctly. Several reasons may cause this problem:

- The ROM is disabled by the network controller configuration.
- An incorrect ROM size is configured on the controller.
- The ROM area overlaps with shared memory or another ROM.
- Another option ROM breaks the interrupt chain. Remove all other option ROMs for testing.
- The ROM is placed at an address outside the range 0xc800-0xdfff.
- The ROM is damaged or has been inserted in the wrong direction.
- Check also with *"Troubleshooting"* on page 253.

If <SPACE> is pressed within 3 seconds, the TCP/IP BOOT-PROM routines will be aborted and the System BIOS bootstrap loader invoked as usual.

According to the configuration of your TCP/IP BOOT-PROM, it may not be possible to boot from a diskette or hard disk drive after the network boot fails.

If you do not press <SPACE> at this point, the TCP/IP BOOT-PROM routines will start and try to boot from the network. If there is no BOOTP server installed the TCP/IP BOOT-PROM ROM-based software will time-out after awhile and pass control to the System BIOS bootstrap loader.

The TCP/IP BOOT-PROM may abort with an error message at this point. In order to save ROM space, ROMs which can only hold 8K byte do not include error messages. Therefore, these ROMs only show the error code and the text must be looked-up in *"Error Messages"* on page 261.

The possible keystrokes at the TCP/IP BOOT-PROM prompt are:

keypress	function
<SPACE>	Abort the TCP/IP BOOT-PROM and boot from the built-in disk
d	Enable diagnostic mode
v	Enable verbose mode

The Diagnostic Mode

If <d> is pressed while the TCP/IP BOOT-PROM displays

```
Press <SPACE> to abort...
```

the diagnostic mode is enabled and will be started after about 3 seconds.

The diagnostic mode is useful to check whether the network controller can be initialized by the TCP/IP BOOT-PROM and establish the network controller's MAC address. Not all ROMs have the full diagnostic mode implemented. All 8K byte ROM versions only show the configuration and MAC address but do not execute a network trace.

In diagnostic mode, the TCP/IP BOOT-PROM will find and initialize the network controller. The WD8ISA ROM displays a message similar to the following:

```
0240 0280 0260 0340 0360 /d000 c400 d400 d800 dc00 /821b ffffffff
0280 d000 00.00.c0.00.80.5d:
ff ff ff ff ff ff 00 00 c0 be f5 06 08 00 45 00
00 5c 14 03 00 00 1e 11 87 29 80 01 01 63 80 01
ff ff 00 7f 00 7d 00 48 ff 47 00 00 00 00 00 00
25 00 00 00 00 00 14 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 03 01 02 00 00 00
00 00 00 00 00 00 64 6b 73 6f 66 74 00 00 00 00
00 00 00 00 00 00 80 01 01 63
ff ff ff ff ff ff 00 00 c0 be f5 06 08 06 00 01
08 00 06 04 00 01 00 00 c0 be f5 06 80 01 01 63
00 00 00 00 00 00 80 01 01 65 00 00 00 00 00 00
25 00 00 00 00 00 14 00 00 00 00 00
```

This debug output contains the following information:

Range Of All Possible I/O Addresses

```
0240 0280 0260 0340 0360
```

The range of i/o ports where the TCP/IP BOOT-PROM searches for the network controller. If no network controller is found at one of theses addresses, it aborts with an error message.

Range Of All Possible Memory Segments

```
d000 c400 d400 d800 dc00
```

The range of segments the TCP/IP BOOT-PROM can use from which the network controller's shared memory is allocated. On network controllers with DMA access, this specifies the DMA channels that can be used.

Feature Settings

```
821b
```

The enabled features of the ROM. Features include read/write protecting diskette drives, locking the system, or defining the bootstrap devices.

IP Broadcast Address

```
ffffffff
```

The IP broadcast address which should be always 255.255.255.255 (hex FF.FF.FF.FF).

Actual Used Settings

```
0280 d000 00.00.c0.00.80.5d
```

The actual settings that the TCP/IP BOOT-PROM uses:

- I/O port at which the network controller was found (0280)
- Shared memory address of the network controller (d000)
- MAC (hardware) address that the network controller uses (00.00.c0.00.80.5d)

Hexdump Of Received Packets

After this, the ROM starts receiving packets from the network. This shows that the TCP/IP BOOT-PROM is able to receive packets and the network controller is setup

correctly. The type of packets received are either broadcast packets or packets with the destination address identical to the network controller's MAC address.

```
ff ff ff ff ff ff 00 00 c0 be f5 06 08 00 45 00
00 5c 14 03 00 00 1e 11 87 29 80 01 01 63 80 01
ff ff 00 7f 00 7d 00 48 ff 47 00 00 00 00 00 00
25 00 00 00 00 00 14 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 03 01 02 00 00 00
00 00 00 00 00 00 64 6b 73 6f 66 74 00 00 00 00
00 00 00 00 00 00 80 01 01 63
ff ff ff ff ff ff 00 00 c0 be f5 06 08 06 00 01
08 00 06 04 00 01 00 00 c0 be f5 06 80 01 01 63
00 00 00 00 00 00 80 01 01 65 00 00 00 00 00 00
25 00 00 00 00 00 14 00 00 00 00 00
```

You can abort the diagnostic mode by pressing <SPACE>.

The Verbose Mode

If you press <v> while the TCP/IP BOOT-PROM displays Press <SPACE> to abort.. the verbose mode is enabled and will start after about 3 seconds.

The 8 kilobyte versions of the TCP/IP BOOT-PROM do not provide enough space to implement the verbose mode program code. Therefore, the verbose mode is not available for these versions.

The verbose mode gives more detailed information about the ROM activities. On Token Ring controllers, the Ring Insert procedure can be monitored. By enabling the verbose mode the following output is displayed when the ROM downloads the RAM disk image via TFTP:

```
reading file /tftpboot/pcnfs.X
to   ether=00.00.c0.3e.3c.40, IP=c1.8d.2f.c6, port=0815
from ether=00.00.c0.e9.16.20, IP=c1.8d.2f.c1, port=1096
```

Boot Image Filename

```
reading file /tftpboot/pcnfs.X
```

This name and location of the boot image file that the TCP/IP BOOT-PROM currently downloads from the TFTP server.

Client Addresses

```
to   ether=00.00.c0.3e.3c.40, IP=c1.8d.2f.c6, port=0815
```

Information about the TCP/IP BOOT-PROM client's addresses:

- Client hardware (MAC) address (00.00.c0.3e.3c.40)
- Client IP address obtained by BOOTP (c1.8d.2f.c6 hex = 193.141.47.198 decimal)
- Client UDP port used for TFTP download (0815 hex = 2069 decimal)

Server Addresses

```
from ether=00.00.c0.e9.16.20, IP=c1.8d.2f.c1, port=1096
```

- Server hardware (MAC) address (00.00.c0.e9.16.20)
- TFTP Server's IP address (c1.8d.2f.c1 hex = 193.141.47.193 decimal)
- Server UDP port used for TFTP download (0815 hex = 4246 decimal)

The verbose mode is useful to find out the filename that the TCP/IP BOOT-PROM client will download from the network server. All numbers are hexadecimal.

Additional DHCP Information

ROMs that implement the DHCP protocol show some more specific information like:

```
DHCP: sip=c2.37.f7.c1, gip=00.00.00.00, yip=c2.37.f7.c6
DHCP: gw0=c2.37.f7.c1, gw1=00.00.00.00, smf=ff.ff.ff.f0
DHCP: bfn=/tftpboot/win95.PX
reading file /tftpboot/win95.PX
to   ether=00.00.c0.3e.3c.40, IP=c2.37.f7.c6, port=0815
from ether=00.00.c0.e9.16.20, IP=c2.37.f7.c1, port=1096
```

In addition to the previous output, this adds the BOOTP or DHCP reply information:

tag	parameter
`sip`	The TFTP server's IP address.
`gip`	The DHCP/BOOTP forwarder's (gateway) IP address.
`yip`	The client's IP address.
`gw0`	The first IP gateway's IP address.
`gw1`	The second (backup) IP gateway's IP address.
`smf`	The IP subnet mask.
`bfn`	The filename of the bootfile.

Remote Boot Concepts

The TCP/IP BOOT-PROM supports both remote boot and remote installation of operating systems over a network. Before setting up a specific configuration, please take some time to read this chapter. Doing so will help you a great deal in understanding the general concepts and the important role that the TCP/IP BOOT-PROM and its related utility programs take in the various remote boot and remote installation environments.

Note that when talking about remote installation, we not only mean first-time installation. Once the installation environment is set up, you can also schedule automated reinstallations with a finger's snap.

Total PC Bootstrap Control

For both remote boot and remote installation environments, the TCP/IP BOOT-PROM was designed to **always** boot the PC from the network. In the case of remote boot, this is obvious. For remote installations, the PC basically only needs to boot from the network when it is actually installed.

But booting the PC from the network does not necessarily mean that the operating system itself is loaded from the network. Instead, the TCP/IP BOOT-PROM downloads a small program called a boot loader, and this boot loader dynamically decides whether to boot the PC locally or to download and execute a boot image from the network server.

After the PC is installed and operational, the boot loader will load only to redirect the boot control back to the BIOS, which will boot the operating system off the local hard disk. But if the administrator decides that the PC should boot in a different way (e.g. for reinstallation or emergency boot), he can schedule this by simply altering the PC's configuration information on the BOOTP/DHCP server. On the next PC reboot, the boot loader will recognize the altered configuration and performs the action that is scheduled by the administrator.

So, the reason for always booting a PC from the network is to have total control over the PC bootstrap and to be able to schedule a reinstallation or emergency boot from the distance without having to walk up to the PC.

Cross-Platform Operation

The TCP/IP BOOT-PROM does not require proprietary services on the server side, nor does it even require specialized server operating systems. As long as the following requirements are met, the TCP/IP BOOT-PROM will work fine:

- The server must implement the TCP/IP network protocol.
- A standard BOOTP (or DHCP) service must be present on one or more servers in the network, and they must be configured to provide configuration information to requesting TCP/IP BOOT-PROM clients.
- A standard TFTP service must be present on one or more servers in the network, and they must be configured to deliver boot loader files and boot images to requesting TCP/IP BOOT-PROM clients.
- It is very common that the client PC, from within the downloaded boot image, wants to connect a network drive to an installation server. In this case, one or more servers in the network must run services that allow network drive connections. The underlying protocol for the network drive connection need not be TCP/IP, it can also be IPX/SPX, NetBEUI or any other transport protocol that the network drive connection requires.

As long as these minimal requirements are met, you can build any remote boot or remote installation environment that suits your needs. For a small network, you can use a single server that acts as BOOTP, TFTP and installation server. In a large company network, you can separate these servers and use multiple BOOTP, TFTP and installation servers for load sharing and fault tolerance.

Also, the TCP/IP BOOT-PROM doesn't mind if you use different server operating systems and/or cross subnet boundaries. Being this flexible and not dependent on specialized and proprietary services, there is no need to change existing company network environments when integrating the TCP/IP BOOT-PROM.

DOS And Windows 3

BOOT-PROMs date back to almost the beginning of the PC era. In those days, hard disk space was expensive, and DOS was the PC operating system of choice. Therefore, BOOT-PROMs were used for "diskless DOS booting".

Since then, the PC and operating system world has rapidly evolved, but there are still a number of interesting uses for remote booting DOS, e.g. in industry PCs,

cash systems, PC-driven airport arrival and departure displays, or for older PCs that are reused as thin clients in terminal server environments. Also, a remote boot configuration serves well as an emergency backup solution when the operating system on the local hard disk is corrupted.

Compared with modern operating systems, DOS and Windows 3.x are small and do not require much PC hardware resources, and there are a number of dedicated application programs that run perfectly well under those "old" operating systems.

In remote boot DOS / Windows 3.x environments, the TCP/IP BOOT-PROM downloads and executes a boot image that reflects a DOS boot disk. The client PCs then boot from the boot image as if it were a bootable floppy disk in drive A:. As it is likely that not all the required operating system and application files will fit into this boot image, there are two common configurations:

- From within the boot image, a network redirector is loaded that establishes a network drive connection to a remote boot server. Over this network drive, additional operating system and application files can be transparently accessed.
- From within the boot image, a ram disk driver is used that creates a virtual drive in the client PC's memory. Then, using the TFTP download function of the TCP/IP BOOT-PROM, a compressed file is downloaded and extracted to the virtual drive. This compressed file contains all the operating system and application files that do not fit into the boot image itself. After being unpacked, these files can be transparently accessed.

For additional sample remote boot DOS / Windows 3.x configurations, please refer to the bootix internet site, http://www.bootix.com.

Windows 95/98/ME/NT/2000

The good news first: Using the TCP/IP BOOT-PROM, it is possible to remotely install all these Microsoft operating systems in much the same manner, and (even better) you can do this completely automated, including hard disk partitioning and formatting and handling multiple reboots.

The bad news is, that it is very difficult to run Windows 95/98/ME truly diskless, and for Windows NT/2000, diskless operation is not possible at all by design[1]. In this manual, we do not provide a sample configuration for diskless 95/98/ME. However, if you are interested in doing so, you may want to have a look at the

1 If you hear about diskless solutions for Windows NT/2000, be sceptical! Mostly, the "diskless solution" will turn out as a thin client / terminal server configuration, and, of course, this is what the TCP/IP BOOT-PROM can do as well.

Microsoft Windows 95 Resource Kit by Microsoft Press. There, a floppy-disk based Windows 95 remote boot solution is shown that you can adapt by converting the floppy disk into a boot image. Note that Microsoft only offers remote boot support for the first version of Windows 95. In all following versions (Windows 95 OSR2, Windows 98 and ME), Microsoft dropped support for diskless operation.

So, let's come back to remote installation of Microsoft operating systems. Once you have performed an automated installation of one of these operating systems, you can easily adapt it so that it works for the other operating systems as well. The basic idea is to use the TCP/IP BOOT-PROM to boot a DOS boot image. From within the DOS boot image, a network redirector is loaded which connects a network drive to the installation server. The installation server hosts a network share that contains all of the operating system distribution files. At this point, you can use the following two techniques to install the operating system:

Scripted Installation

In a scripted (or unattended) installation, the operating system installer program (SETUP.EXE for Windows 95/98/ME, WINNT.EXE for Windows NT/2000) runs on a so-called "answer file" that contains all the information that is needed during the setup process. Instead of querying the user for information like company name, PC host name, network protocol etc., the operating system installer program takes it from the answer file so that the installation can run totally unattended.

In order to reduce the amount of data that is transferred over the network, it is possible to store the operating system installation files in one or more compressed archives. This is especially useful when using multicast technology[1] to distribute the installation files to multiple client PCs at the same time.

Disk Imaging

A disk imaging solution is useful in environments where the PC clients are all equipped with the same hardware and should all be provided with identical configurations, e.g. in a school classroom, in a training center or in a company that has strict hardware standards for client PCs and applications.

For disk imaging, you first setup a reference PC that has all the operating system and application files you want to use, and apply the desired configuration settings to it. Then, you use a disk imaging software to create a sector-by-sector image of the PC's hard disk and place this image on the installation server.

---

1 To transfer files by multicast, bootix offers the MCOPY program which is available as a separate product. If you are interested in this, please contact bootix.

At installation time (after the TCP/IP BOOT-PROM has loaded the DOS boot image and the network redirector has established a network drive connection to the installation server), the disk imaging software is again used to write the sector-by-sector image to the PC's hard disk.

The disk imaging software should also provide mechanisms to apply individual settings to different client PCs, such as the PC's host name, a SID (for Windows NT/2000 machines) and other settings that must be unique for each installed client. It is useful if the disk imaging software is capable of applying differential images, so that handling PC configuration groups is made easier.

Also, when multiple PCs should be installed with the same disk image at the same time, it is of great value to use multicast technology when distributing the hard disk image.

Windows Terminal Server

For Windows NT/2000, diskless operation is not possible by design. When you hear about diskless solutions for Windows NT/2000, mostly the "diskless solution" will turn out as a thin client / terminal server configuration. In such an environment, the operating system on the "thin client" PCs is only used to run a terminal client software. This terminal client software connects to a terminal server machine which actually runs the applications.

So, the "thin client" PC only acts as a terminal which sends the user's keyboard and mouse actions to the server. The server runs the actual application and sends the generated screen output back to the client.

A "thin client" does not need to have fast and expensive hardware, and it also doesn't need to run a "big" operating system (therefore the word "thin"). When equipped with a TCP/IP BOOT-PROM, a "thin client" can boot from the network and doesn't even need a local hard disk.

Using the methods explained in *"DOS And Windows 3"* on page 38, it is possible to setup diskless thin clients that can connect to Windows NT 4.0 or Windows 2000 Terminal Servers.

Linux

The TCP/IP BOOT-PROM can both remote boot and remote install Linux on a client PC. The basic idea is to use the TCP/IP BOOT-PROM to download and execute a specially configured Linux kernel which, in turn, establishes a network connection to the installation server and loads all required files from this server. Once the kernel is loaded, it can execute standard shell scripts that allow you to either

continue remote booting Linux or to perform an automated Linux installation on the client's hard disk.

Diskless X Terminals

Based on a remote boot Linux environment, it is possible to implement low-cost diskless X Terminals. All that has to be done is to set up remote boot Linux PCs and run X Servers on them.

PC Maintenance Operations

The TCP/IP BOOT-PROM can be used to perform PC maintenance operations or "Service boots" on multiple PCs in the enterprise, such as BIOS updates and PC configuration. The basic idea is to convert the bootable maintenance disk into a boot image and use this boot image on multiple PCs.

Remote BIOS Update

Most PC mainboard manufactureres provide DOS utilities that allow to update the PC's BIOS. When you need to perform BIOS updates for a large number of PCs, the TCP/IP BOOT-PROM is of great help. Instead of booting the PCs from a DOS floppy disk, create a single special "BIOS update" boot image and assign this to the PCs you want to update.

Remote BOOT-PROM Update

Using the same method as with remote BIOS update, it is possible to update the TCP/IP BOOT-PROM code itself when this should be necessary. A single special "TCP/IP BOOT-PROM update" boot image will do the job.

Remote PC Configuration

Some PC mainboard manufactureres provide DOS utilities that allow to manipulate the PC's configuration settings. In the same manner as with BIOS and BOOT-PROM updates, it is possible to unify PC configuration throughout the enterprise with a single "PC configuration" boot image.

Windows NT 4 Step-By-Step

This chapter contains a step-by-step example of how to perform a completely automated installation of the Windows NT Workstation 4.0 OS (US version) on a client PC using the BootManage® TCP/IP BOOT-PROM and related utilities.

Note that this is a completely "bare metal" remote installation, beginning with automatically partitioning the hard disk and never requiring a visit to the client machine. With minimal modifications, one can also use this description to install any other 32-Bit Microsoft Windows operating system on a PC equipped with the BootManage® TCP/IP BOOT-PROM.

Environment

As an installation server, we use an Intel-based Windows NT Server 4.0 (US Version) with Service Pack 6a installed. During the real-mode installation phase, the transport protocol is TCP/IP. The client PC is equipped with an Intel Pro/100 series network card which carries a TCP/IP BOOT-PROM of type E100B. We assume that you are already familiar with:

- creating server-based installations for Windows NT Workstation 4.0 machines by using the Network Client Administrator program that is included with the Windows NT Server 4.0 operating system.
- creating an answer file for unattended installation with the Windows NT Setup Manager tool that is included with the Microsoft Windows NT Workstation 4.0 Resource Kit.
- adding OEM components to the automated installation as described in the Microsoft Windows NT Workstation 4.0 Resource Kit.
- installing and configuring the `BOOTPD32` and `TFTPD32` servers for use with the TCP/IP BOOT-PROM as described in the reference section of this manual.

Installation Overview

Before proceeding with the step-by-step installation instructions, please familiarize yourself with this overwiew of the sample installation.

What Microsoft Provides

Microsoft provides a core set of features which enable a nearly hands-free installation of the Windows NT Workstation 4.0 operating system over the network. This is described in the Microsoft Windows NT Workstation 4.0 Resource Kit and involves using the Network Client Administrator and the Windows NT Setup Manager.

The client boots from an MS-DOS boot diskette which contains the (real mode) Microsoft Network Client, connects a network drive to the installation server, and copies all of the installation files to the local hard disk. The user is then instructed to remove the boot diskette, and the system reboots from the local hard disk, which continues the Windows NT installation process.

What The TCP/IP BOOT-PROM Provides

The BootManage® TCP/IP BOOT-PROM builds upon the Microsoft features described above and adds the following features to make the Windows NT installation truly unattended:

- elimination of the boot diskette by using a network boot image
- automatic partitioning and formatting of the client's local hard disk
- custom, per-client configuration parameters communicated from the sever via BOOTP/DHCP options
- installation of multiple clients from the same boot image file
- boot-time network parameters (e.g. Client name, IP address, WINS server, etc.) specified by a central configuration database on the server

How The Installation Process Is Sequenced

Whenever the client PC boots, the TCP/IP BOOT-PROM downloads and executes the `BPBOOT` bootstrap loader. By using the ID field of an unused partition entry as a status flag, `BPBOOT` determines the state of the automated installation and decides whether to boot from the installation boot image or from the local hard disk.

At first, `BPBOOT` does not have any state information, so it defaults to downloading the boot image. Within the boot image, the *PREPARE.BAT* file is used to dis-

patch the different installation preparation tasks. When there is no state information, the REINSTALL section is executed, which contains commands to partition and quick format the hard disk and write the first state information (the status flag) to the hard disk. After that, the PC must be rebooted so that the BIOS can recognize the newly created partition(s).

Upon the next reboot, `BPBOOT` recognizes the previously set state information, but decides that the installation process is not finished yet. So, `BPBOOT` again downloads the boot image. This time, *PREPARE.BAT* recognizes (by the state information) that the hard disk is already partitioned and formatted, so it branches to the OS_INSTALL section. There, the client connects a network drive to the installation server, copies and patches the installation batch file which, in turn, executes the Windows NT setup processor, *WINNT.EXE*. When *WINNT.EXE* finishes its part of the installation (which is to copy all installation files to the client's hard disk), it automatically reboots the PC.

Again, the PC boots, and `BPBOOT` gets control. And again, `BPBOOT` decides that the installation is not finished yet, and downloads the boot image once more. This time, *PREPARE.BAT* branches to the POST_INSTALL section, which double-checks that the installation files have been copied successfully to the client. Also, this section may contain additional commands to modify the installation files on the client and/or to copy additional files to it. At the end of the POST_INSTALL section, the status flag is set to a value that indicates "local boot", and the PC is rebooted.

Once again, `BPBOOT` gets control at PC boot time, but this time, it recognizes the "local boot" state and decides that the remainder of the NT installation has to be done by booting from the local hard disk. So, `BPBOOT` no longer downloads the boot image, but instead boots the client PC from the local hard disk. The NT installation continues with the familiar steps (text mode setup, NTFS conversion and GUI mode setup) until it is finished.

The text-based configuration and batch files are parameterized by BOOTP/DHCP options, so it is possible to install multiple clients using the same boot image. All option values are defined in the BOOTP Server's configuration file. The option values are patched into the text-based configuration files by the `BPUTIL` program after downloading the boot image. This way, although using only a single boot image, each client gets its individual configuration information.

We do not address licensing issues here. Please be sure that you have purchased the appropriate number of operating system client licenses when installing multiple clients over the network.

Step 1: Prepare The Installation Server

Our Windows NT Server 4.0, named NT4SERVER, is member of the workgroup BOOTMANAGE. Attached to a class A IP network, it has the IP address 10.0.0.1 and netmask 255.0.0.0.

On the server, create a shared network installation directory and copy all of the files needed for the client installation to this directory. For this purpose, Microsoft provides the Network Client Administrator that is included with the Windows NT Server 4.0 operating system. Follow the instructions in the chapter entitled *"Microsoft Network Client Administrator"* on page 73.

Step 2: Add Files To Installation Server

For our sample installation, we need to have the DOS *SMARTDRV.EXE* program in the *c:\clients\winnt\tools* directory on the installation server. Therefore, create this directory on the installation server now and copy the *SMARTDRV.EXE* program from an MS-DOS system into it.

When setting up the operating system distribution directory on the server, please pay special attention to the following Microsoft Knowledge Base articles. There, you find helpful information that prevents you from running into different kinds of trouble.

- Q178275: Unattended Installation Using AGP Video Cards
- Q185773: NTFS Corruption on Drives > 4 GB Using ExtendOEMPartition
- Q159203: Unattended Install Prompts for New IP if Zero Is in Address
- Q196288: Unattended Installation Prompts for Install Partition
- Q236158: Winnt32.exe and Setupdd.sys Not Included with SP4, SP5, and SP6

You may want to add additional software packages to the automated installation, e.g. a Windows NT Service Pack, third-party device drivers or a set of standard applications which should be installed together with the operating system. For detailed information about how to add the `BMDRV` device driver and the `BMUTIL32` service, please refer to *"The BMDRV Device Driver"* on page 229 and *"The BMUTIL32 Program"* on page 237.

It is not necessary for our automated installation to add software components, but in case you want to do so, see Chapter 2 of the Windows NT Workstation 4.0 Resource Kit for instructions on how to do this. We recommend to start with a "plain" OS installation and add additional software packages later on.

Step 3: Create Network Client Boot Disk

As described in *"Microsoft Network Client Administrator"* on page 73, use the Microsoft Network Client Administrator to create a network boot diskette. After creating this diskette, you must add the NDIS2 driver for the Intel Pro/100 network adapter and modify the configuration files *PROTOCOL.INI* and *SYSTEM.INI*.

Copy the NDIS2 driver file *E100B.DOS* from the Intel driver diskette to the *\NET* directory of the network boot diskette.

In the network adapter section of the *PROTOCOL.INI* file, modify the drivername line so that it reads:

```
drivername=e100b$
```

In the [network drivers] section of the *SYSTEM.INI* file, modify the netcard line so that it reads:

```
netcard=e100b.dos
```

Step 4: Test Network Client Boot Disk

Test the just-created diskette and verify that it works correctly. You must be able to boot the client PC from the diskette, log-on, and connect a network drive to the *CLIENTS* share on the installation server.

Do not proceed without verifying that the diskette works!

Since no BootManage® TCP/IP BOOT-PROM components have been employed yet, any problems being experienced thus far can only be solved by consulting the appropriate Microsoft documentation or support channels.

Step 5: Add Files To Boot Disk

So far, we have entirely followed the Microsoft way of performing unattended installations over the network, and there was nothing specific to the BootManage® TCP/IP BOOT-PROM.

Before creating a boot image from the installation diskette, we must add the TCP/IP BOOT-PROM's real mode utilities and also modify some configuration files.

Add TCP/IP BOOT-PROM Utilities

On the installation diskette, create the directory *\BIN* and then copy the files *BPUTIL.COM*, *BPUTIL.SYS*, *BMFDISK.EXE*, and *REBOOT.COM* from the BootManage® TCP/IP BOOT-PROM Utility Disk to this directory.

Add/Replace Configuration Files

Copy the following configuration files from the *\SAMPLES\NT4WS* folder of the TCP/IP BOOT-PROM Utility Diskette to the installation diskette, replacing existing files:

CONFIG.SYS
AUTOEXEC.BAT
PREPARE.BAT
NETWORK.BAT
NET\PROTOCOL.INI
NET\SYSTEM.INI

Remove Unneeded Files

There are a number of files on the installation diskette that are not needed for the unattended installation. If you leave these files on the installation diskette, they will do no harm. However, we will not reference them further.

The contents of your installation floppy disk should now look like this:

Root Directory

```
NET            <DIR>
BIN            <DIR>
AUTOEXEC BAT
PREPARE  BAT
NETWORK  BAT
COMMAND  COM
CONFIG   SYS
IO       SYS   (hidden)
MSDOS    SYS   (hidden)
```

Directory BIN

```
BMFDISK  EXE
REBOOT   COM
BPUTIL   COM
BPUTIL   SYS
```

Directory NET

```
IFSHLP   SYS
```

```
NDISHLP  SYS
HIMEM    SYS
NEMM     DOS
TCPDRV   DOS
E100B    DOS
PROTMAN  DOS
EMM386   EXE
NMTSR    EXE
TCPTSR   EXE
TINYRFC  EXE
EMSBFR   EXE
PROTMAN  EXE
NET      EXE
NET      MSG
NETH     MSG
NETBIND  COM
UMB      COM
WFWSYS   CFG
PROTOCOL INI
SYSTEM   INI
TCPUTILS INI
LMHOSTS
NETWORKS
PROTOCOL
```

Step 6: Modify Configuration Files

This sample installation disk configuration has been designed to be as flexible as possible, so that it can be used for a broad variety of systems. However, if you may want to make site-specific configuration changes, use a text editor to modify the configuration files and/or copy additional files to the installation diskette.

To see the contents of the configuration files we use in this example, please refer to *"Configuration And Batch Files Reference"* on page 55.

Step 7: The Installation Batch Files

Connecting a network drive to the installation server allows to access installation batch files that are located on the installation server. Two files are used in this manner:

During the OS_INSTALL phase, the *INSTALL.BAT* file is responsible for copying the operating system files to the client's hard disk.

During the POST_INSTALL phase, the *POSTINST.BAT* file is responsible for verifying that the OS installation was successful. It also allows to copy additional files to the client's hard disk and to further customize the installation process.

Copy the files *INSTALL.BAT* and *POSTINST.BAT* from the *\SAMPLES\NT4WS* folder of the TCP/IP BOOT-PROM Utility Diskette to the *c:\clients\winnt* folder on your installation server.

To see the contents of the configuration files we use in this example, please refer to *"Batch Files On The Installation Server"* on page 59.

Step 8: The unattend.txt File

The *WINNT.EXE* program is used to install the Windows NT operating system on a computer running DOS. Using the `/u` command line option, *WINNT.EXE* derives all setup information from an unattended text file (also called an answer file) instead of interactively querying the user.

See Chapter 2 and Appendix A of the Microsoft Windows NT Workstation 4.0 Resource Kit for instructions on how to create an unattended text file. Microsoft provides the Windows NT Setup Manager program for this purpose, but you can also use a standard text editor and create the unattended text file "by hand".

In the directory *\SAMPLES\NT4WS* on the BootManage® TCP/IP BOOT-PROM Utility Disk, there is a sample *unattend.txt* file which already contains the options that we use in this configuration example.

Copy this *unattend.txt* file to the *c:\clients\winnt* directory of your installation server and modify it to suit your needs. The contents of the unattend.txt file we use in this example is listed in *"Windows NT Setup Answer File (unattend.txt)"* on page 62.

Step 9: Create The Boot Image File

Copy the file *bmimage.exe* from the BootManage® TCP/IP BOOT-PROM Utility Disk to the *%SystemRoot%* directory of the Windows NT Server:

```
C:\> copy a:\bmimage\bmimage.exe %SystemRoot%
```

On the installation server, create the directory *c:\tftpboot* and copy the file *bpboot* from the BootManage® TCP/IP BOOT-PROM Utility Disk to this directory:

```
C:\> mkdir tftpboot
C:\> cd tftpboot
C:\tftpboot> copy a:\bpboot\bpboot .
```

Insert the network installation diskette and use the `BMIMAGE` program to convert this diskette into a boot image file. Name this boot image file *bpboot.X* and place it in the *c:\tftpboot* directory.

```
C:\tftpboot> bmimage -d bpboot.X -F 2880,a:
C:\tftpboot> bmimage -d bpboot.X -i a:\
```

Verify that all of the files from the network installation diskette have been transferred to the boot image.

```
C:\tftpboot> bmimage -d bpboot.X -D
```

Reserve some extra space in the boot image, so that we can copy additional files to it after downloading.

```
C:\tftpboot> bmimage -d bpboot.X -P50
```

Step 10: Install BOOTP/TFTP Servers

On the installation server, create the directory *C:\ETC*.

```
C:\tftpboot> cd \
C:\> mkdir etc
C:\> cd etc
```

From the BootManage® TCP/IP BOOT-PROM Utility Diskette, copy the files *BOOTPD32.EXE* and *TFTPD32.EXE* and also the sample *bootptab* file to the *ETC* directory[1]:

```
C:\etc> copy a:\servers\bootpd32.exe .
C:\etc> copy a:\servers\tftpd32.exe .
C:\etc> copy a:\samples\nt4ws\bootptab .
```

Open the *bootptab* file with a text editor and replace the sample address 00.11.22.aa.bb.cc with the hardware (ethernet or MAC) address of your network adapter. Then, save the modified *bootptab* file[2].

The contents of the bootptab file we use in this example is listed in *"BOOTP Server Configuration File (bootptab)"* on page 61.

You may want to make additional changes customized for your site. The *bootptab* file contains comments that show you the meaning of all custom options.

Step 11: Start BOOTP/TFTP Servers

This sample and discussion assumes that you are using `BOOTPD32` and not the Windows NT DHCP server. In fact, we strongly recommend that for testing and learning purposes, you use `BOOTPD32` instead of DHCP to get started. Therefore, please make sure that the Microsoft DHCP Server is disabled.

---

1 If the BOOTP and TFTP server programs are stored in a compressed archive on the diskette. In this case, you can extract the files from the archive directly to the target directory on the hard disk.

2 The Windows NT text editor (NotePad) tends to add the extension .txt to files when saving them. When using NotePad, make sure that the file is saved as bootptab and not as bootptab.txt!

To start BOOTPD32 and TFTPD32, the server startup batch file *servers.bat* is also provided on the BootManage® TCP/IP BOOT-PROM Utility Diskette. Copy and execute this batch file as follows:

```
C:\etc> copy samples\nt4ws\servers.cmd .
C:\etc> servers
```

The servers.bat file contains the following commands:

```
start bootpd32 -cmd -d -d -d -d -i 10.0.0.1/255.0.0.0 -s -t 0
start tftpd32 -cmd -v 2 -d c:\tftpboot -r
```

Both servers will open an application window that displays status and debug messages. Any BOOTP and TFTP requests and replies will be logged here.

Step 12: Install The Client PC

Turn on the client PC and wait for the TCP/IP BOOT-PROM code to come-up with its copyright message. The TCP/IP BOOT-PROM code should download the file *bpboot*, which is the BootManage® TCP/IP BOOT-PROM boot loader. This boot loader, in turn, should download the boot image file *bpboot.X* and start the automated installation.

The PC will automatically reboot multiple times until the installation is complete.

Scheduling An Automated Reinstallation

After the client PC has been installed, it is possible to schedule a completely automated reinstallation at any time.

User Initiated Reinstallation

In our example, the user is allowed to request a complete reinstallation of the client PC at boot time. Driven by the BpDiS=03 keyword, the BPBOOT boot loader displays the message `Press <SPACE> to start installation services` for 3 seconds at boot time. If the user presses the space bar at this time, BPBOOT downloads the boot image (instead of booting from the hard disk) and completely reinstalls Windows NT on the client PC.

To prevent users from requesting a reinstallation, simply place a hash (#) sign at the beginning of the BpDiS line within the bootptab as follows:

```
# Enable user initiated reinstallation
# :T130="BpDiS=03":\
# end of client record
:
```

Administrator Initiated Reinstallation

The administrator can easily schedule a reinstallation of a client PC by simply "toggling" the BPBOOT condition string in tag 129 between the two preconfigured values.

```
# BPBOOT condition string (toggle for client reinstallation)
:T129=4270426f4f7430f210:\
#:T129=4270426f4f7430e210:\
```

To schedule a reinstallation, place a hash (#) sign at the beginning of the first line, and remove the hash sign from the second line as follows:

```
# BPBOOT condition string (toggle for client reinstallation)
#:T129=4270426f4f7430f210:\
:T129=4270426f4f7430e210:\
```

To again schedule a reinstallation for this PC at a later time, simply revert the changes, so that the BPBOOT condition string has its original value.

Additional Information

Although completely operational, this sample configuration is intended to be used as a starting point for designing your own unattended installation environments.

Installing Multiple Clients

You can install multiple Windows NT Workstation 4.0 client machines by simply adding a host entry for each additional client in the *bootptab* file. For this purpose, you may want to copy and paste the nt4ws1 host entry, rename it to nt4ws2 and adapt the individual tags in the new entry.

Clients With Different Network Adapters

You can use the same boot image to install multiple clients that have different network adapters. For this purpose, use an additional tag that represents the real mode name of the network card (in our example this is E100B). Insert this tag into the *\NET\PROTOCOL.INI* and *\NET\SYSTEM.INI* files and be sure to copy all possible NDIS2 real mode drivers (files ending in "*.DOS*") to the *\NET* directory. You can now set the real mode driver using a tag defined in the bootptab file.

Installing Windows NT Server 4.0 Clients

In addition to installing Windows NT Workstation 4.0 clients, you can also automatically install Windows NT Server 4.0 clients by using the same boot image. All you need to do is to:

- copy the Windows NT Server 4.0 distribution files to the installation server (you may want to use the Microsoft Network Client Administrator to do this).
- copy the INSTALL.BAT and POSTINST.BAT files to the *c:\clients\winnt.srv* directory.
- create an unattend.txt file for automated deployment of Windows NT Server 4.0 and place it in the *c:\clients\winnt.srv* directory.
- add a new template entry (nt4s-common) to the bootptab file, based on the nt4w-common template. In the new template, change the "Installation server path (OS type)" tag T156 to the value "WINNT.SRV".
- add a Windows NT Server 4.0 host entry in the bootptab file which includes the nt4s-common template and test the installation.

Installing Windows 95 And Windows 98 Clients

For installing Windows 95 and 98 clients, you must make additional modifications, but the basic method introduced here remains the same.

Windows 95 and 98 use *SETUP.EXE* instead of *WINNT.EXE* to install the operating system. *SETUP.EXE* has different command line switches, and you also need a different *unattend.txt* file. See the Windows 95 and Windows 98 Resource Kits, respectively, for details.

Using Different Transport Protocols

The real mode network connection can be made using different transport protocols. For example, you could use NetBEUI or IPX/SPX to connect to the server and download the installation files and then use TCP/IP for the remote booting aspect. These protocols are selectable when setting-up the network client boot diskette using the Network Client Administrator.

Using Different Installation Servers

The installation server need not be a Windows NT Server. The only requirement is the capability to connect to a file server via a redirected network drive using the real mode boot image. As such, one is free to use NetWare file servers or Linux systems running the Samba SMB server as their installation server.

Installing From An NFS Server

It is also possible to use NFS instead of a DOS-style redirected network drive to download the installation files from an installation server. The real mode TCP/IP

stack and NFS client software, BPFS, has been certified to work correctly with the installation instructions specified here. Others may work, as well.

Configuration And Batch Files Reference

Here, you find the contents of all configuration and batch files that are used throughout this step-by-step example. Note that all these files are available in the *SAMPLES\NT4WS* folder of the TCP/IP BOOT-PROM Utility Diskette, so you can easily copy them and use them as a starting point for your OS deployment environment.

Files In The Boot Image

The following configuration and batch files are used within the boot image:

A:\CONFIG.SYS

```
dos=high,umb
files=100
lastdrive=z

device=a:\bin\bputil.sys -f
device=a:\net\himem.sys /testmem:off
device=a:\net\emm386.exe noems
device=a:\bin\bputil.sys -x

devicehigh=a:\net\ifshlp.sys
```

In CONFIG.SYS, it is **critically important** to specify `/testmem:off` for *HIMEM.SYS*. Otherwise, *HIMEM.SYS* performs a destructive RAM check that overwrites the RAM area in which the downloaded boot image resides.

The Microsoft Network Client for DOS components occupy a lot of conventional memory. Using *EMM386.EXE* allows us to "load high" some drivers and programs into the UMB area. Depending on your PC's hardware configuration, you may have to instruct *EMM386.EXE* to include and/or exclude UMB memory areas.

A:\AUTOEXEC.BAT

```
@echo off
prompt $p$g
set PATH=a:\;a:\bin;a:\net

rem patch the system preparation and network batch files
bputil -a a:\prepare.bat
bputil -a a:\network.bat

rem execute the (patched) system preparation batch file
prepare.bat
```

The sole purpose of the *AUTOEXEC.BAT* file is to patch and execute other batch files. *AUTOEXEC.BAT* itself does not contain any tags, so it does not need to be patched before execution.

A:\PREPARE.BAT

```
rem check if user requested reinstallation by keypress
if not _#@T254*##### == _BpKeY goto CHECK_ID
cls
echo User requested reinstallation at boot time
goto REINSTALL


rem examine status partition ID to determine installation phase
:CHECK_ID

rem if ID is f0/e0, then hard disk is already partitioned and formatted
bmfdisk -c 80 0 f0
if ERRORLEVEL 1 goto OS_INSTALL
bmfdisk -c 80 0 e0
if ERRORLEVEL 1 goto OS_INSTALL

rem if ID is f1/e1, then the files are already copied to the hard disk
bmfdisk -c 80 0 f1
if ERRORLEVEL 1 goto POST_INSTALL
bmfdisk -c 80 0 e1
if ERRORLEVEL 1 goto POST_INSTALL


rem in all other cases, perform reinstallation
:REINSTALL
echo.
echo Press a key to clean the hard disk and restart installation.
pause

rem clear all partition table entries
bmfdisk -m 80 0 N 00 c -f
bmfdisk -m 80 1 N 00 c -f
bmfdisk -m 80 2 N 00 c -f
bmfdisk -m 80 3 N 00 c -f

rem store BPBOOT condition string in environment variable T129
bputil -s T129 > a:\cond_str.bat
call a:\cond_str.bat
del a:\cond_str.bat

rem create status partition in slot 0
rem determine ID from BPBOOT condition string (f0 or e0)
if "%T129%" == "[9]4270426f4f7430f210" bmfdisk -m 80 0 N f0 8m -f
if "%T129%" == "[9]4270426f4f7430e210" bmfdisk -m 80 0 N e0 8m -f

rem check if a status partition was created
bmfdisk -c 80 0 0
```

```
if ERRORLEVEL 1 goto COND_INVALID

rem create an active FAT16 partition in slot 1
bmfdisk -m 80 1 y 06 #@T158*##m -f

rem quick format FAT16 partition
bmfdisk -q 80 1 fat16 -f

rem write master boot record
bmfdisk -b 80 -f

rem reboot PC to recognize partition table changes
reboot


:OS_INSTALL
cls
echo Connect network drive to installation server
echo.
call network.bat

rem create temporary directory
mkdir c:\temp

rem copy and patch operating system specific install.bat file
copy w:\#@T156*#####\install.bat a:\
bputil -a a:\install.bat
copy a:\install.bat c:\temp\install.bat
del a:\install.bat

rem execute install.bat file, this call does not return
c:\temp\install.bat
goto FILE_NOT_EXIST


:POST_INSTALL
cls
echo Connect network drive to installation server
echo.
call network.bat

rem copy and patch operating system specific postinst.bat file
copy w:\#@T156*#####\postinst.bat a:\
bputil -a a:\postinst.bat
copy a:\postinst.bat c:\temp\postinst.bat
del a:\postinst.bat

rem execute postinst.bat file, this call does not return
c:\temp\postinst.bat
goto FILE_NOT_EXIST


:COND_INVALID
rem invalid condition string, display warning and abort
```

```
echo WARNING: Invalid BPBOOT condition string - check tag/option 129!
goto END


:FILE_NOT_EXIST
rem configuration file does not exist, display warning and abort
echo WARNING: Configuration file does not exist!

:END
```

The *PREPARE.BAT* file handles the "preparation" part of the operating system installation, that is, hard disk partitioning and formatting. The `BMFDISK` utility is used to detect the current installation state and to branch to the corresponding section. *PREPARE.BAT* is patched before execution (from within the *AUTOEXEC.BAT*), so the generic placeholders are replaced with individual information for the client PC.

A:\NETWORK.BAT

```
rem initialize TCP/IP stack
bputil -p a:\net\protocol.ini
bputil -a a:\net\system.ini
bputil -a a:\net\lmhosts
net initialize
netbind
umb
tcptsr
tinyrfc
nmtsr
emsbfr

rem logon to installation server
net logon #@T140*############## #@T141*############## /savepw:no /y
rem connect w: to general installation share
net use w: \\#@T153*##########\#@T145*##########
```

The *NETWORK.BAT* file handles everything that is needed to load the Microsoft Network Client for DOS components and to connect a network drive to the installation server. *NETWORK.BAT* is patched before execution (from within the *AUTOEXEC.BAT*), so the generic placeholders are replaced with individual information for the client PC.

A:\NET\PROTOCOL.INI

```
[network.setup]
version=0x3110
netcard=ms$e100b,1,ms$e100b,1
transport=tcpip,TCPIP
lana0=ms$e100b,1,tcpip

[ms$e100b]
drivername=e100b$
```

```
[protman]
drivername=PROTMAN$
PRIORITY=MS$NDISHLP

[tcpip]
NBSessions=6
DefaultGateway0=#@gw0##########
SubNetMask0=#@smf##########
IPAddress0=#@yip##########
DisableDHCP=1
DriverName=TCPIP$
BINDINGS=ms$e100b
LANABASE=0
```

The client's IP address and subnet mask are replaced by standard BOOTP/DHCP options so that the same boot image can be used for multiple TCP/IP BOOT-PROM clients.

A:\NET\SYSTEM.INI

```
[network]
filesharing=no
printsharing=no
autologon=yes
computername=#@T150###########
lanroot=A:\NET
username=install
workgroup=#@T155##########
reconnect=no
dospophotkey=N
lmlogon=0
logondomain=
preferredredir=basic
autostart=basic
maxconnections=8

[network drivers]
netcard=e100b.dos
transport=tcpdrv.dos,nemm.dos
devdir=A:\NET
LoadRMDrivers=yes
```

As in *PROTOCOL.INI*, we replace individual settings with BOOTP/DHCP options.

Batch Files On The Installation Server

INSTALL.BAT

```
rem cd to operating system specific directory
w:
cd \#@T156*#####
```

```
rem load SmartDrive to speed things up
lh tools\smartdrv 32768 > NUL:
lh tools\smartdrv 16384 > NUL:
lh tools\smartdrv  8192 > NUL:

rem copy and patch unattend.txt file
copy unattend.txt a:\
bputil -a a:\unattend.txt
copy a:\unattend.txt c:\temp\unattend.txt
del a:\unattend.txt

rem cd to installation directory
cd netsetup

rem flag that we are done with the OS installation
bmfdisk -a 80 0 1 -f

rem run Windows NT installation, this program does not return
winnt /b /s:. /u:c:\temp\unattend.txt

:END
```

The *INSTALL.BAT* file handles the operating system specific part of an automated OS installation. *INSTALL.BAT* is patched and executed from within *PREPARE.BAT*. As the last entry in the *INSTALL.BAT* file, the operating system installer executable is launched, and the setup answer file is passed to it.

POSTINST.BAT

```
rem check if OS installation was successful
rem in this example, we only perform a basic check
if not exist c:\$WIN_NT$.~LS\*.* goto INSTALL_ERROR

rem copy additional files to client PC

rem add commands to customize OS installation

rem flag that we are done with the OS installation
bmfdisk -a 80 0 1 -f

rem reboot the PC, the next boot will then be local
reboot


:INSTALL_ERROR
rem the OS installation was not successful
echo The OS installation was not successful!

:END
```

The *POSTINST.BAT* file allows to perform additional actions after the operating system installer has copied the OS installation files to the local hard disk. In this example, we only perform a very simple installation check, but please feel free to

modify the *POSTINST.BAT* file to your liking (implement a detailed installation check, add additional components, etc.). In the *POSTINST.BAT* file, the ID value of the status partition is incremented, so that the next boot will be from the local hard disk.

BOOTP Server Configuration File (bootptab)

```
# global parameters for all clients

bootmanage:\
   :hn:vm=rfc1048:ht=ethernet:ms=1024:\
   :hd="c:/tftpboot":bf=bpboot:\
   :gw=10.0.0.254:\
   :sm=255.0.0.0:


# common entries for all NT4 Workstation clients

nt4w-common:\
   # Include global settings
   :tc=bootmanage:\
   # Installation server name
   :T153="NT4SERVER":\
   # Installation server share
   :T145="CLIENTS":\
   # Installation server path (OS type)
   :T156="WINNT":\
   # Installation pseudo user name
   :T140="instuser":\
   # Password for installation pseudo user
   :T141="instpass":\
   # Windows NT licensing information - User name
   :T151="BootManage User":\
   # Windows NT licensing information - Company name
   :T152="ACME Corporation":\
   # Windows NT licensing information - License key
   :T157="000-1234567":\
   # Workgroup to join
   :T155="BOOTMANAGE":


# entries for individual NT4 Workstation clients

nt4ws1:\
   # Include settings for all NT4 Workstation clients
   :tc=nt4w-common:\
   # Hardware (MAC) and IP Address
   :ha=00.11.22.aa.bb.cc:ip=10.0.1.1:\
   # Computer NetBIOS name
   :T150="NT4WS1":\

   # BPBOOT condition string (toggle for client reinstallation)
```

```
:T129=4270426f4f7430f210:\
#:T129=4270426f4f7430e210:\

# Size of hard disk system partition in Megabytes
:T158="800":\

# Filesystem type, either ConvertNTFS (NTFS) or LeaveAlone (FAT16)
:T159="ConvertNTFS":\
#:T159="ConvertNTFS":\

# Extend partition during NTFS conversion
:T160="0":\
#:T160="1, nowait":\

# Enable user initiated reinstallation
:T130="BpDiS=04":\
# Set debugging level for BPBOOT
:T131="BpDbG=00":\

# end of client record
:
```

To support additional Windows NT Clients, simply add a record for each client, using the "nt4ws1" record as a template. Make sure that each client record has a unique name (e.g. nt4ws2, nt4ws3, ...) and a unique IP address. Change the value of the ha tag to reflect the client's hardware (MAC) address. Also, note that in our example, the client's NetBIOS name is defined in tag number 150.

Windows NT Setup Answer File (unattend.txt)

```
[OEM_Ads]
Banner = "BootManage TCP/IP BOOT-PROM*Setup For Windows NT 4.0"

[Unattended]
OemSkipEula = yes
OemPreinstall = yes
NoWaitAfterTextMode = 1
NoWaitAfterGUIMode = 1
FileSystem = #@T159*########################################
ExtendOEMPartition = #@T160######
AutoPartition = 1
ConfirmHardware = no
NtUpgrade = no
Win31Upgrade = no
TargetPath = WINNT
OverwriteOemFilesOnUpgrade = no
KeyboardLayout = "US"

[UserData]
FullName =      "#@T151*################################################"
OrgName =       "#@T152*################################################"
ComputerName = "#@T150*################################################"
ProductId =     "#@T157*################################################"
```

```
[GuiUnattended]
OemSkipWelcome = 1
OEMBlankAdminPassword = 1
TimeZone = "(GMT-06:00) Central Time (US & Canada)"

[Display]
ConfigureAtLogon = 0
BitsPerPel = 8
XResolution = 640
YResolution = 480
VRefresh = 60
AutoConfirm = 1

[Network]
DetectAdapters = ""
InstallProtocols = ProtocolsSection
InstallServices = ServicesSection
JoinWorkgroup = "#@T155*##############################################"

[ProtocolsSection]
TC = TCParamSection

[TCParamSection]
DHCP = no
IPAddress = #@yip*############################
Subnet =    #@smf*############################
Gateway =   #@gw0*############################

[ServicesSection]
```

In our example, we let the Windows NT setup processor autodetect the video and network adapters. For most modern adapters, Windows NT does not ship with appropriate drivers, and so its built-in detection may fail. One means of adding third-party drivers to an automated Windows NT installation is to use the `BMPCSCAN` utility. For more information about `BMPCSCAN`, please refer to *"The BMPCSCAN Program"* on page 197.

Windows 2000 Step-By-Step

This chapter contains a step-by-step example of how to perform a completely automated installation of the Windows 2000 Professional OS (US version) on a client PC using the BootManage® TCP/IP BOOT-PROM and related utilities.

This example is very similar to the automated Windows NT 4.0 Workstation installation we showed in the previous chapter. Therefore, we will rely on this previous chapter and only show you what has to be changed for Windows 2000.

If you have already setup the sample automated Windows NT 4.0 Workstation installation described in the previous chapter, you will find that adding support for Windows 2000 clients can be done almost at a finger's snap.

Environment

This is exactly the same environment as in the "Windows NT 4 Step-By-Step" chapter, so please refer to *"Environment"* on page 43.

Installation Overview

The automated installation of Windows 2000 works in exactly the same way as the automated installation of Windows NT Workstation 4.0. We use exactly the same boot image file, and the sequence of the installation process is also the same. For an outline of the concept, please refer to *"Installation Overview"* on page 44.

We do not address licensing issues here. Please be sure that you have purchased the appropriate number of operating system client licenses when installing multiple clients over the network.

Step 1: Prepare The Installation Server

Our Windows NT Server 4.0, named NT4SERVER, is member of the workgroup BOOTMANAGE. Attached to a class A IP network, it has the IP address 10.0.0.1 and netmask 255.0.0.0.

Create a shared network installation directory on the server and copy all of the files needed for the client installation to this directory. For this purpose, Microsoft provides the Network Client Administrator that is included with the Windows NT Server 4.0 operating system. Follow the instructions in the chapter entitled *"Microsoft Network Client Administrator"* on page 73.

Although the Microsoft Network Client Administrator does not come with built-in support for Windows 2000 client installations, but it can be adapted quite easily. For details, see *"Windows 2000 Support"* on page 78.

Step 2: Add Files To Installation Server

For our sample installation, we need to have the DOS *SMARTDRV.EXE* program in the *c:\clients\w2kpro\tools* directory on the installation server. Therefore, create this directory on the installation server now and copy the *SMARTDRV.EXE* program from an MS-DOS system into it.

You may want to add additional software packages to the automated installation, e.g. a Windows 2000 Service Pack, third-party device drivers or a set of standard applications which should be installed together with the operating system. For detailed information about how to add the `BMDRV` device driver and the `BMUTIL32` service, please refer to *"The BMDRV Device Driver"* on page 229 and *"The BMUTIL32 Program"* on page 237.

It is not necessary for our automated installation to add software components, but in case you want to do so, see Chapter 5 of the Windows 2000 Professional Resource Kit for instructions on how to do this.

You may want to pay special attention to the following Microsoft Knowledge Base articles:

- Q271791: How to Integrate Service Pack 1 into a Windows 2000 Installation
- Q263125: SP1 Upgrade Does Not Update Recovery Console Files
- Q254078: How to Add OEM Plug and Play Drivers to Windows 2000

Step 3: Create Network Client Boot Disk

See *"Step 3: Create Network Client Boot Disk"* on page 47.

Step 4: Test Network Client Boot Disk

See *"Step 4: Test Network Client Boot Disk"* on page 47.

Step 5: Add Files To Boot Disk

See *"Step 5: Add Files To Boot Disk"* on page 47.

Step 6: Modify Configuration Files

See *"Step 6: Modify Configuration Files"* on page 49.

Step 7: The Installation Batch Files

Connecting a network drive to the installation server allows to access installation batch files that are located on the installation server. Two files are used in this manner:

During the OS_INSTALL phase, the *INSTALL.BAT* file is responsible for copying the operating system files to the client's hard disk.

During the POST_INSTALL phase, the *POSTINST.BAT* file is responsible for verifying that the OS installation was successful. It also allows to copy additional files to the client's hard disk and to further customize the installation process.

Copy the files *INSTALL.BAT* and *POSTINST.BAT* from the *\SAMPLES\W2KPRO* folder of the TCP/IP BOOT-PROM Utility Diskette to the *c:\clients\w2kpro* folder on your installation server.

To see the contents of the configuration files we use in this example, please refer to *"Batch Files On The Installation Server"* on page 69.

Step 8: The unattend.txt File

The *WINNT.EXE* program is used to install the Windows 2000 Professional operating system on a computer running DOS. Using the /u command line option, *WINNT.EXE* derives all setup information from an unattended text file (also called an answer file) instead of interactively querying the user.

See Chapter 5 and Appendix B of the Microsoft Windows 2000 Professional Resource Kit for instructions on how to create an unattended text file. For this purpose, Microsoft provides the Windows 2000 Setup Manager program which is located in the cabinet file *\support\tools\deploy.cab* on the Windows 2000 Professional CD-ROM. The Microsoft Knowledge Base article Q250485 "Setupmgx.dll Could Not Be Loaded or Corrupt Err Msg" contains a detailed description of how to extract and start the Windows 2000 Setup Manager. However, you can also use a standard text editor and create the unattended text file "by hand".

In the directory *\SAMPLES\W2KPRO* on the BootManage® TCP/IP BOOT-PROM Utility Disk, there is a sample *unattend.txt* file which already contains the options that we use in this configuration example.

Copy this *unattend.txt* file to the *c:\clients\w2kpro* directory of your installation server and modify it to suit your needs. The contents of the *unattend.txt* file we use in this example is listed in *"Windows 2000 Setup Answer File (unattend.txt)"* on page 70.

Step 9: Create The Boot Image File

See *"Step 9: Create The Boot Image File"* on page 50.

Step 10: Install BOOTP/TFTP Servers

See *"Step 10: Install BOOTP/TFTP Servers"* on page 51, but use the *bootptab* file from the *samples\w2kpro* directory.

It is easily possible to merge the contents of the Windows NT 4.0 and Windows 2000 *bootptab* files into one single *bootptab* file. This way, you can support Windows NT 4.0 and Windows 2000 clients using the same BOOTP server.

Step 11: Start BOOTP/TFTP Servers

See *"Step 11: Start BOOTP/TFTP Servers"* on page 51.

Step 12: Install The Client PC

See *"Step 12: Install The Client PC"* on page 52.

Configuration And Batch Files Reference

Here, you find the contents of all configuration and batch files that are used throughout this step-by-step example. Note that all these files are available in the *samples\w2kpro* folder of the TCP/IP BOOT-PROM Utility Diskette, so you can easily copy and use them as a starting point for your OS deployment environment.

Files In The Boot Image

The boot image we use for the automated Windows 2000 Professional installation is exactly the same as the boot image for the Windows NT Workstation 4.0 installation. To see the contents of these files, see *"Files In The Boot Image"* on page 55.

Batch Files On The Installation Server

The *INSTALL.BAT* and *POSTINST.BAT* files we use for this Windows 2000 Professional installation are almost identical to the corresponding files used for the Windows NT 4.0 Workstation installation. The only difference is at the end of the *INSTALL.BAT* file, where the Windows 2000 setup program, *WINNT.EXE*, is called without the `/b` switch:

```
rem run Windows 2000 installation, this program does not return
winnt /s:. /u:c:\temp\unattend.txt
```

To see the contents of these files, see *"Batch Files On The Installation Server"* on page 59.

BOOTP Server Configuration File (bootptab)

```
# global parameters for all clients

bootmanage:\
   :hn:vm=rfc1048:ht=ethernet:ms=1024:\
   :hd="c:/tftpboot":bf=bpboot:\
   :gw=10.0.0.254:\
   :sm=255.0.0.0:


# common entries for all Windows 2000 Professional clients

w2kpro-common:\
   # Include global settings
   :tc=bootmanage:\
   # Installation server name
   :T153="NT4SERVER":\
   # Installation server share
   :T145="CLIENTS":\
   # Installation server path (OS type)
   :T156="W2KPRO":\
```

```
   # Installation pseudo user name
   :T140="instuser":\
   # Password for installation pseudo user
   :T141="instpass":\
   # Windows 2000 licensing information - User name
   :T151="BootManage User":\
   # Windows 2000 licensing information - Company name
   :T152="ACME Corporation":\
   # Windows 2000 licensing information - License key
   :T157="AAAAA-BBBBB-CCCCC-DDDDD-EEEEE":\
   # Workgroup to join
   :T155="BOOTMANAGE":


# entries for individual Windows 2000 Professional clients

w2kpro1:\
   # Include settings for all Windows 2000 Professional clients
   :tc=w2kpro-common:\
   # Hardware (MAC) and IP Address
   :ha=00.11.22.aa.bb.cc:ip=10.0.2.1:\
   # Computer NetBIOS name
   :T150="W2KPRO1":\

   # BPBOOT condition string (toggle for client reinstallation)
   :T129=4270426f4f7430f210:\
   #:T129=4270426f4f7430e210:\

   # Size of hard disk system partition in Megabytes
   :T158="1200":\

   # Filesystem type, either ConvertNTFS (NTFS) or LeaveAlone (FAT16)
   :T159="ConvertNTFS":\
   #:T159="LeaveAlone":\

   # Extend partition during NTFS conversion
   :T160="0":\
   #:T160="1":\

   # Enable user initiated reinstallation
   :T130="BpDiS=04":\
   # Set debugging level for BPBOOT
   :T131="BpDbG=00":\

   # end of client record
   :
```

Windows 2000 Setup Answer File (unattend.txt)

```
[Data]
AutoPartition=1
MsDosInitiated="0"
UnattendedInstall="Yes"
```

```
[Unattended]
UnattendMode=FullUnattended
OemSkipEula=Yes
OemPreinstall=Yes
TargetPath=\WINNT
FileSystem = #@T159*##############
ExtendOEMPartition = #@T160#

[GuiUnattended]
AdminPassword=AdminPassword
OEMSkipRegional=1
TimeZone=4
OemSkipWelcome=1

[UserData]
FullName =      "#@T151*################################################"
OrgName =       "#@T152*################################################"
ComputerName = "#@T150*################################################"
ProductId =     "#@T157*################################################"

[Identification]
JoinWorkgroup = "#@T155*################################################"

[Networking]
InstallDefaultComponents=No

[NetAdapters]
Adapter1=params.Adapter1

[params.Adapter1]
INFID=*

[NetClients]
MS_MSClient=params.MS_MSClient

[NetServices]
MS_SERVER=params.MS_SERVER

[NetProtocols]
MS_TCPIP=params.MS_TCPIP

[params.MS_TCPIP]
DNS=No
UseDomainNameDevolution=No
EnableLMHosts=Yes
AdapterSections=params.MS_TCPIP.Adapter1

[params.MS_TCPIP.Adapter1]
SpecificTo=Adapter1
DHCP=No
IPAddress =       #@yip*############################
SubnetMask =      #@smf*############################
DefaultGateway = #@gw0*############################
WINS=No
```

```
NetBIOSOptions=0

[RegionalSettings]
LanguageGroup=1
```

Microsoft Network Client Administrator

The Microsoft Windows NT 4.0 Server includes a utility called the Network Client Administrator. The Network Client Administrator configures an NT Server as a distribution point for over-the-network installation of various client operating systems such as Windows NT Workstation 4.0 and Windows 95. Using the Network Client Administrator, all the operating system installation files are copied to a directory on the server and shared for use by clients on the network. The Network Client Administrator also creates over-the-network installation diskettes that can be easily transferred to boot images through the BootManage® TCP/IP BOOTPROM utilities.

This chapter provides a short tutorial on how to use the Network Client Administrator to setup distribution points and network installation diskettes for Windows NT Workstation 4.0, Windows NT Server, and Windows 95. This will ultimately allow the construction of powerful, timesaving mechanisms to perform unattended installations of these operating systems.

Starting Network Client Administrator

On your Windows NT Server 4.0, you can start the Network Client Administrator program by clicking Start ➜ Programs ➜ Administrative Tools (Common) ➜ Network Client Administrator. The program comes up as shown in *Figure 7-1*.

Select Make Network Installation Startup Disk and then click Continue. Complete the next dialog box as shown in *Figure 7-2* and then click OK (In our example, D: is the CD-ROM drive that holds the Windows NT Server 4.0 CD-ROM). The Network Client Administrator now creates the directory *C:\CLIENTS* and copies all of the available client files to it.

Next, the Network Client Administrator asks you to specify the operating system type for the target workstations, as displayed in *Figure 7-3*.

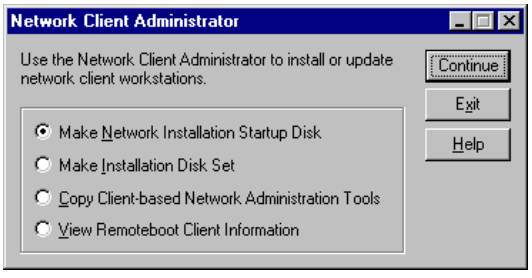


Figure 7-1. The Network Client Administrator

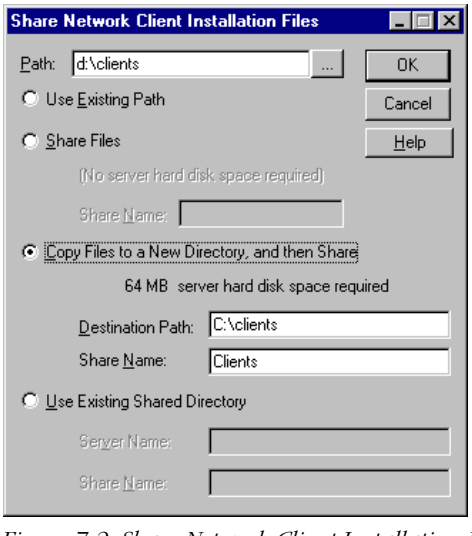


Figure 7-2. Share Network Client Installation Files

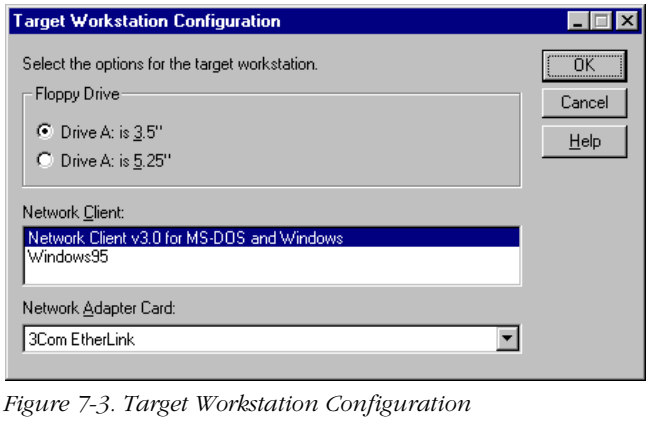


Figure 7-3. Target Workstation Configuration

Adding Windows NT Entries

At this time, the Network Client field only lets you choose between Network Client v3.0 for MS-DOS and Windows, and Windows 95. Since we would like to have Windows NT Workstation 4.0 and Windows NT Server 4.0 available, too, select CANCEL and terminate the Network Client Administrator at this time.

To add an entry for Windows NT Workstation 4.0, insert the Windows NT Workstation 4.0 CD-ROM and execute:

```
xcopy /s /i d:\i386 c:\clients\winnt\netsetup
```

To add an entry for Windows NT Server 4.0, insert the Windows NT Server 4.0 CD-ROM and execute:

```
xcopy /s /i d:\i386 c:\clients\winnt.srv\netsetup
```

For more information about supporting additional operating system clients in Network Client Administrator, read the text file *\CLIENTS\SUPPORT\README.TXT* on the Windows NT Server 4.0 CD-ROM. Also, see *"Adding Support For Windows 2000 Clients"* on page 79.

Creating An Installation Diskette

Again, start the Network Client Administrator and click OK in every dialog (do not change any settings) until you reach the Target Workstation Configuration dialog. This time, it should look like *Figure 7-4*.

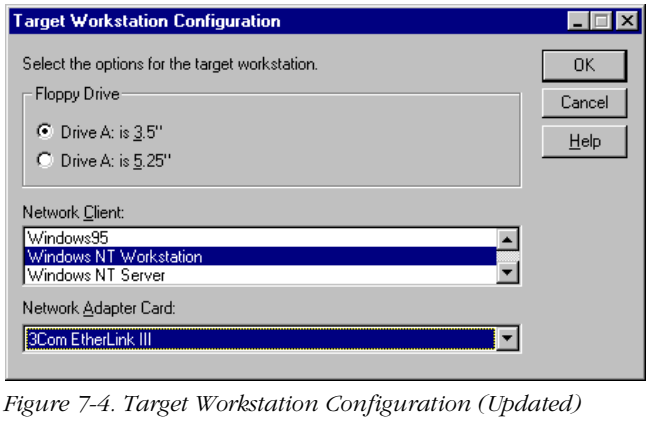


Figure 7-4. Target Workstation Configuration (Updated)

In the Network Client box, select the operating system type you wish to install on the client. For demonstration purposes, we will continue as if you had selected

Windows NT Workstation, but the procedure for other choices is essentially no different. In the Network Adapter Card drop-down box, you can choose which network driver will be copied to the installation diskette. If the network card you use in the client PC is not listed here (e.g. Intel Pro/100), just select 3Com EtherLink III at this time. Later, you can manually change the adapter type to suit your needs. Click OK, read the license screen, and then advance to the Network Startup Disk Configuration dialog, which is displayed in *Figure 7-5*.

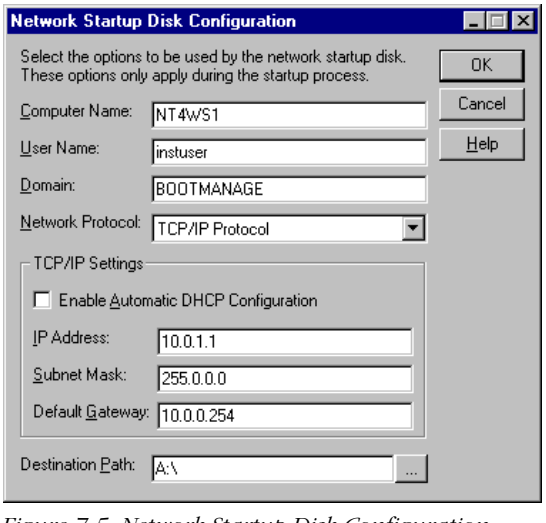


Figure 7-5. Network Startup Disk Configuration

The name of our client PC is NT4WS1, and it belongs to the workgroup (or Windows NT domain) named BOOTMANAGE. During unattended setup, the client uses the TCP/IP transport protocol to connect to the installation server and logs-on as user instuser. You may want to choose Enable Automatic DHCP configuration, but in our examples we intentionally refrain from doing so in order to demonstrate the BootManage® TCP/IP BOOT-PROM's patching capabilities.[1]

For testing purposes, we have provided the client's IP address and subnet mask here. Later, we will replace these fixed values with BOOTP/DHCP variables so that multiple clients can be booted using the same boot image.

After clicking OK, you are requested to enter a bootable DOS system disk in drive A:. You can easily create such a disk by executing the following command on a native DOS (preferably MS-DOS 6.22) system[2]:

---

1 If you use a BOOTP server (such as BOOTPD32), you cannot use automatic DHCP configuration here. If you use a DHCP server (such as the Microsoft DHCP Server), you can choose between automatic and manual configuration.

2 You cannot create a DOS system diskette by using the Windows NT disk formatting dialog!

```
FORMAT A: /U /S
```

The Network Client Administrator will copy all of the files to the disk that are necessary for starting the client, connecting to the installation server, and then starting the Windows NT installation. After all of the files have been copied, terminate the Network Client Administrator.

Before proceeding, make absolutely sure that:

- network clients can access the CLIENTS share on the installation server using the username and password assigned for the unattended installation "user". ("intuser" was used above.)
- the directory *C:\CLIENTS\WINNT\NETSETUP* contains all of the Windows NT 4.0 Workstation installation files.
- the directory *C:\CLIENTS\WINNT.SRV\NETSETUP* contains all of the Windows NT 4.0 Server installation files[1].
- the directory *C:\CLIENTS\MSCLIENT* contains all of the Microsoft Network Client v3.0 for MS-DOS files
- the file *C:\CLIENTS\NCADMIN.INF* contains configuration information for the Network Client Administrator program

Replace Real Mode Network Card Driver

While creating the network installation floppy disk using the Network Client Administrator, we chose the 3Com EtherLink III network card because the Intel Pro/100 type was not available. Now, we will make the necessary changes manually:

First, copy the NDIS2 driver file *E100B.DOS* to the subdirectory *NET* of your just-created network install disk. You will find the file *E100B.DOS* on the driver disk that came with your network adapter.

Open the files *NET\PROTOCOL.INI* and *NET\SYSTEM.INI* with a standard text editor (e.g. notepad). Use the text editor's search and replace function to globally replace each occurrence of the string ELNK3 with the string E100B.

1 Only needed if you want to install Windows NT Server clients.

It is possible to integrate additional network drivers in the Microsoft Network Client for DOS distribution point, so that the additional network drivers are listed in the "Target Workstation Configuration" dialog box.

However, this is beyond the scope of this manual. If you want to do this, please read the Microsoft Knowledge Base article Q128800: "How to Provide Additional NDIS2 Drivers for Network Client 3.0".

Test The Installation Floppy Disk

You are now ready to perform your first connection test. If you have not already done so, use the Windows NT User Manager to define the installation user "instuser" who will only need to have read access to the CLIENTS share.

Boot the client computer using the just-created network installation disk. It should ask you for the password of the installation user, connect a network drive to the installation server, and start the Windows NT setup program. This process is controlled by the *AUTOEXEC.BAT* file that was created by the Network Client Administrator during creation of the network installation disk.

Windows 2000 Support

In combination with Windows 2000, you may want to do the following:

- Run Network Client Administrator on a Windows 2000 machine
- Add support for automated installation of Windows 2000 clients in Network Client Administrator

Running On Windows 2000

Microsoft does not ship the Network Client Administrator in any of their Windows 2000 operating system distributions. However, if you have a Windows NT 4.0 Server CD at hand, Microsoft Knowledge Base article Q252448 describes "How to Create an MS-DOS Network Startup Disk in Windows 2000". For convenience, we repeat the contents of this article as follows:

1. On the Windows 2000 system, create a folder named *c:\ncadmin*.
2. Copy the files *ncadmin.ex_*, *ncadmin.hl_* and *ncadmin.cn_* from the I386 folder on the Windows NT Server 4.0 CD-ROM to your *c:\ncadmin* folder.
3. At a command prompt, change to the *c:\ncadmin* folder and expand the three files using the following command:

```
c:\ncadmin> expand -r ncadmin.*
```

4. The compressed files *ncadmin.ex_*, *ncadmin.hl_* and *ncadmin.cn_* are no longer needed and can be deleted:

```
c:\ncadmin> attrib -r ncadmin.??_
c:\ncadmin> del ncadmin.??_
```

5. Start Network Client administrator by executing the *ncadmin.exe* file. For convenience, you may want to create a shortcut to *ncadmin.exe* on the desktop or in the start menu.

```
c:\ncadmin> ncadmin
```

Not that Network Client Administrator needs access to the Windows NT Server 4.0 CD-ROM in order to copy the contents of the CLIENTS directory to the hard disk.

Adding Support For Windows 2000 Clients

In *"Adding Windows NT Entries"* on page 75, you have already seen how one can add support for Windows NT 4.0 Workstation and Server systems to Network Client Administrator. There, you only needed to create a new directory and copy the operating system distribution files to it. But how did the Network Client Administrator know what to display in its dialog boxes, and why did it know how to create a network installation diskette for Windows NT 4.0?

The Network Client Administrator Information File

The answer is: Network Client Administrator reads the information file *ncadmin.inf* which is located in the *CLIENTS* share. From this file, it reads information about the supported client operating systems. Additionally, it checks if the distribution point directories are actually present. A client operating system is only listed in the "Target Workstation Configuration" dialog (see *Figure 7-4* on page 75) if the following two conditions are both met:

- An entry for the client OS must be present in the *ncadmin.inf* file
- The OS's distribution directory (referred to in the *ncadmin.inf* file) must exist.

For Windows NT 4.0 Workstation and Server, the *ncadmin.inf* file already contains configuration entries. So all that had to be done in *"Adding Windows NT Entries"* on page 75 was to create the distribution directory.

Extending The Information File For Windows 2000

To add support for Windows 2000 clients in Network Client Administrator, you must mofify the *ncadmin.inf* file with a text editor. Explaining the syntax of the *ncadmin.inf* file in detail is beyond the scope of this manual, but you can find a sample *ncadmin.inf* file in the *samples\w2kpro* folder of the TCP/IP BOOT-PROM

utility disk. This sample file already contains the additional entries required for Windows 2000 clients.

Copying Windows 2000 Professional Distribution Files

All that remains to be done is to copy the distribution files from the I386 folder on the Windows 2000 Professional CD-ROM to the *c:\clients\w2kpro\netsetup* folder on the installation server. To do this, insert the Windows 2000 Professional CD-ROM and execute:

```
xcopy /s /i d:\i386 c:\clients\w2kpro\netsetup
```

In this example, we assume that the CD-ROM drive is D:.

BOOTP Servers

The BOOTP Standard

The Bootstrap Protocol BOOTP is a simple protocol that allows a booting client to receive configuration information from a network server. This configuration information includes the client's IP address and subnet mask, the IP address of a server host, the name and location of a boot image file, and other network related configuration information.

Before its operating system is loaded, a booting client only knows its network hardware (MAC) address. The client broadcasts a BOOTREQUEST packet containing this MAC address. Every BOOTP Server which receives the BOOTREQUEST packet looks up the MAC address in a database. If a matching entry is found, the BOOTP Server collects all configuration information for the client from its database and sends it in a BOOTREPLY packet back to the client.

From then on, the client knows its IP address, subnet mask and other network information, and may continue with its bootstrap by downloading and executing a boot image file from a TFTP Server. Therefore, the BOOTP protocol is especially useful in remote boot and remote installation environments.

The BOOTP protocol is an open and vendor-independent internet standard defined by the Internet Engineering Task Force (IETF) in a number of Request For Comments (RFC) documents[1]:

Document	Title
RFC 951	Bootstrap Protocol
RFC 1048	BOOTP Vendor Information Extensions
RFC 1084	BOOTP Vendor Information Extensions
RFC 1395	BOOTP Vendor Information Extensions
RFC 1497	BOOTP Vendor Information Extensions
RFC 1542	Clarifications and Extensions for the Bootstrap Protocol

Most UNIX operating systems already contain a BOOTP server as part of the operating system distribution, however this BOOTP server may not always be full-featured. Other server operating systems like Windows NT, NetWare, and OS/2 do not provide a BOOTP server.

The BootManage® BOOTP Server

The BootManage® BOOTP Server is a standards-conforming server implementation of the BOOTP protocol as defined in the IETF RFC document 951, with support for vendor information extensions as defined in RFC's 951, 1048, 1084, 1395, 1497 and 1533. More, the BootManage® BOOTP Server provides a number of custom enhancements which provide extended functionality, especially in combination with the BootManage® TCP/IP BOOT-PROM and the BootManage® PXE Toolkit.

Implementations are available for the following operating system platforms:

- Windows 2000 Professional and Server
- Windows NT 3.x and 4.0 (Workstation and Server)
- Windows 95/98/ME
- Windows 3.x, Windows for Workgroups 3.1x
- NetWare 3.x, 4.x, 5.x
- OS/2

The amount of extended BOOTP Server functionality varies for the different operating systems. Not every extended function is available for every listed operating system.

A Word About Licensing

The BootManage® BOOTP Server is not free software, and you need a license to use it. However, the BootManage® TCP/IP BOOT-PROM and the BootManage® PXE Toolkit products already contain a license for the BootManage® BOOTP Server. If you purchased one of these products, and if you use the BootManage® BOOTP Server in conjunction with any of these products, you do not need a separate license for the BootManage® BOOTP Server.

1 There are more documents available about BOOTP (and DHCP, the successor to BOOTP). For more information about the IETF and RFC documents, point your web browser to http://www.ietf.org.

Server Implementations

To explain the functions of the BootManage® BOOTP Servers, we will use the 32-Bit Windows implementation BOOTPD32 as a reference. Later in this chapter, we will explain the implementation specific issues for the other BootManage® BOOTP Servers.

Windows 95/98/ME/NT/2000

BOOTPD32 is a BOOTP server for Windows 32-Bit platforms which supports the following special features:

- Allows large BOOTREPLY packets.
- Simultaneous operation on multiple network interfaces.
- Supports custom tags that can hold ASCII or binary information.

BOOTPD32 derives configuration information for requesting BOOTP clients from a single text based configuration file, commonly named *bootptab*. Those who have worked with the *bootptab* file on UNIX before will find the file format familiar.

Running BOOTPD32 As Application

First, copy the file *bootpd32.exe* to the local hard disk of a Windows 95/98/ME, Windows NT Workstation/Server, or Windows 2000 Professional/Server machine.

```
copy bootpd32.exe %WINDIR%                  (on Windows 95/98/ME)
copy bootpd32.exe %SystemRoot%\system32     (on Windows NT/2000)
```

BOOTPD32 will look up the port number for the bootps (BOOTP Server) and bootpc (BOOTP Client) ports in the file *%SystemRoot%\system32\drivers\etc\services*. If this entry is not present, BOOTPD32 will use the following defaults:

```
bootps  67/udp  bootp   # bootp server
bootpc  68/udp           # bootp client
```

If you want to start BOOTPD32 as a Win32 application, call it from the command line using the -run option and all other options that suit your needs, e. g.:

```
bootpd32 -run -d -d -d -d -i 195.4.136.167/255.255.255.224 -t 0
```

You must at least specify one network interface on the command line using the -i option. For a detailed list of all supported options, please see *"BOOTPD32 Command Line Options"* on page 84.

On the Windows NT/2000 Server, BOOTPD32 and Microsoft's DHCP Server cannot run simultaneously. This is because both servers use the same TCP/IP port (bootps, UDP 67). Trying to start BOOTPD32 on a machine which is already running Microsoft's DHCP server will result in a "bind error".

Installing BOOTPD32 As A Service

If you want to install BOOTPD32 as a Windows NT/2000 service, use the -install option:

```
bootpd32 -install -i 195.4.136.167/255.255.255.224 -t 0
```

When installed as a service, BOOTPD32 will automatically start at every Windows NT/2000 system startup. Like any other service, you can start and stop BOOTPD32 using the Windows NT/2000 Service Control Manager. In addition, you can control BOOTPD32 from the command line:

```
net start bootpd            (start BOOTPD32 service)
net stop bootpd             (stop BOOTPD32 service)
```

BOOTPD32 Command Line Options

The following arguments can be given to BOOTPD32:

```
bootpd32 -run options           (run as Win32 application)
bootpd32 -install options       (install as Windows NT/2000 service)
bootpd32 -remove                (remove service)
bootpd32 -?                     (display command line syntax)
```

With the -run or -install argument, the following options are provided:

Option	Description
-a	Send BOOTREPLY packet as broadcast
-b	enable database interface
-c *port*	specify client port number
-d	Increase debugging output. This option can be used up to four times.
-h	ignore IP address in BOOTP request and lookup hardware address only.
-i *ipaddr*[/*subnetmask*]	The IP address of the network interface that is to be used with BOOTPD32. You need to add the subnet mask if you are using subnetting. It is possible to enter multiple -i options for multiple interfaces.

Option	Description
`-i` *ipaddr*`[:`*subnetbits*`]`	Same as above but allows one to specify the subnet by number of subnet bits instead of the subnet mask. You need to add the number of subnet bits if you are using subnetting. It is possible to enter multiple `-i` options for multiple interfaces.
`-n`	ignore all requests with non-zero source IP address
`-s`	Enable stand-alone mode (always true).
`-t` *timeout*	terminate `BOOTPD32` after *timeout* minutes of inactivity. Setting this value to 0 causes `BOOTPD32` to never timeout.
`-u`	send BOOTP reply to source IP address if set
`-z`	ignore all BOOTP requests with empty source IP address
configfile	Filename which holds the `BOOTPD32` configuration information (default is *c:\etc\bootptab*).
dumpfile	Filename which `BOOTPD32` uses to dump the in-memory configuration to (default is *c:\etc\bootpd.dmp*).
`@`*argumentsfile*	Read command line arguments from *argumentsfile*.

Network Interfaces

Specify IP Address And Subnet Mask (-i)

`BOOTPD32` needs to know on what network interfaces it should listen for incoming BOOTREQUEST packets. With the `-i` command line option, you must explicitly specify every network interface `BOOTPD32` should use, even if you have only a single network interface.

The following example instructs `BOOTPD32` to use only the network interface 193.141.47.195:

```
bootpd32 -i 193.141.47.195 -t 0
```

To specify multiple network interfaces, simply use multiple `-i` options on the command line as in the following example:

```
bootpd32 -i 193.141.47.195 -i 193.141.48.1 -i 193.141.49.2
```

When specifying only the IP address with the `-i` command line option, `BOOTPD32` assumes the corresponding subnet mask. If you use subnetting, you must also specify subnet information to the `-i` option. This can be done by the explicit subnet mask or the number of subnet bits as in the following examples:

```
bootpd32 -i 193.141.47.195/255.255.255.240   (specify entire subnet mask)
bootpd32 -i 193.141.47.195:5              (specify number of subnet bits)
```

Sending And Receiving Packets

Send BOOTREPLY As Broadcast (-a)

The `-a` command line option instructs `BOOTPD32` to send BOOTREPLY packets as broadcasts. Without this option, `BOOTPD32` sends BOOTREPLY packets directed to the client's MAC address.

Send BOOTREPLY To IP Address (-u)

When the `-u` command line option is given, `BOOTPD32` checks if the client IP address is set in incoming BOOTREQUEST packets. If it is, `BOOTPD32` uses this IP address when sending a BOOTREPLY packet to the client.

Ignore BOOTREQUESTs With Non-Zero IP Address (-n)

When the `-n` command line option is given, `BOOTPD32` checks if the client IP address is set (non-zero) in incoming BOOTREQUEST packets. If it is, `BOOTPD32` simply ignores this BOOTREQUEST packet.

That means, `BOOTPD32` only sends BOOTREPLY packets to clients which use an IP address of 0.0.0.0 in a BOOTREQUEST packet.

Ignore BOOTREQUESTs With Zero IP Address (-z)

When the `-z` command line option is given, `BOOTPD32` checks if the client IP address is 0.0.0.0 in incoming BOOTREQUEST packets. If it is, `BOOTPD32` simply ignores this BOOTREQUEST packet.

That means, `BOOTPD32` only sends BOOTREPLY packets to clients which use a non-zero IP address in a BOOTREQUEST packet.

Ignore IP Address In BOOTREQUESTs (-h)

The `-h` command line option instructs `BOOTPD32` to ignore the client IP address in BOOTREPLY packets and lookup the client's MAC address only. As opposed to the `-n` and `-z` options, `BOOTPD32` will always send a BOOTREPLY to the client.

As this is also the standard behaviour of BOOTPD32, the `-h` option is obsolete.

Specify Client UDP Port (-c)

With the `-c` option, you can define the destination UDP port that BOOTPD32 uses when sending BOOTREPLY packets to the client.

Configuration File

BOOTPD32 reads information about clients from a configuration file which is commonly known as *bootptab*. If no configuration file name is given on the command line, BOOTPD32 uses its default, *c:\etc\bootptab*. See *"The bootptab File"* on page 93 for a detailed description of the configuration file's syntax.

Every time it receives a BOOTREQUEST packet, BOOTPD32 checks the date/time stamp of the configuration file. If it has changed since the last BOOTREQUEST packet was received, BOOTPD32 re-reads the configuration file. So, you do not need to restart BOOTPD32 after modifying the configuration file.

Dump File

BOOTPD32 can dump its current in-memory configuration information to a dump file. If no dump file name is given on the command line, BOOTPD32 uses its default, *c:\etc\bootpd.dmp*. To trigger the dump, select `File` ➜ `Dump bootptab` from the BOOTPD32 application window.

Logging

At this time, BOOTPD32 does not provide a means of logging information to a file. You can specify the amount of log information that BOOTPD32 writes to the application window on the screen.

Verbosity Level (-d)

With the `-d` option, you can control the amount of logging information that BOOTPD32 writes to the on-screen application window. Use the `-d` option up to four times to increase verbosity.

Using the `-d` option as the first one on the command line ensures that BOOTPD32 already uses logging when parsing the rest of the command line arguments.

Special Settings

Enable Database Interface (-b)

This option was present to allow BOOTPD32 to communicate with databases. It is no longer used and obsolete.

Using An Arguments File

If the command line is longer than your operating system is able to accept, then you can write all arguments and comments into a file, e.g. *c:\etc\bootpd.cnf*:

```
# Ethernet interface
-i 193.141.47.195/255.255.255.240
# Token-Ring interface
-i 193.141.47.209/255.255.255.240
# other options
-d -d -d -d -t 0 -s
```

and pass them to BOOTPD32 using the @ option:

```
bootpd32 @c:\etc\bootpd.cnf
```

Special Features

The BOOTPD32 daemon provides the following special features:

- The ms tag to increase the BOOTP reply size.
- Automatic detection of maximum usable BOOTP reply size.
- The mw tag to delay a BOOTP reply.
- The sa tag to overwrite the boot server's IP address.

Increasing The BOOTP Reply Size

The standard BOOTP reply is limited to 300 bytes, which includes 64 bytes of vendor-specific information. Using the tag ms, you can increase the size of the BOOTP reply. This allows you to pass more vendor-specific information with the BOOTP reply.

For example: the default BOOTP reply size is 328 bytes (300 bytes of BOOTP reply structure plus 28 bytes of UDP/IP information). By changing the BOOTP reply size to 776 bytes, you get 512 bytes of vendor specific information. To increase the BOOTP reply size, use the ms tag:

```
# template for vanilla Ethernet class C network
ether:\
    :hn:sm=255.255.255.0:vm=rfc1048:ht=ethernet:

# sample entry
diskless:\
    :tc=ether:ha=0000c00756bf:ip=193.141.47.198:\
    :hd=/tftpboot:bf=ramd.X:ms=776:
```

Delaying BOOTP Replies

You can delay a BOOTP reply until the requesting client exceeds a threshold. If you set a threshold (seconds) using the tag `mw`, then the BOOTP daemon starts sending BOOTP replies only after that limit is reached.

In this example, the BOOTP server will wait until the client sends BOOTP requests which are time-stamped for 8 seconds since the client started sending BOOTP requests. After that, the BOOTP server will reply to the BOOTP request:

```
# template for vanilla Ethernet class C network
ether:\
    :hn:sm=255.255.255.0:vm=rfc1048:ht=ethernet:

# sample entry
diskless:\
    :tc=ether:ha=0000c00756bf:ip=193.141.47.198:\
    :hd=/tftpboot:bf=ramd.X:mw=8:
```

This feature is useful if you have two BOOTP servers running in one network, where the second BOOTP server delays its BOOTP replies. In this configuration, the second BOOTP server only answers BOOTP replies if the first (undelayed) BOOTP server is down. This allows the second BOOTP server to act as a backup BOOTP server.

Changing The Boot Server's IP Address

By default, `BOOTPD32` uses its own IP address as the boot server's IP address and may be used for a later TFTP transfer. You can overwrite the boot server's IP address using the tag `sa`.

This example sets the server 193.141.47.193 as the boot server from which the bootfile */tftpboot/ramd.X* is transferred:

```
# template for vanilla Ethernet class C network
ether:\
    :hn:sm=255.255.255.0:vm=rfc1048:ht=ethernet:

# sample entry
diskless:\
    :tc=ether:ha=0000c00756bf:ip=193.141.47.198:\
    :hd=/tftpboot:bf=ramd.X:sa=193.141.47.193:
```

Windows 3

For Windows 3.x and Windows for Workgroups 3.1x, the BootManage® BOOTP Server exists as a 16-Bit Windows application, based on the Windows Sockets network interface.

To install, copy the file *bootpd.exe* to your system's Windows directory, normally *c:\windows*. Note that you need to have a Windows Sockets compliant TCP/IP stack installed on this machine.

In the services file, check that the required ports are listed as follows:

```
bootps      67/udp
bootpc      68/udp
```

If you do not have a services file, or the bootps and bootpc ports are not listed, the BOOTP server will use its built-in default values.

Then, use the Windows file manager to start BOOTPD:

```
bootpd -i 193.141.47.195 -d -d -d -d -t 0
```

Further information on the options for BOOTPD can be found earlier in this chapter.

Instead of running the BOOTP Server via the file manager's Run menu, you can also create a Program Item Property:

```
Description:       BOOTPD
Command Line:      c:\windows\bootpd -i 193.141.47.195 -d -d -d -d -t 0
Working Directory: c:\etc
Shortcut Key:      none
```

The format of the *c:\etc\bootptab* file is identical to the CMU BOOTP daemon. A sample entry would be:

```
# template for vanilla Ethernet class C network
ether:\
    :hn:sm=255.255.255.0:vm=rfc1048:ht=ethernet:

# sample entry
diskless:\
    :tc=ether:ha=0000c00756bf:ip=193.141.47.198:\
    :hd=/TFTPBOOT:bf=ramd.X:
```

Novell NetWare

For Novell NetWare, the BootManage® BOOTP Server exists as a NetWare Loadable Module (NLM).

To install, copy the file *bootpd.nlm* to your NetWare server's *sys:\system* directory. In order to run the BOOTPD NLM, the NetWare TCP/IP stack must be installed on the server.

In the sys:\etc\services file, make sure that the bootps and bootpc ports are listed:

```
bootps      67/udp
bootpc      68/udp
```

If you do not have a services file, or the bootps and bootpc ports are not listed, the BOOTP server will use its built-in default values.

Now, edit the *sys:\system\autoexec.ncf* file to start the BOOTP server NLM at system startup like in the following sample:

```
file server name nw
ipx internal net deadbeef

load smcnw311 port=240 int=3 mem=d4000 frame=ETHERNET_II name=en0
bind ipx to en0 net=1

load tcpip forward=yes rip=yes
bind ip to en0 addr=193.141.47.195

load bootpd -i 193.141.47.195 -d -d -d -d -t 0
```

Further information on the options for the BOOTP server can be found earlier in this chapter.

The format of the *sys:\etc\bootptab* file is identical to the CMU BOOTP daemon. A sample entry would be:

```
# template for vanilla Ethernet class C network
ether:\
    :hn:sm=255.255.255.0:vm=rfc1048:ht=ethernet:

# sample entry
diskless:\
    :tc=ether:ha=0000c00756bf:ip=193.141.47.198:\
    :hd="sys:/tftpboot":bf=ramd.X:
```

If the NetWare `BOOTPD` NLM can not find an interface to send the BOOTP reply to, it broadcasts the reply over all interfaces which have been enabled by the `-i` option.

Note that if multiple network adapters are used, you may have to use the `@` option of `BOOTPD`. This is because the length of the command line on NetWare is limited and the editor cannot hold enough characters in one line to pass all necessary arguments to the `BOOTPD` NLM.

The option `@` can be used to pass a filename to the `BOOTPD` NLM that holds all necessary arguments. This is a sample file named *sys:\etc\bootpd.cnf*:

```
-d -d -d -d -t 0 -s
-i 193.141.47.195/255.255.255.240
-i 193.141.47.211:3 -i 193.141.47.219/255.255.255.248
```

Arguments can be separated by newline characters or spaces. The `BOOTPD` NLM can then be loaded using the following command:

```
load bootpd @sys:\etc\bootpd.cnf
```

UNIX

Almost every UNIX system comes with a BOOTP server as part of the operating system distribution. Some of these BOOTP servers are limited in several ways, e. g. they do not support vendor options or only support very small reply sizes. To overcome these limitations, the BootManage® BOOTP Server been ported to a number of UNIX systems. For UNIX, the BootManage® BOOTP Server exists as a so-called BOOTP daemon (bootpd).

To install, copy the file *bootpd* to your system's binary directory, e.g. */bin*.

In the */etc/services* file, check that the bootps and bootpc ports are listed as follows:

```
bootps      67/udp
bootpc      68/udp
```

If you do not have a *services* file, or the bootps and bootpc ports are not listed, the BOOTP server will use its built-in default values.

If you want to replace the existing BOOTP daemon, remove its entry from */etc/inetd.conf* and restart your inetd. Then create a startup script to start the BOOTP daemon at each system start. You may want to put this script into */etc/rc3.d*:

```
/etc/bootpd -i 193.141.47.195 -d -d -d -d -s -t 0
```

Further information on the options for the BOOTP daemon can be found earlier in this chapter.

OS/2

The BootManage® BOOTP Server has been ported to the OS/2 operating system. You need to install OS/2 TCP/IP in order to operate the BootManage® BOOTP Server.

To install, copy the file *bootpd.exe* to your OS/2 system's binary directory, e.g. *c:\bin*.

In the *c:\tcpip\etc\services* file, check that the bootps and bootpc ports are listed as follows:

```
bootps      67/udp
bootpc      68/udp
```

If you do not have a *services* file, or the bootps and bootpc ports are not listed, the BOOTP server will use its built-in default values.

Now create a file which starts the BOOTP server, e.g. *c:\bin\bootpd.cmd*, with the following contents:

```
/* BOOTP server start script */
'@echo off'
say "Starting BOOTPD..."
'bootpd -i 193.141.47.204/255.255.255.240 -d -d -d -s -t 0'
exit 0
```

After this, execute the *bootpd.cmd* file to load the BOOTP server.

Further information on the options for the BOOTP daemon can be found earlier in this chapter.

Other BOOTP Servers

When using BOOTP servers other than the BootManage® BOOTP Server, there may be some special things to observe. As the BOOTP standard has evolved in several RFC's, especially older BOOTP servers only implement part of the specified functionality.

You may observe that a BOOTP server is limited in the amount of information that can be sent in a BOOTREPLY message, or that there is only limited support for vendor tags.

As the BOOTP protocol does not specify the format of the server's configuration database (the *bootptab* file), BOOTP servers offer different levels of tag support in the database. It may be the case that

- tags are not recognized at all, e.g. the hd tag is not supported, and you must specify the full path/filename using the bf tag.
- the number of tc tags in a host entry or the level of nested templates is limited.

If you encounter problems when using a BOOTP server, please check with the server's manual if any of these restrictions apply to your implementation.

The bootptab File

When it receives a BOOTREQUEST packet from a booting client, the BootManage® BOOTP Server looks up the client's hardware (MAC) address in a configuration database. If an entry is found, all the configuration information that is associated with this entry is put together in a BOOTREPLY packet which the BOOTP server sends back to the client.

The configuration database is an ASCII text file named *bootptab*, in which two-character case-sensitive tag symbols are used to represent configuration parame-

ters. These parameter declarations are separated by colons, with a general format of:

```
hostname:tag=value:tag=value:tag=value:...
```

A simple entry may look like:

```
pc301:ht=ethernet:ha=0000c00756bf:ip=193.141.47.198:sm=255.255.255.0
```

Adding comments and splitting an entry over multiple lines increases readability of the bootptab file:

```
# Entry for PC No. 301
pc301:\
    ht=ethernet:ha=0000c00756bf:\
    ip=193.141.47.198:sm=255.255.255.0
```

Blank lines and lines beginning with "#" are ignored in the configuration file. Host entries are separated from one another by newlines; a single host entry may be extended over multiple lines if the lines end with a backslash "\".

Make sure that the backslash is the very last character on the line. If there are any spaces, tabulators or other characters between the backslash and the end-of-line character, line extension will not work for this line!

It is also acceptable for lines to be longer than 80 characters. Tags may appear in any order, with the following two exceptions:

- The hostname must be the very first field in an entry.
- The hardware type must precede the hardware address.

Reference Of Supported Tags

The BootManage® BOOTP Server supports the following tags:

Tag	Description
bf	Boot file name
bs	Boot file size in 512-byte-blocks
cs	Cookie server IP address list
ds	Domain name server IP address list
gw	Gateway (router) IP address
ha	Client hardware (MAC) address
hd	Boot file home directory
hn	send client's host name

Tag	Description
ht	Hardware type
im	Impress server IP address list
ip	Client IP address
lg	Log server IP address list
lp	LPR print server IP address list
mw	BOOTREPLY message wait time
ms	BOOTREPLY message size
ns	IEN-116 name server IP address list
rl	Resource location protocol server IP address list
sa	TFTP server IP address
sm	Client subnet mask
tc	Table continuation (points to template entry to include)
to	Time offset in seconds from UTC
ts	Time server IP address list
vm	Vendor magic cookie selector
Tnnn	Generic (custom) tag number nnn

Boot File Name (bf)

The boot file name is an ASCII string which may be optionally surrounded by double quotes. This specifies the name of the file that the client should use as boot file when booting from the network.

Boot File Size (bs)

The bootfile size `bs` may be either a decimal, octal, or hexadecimal integer specifying the size of the bootfile in 512-octet blocks, or the keyword `auto` which causes the server to automatically calculate the bootfile size at each request. As with the time offset, specifying the `bs` symbol as a boolean has the same effect as specifying `auto` as its value.

Cookie Server IP Address List (cs)

This tag takes a whitespace-separated list of IP addresses as its value. It tells the client the IP addresses of all cookie servers that the client should use.

Domain Name Server IP Address List (ds)

This tag takes a whitespace-separated list of IP addresses as its value. It tells the client the IP addresses of all domain name servers that the client should use.

Gateway (Router) IP Address List (gw)

This tag takes a whitespace-separated list of IP addresses as its value. It tells the client the IP addresses of all routers/gateways that the client should use.

Client Hardware (MAC) Address (ha)

The `ha` tag specifies the client's hardware (MAC) address. When the BOOTP server receives a BOOTREQUEST, it reads the client's hardware address from the BOOTREQUEST and scans all the records in the *bootptab* file until it finds one that contains a `ha` tag whose value matches the given hardware address. This way, the BOOTP server finds the BOOTP configuration information that is sent back to the client in a BOOTREPLY packet.

The hardware address must be specified as hexadecimal. To enhance readability, dots can be used as separator characters. The `ha` tag must be preceded by the `ht` tag.

Boot File Home Directory (hd)

The home directory is an ASCII string which may be optionally surrounded by double quotes. This is the directory that contains the boot file.

Send Client's Host Name (hn)

The `hn` tag is strictly a boolean tag, it does not take the usual equals-sign and value. It's presence indicates that the client's hostname should be included in the BOOTREPLY packet.

Hardware Type (ht)

The `ht` tag specifies the hardware type code as a symbolic name. For Ethernet networks, use `ethernet` or `ether`, for Token-Ring networks use `token-ring` or `tr`. The `ha` tag takes a hardware address in hexadecimal numeric form, optional periods may be included for readability. The `ha` tag must be preceded by the `ht` tag.

Impress Server IP Address List (im)

This tag takes a whitespace-separated list of IP addresses as its value. It tells the client the IP addresses of all impress servers that the client should use.

Client IP Address (ip)

This tag takes a single IP address as its value. It is the IP address that the client should use.

Log Server IP Address List (lg)

This tag takes a whitespace-separated list of IP addresses as its value. It tells the client the IP addresses of all log servers that the client should use.

LPR Print Server IP Address List (lp)

This tag takes a whitespace-separated list of IP addresses as its value. It tells the client the IP addresses of all LPR print servers that the client should use.

BOOTREPLY Message Wait Time (mw)

This tag can be used to delay the BOOTP server's reply. See *"Delaying BOOTP Replies"* on page 89 for details.

BOOTREPLY Message Size (ms)

This tag can be used to increase the BOOTP reply size. See *"Increasing The BOOTP Reply Size"* on page 88 for details.

IEN-116 Name Server IP Address List (ns)

This tag takes a whitespace-separated list of IP addresses as its value. It tells the client the IP addresses of all IEN-116 name servers that the client should use.

RLP Server IP Address List (rl)

This tag takes a whitespace-separated list of IP addresses as its value. It tells the client the IP addresses of all RLP servers that the client should use.

TFTP Server IP Address (sa)

The `sa` tag may be used to specify the IP address of the particular TFTP server you wish the client to use. In the absence of this tag, the client performs TFTP to the same machine the BOOTP server is running on.

Client Subnet Mask (sm)

This tag takes a single IP address as its value. It is the IP subnet mask that the client should use.

Table Continuation, Template Pointer (tc)

Often, many host entries share common values for certain tags (such as name servers, etc.). Rather than repeatedly specifying these tags, a full specification can be listed for one host entry and shared by others via the tc (table continuation) mechanism. Often, the template entry is a dummy host which doesn't actually exist and never sends BOOTP requests. The `tc` tag symbol can appear anywhere in the host

entry. Information explicitly specified for a host always overrides information implied by a `tc` tag symbol, regardless of its location within the entry. The value of the `tc` tag may be the hostname of any host entry previously listed in the configuration file.

Time Offset (to)

The time offset to may be either a signed decimal integer specifying the client's time zone offset in seconds from UTC, or the keyword auto which uses the server's time zone offset. Specifying the `to` symbol as a boolean has the same effect as specifying auto as its value.

Time Server IP Address List (ts)

This tag takes a whitespace-separated list of IP addresses as its value. It tells the client the IP addresses of all time servers that the client should use.

Vendor Magic Cookie Selector (vm)

The vendor magic cookie selector (the `vm` tag) may take one of the following keywords: `auto` (indicating that vendor information is determined by the client's request), `rfc1048` or `rfc1084` (which always forces an RFC1084-style reply), or `cmu` (which always forces a CMU-style reply).

Generic (Custom) Tag Number nnn (Tnnn)

There is also a generic tag, `Tn`, where `n` is an RFC1084 vendor field tag number. Thus it is possible to immediately take advantage of future extensions to RFC1084 without being forced to modify the BOOTP server first. Generic data may be represented as either a stream of hexadecimal numbers or as a quoted string of ASCII characters. The length of the generic data is automatically determined and inserted into the proper field(s) of the RFC1084-style BOOTP reply.

As of this writing, you can use custom tag numbers between 128 and 254, inclusive.

In conjunction with the BootManage TCP/IP BOOT-PROM and the BootManage PXE Toolkit, generic tags are especially used for custom information and magic keywords.

A Sample bootptab File

The following sample bootptab file illustrates the tag syntax and demonstrates some of the features that assist you in structuring client configuration information.

In this example, the lines have been numbered so we can refer to them in the following explanation. Note that a "real" bootptab file must not contain these line numbers.

```
 1   # global parameters for the entire network
 2
 3   global:\
 4      :hn:vm=rfc1048:ht=ethernet:ms=1024:\
 5      :sm=255.255.0.0:
 6
 7
 8   # common parameters for the sales department
 9
10   sales-common:\
11      :tc=global:\
12      :gw=10.0.1.254:
13
14
15   # PC sa001 in the sales department
16
17   sa001:\
18      :tc=sales-common:\
19      :ha=00.a0.c9.42.da.23:ip=10.0.1.1:\
20      # Hard disk partition size in Mbytes
21      :T200="400":\
22      # Filesystem type
23      :T201="ConvertNTFS":\
24      # end of client record
25      :
```

As Lines 1, 8 and 15 start with the '#' character, they contain only comments and do not have any functional relevance for the BOOTP Server. You can also use comment lines within an entry, as shown for lines 20, 22 and 24.

Lines 3 and 10 contain names for template entries, line 17 contains the name of a host entry. Template and host entries can be distinguished by the ha tag. A host entry must contain a ha tag, whereas a template entry must not contain a ha tag.

Lines 11 and 18 demonstrate how a template is included in a configuration record. Host sa001 includes the settings of the template sales-common which, in turn, includes the settings of the template global. Therefore, host sa001 inherits the settings of both templates.

Lines 21 and 23 contain custom tags which can be freely defined in the range of T128 ... T254.

Note that each line of a configuration entry is terminated with a backslash '\' excluding the last one. Using a single colon as the last line of an entry (see line 25) allows you to rearrange the lines within an entry without worrying about the fact that the last line of an entry must not have a terminating backslash.

DHCP Servers

The DHCP Standard

The Dynamic Host Configuration Protocol DHCP is, like BOOTP, a simple protocol that allows a booting client to receive configuration information from a network server. As with the BOOTP protocol, this configuration information includes the client's IP address and subnet mask, the IP address of a server host, the name and location of a boot image file, and other network related configuration information.

Basically, DHCP uses the same packet structure as BOOTP. Through additional configuration options and a "lease mechanism", DHCP allows to dynamically hand out IP addresses to requesting clients. Once the client has acquired an IP address, it can use it for a specific amount of time.

A normal DHCP protocol handshake now has four packets:

6. The DHCP client broadcasts a DHCPDISCOVER packet that contains its network hardware (MAC) address.
7. The DHCP server sends a DHCPOFFER to the client in which it offers the client an IP address.
8. The DHCP client sends a DHCPREQUEST to the DHCP server in which it explicitly requests the offered IP address.
9. The DHCP server sends a DHCPACK to the DHCP client in which it grants the client the requested IP address.

Only after a DHCP client has received a DHCPACK, it is allowed to use the IP address that was granted in the DHCPACK.

The DHCP protocol is useful in environments where client computers should be automatically provided with IP addresses from an IP address pool, e.g. notebook computers in an insurance company.

The DHCP protocol is an open and vendor-independent internet standard defined by the Internet Engineering Task Force (IETF) in a number of Request For Comments (RFC) documents[1]. DHCP is closely related to BOOTP, so some RFC documents refer to both protocols.

Document	Title
RFC 2131	Dynamic Host Configuration Protocol
RFC 2132	DHCP Options and BOOTP Vendor Extensions
RFC 1534	Interoperation Between DHCP and BOOTP

Most operating systems already contain a DHCP server as part of the operating system distribution. This is true for most UNIX systems, Microsoft Windows NT Server 4.0, Windows 2000 Server, and NetWare 5.

Therefore, bootix does not provide a DHCP server of its own.

BOOTP Versus DHCP Servers

You can use either a BOOTP or DHCP server to provide configuration information to a booting client. Actually, BOOTP and DHCP are basically the same protocol, i.e. both use the same packet structure and bind to the same UDP port. To be more precise, DHCP is the successor to BOOTP, the main intention behind DHCP was to provide mobile client computers with dynamic IP addresses.

Client PCs that start up using the BootManage® TCP/IP BOOT-PROM will accept replies from both BOOTP and DHCP servers. Only if you want to hand out IP addresses to clients dynamically, you must use DHCP.

Following, we will shortly introduce some DHCP servers that ship with the corresponding operating system.

Microsoft DHCP Server

Microsoft DHCP Server is included in the Windows NT 4.0 Server and also in the Windows 2000 Server; it is implemented as a Windows NT/2000 service.

Following, we want to show you how to install and configure the Microsoft DHCP Server on Windows NT 4.0 Server.

1 There are more documents available about DHCP). For more information about the IETF and RFC documents, point your web browser to http://www.ietf.org.

Installing Microsoft DHCP Server

The Microsoft DHCP Server is included in the Microsoft Windows NT 4.0 Server but, by default, is not automatically installed with the operating system. To install the Microsoft DHCP Server, select Start ➜ Settings ➜ Control Panel and double-click the Network Icon. After selecting the Add button in the Services tab, the Select Network Service dialog opens and presents you with a list of network services. Select Microsoft DHCP Server and click OK. After the files have been copied from the Windows NT Server CD-ROM to your system's hard disk, the Microsoft DHCP Server should be listed in the Services tab as shown in *Figure 9-1*, and you are then asked to reboot the NT Server.

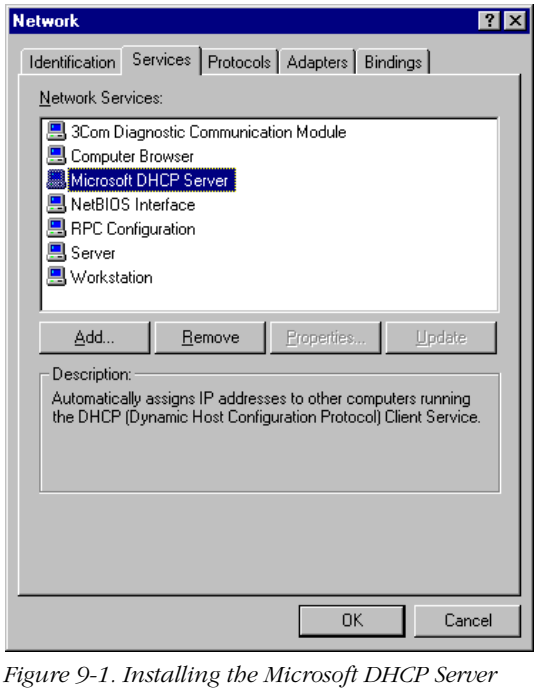


Figure 9-1. Installing the Microsoft DHCP Server

The Windows NT Server 4.0 machine that runs the Microsoft DHCP Server must have a static IP address assigned, it cannot work as a DHCP client!

After rebooting, select Start ➜ Settings ➜ Control Panel and double-click on the Services Icon. Make sure that the Microsoft DHCP Server appears in the list with Status set to Started and Startup set to Automatic, as shown in *Figure 9-2*.

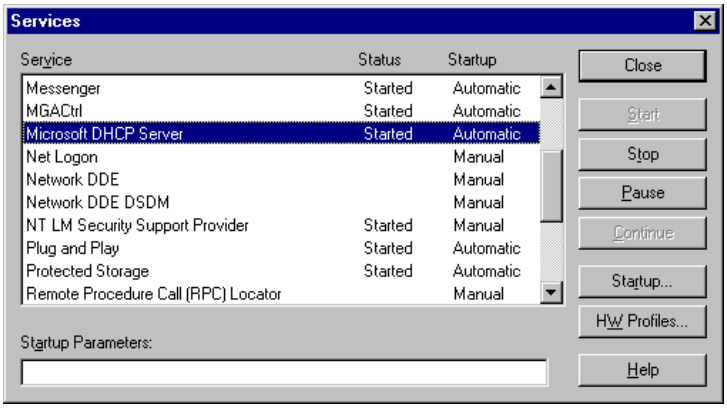


Figure 9-2. Services Control Panel

Starting And Stopping Microsoft DHCP Server

Like any other Windows NT service, Microsoft DHCP Server can be started and stopped from the services applet of the control panel. In the service window, locate "Microsoft DHCP Server", and then select the Start or Stop button. The Status column will tell you whether or not the service is running.

Also, you can start and stop Microsoft DHCP Server from the Windows NT command line:

```
net start dhcpserver
net stop dhcpserver
```

Configuring Microsoft DHCP Server

The Microsoft DHCP Server can be configured through both a graphical interface and through a command line program.

The graphical interface, called DHCP Manager, is installed along with the Microsoft DHCP Server. We will use DHCP Manager for the examples in this chapter.

The command line program, `DHCPCMD`, does not ship with the Windows NT 4.0 Server, but can be found in the Microsoft Windows NT 4.0 Server Resource Kit. Using `DHCPCMD`, you can perform automated configuration from within batch files. For more information, please consult the Microsoft Windows NT 4.0 Server Resource Kit documentation.

Starting DHCP Manager

To start DHCP Manager, select Start ➜ Programs ➜ Administrative Tools (Common) ➜ DHCP Manager. The program will start up as shown in *Figure 9-3*.

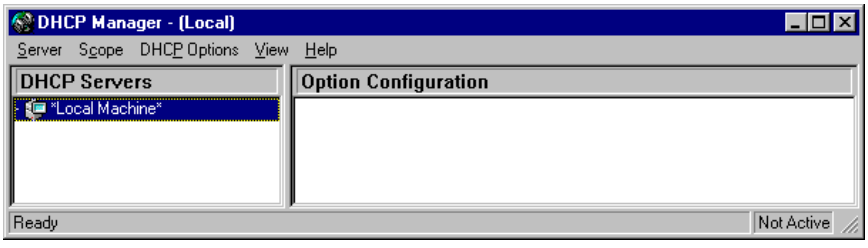


Figure 9-3. DHCP Manager

Creating And Activating A DHCP Scope

To create a DHCP scope, double-click Local Machine in the DHCP Servers pane and select Create from the Scope menu. In the Create Scope Dialog shown in *Figure 9-4*, enter the Scope information. Note that the shown IP addresses are just sample values. Make sure that you use addresses that match your network environment. Click OK to accept your scope information and close the dialog box.

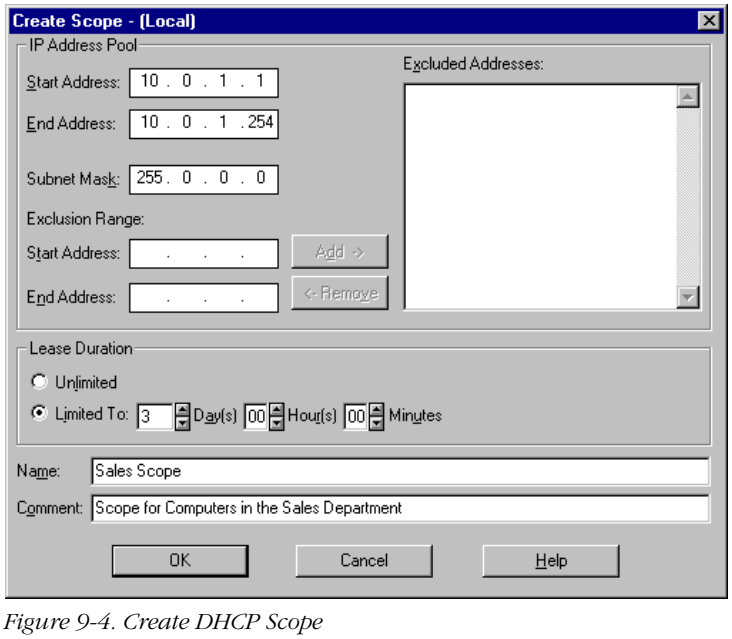


Figure 9-4. Create DHCP Scope

A dialog box is displayed asking if you want to activate the just-created scope. If you choose NO here, the scope is not activated. You must then activate the scope when all configuration is cone. Note that if a scope is deactivated, Microsoft DHCP Server will not answer DHCP requests from booting TCP/IP BOOT-PROM clients.

Adding The Bootfile Name Option

The TCP/IP BOOT-PROM needs to know the name and location of a boot file, which can be either a boot image or a boot loader (such as BPBOOT). To achieve this, the Bootfile Name option (067) is used.

Highlight the just-created scope in the DHCP Servers pane and then select DHCP Options ➜ Scope. In the Unused Options box, select 067 Bootfile Name and click Add. Then, click Value and enter the name of the boot file in the text field, as shown in *Figure 9-5*.

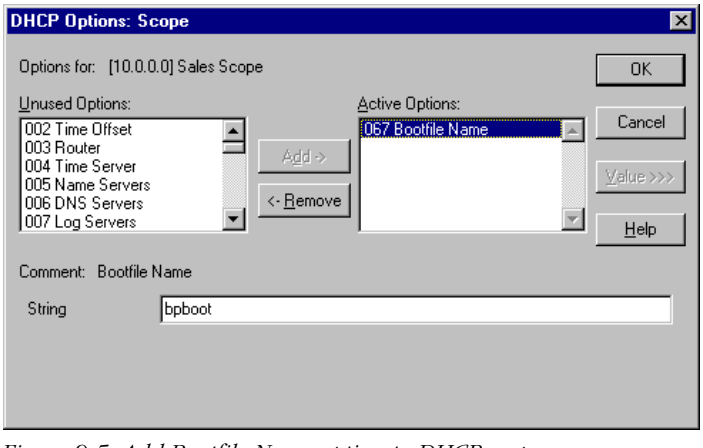


Figure 9-5. Add Bootfile Name option to DHCP scope

After adding the new option to your scope by clicking OK, TCP/IP BOOT-PROM clients will be told to download and execute the BPBOOT boot loader.

Using the same method, you may want to add other standard options such as Routers, DNS Servers, WINS Servers and the like.

If you boot the TCP/IP BOOT-PROM client now, it downloads and executes the BPBOOT boot loader. However, at this time, BPBOOT complains that it does not find any configuration information in the DHCP reply. Following, we show how to include this configuration information within custom DHCP options.

Implementing Custom Options

The Microsoft DHCP allows to define custom options that can be used to provide special information to TCP/IP BOOT-PROM clients. Also, these custom options can be used to hold magic keywords that control the BPBOOT bootstrap loader. Custom options can have any number between 128 and 254, inclusive.

To add a new custom option, select DHCP Options ➜ Defaults. The DHCP Options: Default Values dialog pops up. Select New... to open the Add Option Type dialog. There, enter the new custom option as shown in *Figure 9-6*, making sure that the Data Type field is set to String.

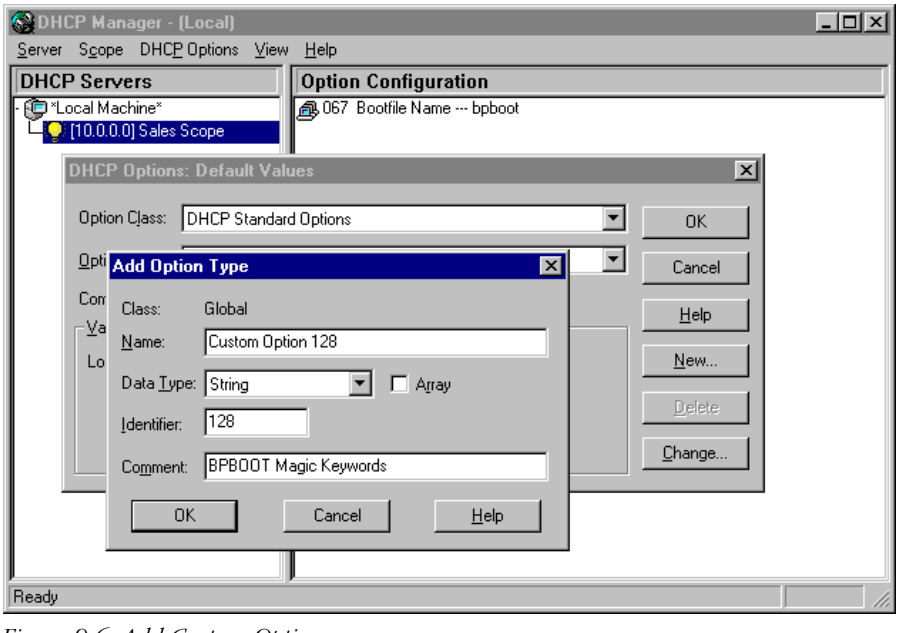


Figure 9-6. Add Custom Option

Close the dialog windows by clicking OK, and the new custom option is defined, but not yet present in your DHCP scope. Do this next, using the same procedure as for adding the boot file name. In the String field, you may enter any string value of your liking. This value will be then available from within the boot image, and you can use it to whatever suits your needs.

To define more custom options, just repeat the above procedure and select a different option number.

The Microsoft DHCP Server provides only a very limited amount of space for custom options. If you experience that not all of your custom option information is received by the client, this may be because the Microsoft DHCP Server truncated it. In this case, consider using the options files feature of the BPBOOT boot loader.

BPBOOT Magic Keywords In Custom Options

A custom option is the right place to hold magic keywords (such as BpOpT or BpDiS) for the BPBOOT bootstrap loader. You can either use one custom option per magic keyword, or you can place multiple magic keywords in a single custom option.

Figure 9-7 shows both methods: Custom option 128 holds a single magic keyword, and custom option 129 holds two magic keywords, separated by a semicolon.

For more information about magic keywords, please refer to *"BPBOOT's Magic Keywords"* on page 213.

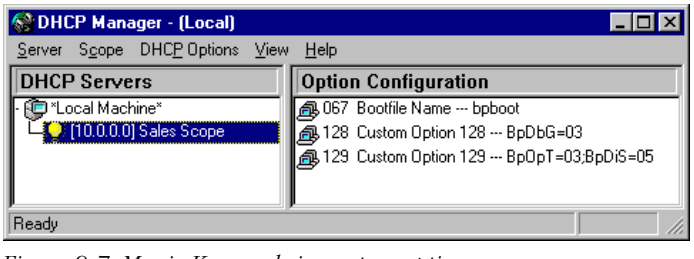


Figure 9-7. Magic Keywords in custom options

Activating The DHCP Scope

If you have not already done so, you must activate the DHCP scope so that the Microsoft DHCP Server will actually send configuration information to booting TCP/IP BOOT-PROM clients. To activate the scope, just highlight it in the DHCP Servers window and select Scope ➜ Activate. The lightbulb symbol to the right of the scope becomes lit, and the scope is activated.

ISC DHCP Server (dhcpd)

The Internet Software Consortium (ISC) offers a freely available[1] DHCP Server (dhcpd) which has become the standard DHCP Server in the NetBSD and Red Hat Linux operating system distributions.

At the time of writing, ISC dhcpd v2.0 is the current stable version which will do a good job in working together with the TCP/IP BOOT-PROM. ISC dhcpd v3.0 adds a lot of functionality, but is available as a beta version only.

1 You can download the ISC DHCP Server source distribution from the Internet Software Consortium's web site, http://www.isc.org.

ISC dhcpd v2.0 has all the features that the TCP/IP BOOT-PROM needs in order to make use of its full functionality. If you are looking for a recommendation of what version to use, then get v2.0.

Installing ISC DHCP Server

If the ISC DHCP Server is not already included in your operating system distribution, download the source distribution and compile it on your system. Explaining how to do this is beyond the scope of this manual, please consult the documentation within the ISC dhcpd source distribution.

Starting And Stopping ISC DHCP Server

The ISC DHCP Server, dhcpd, runs as a standalone server and should not be started by the inetd daemon. On Red Hat Linux, the script /etc/init.d/dhcpd is used to automatically start dhcpd when the system boots up.

As the root user, you can call this script in the following ways:

```
/etc/init.d/dhcpd start      (starts dhcpd)
/etc/init.d/dhcpd stop       (stops dhcpd)
/etc/init.d/dhcpd restart    (stops dhcpd, then starts it again)
```

Configuring ISC DHCP Server

By default, the ISC DHCP Server's configuration files are located in the /etc directory. Whenever dhcpd starts up, it reads the ASCII text file /etc/dhcpd.conf, which contains all configuration information. The dhcpd.conf file can be modified with any text editor.

Whenever the dhcpd.conf file is modified, dhcpd must be restarted in order to recognize the altered configuration. To do so, terminate dhcpd by sending it a SIGTERM signal, and then re-invoke dhcpd. It is not sufficient to just send a SIGHUP signal!

The ISC DHCP Server must be able to keep track of what IP addresses are leased to what clients across system reboots, it uses the ASCII text file dhcpd.leases. On Linux systems, this file may be either in /etc or in /var/state/dhcp. In contrast to dhcpd.conf, dhcpd.leases should not be edited by hand, but can be read in order to get information about the client computers that have been configured by DHCP.

ISC DHCP Server Capabilities

The ISC DHCP Server is very flexible in the way that it can be configured to serve up configuration information for TCP/IP BOOT-PROM client PCs. It is possible to:

- group clients that share common configuration information
- support multiple IP subnets on the same physical network
- hand out IP addresses dynamically or statically
- support BOOTP clients with dynamic IP addresses
- support custom tags/options in ASCII and binary form

Some common configuration types are:

Static IP Address, Client Known By Hardware (MAC) Address

All clients must be listed with their hardware (MAC) address in the *dhcpd.conf* file. The ISC DHCP Server will not send replies to unknown clients. Also, every known client has a statically assigned IP address which can be explicitly given in the configuration file. If no client IP address is given, the ISC DHCP Server will use the client's host and domain name to lookup the IP address "in the background".

Dynamic IP Address, Client Known By Hardware (MAC) Address

All clients must also be listed with their hardware (MAC) address in the *dhcpd.conf* file. The ISC DHCP Server will not send replies to unknown clients. However, IP addresses will be assigned to clients dynamically from an "IP address pool".

Dynamic IP Address, New (Unknown) Client

Clients need not be listed in the *dhcpd.conf* file. The ISC DHCP Server sends replies to all requesting clients. As with in the previous example, IP addresses will be assigned to clients dynamically from an "IP address pool".

Note that this method does not allow to provide clients with individual options; all requesting clients receive the same generic information.

For more information, please refer to the documentation that comes with ISC DHCP Server.

NetWare DHCP Server

NetWare DHCP Server is known to work with the TCP/IP BOOT-PROM, but at this time, we do not have any further information or sample configuration.

Other DHCP Servers

There are a number of other DHCP Servers available, but providing detailed information and configuration esamples for all of them would be beyond the scope of this manual. However, there is some general advice when using the TCP/IP BOOT-PROM with one of the “other” DHCP Servers:

- Start with a simple configuration so that you can successfully download and execute a DOS boot image on a TCP/IP BOOT-PROM client PC.
- Check if the DHCP Server supports custom options. Define some custom options and use “bputil -s” on the client to see if these custom options have been correctly transferred.
- In case that custom options are not (sufficiently) supported, use the options files feature of the BPBOOT boot loader to implement custom options.

TFTP Servers

The TFTP Standard

The Trivial File Transfer Protocol TFTP is a simple protocol to transfer files, designed to be small and easy to implement. Therefore, it lacks most of the features of a regular file transfer protocol. The only thing it can do is read and write files from/to a remote server, where each packet is acknowledged separately.

The TFTP protocol is especially useful in remote boot and remote installation environments where a client PC needs to download a boot image file over the network from a boot server.

The TFTP protocol is an open and vendor-independent internet standard defined by the Internet Engineering Task Force (IETF) in a number of Request For Comments (RFC) documents[1]:

Document	Title
RFC 1350	The TFTP Protocol (Revision 2)
RFC 1785	TFTP Option Negotiation Analysis
RFC 2347	TFTP Option Extension
RFC 2348	TFTP Blocksize Option
RFC 2349	TFTP Timeout Interval and Transfer Size Options

Most UNIX operating systems already contain a TFTP server as part of the operating system distribution, however this TFTP server may not always be full-featured. Other server operating systems like Windows NT, NetWare, and OS/2 do not provide a TFTP server.

1 There are more RFC documents on TFTP available, but most of them have been obsoleted by the documents shown in our table. For more information about the IETF and RFC documents, point your web browser to http://www.ietf.org.

The BootManage® TFTP Server

The BootManage® TFTP Server is a standards-conforming server implementation of the TFTP protocol as defined in the IETF RFC document 1350. More, the BootManage® TFTP Server provides a number of custom enhancements which provide extended functionality, especially in combination with the BootManage® TCP/IP BOOT-PROM and the BootManage® PXE Toolkit.

Implementations are available for the following operating system platforms:

- Windows 2000 Professional and Server
- Windows NT 3.x and 4.0 (Workstation and Server)
- Windows 95/98/ME
- Windows 3.x, Windows for Workgroups 3.1x
- Several UNIX platforms, including Linux
- NetWare 3.x, 4.x, 5.x
- OS/2

The amount of extended TFTP Server functionality varies for the different operating systems. Not every extended function is available for every listed operating system.

A Word About Licensing

The BootManage® TFTP Server is not free software, and you need a license to use it. However, the BootManage® TCP/IP BOOT-PROM and the BootManage® PXE Toolkit already contain a license for the BootManage® TFTP Server. If you purchased one of these products, and if you use the BootManage® TFTP Server in conjunction with any of these products, you do not need a separate license for the BootManage® TFTP Server.

Server Implementations

To explain the functions of the BootManage® TFTP Servers, we will use the 32-Bit Windows implementation `TFTPD32` as a reference. Later in this chapter, we will explain the implementation specific issues for the other BootManage® TFTP Servers.

Windows 95/98/ME/NT/2000

TFTPD32 is a TFTP server for Windows 32-Bit platforms which adds several features that are not available in standard TFTP implementations:

- Allows restricted file access for security.
- Provides several means of controlling server and network load.
- Allows larger TFTP segment sizes (LTFTP) for higher throughput, fewer network transactions the possibility to download larger files.
- Supports multicast file transfer (MTFTP) for faster concurrent booting of multiple clients.
- Supports filename mapping (in combination with the BPBOOT bootstrap loader from the BootManage® TCP/IP BOOT-PROM distribution).

Running TFTPD32 As Application

First, copy the file *tftpd32.exe* to the local hard disk of a Windows 95/98/ME, Windows NT Workstation/Server, or Windows 2000 Professional/Server machine.

```
copy tftpd32.exe %WINDIR%                  (on Windows 95/98/ME)
copy tftpd32.exe %SystemRoot%\system32     (on Windows NT/2000)
```

TFTPD32 will look up the port number for the tftp port in the file *%SystemRoot%\system32\drivers\etc\services*. If this entry is not present, TFTPD32 will use the following default:

```
tftp    69/udp
```

If you want to start TFTPD32 as a Win32 application, simply invoke it from the command line using the -cmd option and any other options which suit your needs, e. g.:

```
tftpd32 -cmd -v 2
```

If you do not specify any options on the command line, TFTPD32 will read the options from the text file *c:\etc\tftpd.cnf*. For a detailed list of all supported options, please see *"TFTPD32 Command Line Options"* on page 116.

If TFTPD32 should operate as a standard TFTP Server, it is not necessary to specify any options on the command line beyond -cmd. However, to activate one or more of the advanced features TFTPD32 provides, you must specify the corresponding option(s).

Installing TFTPD32 As A Service

If you want to install TFTPD32 as a Windows NT/2000 service, use the -install option:

```
tftpd32 -install
```

Please note that when using -install, you cannot specify any other options on the command line. Therefore, be sure to place any desired options in the text file, *c:\etc\tftpd.cnf*, where TFTPD32 will read them.

When installed as a service, TFTPD32 will automatically start at every Windows NT/2000 system startup. Like any other service, you can start and stop TFTPD32 using the Windows NT/2000 Service Control Manager. In addition, you can control TFTPD32 from the command line:

```
net start tftpd              (start TFTPD32 service)
net stop tftpd               (stop TFTPD32 service)
```

TFTPD32 Command Line Options

The following arguments can be given to the TFTPD32 server:

```
tftpd32 -cmd options          (run as Win32 application)
tftpd32 -install              (install as Windows NT service)
tftpd32 -remove               (remove service)
tftpd32 -?                    (display command line syntax)
```

With the -cmd argument, the following options are provided:

Option	Description
-a [*file*]	Read table of mapped filenames from *file*
-b *num*	Limit TFTP bandwidth to approximately *num* KBytes/s
-c *num*	Number of concurrent TFTP transfers (max. 128)
-d [*path*]	Restrict TFTP access to directory *path* and its subdirectories.
-h	Enable read-ahead buffer.
-i *num*	Exit after *num* seconds of inactivity.
-k *num*	Set TFTP transmission keep-alive time.
-l [*file*]	Log all messages to a file.
-m *file port*	MTFTP configuration file and listen port.
-p *num*	Set TFTP listen port.
-r	Restrict file access to current directory.
-s *num port*	Set TFTP segment size and listen port.
-t *num*	Set Multicast Time-to-live value to *num* seconds
-u *num*	Number of MTFTP unicasts to be sent (default: 4).
-v *level*	Set level of verbosity (0, 1, or 2).

Option	Description
`-w`	Enable writing to existing files on the TFTP server
`-x`	If `-w` is also set, allow creation of files on the TFTP server

Controlling Server And Network Load

The following options allow to control performance related parameters of `TFTPD32`.

Limit Number Of Concurrent TFTP Transfers (-c)

The number of concurrent TFTP transfers can be limited by using the `-c` option. The following example limits the number of concurrent TFTP transfers to 4:

```
tftpd32 -c 4
```

Enable Read-Ahead Buffers (-h)

This option enables read-ahead buffers for the TFTP file transfer. That is, `TFTPD32` reads more data than is needed and buffers the additional data for later use. In some environments, the transfer speed increases by enabling the `-h` option. By default, `TFTPD32` does not read-ahead.

Terminate TFTP Server After Inactivity Period (-i)

`TFTPD32` can be automatically terminated by using the `-i` option. If there is no TFTP activity for the number of seconds specified, `TFTPD32` terminates. The following example aborts `TFTPD32` after 45 seconds of inactivity:

```
tftpd32 -i 45
```

Set TFTP Transmission Keepalive Time (-k)

`TFTPD32` retains all information from a TFTP transfer for a specified amount of time so that a client can repeat a prior transfer. The `-k` option allows one to change the default time of 15 seconds.

Security Related Options

Security related options control in what way TFTP clients can access files on the TFTP server.

Set TFTP Server Root Directory (-d)

A default path can be added to each request issued by a TFTP client. This way, `TFTPD32` restricts access to files below a certain directory on the server. If the `-d`

option is given without a directory, this defaults to *c:\tftpboot*. The following examples restrict TFTP access to both files and subdirectories below *c:\tftpboot*:

```
tftpd32 -d /tftpboot
tftpd32 -d
```

Restrict TFTP Access To Directory (-r)

The `-r` option restricts all file access to the current directory and its subdirectories. That is, no files from any other locations may be transmitted. The current directory can be changed using the `-d` option.

The following example limits all file transmissions to the directory */tftpboot* and its subdirectories:

```
tftpd32 -d c:\tftpboot -r
```

Enable TFTP Write Access (-w)

Normally, `TFTPD32` only permits read operations, i.e. a TFTP client can only read a file, not write one to a TFTP server. The `-w` option enables TFTP write access so that a TFTP client can write to existing files on the TFTP server. Unless the `-x` option is also specified, a TFTP client can only write to existing files and cannot create new files.

Allow Creation Of Files (-x)

Together with the `-w` option, the `-x` option allows a TFTP client to create new files on the TFTP server.

Logging

`TFTPD32` can log its activity to the application window on the screen, to the Windows NT event log and/or to a log file.

Log TFTP Server Activity (-l)

All output of `TFTPD32` can also be logged to a file. The `-l` option enables logging and allows one to specify a filename. If no filename is specified, then the default is *c:\tftpboot\tftpd.log*. The following example enables full debugging and logs all output to *c:\tmp\tftpd.log*:

```
tftpd32 -v 2 -l c:\tmp\tftpd.log
```

Set Verbosity Level (-v)

The debugging and activity output of `TFTPD32` can be changed by the `-v` option. The default is 1, and increasing the number generates more output. A value of 0 does not create any output, and a value of 4 displays very detailed information.

Multicast TFTP Operation

Enable Multicast TFTP (MTFTP) Operation (-m)

The `-m` option enables the `TFTPD32` multicast file transfer support (MTFTP). Two arguments are passed to the `-m` option: The first argument specifies a configuration file, the second argument specifies the UDP port number where `TFTPD32` listens for MTFTP file transfer requests. The following example uses port number 75 for MTFTP requests:

```
tftpd32 -m c:\etc\mtftptab 75
```

The configuration file has the following format:

```
c:\tftpboot\pcnfs.X      225.0.0.1     76
c:\tftpboot\ramd.X       225.0.0.2     76
```

Since there are no reserved port numbers for multicast TFTP, it cannot be guaranteed that the default ports of 75 and 76 will be available on all systems.

The first column holds the filename to be transmitted. If `TFTPD32` receives an MTFTP request for that filename on the specified MTFTP server port (75 in the above example), then it sends the MTFTP packets to the IP address specified in the second column (225.0.0.1). The third column holds the MTFTP client port number where all data is sent.

Set Number Of Unicast Packets In MTFTP Mode (-u)

The `-u` option is used in conjunction with multicast file transfer (MTFTP). The `-u` option specifies the number of unicast packets that `TFTPD32` sends to the MTFTP client after a transmission has been initiated.

The default is 4 packets and should not be changed.

Large Block TFTP Operation

Enable Large Block TFTP (LTFTP) Operation (-s)

This option allows one to set the TFTP segment size. The standard TFTP protocol specifies a fixed segment size of 512 bytes and a 16-Bit packet counter. By increasing the segment size, fewer network transactions are needed to transfer a given

file and, therefore, faster file transfer can be achieved. Also, it is possible to transfer larger files[1].

Increasing the TFTP segment size makes the TFTP transmission incompatible with standard TFTP clients and the RFC specification. Therefore, TFTPD32 only sends larger packets to clients which open the TFTP transmission at the specified UDP port:

```
tftpd32 -s 1408 59
```

This adds support for larger TFTP packets (here 1408 bytes) if the connection is opened at UDP port 59 instead of the standard TFTP port 69.

To retain compatibility with the standard TFTP protocol, TFTPD32 will still reply to incoming standard TFTP requests on port 69 with 512-byte-packets. At the same time, opening a TFTP transmission at the non-standard UDP port 59 allows TFTPD32 to send up to 1408 bytes in TFTP packets.

Custom Settings

Specify TFTP Server Listen Port (-p)

By default, TFTPD32 uses UDP port number 69 to listen for TFTP file transfer requests. This can be overridden by using the -p option. TFTPD32 can also be used together with an existing TFTP server. In this case, TFTPD32 is used only for extended features.

To disable the standard TFTP functionality of TFTPD32, bind the TFTP UDP port number to an unused UDP port, e.g.:

```
tftpd32 -s 1408 59 -m c:\etc\mtftptab 75 -v 1 -h -p 21435
```

Using Mapped Filenames

Enable Mapped Filenames (-a)

The -a option enables the TFTPD32 filename mapping feature. As an argument to the -a option, the filename of a map configuration file must be passed. The pur-

1 With standard TFTP, the maximum file size is 2^{16} x 512 bytes = 32 MBytes. Increasing the TFTP segment size to 1408 allows a maximum file size of 2^{16} x 1408 bytes = 88 MBytes.

pose of the map configuration file is to substitute a requested filename with a mapped filename using the following format:

```
<requested-filename-1>   <mapped-filename-1>
<requested-filename-2>   <mapped-filename-2>
<requested-filename-3>   <mapped-filename-3>
...
$DEFAULT$                <default-filename>
```

Whenever TFTPD32 receives a TFTP request for a file/pathname which starts with the character string *MAP*, it strips the *MAP* prefix and looks up the remaining file/pathname in the left-hand side of the map configuration file.

If a matching file/pathname is found, TFTPD32 replaces the requested file/pathname with the corresponding (mapped) file/pathname listed in the right-hand side of the map configuration file. Then, TFTPD32 sends the mapped file (instread of the requested file) to the TFTP client.

If a matching file/pathname is not found, TFTPD32 checks if a $DEFAULT$ entry is present at the end of the map configuration file. If it finds one, TFTPD32 uses the mapped file/pathname associated with the $DEFAULT$ entry.

If a matching file/pathname is not found, and TFTPD32 also does not find a $DEFAULT$ entry at the end of the map configuration file, it simply uses the (original) requested file/pathname, but without the *MAP* prefix.

TFTPD32 will only try to perform filename mapping if the requested filename starts with the string *MAP*. TFTPD32 will not perform any filename mapping for files that do not start with the string *MAP*. Files in the left-hand side of the map file must be specified without the leading *MAP* string.

The filename mapping feature is especially useful in combination with the BPBOOT bootstrap loader which is part of the BootManage® TCP/IP BOOT-PROM distribution. BPBOOT can generate individual filenames using the client's MAC address. This allows to easily define "boot image groups" for PC clients in a DHCP environment without the need of specifying individual boot images in the DHCP server's configuration.

Mapped Filenames Example

The following example illustrates the filename mapping feature. Let us assume that the map file *c:\etc\tftpd.map* contains the following information:

```
c:\tftpboot\112233445566.X   c:\tftpboot\winnt40.X
c:\tftpboot\445566778899.X   c:\tftpboot\winnt40.X
c:\tftpboot\778899aabbcc.X   c:\tftpboot\win95.X
c:\tftpboot\aabbccddeeff.X   c:\tftpboot\win98.X
$DEFAULT$                    c:\tftpboot\bootfile.X
```

To enable filename mapping, TFTPD32 must be invoked as follows:

```
tftpd32 -cmd -a c:\etc\tftpd.map -v 2
```

The command line option -v is not needed for mapped filename operation, but it is very useful for locating problems.

When receiving a TFTP read request for file *MAPc:\tftpboot\778899aabbcc.X*, TFTPD32 strips the leading *MAP* and looks up the remaining filename *c:\tftpboot\778899aabbcc.X* in the left-hand side of the map file *c:\etc\tftpd.map*. The filename is found in the third row, so TFTPD32 substitutes it with the corresponding entry *c:\tftpboot\win95.X* on the right-hand side and sends this file back to the client. Likewise, when a TFTP client requests *MAPc:\tftpboot\aabbccddeeff.X*, TFTPD32 sends *c:\tftpboot\win98.X* instead.

The first two rows of the map file *c:\etc\tftpd.map* demonstrate how to group boot image files. Clients request different files (*MAPc:\tftpboot\112233445566.X* and *MAPc:\tftpboot\445566778899.X*), but both are substituted by the same file (*c:\tftpboot\winnt40.X*).

To demonstrate the $DEFAULT$ entry, let us assume that TFTPD32 receives a read request for a file named *MAPc:\tftpboot\123456789abc.X*. This file is not listed in our sample map file, but it starts with the string *MAP*. In this case, TFTPD32 looks for a $DEFAULT$ entry on the left-hand side at the end of the map file. In our map file, a $DEFAULT$ entry is present, so TFTPD32 uses this for substitution and sends the file *c:\tftpboot\bootfile.X* back to the client. If there were no $DEFAULT$ entry, TFTPD32 simply strips the *MAP* prefix and sends the file *c:\tftpboot\123456789abc.X* back to the client instead.

Windows 3

For Windows 3.x and Windows for Workgroups 3.1x, the BootManage® TFTP Server exists as a 16-Bit Windows application, based on the Windows Sockets network interface.

To install, copy the file *tftpd.exe* to your system's Windows directory, normally *c:\windows*. Note that you need to have a Windows Sockets compliant TCP/IP stack installed on this machine.

In the services file, check that the TFTP port is listed as follows:

```
tftp    69/udp
```

If you do not have a services file, or the TFTP port is not listed, the TFTP server will use its built-in default value of 69/udp.

Then, use the Windows file manager to start `TFTPD`:

```
tftpd -r -v 1 -s 1408 59
```

Further information on the options for `TFTPD` can be found earlier in this chapter.

Instead of running the TFTP Server via the file manager's Run menu, you can also create a Program Item Property:

```
Description:       TFTPD
Command Line:      c:\windows\tftpd -r -v 1 -s 1408 59
Working Directory: c:\etc
Shortcut Key:      none
```

Novell NetWare

For Novell NetWare, the BootManage® TFTP Server exists as a NetWare Loadable Module (NLM).

To install, copy the file *tftpd.nlm* to your NetWare server's *sys:\system* directory. In order to run the `TFTPD` NLM, the NetWare TCP/IP stack must be installed on the server.

In the *sys:\etc\services* file, make sure that the TFTP server port is listed:

```
tftp    69/udp
```

If you do not have a *services* file, or the TFTP port is not listed, the TFTP server will use its built-in default value of 69/udp.

Now, edit the *sys:\system\autoexec.ncf* file to start the TFTP server NLM at system startup like in the following sample:

```
file server name nw
ipx internal net deadbeef

load smcnw311 port=240 int=3 mem=d4000 frame=ETHERNET_II name=en0
bind ipx to en0 net=1

load tcpip forward=yes rip=yes
bind ip to en0 addr=193.141.47.195

load tftpd -v 1 -r -d sys:\tftpboot -s 1408 59
```

Further information on the options for the TFTP server can be found earlier in this chapter.

UNIX

The BootManage® TFTP Server has been ported to a number of UNIX systems, including Linux. For UNIX, the BootManage® TFTP Server exists as a so-called TFTP daemon (tftpd).

Most UNIX systems already ship with a TFTP server as part of the operating system distribution. You can use the operating system's TFTP server or replace it with the BootManage® TFTP Server to make use of its advanced features.

To install, copy the file *tftpd* to your system's binary directory, e.g. */bin*.

In the */etc/services* file, check that the TFTP port is listed as follows:

```
tftp    69/udp
```

If you do not have a services file, or the TFTP port is not listed, the TFTP server will use its built-in default value of 69/udp.

If you want to replace the existing TFTP daemon, remove its entry from */etc/inetd.conf* and restart your inetd. As the BootManage® TFTP Server cannot run under control of inetd, you must create a startup script to start the TFTP daemon at each system start. You may want to put this script into */etc/rc3.d*:

```
/etc/tftpd -v 0 -d /tftpboot -h -r -s 1408 59 &
```

If you do not want to replace the existing TFTP daemon do not remove its entry from */etc/inetd.conf* and, instead, create the following script:

```
/etc/tftpd -v 0 -d /tftpboot -h -r -s 1408 59 -p 21435 &
```

Further information on the options for the TFTP daemon can be found earlier in this chapter.

OS/2

The BootManage® TFTP Server has been ported to the OS/2 operating system. You need to install OS/2 TCP/IP in order to operate the BootManage® TFTP Server.

To install, copy the file *tftpd.exe* to your OS/2 system's binary directory, e.g. *c:\bin*.

In the *c:\tcpip\etc\services* file, check that the TFTP port is listed as follows:

```
tftp            69/udp
```

If you do not have a *services* file, or the TFTP port is not listed, the TFTP server will use its built-in default value of 69/udp.

Now create a file which starts the TFTP server, e.g. *c:\bin\tftpd.cmd*, with the following contents:

```
/* TFTP server start script */
'@echo off'
say "Starting TFTPD..."
'tftpd -d /tftpboot -h -r -s 1408 59 -v 2'
exit 0
```

After this, execute the *tftpd.cmd* file to load the TFTP server.

Other TFTP Servers

When using TFTP servers other than the BootManage® TFTP Server, there may be some special things to observe, mainly dealing with different implementation of security issues.

Some systems have security options for the TFTP daemon. Check with your manual on how to activate them. Also verify if the BOOTP daemon has the same options.

For example, some operating systems set options like the following in */etc/inetd.conf*:

```
tftp    dgram   udp   wait    root    /etc/tftpd   tftpd -s /tftpboot
```

This restricts access of TFTP transfers to the directory */tftpboot*.

Some implementations are known not to work in conjunction with the BOOTP daemon. For example, if the user wants to transfer the file */tftpboot/ramd.X*, the TFTP daemon looks for it in */tftpboot/tftpboot/ramd.X* instead of */tftpboot/ramd.X*.

This can be solved by linking the file */tftpboot/ramd.X* into the directory */tftpboot/tftpboot/ramd.X*:

```
# mkdir /tftpboot
# chmod a+rx /tftpboot
# cd /tftpboot
# mkdir tftpboot
# chmod a+rx tftpboot
# cp /tmp/ramd.X .
# chmod a+r ramd.X
# ln ramd.X tftpboot
# _
```

Also, make sure that the TFTP daemon is able to access the file */etc/tftphosts* or */etc/hosts.tftp*. These list all of the hosts which are allowed to access TFTP on your system.

Some systems need to create a user named tftp in order to enable TFTP transfers.

Multicast TFTP

The TCP/IP BOOT-PROM is capable of using IP multicast to receive TFTP data. This dramatically reduces the network load in environments where multiple PC clients are booting simultaneously using the same boot image. To use multicast, your TFTP server must be multicast-enabled.

The Multicast Standard

Multicast is a standard TCP/IP feature that is described in the Internet Request for Comments document RFC1054.

IP multicasting is defined as the transmission of an IP datagram to a host group, i.e. a set of zero or more hosts identified by a single IP destination address. The membership of a host group is dynamic, so hosts may join and leave groups at any time. There is no restriction on the location or number of members in a host group. A host may be a member of more groups at a time.

A host group may be permanent or transient. A permanent group has a well-known, administratively assigned IP address. It is the address that is permanent, not the membership of the group. At any time, a permanent group may have any number of members, even zero.

Using Multicast With TFTP

The TCP/IP BOOT-PROM is able to use multicast when receiving TFTP packets. This allows a multicast-enabled TFTP server to send TFTP packets only once over the network while multiple boot clients collect them. You should consider using multicast in environments where multiple boot clients are turned on simultaneously and also download the same boot image.

Although multicast is standardized in an RFC, doing TFTP multicast is not. At this time, TFTP multicasting is a proprietary extension specific to the TCP/IP BOOT-PROM.

The TCP/IP BOOT-PROM And Multicast

For allowing TFTP multicast, there need not be any changes to the standard TFTP protocol, only to the packet headers under which this data is sent. Also, there are not many changes that need to be made to a standard TFTP daemon, and most of the changes are administrative rather than technical.

The key difference is that all packets go to a specific multicast IP address rather than to the IP address of the requesting client. This multicast IP address is determined by the system administrator, and for every file that can be requested by multicast TFTP, there must be one address.

The TCP/IP BOOT-PROM derives all multicast IP related information from a custom tag in the vendor specific section of the BOOTP or DHCP reply. The following excerpt of a bootptab file shows TFTP multicast usage:

```
# template for classroom 203

class203:\
   :hn:vm=rfc1048:ht=ethernet:sm=255.255.255.0:\
   :hd=/tftpboot:bf=class203.PX:\
   :T160=4d74467450e1000003004b004c0000:


# entries for the PC clients in classroom 203

pc203-1:tc=class203:ha=0000c0595f66:ip=193.141.47.1:
pc203-2:tc=class203:ha=0000c0595f67:ip=193.141.47.2:
pc203-3:tc=class203:ha=0000c0595f68:ip=193.141.47.3:
# ...
```

In this example, the custom tag T160 is used to hold the multicast TFTP configuration information, but you may use any custom tag number starting at 128 that you wish. The TCP/IP BOOT-PROM recognizes the multicast TFTP configuration tag by the magic ID number 4d74467450 (the hexadecimal representation of the ASCII string MtFtP) and not by the tag number itself.

Following this magic ID comes the information that the TCP/IP BOOT-PROM needs to download the boot image using multicast TFTP.

Multicast IP Address

The four bytes directly following the magic ID hold the hexadecimal multicast IP address that the TCP/IP BOOT-PROM should use to listen for TFTP packets.

```
:T160=4d74467450e1000003004b004c0000:
                  ^^^^^^^^
                  multicast IP address
```

In our example, this is e1000003. To get the corresponding IP address in standard decimal notation, you have to split the given number into four bytes (e1.00.00.03) and convert each of these bytes to decimal (225.0.0.3).

Basically, any multicast IP address starting at 224.0.0.0 can be used. Due to reservations, only addresses starting at 225.0.0.0 should be used.

Multicast TFTP Server Port

The next two bytes hold the hexadecimal UDP port number that the TCP/IP BOOT-PROM uses to send requests to the multicast TFTP server.

```
:T160=4d74467450e1000003004b004c0000:
                         ^^^^
                         multicast TFTP server port
```

This must be the same port number that the multicast-enabled TFTP server uses to listen for incoming requests. In our example, the port number is 004b (decimal 75).

Multicast TFTP Client Port

The next two bytes hold the hexadecimal UDP port number that the TCP/IP BOOT-PROM uses to listen for incoming multicast TFTP packets.

```
:T160=4d74467450e1000003004b004c0000:
                             ^^^^
                             multicast TFTP client port
```

This must be the same port number that the multicast-enabled TFTP server uses to send outgoing packets to. In our example, the port number is 004c (decimal 76).

TFTP Transmission Timeout

The next byte holds the hexadecimal TFTP transmission timeout value in seconds.

```
:T160=4d74467450e1000003004b004c0000:
                                 ^^
                                 TFTP transmission timeout
```

If this value is 00 (as in our example), the TCP/IP BOOT-PROM uses the default of two seconds.

TFTP Opening Timeout

The last byte specifies the hexadecimal timeout for the opening of a TFTP transfer in seconds. Before initiating a multicast TFTP transfer, the client will first listen for a certain time if the transmission is already in progress, that is, if another multicast TFTP client has already requested to download the same file by multicast TFTP. If

this is the case, then it is not necessary to initiate another transfer. Instead, the client only needs to collect the multicast TFTP packets as well.

If no multicast TFTP transfer for the desired file is detected within the TFTP opening timeout, the client initiates a new transfer.

```
:T160=4d74467450e1000001004b004c0000:
                                    ^^
                                    TFTP opening timeout
```

If this value is 00 (as in our example), the TCP/IP BOOT-PROM uses the default of four seconds.

TFTP Server Multicast Configuration

Since doing multicast TFTP transfers is not a standard, you need a special multicast-enabled TFTP server. The TFTP server that comes with the TCP/IP BOOT-PROM is multicast-enabled, and by default it uses the file */etc/mtftptab* to assign filenames to multicast IP addresses and client UDP ports.

Within the configuration file */etc/mtftptab*, the first column holds the filename, the second column holds the multicast IP address and the third column holds the UDP destination port to be used. The UDP ports should be the same for all files. Columns are separated by spaces.

```
/tftpboot/class201.PX   225.0.0.1     76
/tftpboot/class202.PX   225.0.0.2     76
/tftpboot/class203.PX   225.0.0.3     76
/tftpboot/teachers.PX   225.0.1.100   76
```

Each line in this configuration file defines a separate multicast IP address assignment. The entries are to be read as follows:

- If a client issues a multicast TFTP request to download the file */tftpboot/class201.PX*, the TFTP server sends this file to destination IP address 225.0.0.1 using destination UDP port 76.
- If a client issues a multicast TFTP request to download the file */tftpboot/teachers.PX*, the TFTP server sends this file to destination IP address 225.0.1.100 using destination UDP port 76.

For multicast IP addressing, the IP address range 224.0.0.0 to 239.255.255.255 has been reserved. These addresses should only be used for multicast and not be assigned to individual hosts.

TFTP Server Command Line Options

Upon startup, the multicast-enabled TFTP server must be supplied with the full pathname to the */etc/mtftptab* file and the UDP port it should use to listen for incoming multicast TFTP requests. For this purpose, the `-m` command line option has been provided:

```
tftpd -m /etc/mtftptab 75
```

The following shows a possible use of the TFTP server:

```
tftpd -d /tftpboot -h -m /etc/mtftptab 75 -r -s 1408 59 -v 2
```

Multicast TFTP is only supported on TCP/IP BOOT-PROMs starting with version 1.52. Also, not all TCP/IP BOOT-PROM types support multicast TFTP. When a specific TCP/IP BOOT-PROM type supports multicast, its driver's version information string is tagged by a capital M (e.g. TCP/IP BOOT-PROM 1.52 WD8ISA 1.52M)

Servers Without Multicast Support

The TCP/IP stack of the server (on which your multicast-enabled TFTP server runs) has to support multicast, as well. If it does not, then the multicast address is seen as a host address in a different network for which the TCP/IP stack does not have a route.

In this case, you can use multicast by employing following workaround:

- Using the `arp -s` command, create a static ethernet-to-IP address mapping for a "broadcast pseudo host". As the IP address, use a free address in your network. As the ethernet address, use FF:FF:FF:FF:FF:FF.
- Use the route add command to create a static route for every multicast network address you use. As the route's destination, specify the IP address of the "broadcast pseudo host".

This way, all packets with multicast IP addresses will be sent as broadcasts on the local network.

Routing And Multicast

If you want to route multicast packets across subnet boundaries, your routers must know what packets to route into what subnets. This is necessary for all multicast IP packets, not only to multicast TFTP packets.

In order to route multicast packets, the routers must support the Intenet Group Multicasting Protocol (IGMP).

Explaining the details of routing IP multicast packets is beyond the scope of this manual. Please refer to the IETF document RFC 1122 titled *Host Extensions for IP Multicasting*.

The BMIMAGE Program

BMIMAGE is a DOS command line program designed to create, restore, and modify DOS boot image files without requiring an actual diskette. BMIMAGE accomplishes this by directly reading and modifying DOS files within the boot image.

Using BMIMAGE, one can

- insert single files or complete directory trees into a DOS boot image.
- extract single files or complete directory trees from a DOS boot image.
- create optimized DOS boot images which only allocate the actual storage space occupied by the included files.
- create DOS boot images of all common diskette formats up to 2.88 MB without the need for an actual boot diskette.
- write DOS batch files that automate the process of creating or updating multiple DOS boot images.

BMIMAGE operates under the following operating systems:

- native DOS
- Windows 3.x, Windows for Workgroups 3.x (DOS Window)
- Windows 95/98/ME
- Windows NT Workstation and Server
- Windows 2000 Professional and Server

Common BMIMAGE Operations

The most common operations of BMIMAGE are to create and restore entire DOS boot images, and to copy single files into and out of a boot image. In the following short tutorial, you will learn how to perform these tasks step by step.

Creating A DOS Boot Image

To create your first DOS boot image, place a bootable DOS system disk in drive A: and invoke BMIMAGE with the -d and -F options:

```
C:\> bmimage -d simple.X -F 1440,A:
```

This instructs BMIMAGE to create a new DOS boot image file named *simple.X* in the current directory. This boot image can hold up to 1.44 MBytes, but its file size is adapted to the required space of the included files. The boot sector and system programs like *IO.SYS*, *MSDOS.SYS*, and *COMMAND.COM* are read from a bootable DOS diskette in drive A:.

Let us have a look at the contents of the just-created DOS boot image file. The -D option instructs BMIMAGE to display the files and directories contained in *simple.X*:

```
C:\> bmimage -d simple.X -D

Filename                                          Size Date      Time  Attrib
-------------------------------------------------------------------------------
\IO.SYS                                         41.055 31.MAY.94 06:22
\MSDOS.SYS                                      38.186 31.MAY.94 06:22
\COMMAND.COM                                    57.377 31.MAY.94 06:22
```

With *simple.X*, you have created a DOS boot image (albeit a very simple one) which can be downloaded using the TCP/IP BOOT-PROM or the PXE Toolkit in order to remotely boot your PC client.

To do something useful, a boot image must contain more than just the minimum DOS system files.

But before adding files and directories to the DOS boot image, we will show you how to use *simple.X* as a reference to create other boot images:

```
C:\> bmimage -d n:\tftpboot\bootfile.X -F 2880,simple.X
```

This creates a new DOS boot image named *bootfile.X* in the subdirectory *tftpboot* of the network drive N:. The boot sector and DOS system files are taken from an already existing boot image (*simple.X*), thus eliminating the need for a physical DOS boot diskette. Note that *bootfile.X* can hold up to 2.88 MBytes, whereas *simple.X* can only hold 1.44 MBytes.

The BMIMAGE Environment Variable

The BMIMAGE program uses the -d option to determine the name and location of the DOS boot image file. As an alternative, the environment variable BMIMAGE can be used for this purpose.

Whenever the BMIMAGE program is invoked without a -d option, it tries to determine the name and location of the DOS boot image file from the environment variable BMIMAGE.

When the -d option is present, the BMIMAGE environment variable is ignored. The following example demonstrates this behaviour.

```
C:\> set BMIMAGE=n:\tftpboot\bootfile.X
C:\> bmimage -D               (displays the contents of bootfile.X)
C:\> bmimage -d simple.X -D   (displays the contents of simple.X)
```

Inserting And Extracting Files

To make our *bootfile.X* boot image do something useful, we have to add several files to it. For the following examples, we assume that you have set the BMIMAGE environment variable as described above and, therefore, you do not need to specify the -d option.

Create a directory named *image* on the root of your local hard disk. Into this directory, copy all of the files and subdirectories you want to have in your boot image. Then, use the following command to insert the entire directory tree into your boot image:

```
C:\bp> bmimage -i c:\image
```

Using the -D option, display the contents of the boot image. Note that the directory *image,* itself, does not appear in the listing, but everything within this directory does. Also, note that the boot image's file size has increased.

Next, we want to access individual files in the boot image. First, insert the file *m:\myfiles\start.bat* into the *network* subdirectory of the boot image:

```
C:\> bmimage -I \network\start.bat,m:\myfiles\start.bat
C:\> bmimage -I \autoexec.bat,c:\image\autoexec.bat
```

If the subdirectory *network* does not exist in the boot image, BMIMAGE will create it.

Then, extract the file *autoexec.bat* from the boot image, edit it with a text editor and insert the edited file back into the boot image:

```
C:\> bmimage -O \autoexec.bat,c:\image\autoexec.bat
C:\> edit c:\image\autoexec.bat
C:\> bmimage -I \autoexec.bat,c:\image\autoexec.bat
```

If the file is already present in the boot image, it will be overwritten.

To delete the file *start.bat* from the *network* subdirectory, use:

```
C:\> bmimage -E \network\start.bat
```

Last, we want to restore the entire contents of a boot image to a directory on the local hard disk:

```
C:\> bmimage -o c:\image2
```

BMIMAGE Command Line Options

The following command line options can be used with BMIMAGE:

Option	Description
-d *file*	The filename of the RAM disk image.
-D	Show directory listing.
-T *file*	Display file contents.
-F *dsize, bsec*	Create a new image with *dsize* Kbytes, where dsize can be one of 320, 360, 640, 720, 1200, 1440 or 2880. *bsec* points to a drive or boot image file that holds the boot sector to use.
-P *dsize*	Enlarge (pad) an existing image by *dsize* Kbytes.
-I *dst*[,*src*]	Copy file *src* from DOS to image file *dst*.
-i *dir*	Copy all files (recursive) from directory *dir* to image
-O *src*[,*dst*]	Copy file *src* from image to DOS file *dst*
-o *dir*	Copy all files (recursive) from image to directory *dir*
-E *file*	delete *file* from image
-C	delete all files on an already existing image
-M *dir*	create a subdirectory *dir* in image
-v	give more information on all activities

The -C Option

The -C option is used to erase all files from a DOS boot image. As the DOS system files are also erased, you can no longer boot from this image. The following example will remove all files from a DOS boot image:

```
bmimage -d ramd.X -C
```

The -d Option

This option identifies the filename of the DOS boot image. The filename can be an existing image or an image that should be created. The filename can also include a drive letter, e.g. *f:\tftpboot\ramd.X*.

If the -d option is not given, BMIMAGE takes the image that is defined by the environment variable BMIMAGE. If neither the -d option is given nor the BMIMAGE environment variable is defined, BMIMAGE cannot locate the image and displays its help message.

The -D Option

The `-D` option shows a recursive directory listing of all files in the DOS boot image:

```
bmimage -d ramd.X -D
```

The -E Option

This option removes a file from the DOS boot image. This example deletes the file *autoexec.bat* from the DOS boot image *ramd.X*:

```
bmimage -d ramd.X -E autoexec.bat
```

This deletes the file *hosts* from the etc directory in the DOS boot image:

```
bmimage -d ramd.X -E \etc\hosts
```

The -F Option

This option formats (creates) a new DOS boot image. The name of the DOS boot image to be created must be defined by the `-d` option or by the environment variable `BMIMAGE`. If the DOS boot image already exists, it will be overwritten.

The first argument specifies the format of the DOS boot image. This can be one out of the following: 160, 180, 320, 360, 640, 720, 1200, 1440 or 2880. Initially, `BMIMAGE` will not allocate the space needed for the complete DOS boot image. Instead, space will be allocated as needed when files are later copied into the DOS boot image.

The second argument points to a file or a drive which holds the bootsector and the system files for the DOS boot image. The two arguments must be separated by a comma:

```
bmimage -d ramd.X -F 1440,A:
```

This creates a 1.44 MB type DOS boot image that takes the bootsector and system files from the diskette in drive A:. The bootsector and system files can also be read out of an existing DOS boot image:

```
bmimage -d ramd.X -F 1440,f:\tftpboot\dos.X
```

If the following system files exist on the diskette or the existing DOS boot image, then they will be copied into the new DOS boot image:

IO.SYS and *MSDOS.SYS*
IBMBIO.COM and *IBMDOS.COM*
COMMAND.COM

The -I Option

This option copies a file into an existing DOS boot image.

The name assigned to the file in the DOS boot image is provided as the first argument and the source of that file is optionally provided as the second argument. If the second argument is omitted, then the first argument identifies both the source and destination, except that the directory path is removed:

```
bmimage -d ramd.X -I config.sys,config
```

This copies the file *config.txt* into the DOS boot image using the name *config.sys*. The following examples do the same:

```
bmimage -d ramd.X -I \config.sys,config.txt
bmimage -d ramd.X -I /config.sys,config.txt
```

This copies the file *c:\files\hosts* into the directory *etc* of the DOS boot image file *ramd.X*:

```
bmimage -d ramd.X -I \etc\hosts,c:\files\hosts
```

This copies the file *hosts* from the current directory into the directory *etc* of the DOS boot image file *ramd.X*. Note that the omitted destination path is defined by using the source filename:

```
bmimage -d ramd.X -I \etc\hosts
```

The -i Option

This copies all of the files in the directory defined by the argument and the files in its subdirectories into the DOS boot image. If you want to copy all of the files from the current directory, use a dot as the argument.

This copies all of the files and directories out of the current directory into the DOS boot image at *f:\tftpboot\ramd.X*:

```
bmimage -d f:\tftpboot\ramd.X -i .
```

This copies all of the files from diskette drive A: into the DOS boot image *ramd.X*:

```
bmimage -d ramd.X -i a:\
```

The -M Option

The `-M` option creates a new subdirectory in the DOS boot image.

Each of these examples creates a new subdirectory *dos* in the DOS boot image:

```
bmimage -d ramd.X -M dos
bmimage -d ramd.X -M \dos
bmimage -d ramd.X -M /dos
```

This creates a new subdirectory *subdir* in the existing directory *dos*. All parent directories must already exist:

```
bmimage -d ramd.X -M \dos\subdir
```

The -O Option

Similar to option `-I` but copies out of the DOS boot image.

The -o Option

Similar to option `-i` but copies out of the DOS boot image.

The -P Option

The `-P` option pads (adds) dummy data to the end of the DOS boot image. This is useful for creating space into which a program may write files in the DOS boot image, which is a RAM disk at runtime. Do not try to exceed the boot image size beyond its maximum size defined with the `-F` option, or the client will fail when trying to boot from this image.

This creates 64 KBytes of additional space in the DOS boot image:

```
bmimage -d ramd.X -P 64
```

The -T Option

The option `-T` shows the contents of a file in the DOS boot image defined by the `-d` option. To display the contents of the file *autoexec.bat* on the screen, use:

```
bmimage -d ramd.X -T autoexec.bat
```

The following example displays the contents of the file *hosts* in the subdirectory *etc* on the screen:

```
bmimage -d ramd.X -T \etc\hosts
```

The -v option

The `-v` option gives more technical feedback about `BMIMAGE` operations. It can be used together with all options or alone.

The following example displays information about the DOS boot image structure when the DOS boot image is created:

```
bmimage -d ramd.X -F 1440,A: -v
```

The BPSHELL Program

BPSHELL is a menu-based DOS program used to:

- create boot image files from physical diskettes
- restore boot image files to physical diskettes
- create and maintain BOOTP configuration files
- create passwords for TCP/IP BOOT-PROM DOS diskette drive protection.
- change TCP/IP BOOT-PROM configuration parameters.
- create and view boot menus for use with the BPMENU loader.
- display TCP/IP BOOT-PROM documentation.

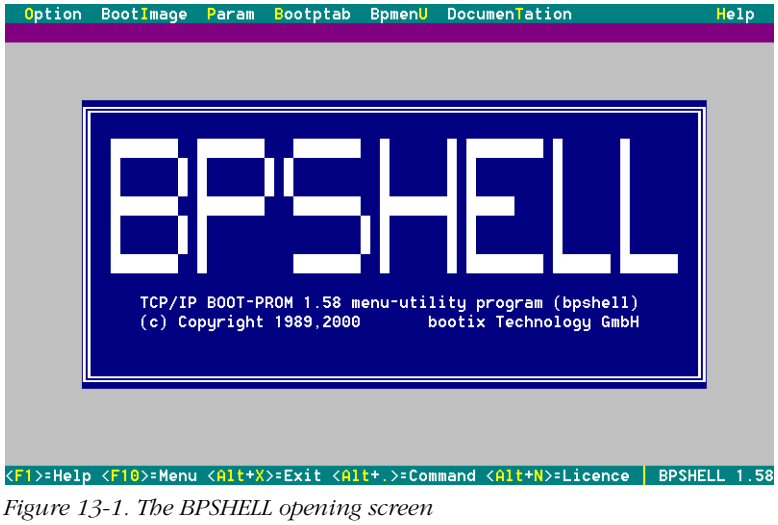


Figure 13-1. The BPSHELL opening screen

Some of BPSHELL's features have been replaced by more sophisticated programs and documentation:

- To perform advanced operations on DOS boot image files or to automate DOS boot image manipulation through batch files, use the BMIMAGE program.
- For advanced editing and checking of BOOTP configuration files, consider using the BootManage® BOOTPtab Editor which is available as a separate product.
- To change TCP/IP BOOT-PROM configuration parameters, the BPCONFIG utility is now available. In order to obtain BPCONFIG, please contact bootix.
- The TCP/IP BOOT-PROM documentation is available in various forms (printed on paper, as Adobe Acrobat PDF file, and as HTML pages on the bootix web site, http://www.bootix.com).

Installing BPSHELL

In order to use BPSHELL, execute the program file *bpshell.exe* and make sure that the help file *bpshell.hlp* is located in the same directory.

The Option Menu

The Option menu offers the selections shown in *Figure 13-2*.

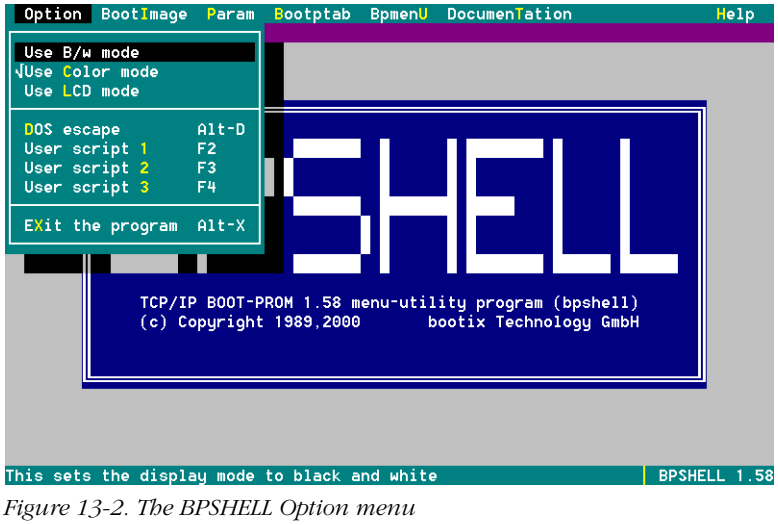


Figure 13-2. The BPSHELL Option menu

From the Option menu, you can perform the following tasks:

Use B/W mode, Use Color mode / Use LCD mode
: The video color mode to be used. On exit this value is stored in the file *bpshell.cnf* and read each time the BPSHELL program is started. If *bpshell.cnf* cannot be read, color mode is used as the default.

  On color systems, select the color mode. The b/w mode uses shades of gray instead of colors. The LCD mode uses white on black and reverse characters.

DOS escape
: The DOS escape temporarily suspends BPSHELL and invokes an MS-DOS command interpreter. Enter exit at the DOS prompt to return to BPSHELL.

User script 1, 2 and 3
: User scripts are MS-DOS batch files which can be called via a function key. BPSHELL searches for these batch files in the same directory where the BPSHELL program is placed. User scripts are named *USER1.BAT*, *USER2.BAT*, and *USER3.BAT*, depending on the function key used.

  These scripts can be used to copy and install RAM disk images to and from the server without leaving BPSHELL.

Exit the program
: Exit BPSHELL and return to MS-DOS.

Create And Restore Boot Images

The BPSHELL program can be used to create and restore boot images. A boot image is a copy of a bootable diskette which contains all the data found on the diskette bundled into a single file. Boot images may also be called RAMdisk images or container files.

Create A Boot Image

Creating a boot image is done in two steps:

1. Create a bootable diskette with the operating system and network software. Examples on how to configure this diskette for various network software are shown later in this manual.
2. Create a boot image file from this diskette. This file is then uploaded and installed on the network server.

The next paragraphs describe how to create a boot image. The procedure can be used to create DOS or other operating system based boot images.

From the BPSHELL menu, select BootImage ➜ Create from diskette. The window shown in *Figure 13-3* can also be entered directly by invoking BPSHELL -C from the system prompt.

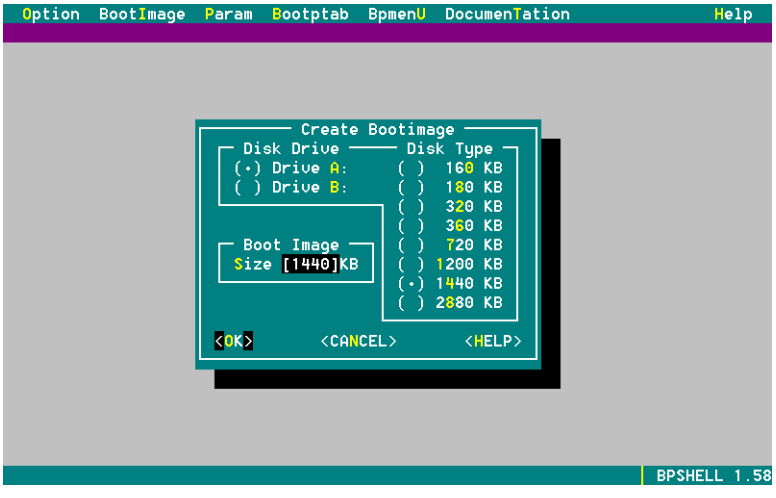


Figure 13-3. Create boot image with BPSHELL

This window permits the selection of the diskette drive and the size to be read. The size of the boot image file can also be set. In detail, these options are:

Disk Drive
: The diskette drive in which the bootable disk is inserted.

Disk Type
: The physical size of the diskette. BPSHELL checks the bootsector of the diskette to determine its characteristics. This default value may be overridden by selecting the size that meets the characteristics of your diskette.

Boot Image
: In order to decrease the time required to download the boot image, you can make the boot image smaller than the size of the diskette. If a value less than the size of the diskette is entered, BPSHELL will only read this number of bytes from the diskette. If a value larger than the size of the diskette is entered, the boot image will be padded with random data.

When modifying the boot image size, be sure that all data resides at the beginning of the diskette. Otherwise, BPSHELL may truncate some of the data. Programs like DEFRAG from MS-DOS 6.0 can be used to reorganize the diskette structure in this manner.

The default is to create a boot image equal to the maximum size of the diskette.

After all the selections have been made, select `OK`. The file selection window now appears as shown in *Figure 13-4*.

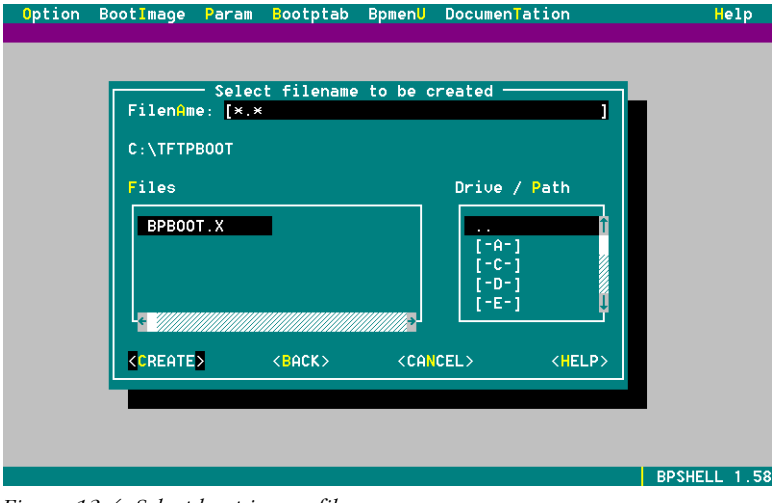


Figure 13-4. Select boot image filename

In the `FilenAme` field, enter the name of the boot image to be created. To overwrite an already existing boot image file, select the existing file in the `Files` subwindow.

Now press `CREATE` to start the boot image creation procedure. You will see a progress bar which indicates the progress of the boot image creation process.

After the boot image file has been created, you can transfer the boot image file to the server by copying it to a network drive or using `ftp`. Or, simply create the boot image directly on an attached or mounted network drive.

Once a boot image has been created, it must be stored on the TFTP server so that the TCP/IP BOOT-PROM client can access it during the network boot process. If you want to modify the boot image, either the original diskette is needed or use `BPSHELL` to re-create the boot image on a diskette. Alternatively, `BMIMAGE` can be used to directly manipulate the individual files contained in the boot image. See *"The BMIMAGE Program"* on page 133 for more information.

Restore A Boot Image

To restore a boot image, invoke the `BPSHELL` program and select the `BootImage` ➜ `Restore` to diskette window. The window shown in *Figure 13-5* can be entered directly by typing `BPSHELL -R` from the system prompt.

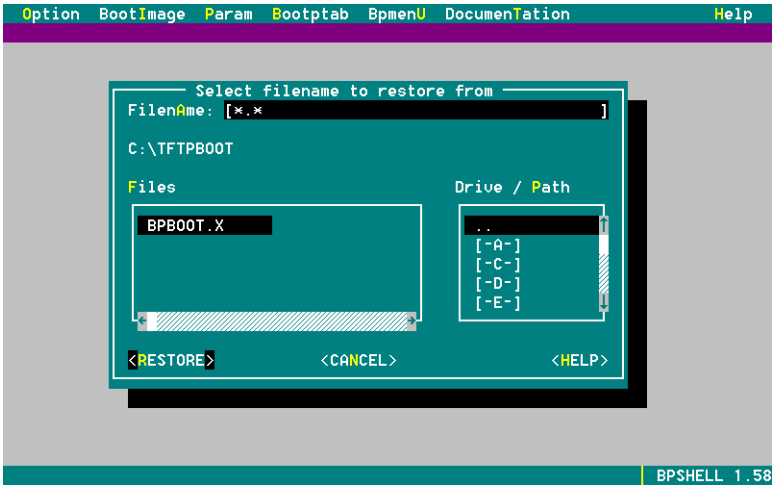


Figure 13-5. Restoring a boot image to diskette

Restoring a boot image to a diskette will overwrite all of the original contents of the diskette.

Select the boot image you want to restore and select `RESTORE`. The `Restore Bootimage` dialog will now appear as seen in *Figure 13-6*.

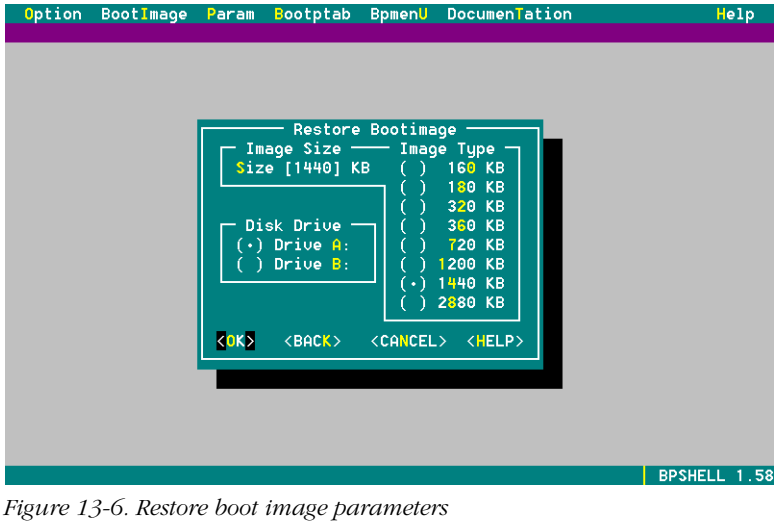


Figure 13-6. Restore boot image parameters

The `Restore Bootimage` dialog allows you to override the default parameters `BPSHELL` set while scanning the diskette drive. These parameters are:

Image size
: The amount of data in kilobytes (1024 byte blocks) to be restored from the boot image file to the diskette. If the value in the size field is less than the size of the boot image file, then the remaining data of the boot image file is ignored.

  If the value in the size field is larger than the size of the boot image file, then the diskette will be padded with random data.

Image type
: The physical size of the diskette. `BPSHELL` checks the diskette in order to determine its size. You may want to overwrite this value for unformatted diskettes.

Disk drive
: The diskette drive in which the diskette is inserted.

Now press `OK` to start the restore procedure. You will see a progress bar which indicates the progress of the boot image restoration process.

TCP/IP BOOT-PROM Parameters

The TCP/IP BOOT-PROM can be configured by a number of parameters. Although the default configuration will work in most environments, you may want to change one or more of the TCP/IP BOOT-PROM configuration parameters. To do so, there are three possibilities:

1. Specify the desired configuration parameters when ordering TCP/IP BOOT-PROMs. This way, you receive the TCP/IP BOOT-PROMs preconfigured with the parameters you specified.
2. With a FLASH programming device, read the TCP/IP BOOT-PROM's contents into a binary file and use the `BPSHELL` program to modify the parameters within this FLASH image file. Then, again with the FLASH programming device, write the modified FLASH image file back into the FLASH.
3. The new `BPCONFIG` utility is capable of changing the TCP/IP BOOT-PROM parameters directly by reprogramming the FLASH in the network adapter. Note that `BPCONFIG` only works if the TCP/IP BOOT-PROM code is stored in a FLASH device (not in an EPROM) and the network adapter also allows programming of the FLASH device.

Using BPSHELL, it is not possible to change the configuration parameters of a TCP/IP BOOT-PROM directly. If you want to do so, you must use the BPCONFIG program instead. To obtain BPCONFIG, please contact bootix.

Configuration Parameters

The TCP/IP BOOT-PROM offers a variety of configuration parameters. We will use BPSHELL's configuration menu to introduce these parameters and to explain their meaning.

We do not recommend to use BPSHELL when configuring TCP/IP BOOT-PROMs. Instead, either specify the configuration parameters when ordering TCP/IP BOOT-PROMs or use the BPCONFIG utility.

BPSHELL's configuration menu is only used for the purpose to explain the available configuration parameters.

To configure the parameters of a TCP/IP BOOT-PROM code image, select Param ➜ Change ROM parameters from the menu bar or invoke BPSHELL -G from the system prompt. In the opening file selector box, either enter the name of the ROM image you wish to modify in the filename field or select one from the files in the sub-window. This must be the name of an existing image.

By default, BPSHELL searches for files with the extension *.ROM*. You can override this by entering something like **.BIN* in the filename field.

The file format of the ROM image is a binary copy of the contents of the ROM. It does not use any checksumed format such as Intel hex.

After pressing RETURN, the PROM configuration dialog appears as shown in *Figure 13-7*.

Startup Procedure

When the TCP/IP BOOT-PROM starts-up it checks for bootable drives using a default sequence. This sequence can be customized by selecting one of the following options:

Disk A: → network
: Check if drive A: is bootable, if not boot from the network.

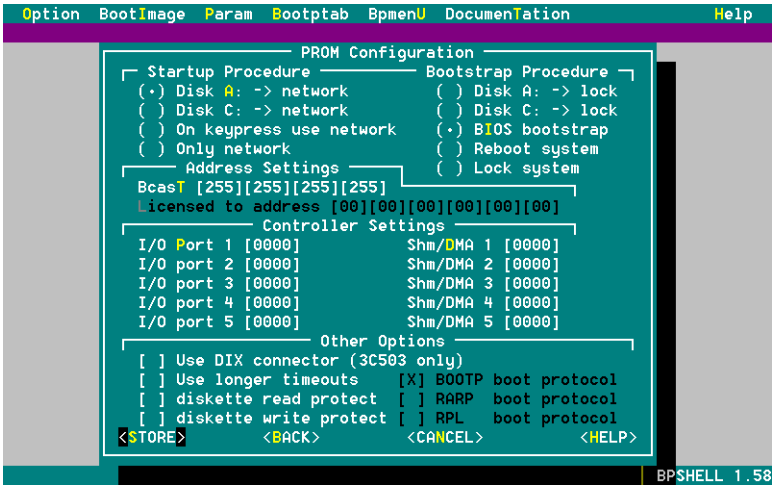


Figure 13-7. TCP/IP BOOT-PROM configuration

Disk C: → network
: Check if drive C: is bootable, if not boot from the network.

On keypress use network
: Only boot from the network if the user presses SPACE.

Only network
: Prevent booting from any drive, allow only network boot.

Bootstrap Procedure

Only if the startup procedure fails[1], the TCP/IP BOOT-PROM checks the bootstrap procedure option about how to continue. The following bootstrap procedure options are available:

Disk A: → lock
: Check if drive A: is bootable otherwise lock the system.

Disk C: → lock
: Check if drive C: is bootable otherwise lock the system.

BIOS bootstrap
: Execute a regular System BIOS bootstrap.

Reboot system
: Reboot the system.

1 For example, the startup procedure may fail if the TCP/IP BOOT-PROM is not able to boot from the network or if the user aborts the network boot by pressing a key.

Lock system
Lock the system. The user must reboot via RESET.

Address Settings

Bcast
The IP broadcast address defaults to 255.255.255.255, that is, all bits are set. Older BSD systems use all bits clear for broadcasts, i.e. the broadcast address is 0.0.0.0. One may also want to change this setting to a subnet broadcast address.

Licensed to address
Some TCP/IP BOOT-PROMs are copy protected. The copy protection embeds the network controller's address in the ROM, making it impossible to use this ROM in any other controller. This setting cannot be changed.

Controller settings

I/O Port 1 ... 5
The TCP/IP BOOT-PROM searches for a network controller at up to 5 i/o port addresses. If the default port addresses do not match your configuration, then they must be changed. Unused addresses are set to zero, and on auto-detecting network drivers all addresses are set to zero.

Shm/DMA 1 ... 5
If the network driver uses shared memory, all possible shared memory segments can be set here. The ROM scans these segments to determine whether they are allocated to another device. If they are not, then the ROM sets the shared memory area to one of these segments. On network controllers with DMA transfer, this setting is used as the DMA channel. Unused segments or channels are set to zero. On auto-detecting network drivers all addresses are set to zero.

Other Options

Use DIX connector (3C503 only)
This setting is only used on the 3Com 3C503 (Etherlink II) controller. By default, the TCP/IP BOOT-PROM uses the BNC connector. If this option is set it uses the DIX connector.

Use longer timeouts
To prevent TFTP time-outs on busy networks or loaded servers the time-out can be increased. If this option is set, the time-out is multiplied by 4 for the TFTP protocols.

Diskette read protect / Diskette write protect
: After a successful network boot the diskette drives can be protected. These settings define the type of protection. For read and write protection both options must be set.

BOOTP / RARP / RPL boot protocol
: This setting defines the protocol that the TCP/IP BOOT-PROM should use. The TCP/IP BOOT-PROM only supports the BOOTP boot protocol. The RARP and RPL protocols are not supported. These settings cannot be changed[1].

After completing all selections, press `RETURN` to write the changes to the ROM image file.

Encode A Password

Diskette drives can be read and write protected after a successful network boot. To remove this protection, a password must be given to `BPUTIL`. These passwords can be generated from the `Param` ➜ `Crypt` window shown in *Figure 13-8*.

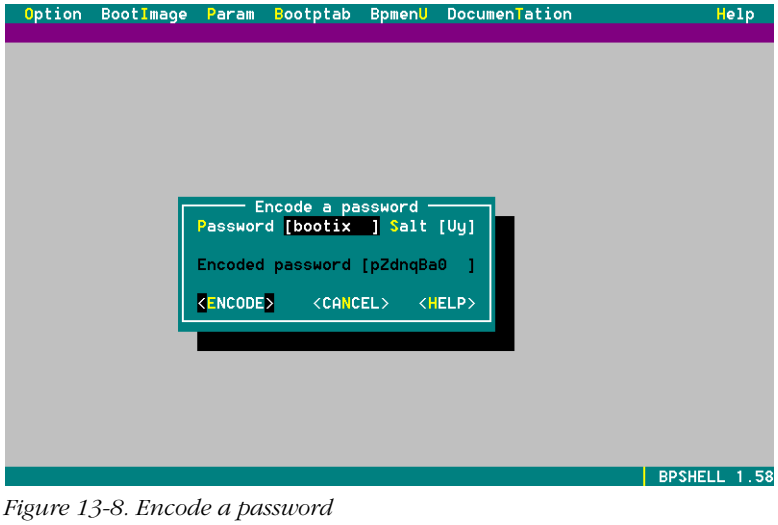


Figure 13-8. Encode a password

The window can be entered directly by typing `BPSHELL -A` from the system prompt.

To encrypt a password, enter it in the password field. A password can only use the characters a-z, A-Z and 0-9.

1 RARP and RPL have been supported by the TCP/IP BOOT-PROM Plus, which is no longer available.

The salt field permits modifications to the password encryption algorithm. BPSHELL will choose two random characters so that two equal password do not have the same encrypted result.

After selecting ENCODE, BPSHELL encrypts the password and displays the encrypted password in the encrypted password field. This can then be used as the password argument to BPUTIL -U U *cpasswd*.

Diskette drive protection only works when the diskette drive is accessed through the BIOS. It does not work for operating systems that use drivers to directly access the diskette drive controller by circumventing the diskette BIOS routines.

The Integrated Editor

Files can be edited from BPSHELL. The Bootptab ➜ Edit menu allows editing of files which follow the */etc/bootptab* structure. Bpmenu ➜ Edit edits BPMENU scripts.

If the edit function is started for the first time, the window shown in *Figure 13-9* appears.

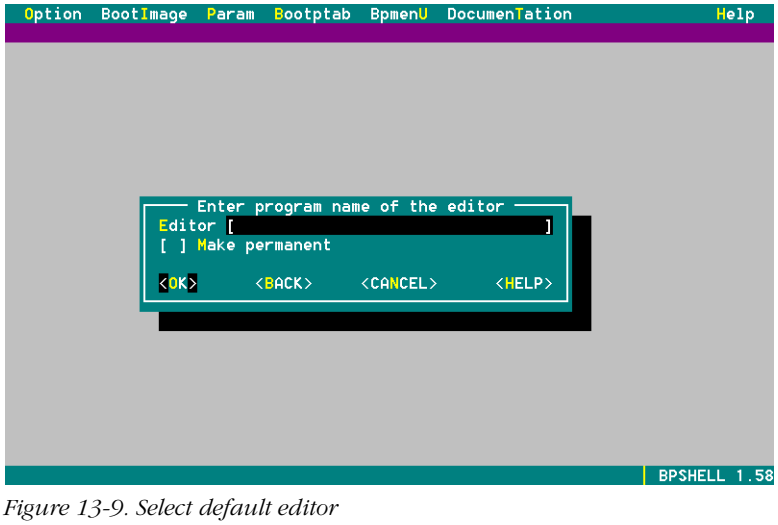


Figure 13-9. Select default editor

The preferred editor is specified in this window. If the editor field is left empty, BPSHELL uses its own built-in editor. Otherwise BPSHELL will call the specified external editor every time an edit menu is selected.

Once set, this option is active in memory for all following calls. To make this setting permanent, set the permanent option. This will write the chosen setting to the *BPSHELL.CNF* file, which is read on subsequent BPSHELL invocations.

The BPSHELL built-in editor has limited capabilities in editing ASCII files.

The maximum number of lines the BPSHELL editor can hold is 750.

Key	Operation
Cursor Left	Move cursor left one char.
Ctrl CurLeft	Move cursor to previous word.
Cursor Right	Move cursor right one char.
Ctrl CurRight	Move cursor to next word.
Cursor Up	Move cursor up one line.
Cursor Down	Move cursor down one line.
Home	Move cursor to first character in current line.
Ctrl Home	Move cursor to first line.
End	Move cursor to last character in current line.
Ctrl End	Move cursor to last line.
Ins	Toggle between insert and delete mode.
Ctrl Y	Delete current line.
Del	Delete character under cursor.
Backspace	Delete character before cursor.
F6	Start/stop selecting text.
Shift F6	Hide text selection.
Ctrl K	Copy selected text to current cursor position.
Ctrl L	Delete selected text.
Ctrl V	Move selected text to current cursor position.
ESC	Exit the editor without saving.
F1	Show help text.
F2	Zoom/unzoom editor window.
Shift Tab	Leave editor and step to next button.

Display BPMENU Script

BPSHELL can display BPMENU scripts. By selecting the Bpmenu ➜ Display windows, selected scripts can be displayed as shown in *Figure 13-10*.

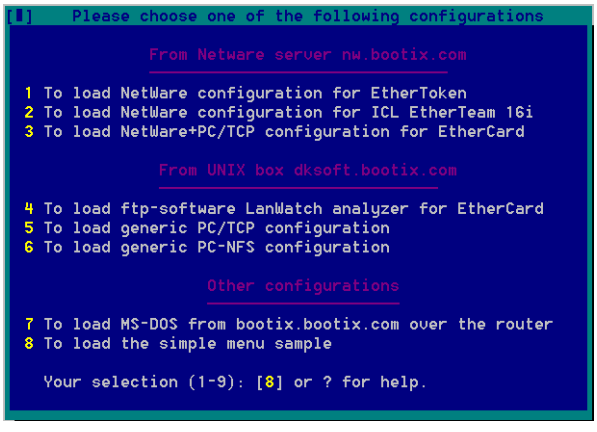


Figure 13-10. Sample BPMENU script

Typing BPSHELL -S from the command line enters the BPMENU display window directly.

By default, all BPMENU scripts have the extension *.M*. This must not be changed because the BPMENU script interpreter program only accepts this extension.

BPSHELL can only interpret the output routines of a BPMENU script. It does not allow keyboard input or RAM disk image downloads. This is useful for debugging or writing BPMENU scripts without rebooting the diskless Personal Computer after each change.

After a script is displayed press RETURN to return to BPSHELL.

BPSHELL Command Line Options

You can start BPSHELL with one of the following command line options:

```
  Version: TCP/IP BOOT-PROM 1.58 menu-utility program (bpshell)
Copyright: (c) Copyright 1989,2000       bootix Technology GmbH

Usage    : bpshell [-a|c|e|g|m|n|r|s]
Where    :
    -a   = Encode a password
    -c   = Create a RAM disk image
    -e   = Edit a bootptab file
    -g   = Change ROM parameters
    -m   = Modify a bootptab file
    -n   = Read License Agreement
    -r   = Restore a RAM disk image
    -s   = Display a BPMENU script

No argument starts BPSHELL in interactive mode
```

This causes BPSHELL to directly come up with the corresponding menu. Note that you can not control BPSHELL entirely from the command line. After coming up with the selected menu, BPSHELL has to be used interactively.

The BPUTIL Program

The BPUTIL program is a multi-purpose program that works in conjunction with the TCP/IP BOOT-PROM to:

- Restore memory allocated by the TCP/IP BOOT-PROM
- Patch information given by the BOOTP reply into ASCII and binary files
- Transfer files from the network server to the client using the TFTP protocol
- Protect diskette drives after a remote boot
- Use the TCP/IP BOOT-PROM with EMM386 and HIMEM.
- Restart the remote boot client and other miscellaneous functions.

BPUTIL is available in four different versions:

- BPUTIL.COM, a DOS program that can be executed from within batch files such as *AUTOEXEC.BAT*.
- BPUTIL.SYS, a DOS device driver that is loaded from *CONFIG.SYS*. After completion, the driver releases all of its resources so that memory is not permanently allocated.
- BPUTIL.144, a special binary loader version to patch the *CONFIG.SYS* file of 1.44 MByte boot image files.
- BPUTIL.288, a special binary loader version to patch the *CONFIG.SYS* file of 2.88 MByte boot image files.

The two special binary loader versions BPUTIL.144 and BPUTIL.288 have been designed to patch the *CONFIG.SYS* file before it is executed by the DOS kernel. If you need to do this, see *"BPUTIL.144 and BPUTIL.288"* on page 173.

BPUTIL Command Line Options

All command line options use the leading character "-", e.g. `-x`. Options cannot be stacked, i.e. `-x -f`; multiple calls must be made instead. The following table shows all the BPUTIL options in alphabetical order:

Option	Description
`-a` *fnam*	Patch the BOOTP reply into ASCII file.
`-b` *tnum fnam*	Patch the BOOTP reply into binary file.
`-c`	Display the TCP/IP BOOT-PROM configuration
`-C`	Set ERRORLEVEL if the PC was not booted over the network.
`-d` *dnum*	Checksum the diskette drive.
`-e`	Reboot the PC via System BIOS call.
`-E`	Reboot the PC via `Ctrl - Alt - Del`
`-f`	Fix MS-DOS memory allocation for use with EMM386.
`-h`	Install the XMS handler entrypoint.
`-i` *fnam*	Include a file within BOOTP reply to extend the BOOTP reply block.
`-i` [*gip*] *sip fn*	Retrieve a file via TFTP and include it within the BOOTP reply to extend the BOOTP reply block.
`-m`	Display MS-DOS memory control blocks.
`-o` *drv*	Set the boot drive
`-p` *fnam*	Same as -a but uses a space to separate IP addresses.
`-r`	Restore TCP/IP BOOT-PROM memory and diskette drive A:.
`-s` [*tag* [*tags*]]	Display all tags from the BOOTP reply. If one or more tags are specified, display only those tags.
`-S`	Display only those tags from the BOOTP reply that have a value assigned.
`-t` [*gip*] *sip src dst*	TFTP transfer file *src* to file *dst* from the network server with IP address sip.
`-u r|w|rw|u` *pwd*	Protect or unprotect diskette drives.
`-v` *addr32*	Relocate the RAM disk to memory address addr32.
`-x`	Protect the RAM disk via HIMEM.
`-z`	Remove RAM disk but not the TCP/IP BOOT-PROM memory.
`-?`	Display the BPUTIL options.

All arguments to `BPUTIL` can be passed using tags. If `BPUTIL` finds that an argument includes the special ID "#@", it first looks up that tag in the BOOTP/DHCP reply and then uses the resulting string as its argument.

For example:

```
bputil -t 193.141.47.197 /tftpboot/network.drv network.drv
```

transfers the file */tftpboot/network.drv* from the server 193.141.47.197 to the current drive. You can now specify the name of the file to be transferred by storing the filename in a tag in the BOOTP reply:

```
bputil -t 193.141.47.197 #@t130################ network.drv
```

`BPUTIL` will then lookup the value of the tag t130 and use its contents as the filename to be transferred. This can be used for variables in *CONFIG.SYS* because *CONFIG.SYS* cannot be patched before it is being executed.

Also, the network driver to be loaded can be remotely configureable:

```
device=bputil.sys -t #@sip############ #@t130############## netdrv.sys
device=protman.sys
device=netdrv.sys
```

The file *netdrv.sys* must already exist as a dummy or empty file before you can overwrite it using the `-t` option. Otherwise, MS-DOS cannot find the file while executing from *CONFIG.SYS*.

Restoring Occupied Memory

The -c Option

This option is intended for debugging only. The `-c` option returns the location of the BOOTP reply, original System BIOS boot and disk vectors, and the RAM size detected by the System BIOS at system start.

```
bootp= 9400:2292 (hex)
bootv= c800:102e (hex)
diskv= c800:10de (hex)
ramds= 00110000 (hex memory)
ramdl= 2791 (512 byte sectors)
ramsz= 640
xms  = 15360
xentr= 0000:0000
xhand= 43262
iop  = 0280 (hex)
shm  = d000 (hex)
dma  = 0000 (hex)
eth  = 0001 (hex)
feat = 821b (hex)
ver  = 1.48
```

The meaning of the fields are:

Variable	Description
bootp	Memory location of the BOOTP reply (512 bytes).
bootv	Entrypoint of the original System BIOS boot vector.
diskv	Entrypoint of the original System BIOS disk vector.
ramds	Memory location (32 bits address) of the RAM disk data.
ramdl	Length of the RAM disk in sectors (512 bytes).
ramsz	Original System BIOS Base Memory size.
xms	Flag if XMS is used.
xentr	Entry point of the XMS handler (HIMEM).
xhand	XMS handle.
iop	I/O port used on the network controller.
shm	Shared memory segment used on the network controller.
dma	DMA channel used on the network controller.
eth	Flag if the 3C503 does use BNC connector.
feat	Coded features bits of the TCP/IP BOOT-PROM.
ver	The version of the ROM (not the version of BPUTIL).

The -C option

The `-C` option checks if the PC was booted over the network using the TCP/IP BOOT-PROM. If the PC was booted over the network, the MS-DOS environment variable `ERRORLEVEL` is set to 0. Otherwise, `ERRORLEVEL` is set to 1.

The -r Option

Use the `-r` option of `BPUTIL` to remove the RAM disk drive and restore the memory used by the ROM and the RAM disk. The `-r` option also restores the physical diskette drive A:.

If -r is used, the RAM disk is removed and replaced by the physical diskette drive A:. If no diskette is inserted in this drive, MS-DOS will prompt with a read error. To avoid this, call -r only when the current directory is on a drive different from A:.

If you are unable to load programs into high memory or the result returned by the `MEM /C` command no longer displays programs which are loaded into high memory, see the `-f` option for more information.

The -z Option

The `-z` option is equivalent to the `-r` option but does not restore the memory used by the TCP/IP BOOT-PROM. It only restores the memory of the RAM disk and the physical diskette drive.

It is not recommended that one use this option because it does not free the Base Memory allocated by the TCP/IP BOOT-PROM.

This option is intended for debugging only.

Patching Files

The number of bytes that can be used by tags and pre-defined vendor fields is limited to 64 bytes. This is specified by the RFC1048 BOOTP protocol. BPUTIL will issue a warning if the size of the BOOTP/DHCP reply exceeds the protocol definition. The -i option overcomes this limitation. The TCP/IP BOOT-PROM and BPUTIL are able to use vendor fields of up to 1024 bytes if the BOOTP/DHCP server is able to support longer vendor fields.

The -a Option

`BPUTIL -a` patches information given by the BOOTP server into ASCII files. The `BPUTIL` program searches within the given file for a special ID (the string `#@`) and replaces it with a string defined by the tag name following the @ character.

For example the file

```
# /etc/hosts Internet address database
#
127.0.0.1       localhost local
#@yip########## #@chn###########
```

contains the two tags `#@yip` and `#@chn`. When running `BPUTIL -a` on the file, the two tags will be replaced by strings given by the BOOTP reply. The `#@chn` will be replaced by the hostname of the remote boot client PC, the tag `#@yip` will be replaced by the Internet address. The length of the field is given by the number of `#` characters and the length of the tag.

If the replacement string is longer than the length of the field, it will truncate the string to meet the requested size. If the string is smaller it will be padded with spaces.

After patching the above file, it might look like:

```
# /etc/hosts Internet address database
#
127.0.0.1       localhost local
128.1.1.99      bozo
```

The filename must always contain the drive's letter and the full path (i.e. a:\etc\hosts). Also, make sure that every line of the patched file is terminated by an end-of-line (CR + LF), especially the last line of the file.

The following tags can be used with the `BPUTIL` program:.

Tag	RFC951 BOOTP Reply
`#@bfn#`	Bootfilename.
`#@caa#`	Client hardware address.
`#@ca8#`	Last 8 digits of client hardware address.
`#@cha#`	Client hardware address, separated by dots.
`#@gip#`	BOOTP gateway IP address, not the IP gateway IP address (gwf) !
`#@shn#`	Server hostname.
`#@sip#`	Server IP address.
`#@yip#`	Your (client) IP address.

Tag	RFC1048 BOOTP Vendor Field
`#@bfs#`	Bootfile filesize.
`#@is0#`	First impress server IP address.
`#@isf#`	All impress server IPs, separated by spaces.
`#@ds0#`	First DNS domain name server IP address.
`#@ds1#`	Second DNS domain name server IP address.
`#@dsf#`	All DNS domain name server IPs, separated by spaces.
`#@gw0#`	First gateway IP address.
`#@gw1#`	Second gateway IP address.
`#@gwf#`	All gateway IPs, separated by spaces.
`#@ls0#`	First log server IP address.
`#@lsf#`	All log server IPs, separated by spaces.
`#@lp0#`	First line printer server IP address.
`#@lpf#`	All line printer server IPs, separated by spaces.
`#@mdf#`	Merit dump file.
`#@ns0#`	First name server IP address.

Tag	RFC1048 BOOTP Vendor Field
#@ns1#	Second name server IP address.
#@nsf#	All name server IPs, separated by spaces.
#@qs0#	First quote/cookie server IP address.
#@qsf#	All quote/cookie server IPs, separated by spaces.
#@rs0#	First RLP server IP address.
#@rsf#	All RLP server IPs, separated by spaces.
#@smf#	Subnet mask.
#@tof#	Time offset field.
#@ts0#	First time server IP address.
#@tsf#	All time server IPs, separated by spaces.
#@txxx#	User defined tag number xxx.
#@Txxx#	User defined tag number xxx, / is converted to \.

Tag	RFC1084 BOOTP Vendor Field
#@chn#	Client host name.

Tag	RFC1395 BOOTP Vendor Field
#@cdn#	Client domain name.
#@rfn#	Root pathname.
#@swp#	Swap server IP address.

Tag	RFC1497 BOOTP Vendor Field
#@efn#	Extensions path.

Tag	RFC1533 BOOTP Vendor Field
#@bca#	Broadcast IP address.
#@nid#	NIS domain name.
#@ni0#	First NIS server IP address.
#@nif#	All NIS server IPs, separated by spaces.
#@nt0#	First ntp server IP address.
#@ntf#	All ntp server IPs, separated by spaces.

Tag	Miscellaneous Parameters
#@iop#	I/O port used by the PROM (hex).
#@dhc#	Flag indicating which protocol was used: BOOTP (0) or DHCP(1).
#@dma#	DMA port used by the PROM (hex).
#@eth#	PROM used BNC (1) or AUI (0) port on 3C503.
#@shm#	Shared memory address used by the PROM (hex).

Tag	Miscellaneous Parameters
#@typ#	Type of controller (hexadecimal number).
#@ver#	TCP/IP BOOT-PROM code version number

Not all of the above tags may be defined on your system. The number of available tags depends on the features of the BOOTP daemon used.

The alignment of a string replacing a tag can be specified. Fields like #@t123####### are replaced with a left aligned string. A '-' sign after the tag specifies left and a '+' sign right alignment. A '*' sign will left align and move all spaces to the end of the line.

If no sign is given, left alignment is default. This option can be used to append filename extensions to tags:

```
#@t128###################
#@t128-##################
#@t128+##################
#@t130##.COM
#@t130-#.COM
#@t130+#.COM
#@t130*#.COM specifies the filename
#@chn+########|#@chn*########|
My hostname is #@chn*########. I am diskless !
Concatenated: #@chn*#########@chn*########
```

This will be changed to:

```
/usr5/bp/dos
/usr5/bp/dos
            /usr5/bp/dos
HELLO    .COM
HELLO    .COM
   HELLO.COM
HELLO.COM specifies the filename
      diskless|diskless|
My hostname is diskless. I am diskless !
Concatenated: disklessdiskless
```

BPUTIL does not change the size of a patched file in order to be compatible with DOS when BPUTIL is executed from CONFIG.SYS. Therefore, all spaces are moved either to the end of line or padded to the end of the field.

The -b Option

The `-b` option can be used to patch binary information into a file. The usage of the binary patch option is "-b tagnum filename" where filename is the name of the binary file (program, driver...) to be patched. Tagnum specifies the tag which contains the information that is to be patched into the file.

The tag is interpreted in a special way. The first three bytes contain in network order (most-significant byte first), the file offset where the data is to be placed. The following bytes contain the data, itself. Note that the file offset begins with 0, that is, 0 is the file offset of the first byte in a file.

Example: The serial number of the PC-NFS *PCNFS.DRV* driver has to be changed before it is loaded. First, determine if there are differences between the two *PCNFS.DRV* files using the MS-DOS FC command:

```
C:\> fc /b pcnfs.1 pcnfs.2
Comparing files PCNFS.1 and PCNFS.2
00000031: 31 32
00000032: 32 23
00000033: 33 36
00000034: 34 33
00000035: 35 33
00000036: 36 34
00000037: 37 37
00000038: 38 36
C:\> _
```

The `BPUTIL` program does not create or remove serial number protection from programs. It only patches information into files. This information can be a serial number which already exists.

Assume that the serial number starts at file-offset 49 (hex 0x31) and the serial number is ASCII 12345678. First, all numbers must be converted to hexadecimal notation and then added to the */etc/bootptab* file:

- file offset 49 = 000031 (hex)
- ASCII 12345678 = 31 32 33 34 35 36 37 38 (hex)

Thus, the entry in the */etc/booptab* file would be:

```
diskless:\
    :tc=pcnfs:ha=0000c0e23324:ip=193.141.47.198:\
    :bf=pcnfs.X:T132=0000313132333435363738:
                      \____/\_____________/
                        |          |
                      file    binary patch
                     offset   information
```

The BPUTIL program must be called from *CONFIG.SYS* in order to patch the serial number before the *PCNFS.DRV* is loaded:

```
device=a:\bin\bputil.sys -b 132 a:\nfs\pcnfs.sys
device=a:\nfs\pcnfs.sys
```

The -i Option

The -i option includes a file into the BOOTP reply in order to overcome the limited size of the BOOTP vendor field. The format of the included file is the same as the BOOTP vendor field defined in RFC 1497. This option may be used to include a file given by the 'Extensions Path' tag.

BPUTIL option -t can be used to transfer a file given by a user tag or by the efn tag from BOOTP. Tags that are already defined in RFC 951's vendor unique field are overwritten by tags in the file. Tags that are not defined are added.

The -i option can be used in conjunction with the -a, -b, -s and -S options. It must be the first option if multiple options are used.

Example:

```
bputil -t 193.141.47.193 /tftpboot/bootp.add bootp.add
bputil -i bootp.add -a a:\startup.bat
```

The -s Option

The reply package given by the BOOTP daemon can be displayed by using the -s option of the BPUTIL program. This option displays all of the fields available to the BPUTIL program and any additionally defined tag fields.

```
A:\> bputil -s
set bfn=/usr/bp/img/pcnfs.X
set cdn=
set cha=00.00.c0.3e.3c.40
set chn=opus
set dma=0000
set ds0=
set dsf=
set eth=0001
set gf0=
set gip=193.141.47.193
set gw0=
set gwf=
set iop=0280
set lp0=
set lpf=
set ns0=
set nsf=
set rfn=
set shm=d000
```

```
set shn=
set sip=193.141.47.193
set smf=255.255.255.0
set swp=
set tof=
set ts0=
set tsf=
set typ=0009
set yip=193.141.47.198
set t128=dksoft
set t130=/usr/bp/pcnfs
set t132=[11]0000313233343536373839
set dmp1=638253630104ffffff000304 ... c046f7075738006646b736f6674820d
set dmp2=2f7573722f62702f70636e66 ... 3132233343536373839ff0000000000
A:\> _
```

You can also specify only the variables you want to see:

```
A:\> bputil -s yip sip
set yip=193.141.47.198
set sip=193.141.47.193
A:\> _
```

The format is designed to work in conjunction with the MS-DOS command interpreter so that environment variables can be easily set:.

```
A:\> bputil -s yip sip > tmp.bat
A:\> call tmp.bat
A:\> _
```

The `-s` option also displays binary tags. If a tag contains binary characters (less than ASCII 32 or greater than ASCII 127), the tag is displayed as follows:

```
t132= [5] f0023489f1
```

Where [5] indicates that the length of the tag is 5 bytes. The bytes themselves follow in hexadecimal notation.

The last two fields dmp1 and dmp2 of the `BPUTIL` output show a hex-dump of the BOOTP reply vendor field.

The -S Option

The `-S` option lists the same tags as the -s option but displays only those which contain data.

The -p Option

The `-p` option is equivalent to the `-a` option but uses spaces instead of dots to separate IP and hardware addresses, e.g. `-a` produces 193.141.47.193, whereas `-p` produces 193 141 47 193.

This is useful for Digital's PathWorks.

Downloading Files By TFTP

The -t and -T Options

These options both use the PROM's built-in TFTP feature to download a file from a TFTP server. This allows you to retrieve additional files from a network server without having to install a network redirector.

When using the `-t` option, the transfer can be aborted by pressing a key. The `-T` option ignores keyboard input and, therefore, cannot be aborted.

This option should not be used after the network controller driver has been loaded because the `-t` and `-T` options initialize the network controller.

Three arguments are given to the `-t` option. These are:

- Server IP address.
- Server filename.
- Local filename.

The server IP address must be given in decimal format. All filenames are converted to lowercase since the device driver version of `BPUTIL` cannot handle lower/uppercase arguments due to limitations in MS-DOS.

Here are some sample uses:

```
device=a:\bin\bputil.sys -t 193.141.47.193 /etc/passwd pwords
```

If a gateway is to be used, it must precede the TFTP server IP address:

```
bputil -t 193.141.47.199 128.1.3.12 /tftpboot/foo foobar
```

This will download the file */tftpboot/foo* from TFTP server 128.1.3.12 using the gateway or router 193.141.47.199. If no gateway is specified, `BPUTIL` expects the TFTP server to be on the local subnet:

```
bputil -t 193.141.47.197 /tftpboot/foo foobar
```

The gateway can be determined by the BOOTP reply. The #@gw0 tag holds the first gateway IP address that is given by the BOOTP reply:

```
bputil -t #@gw0############### 128.1.3.12 /tftpboot/foo foobar
```

If no gateway is given, the #@gw0 tag is empty.

Protecting Diskette Drives

The -u Option

Diskette drives can be protected after a remote boot. Three protection levels are possible:

- Protect for reading.
- Protect for writing.
- Protect for reading and writing.

The protection level can be set by the `BPSHELL` program. Additionally, `BPUTIL` can remove and add protection levels via the `-u` option. Four arguments can be given to the `-u` option:

Option	Description
`-u r`	Add read protection.
`-u w`	Add write protection.
`-u rw`	Add read and write protection.
`-u u` *`cpasswd`*	Remove all protections.
`-u u` *`tagnum`*	Remove all protections.

Protections are defined simultaneously for diskette drive A: and B:. You cannot protect the TCP/IP BOOT-PROM's RAM disk or fixed disk drives.

Here is an example which adds read and write protection to both diskette drives from *CONFIG.SYS*:

```
device=a:\bin\bputil.sys -u rw
```

The remove option needs a password for validation. The password is encrypted and can be created using the BPSHELL program. In our example, the encryption of the password "dirk" is "Pama5si", thus, the encrypted password of "dirk" can be written as follows:

```
device=a:\bin\bputil.sys -u u Pama5si
```

or placed in a BOOTP user-defined tag. In the latter case the tag number must be named, e.g.:

```
device=a:\bin\bputil.sys -u u 199
```

and the */etc/bootptab* should contain an entry such as:

```
diskless:\
    :tc=pcnfs:ha=0000c0e23324:ip=193.141.47.198:\
    :bf=pcnfs.X:T199="Pama5si":
```

In both cases the BPUTIL program prompts the user for a password. If the password entered does not match the encrypted one, the protection is not removed:

```
A:\>bputil -u u Pama5si
Enter password : xxxx
BPUTIL: incorrect password !
A:\> _
```

EMM386.EXE and HIMEM.SYS

The -f Option

If the Upper Memory can no longer be accesed after restoring the RAM disk and diskvectors with the -r option (i.e. MEM /C does not show UMBs), then use the -f option.

Execute the -f option before loading HIMEM and EMM386:

```
switches=/w
dos=high,umb
device=a:\bin\bputil.sys -f
device=a:\dos\himem.sys /testmem:off
device=a:\dos\emm386.exe NOEMS /y=c:\dos\emm386.exe
device=a:\bin\bputil.sys -x
```

The -f option installs the TCP/IP BOOT-PROM as a TSR program so that EMM386 does not overwrite its memory space.

The -x Option

To prevent other MS-DOS programs from overwriting the RAM disk image in Extended Memory, use the -x option. An extended memory service driver such as EMM386 must also be installed.

The -x option reserves the space used by the RAM disk image, and must be set immediately after the XMS driver is loaded. A sample fragment of the *CONFIG.SYS* file could be:

```
device=a:\dos\himem.sys /testmem:off
device=a:\bin\bputil.sys -x
```

The -r option of the BPUTIL program frees any Extended Memory in use. Using the -x option on an XT or non-Extended Memory based system causes BPUTIL to ignore this option.

If you do not use the -x option, the RAM disk image could be overwritten by other MS-DOS programs. The ROM will issue a warning if it was unable to protect the RAM disk image but will continue operating as if the RAM disk image were protected.

Miscellaneous BPUTIL Options

The -e and -E Options

The `-e` option reboots a Personal Computer by calling the System BIOS reboot routine. The `-E` option reboots the Personal Computer by inserting the key sequence `Ctrl - Alt - Del` into the keyboard buffer.

Both options can be used in an MS-DOS batch file if the installation of the network software failed.

The -d Option

To verify the RAM disk image, checksum it via the `-d` option. The `BPUTIL` program then checksums the RAM disk in blocks of 10 cylinders. This option is useful to determine if the RAM disk has been overwritten by another program.

For example:

```
device=a:\bin\bputil.sys -d 0
device=a:\memhack.sys
device=a:\bin\bputil.sys -d 0
```

checksums the RAM disk image and then loads a program which may overwrite the memory. The second call to `BPUTIL` gives the checksum of the RAM disk image after the installation of the program *MEMHACK.SYS*. If the checksums differ, then the RAM disk should be moved to another location via the `-v` option. The argument to the `-d` option defines the driver on which a checksum is to be made. Thus, a 0 checksums drive A:, a 1 drive B:.

The -h Option

This option is intended for debugging, only. If the System BIOS memory move function fails, the TCP/IP BOOT-PROM can be configured to use the `HIMEM` memory copy function.

The `-h` option installs the XMS handler entrypoint of the TCP/IP BOOT-PROM.

The -m Option

This option in intended for debugging, only. The MS-DOS memory control blocks can be monitored by the `-m` option:

```
A:\> bputil -m
id=M, maddr=0x000025b0, psp=0x0008, size=0xd53c, len=0x02b0, owner=
id=M, maddr=0x000050c0, psp=0x0008, size=0xd53c, len=0x0004, owner=
id=M, maddr=0x00005110, psp=0x0512, size=0x9400, len=0x00a4, owner=
id=M, maddr=0x00005b60, psp=0x0000, size=0x097f, len=0x0004, owner=
id=M, maddr=0x00005bb0, psp=0x0512, size=0x9400, len=0x0040, owner=
id=M, maddr=0x00005fc0, psp=0x0000, size=0x097f, len=0x0007, owner=
id=M, maddr=0x00006040, psp=0x0605, size=0x0738, len=0x0133, owner=
id=M, maddr=0x00007380, psp=0x097f, size=0x9400, len=0x0008, owner=
id=M, maddr=0x00007410, psp=0x0742, size=0x097e, len=0x023c, owner=
id=M, maddr=0x000097e0, psp=0x097f, size=0x9400, len=0x1000, owner=
id=M, maddr=0x000197f0, psp=0x0000, size=0x097f, len=0x7a80, owner=
id=Z, maddr=0x00094000, psp=0x9400, size=0xfe94, len=0x0bfe, owner=
id=M, maddr=0x0009fff0, psp=0x0008, size=0xd53c, len=0x353a, owner=
id=M, maddr=0x000d53a0, psp=0x0008, size=0xd53c, len=0x126b, owner=
id=M, maddr=0x000e7a60, psp=0x0000, size=0x097f, len=0x0007, owner=
id=M, maddr=0x000e7ae0, psp=0xe7af, size=0xe8d3, len=0x0124, owner=
id=Z, maddr=0x000e8d30, psp=0x0000, size=0x097f, len=0x072c, owner=
BIOS RAM size= 640 K byte
XMS available= 13624 K byte, XMS largest= 13624 K byte
A:\> _
```

The -o Option

When the TCP/IP BOOT-PROM is used to boot the PC from a DOS boot image, DOS internally records that its boot drive is “A:”. The `-o` option is used to change the DOS boot drive, that is, it tricks DOS into thinking that it was booted from a different drive (e. g. C:).

This is useful when you diskless booted Windows 3.x and, after removing the TCP/IP BOOT-PROM RAM disk, want to format a floppy diskette in drive A:. When Windows complains that “the boot drive cannot be formatted”, use the following command before removing the TCP/IP BOOT-PROM RAM disk:

```
bputil -o c
```

The -v Option

The `-v` option is used to move the RAM disk image within the PC's memory. It takes a 32-bit memory address as its argument. The RAM disk image is then moved to this location.

For example:

```
device=a:\bin\bputil.sys -v 200000
```

This moves the RAM disk image from its current location to the hexadecimal memory address 0x200000 (2 Mbyte).

BPUTIL.144 and BPUTIL.288

As opposed to `BPUTIL.COM` and `BPUTIL.SYS`, the binary loader versions `BPUTIL.144` and `BPUTIL.288` do not operate from within the boot image. Instead, they must be merged with the boot image. This is necessary so that the *CONFIG.SYS* file can be patched before it is processed by the DOS kernel.

You only need to make use of the binary loader versions of `BPUTIL` if you must patch the CONFIG.SYS file. In any other case, use `BPUTIL.SYS` and/or `BPUTIL.COM` instead.

To merge the binary loader with the boot image, you can use the DOS `COPY` command in binary mode as in the following example:

```
copy /b bputil.144 + /b bootimg.X bootimg2.X
```

This will take the existing 1.44 MByte boot image named *bootimg.X*, prepend it with the binary loader *bputil.144* and store the new boot image as *bootimg2.X*.

Similarly, for a 2.88 MByte boot image, use:

```
copy /b bputil.288 + /b bootimg.X bootimg2.X
```

After the binary loader and the boot image have been merged, you can no longer maintain the merged boot image using `BMIMAGE` or `BPSHELL`. Instead, you have to create a new boot image every time you want to make changes to it.

The BMFDISK Program

BMFDISK is a DOS program which can be used to create, remove, and modify partitions of any type on a PC's local hard disks. BMFDISK can also query for information about partitions and set the DOS ERRORLEVEL to allow conditional processing in batch files. BMFDISK provides options to write a master boot sector to the hard disk and quick format a FAT16 partition. In addition, BMFDISK can read and write hard disk blocks directly. Since BMFDISK is entirely controlled by command line options, it can be used in DOS batch files to automate partitioning, formatting, and low level disk operations.

Introducing BMFDISK

BMFDISK is a DOS program that uses BIOS Int 13h (standard and extended) system calls to directly access data blocks on a PC's local hard disk or floppy disk. By supporting the Int 13h extensions, BMFDISK is able to handle hard disks of up to 2048 GBytes in size.

BMFDISK provides the following functions:

- create, delete, and modify partition entries in a hard disk's partition table
- retrieve information about partitions
- locate partitions according to their ID value
- write a master boot sector to a hard disk
- quick format a FAT16 partition
- read raw blocks from the hard disk and write them to a file
- write raw blocks to the hard disk by reading them from a file
- clear a hard disk (area) by writing 'zero' blocks to it
- return result codes using DOS ERRORLEVEL

Installing BMFDISK

BMFDISK is a DOS program that does not need to be installed. Instead, simply copy the file *bmfdisk.exe* to the place where you need it: a boot image, a floppy disk, or the PC's local hard disk.

BMFDISK is available in three different versions, all having full functionality but different restrictions:

1. If you received BMFDISK as part of the BootManage TCP/IP BOOT-PROM distribution or as part of the BootManage PXE Toolkit distribution, it will only work from within a DOS boot image when the PC was booted using either of the above products.
2. If you downloaded the standalone BMFDISK tryout version, it will show a licensing message at every invocation and also will ask you to press a key before cintinuing the command execution.
3. If you purchased the personalized standalone BMFDISK version, you can use it (according to the licensing terms) without any of the above restrictions.

Technical Background Information

To access physical blocks on the hard disk, BMFDISK uses both the standard and extended functions of the Int 13h BIOS interface.

The standard Int 13h BIOS interface functions see the hard disk in terms of track[1], head, and sector numbers.

The extended Int 13h BIOS interface functions (commonly known as Int 13h Extensions) see the hard disk in terms of absolute block numbers.

As there can be multiple hard disks present in a PC, all BIOS Int 13h functions use drive numbers to distinguish the hard disks.

Logically, a hard disk can be subdivided into multiple partitions which are identified by their partition number. BIOS Int 13h functions do not know about partitions; it is the duty of BMFDISK to translate partition information into track/head/sector or block numbers.

1 Track numbers are also known as cylinder numbers, both terms mean the same thing.

Drive Numbers

The PC system BIOS uses hexadecimal drive numbers to identify local mass storage devices like floppy drives and hard disks in the following way:

Drive Number	Device
00	first floppy disk
01	second floppy disk
80	first hard disk
81	second hard disk
82	third hard disk
83	fourth hard disk

Some BIOS types allow the mapping of CD-ROM drives to hard disk drives. On these systems, you can use a drive number to access the CD-ROM drive. For example, on a system with one hard disk and one CD-ROM device, drive number 80 is assigned to the hard disk and drive number 81 is assigned to the CD-ROM drive.

In some PC systems, it may be possible to change the drive numbers from within the System Setup. E. g. if you have one EIDE and one SCSI hard disk in a system that supports both drive types, it may be possible to change drive number assignment from within the System Setup by toggling between the settings "Boot from EIDE first" and "Boot from SCSI first".

Do not mix up BIOS physical drive numbers with DOS/Windows logical drives, which are associated with partitions and identified by drive letters (such as "C:" and "D:"). Therefore, physical drive number 80 may contain the DOS/Windows logical drives C: and D:.

Partition Numbers

The PC's system BIOS is capable of subdividing a hard disk drive into multiple primary partitions, up to a maximum of four, numbered from 0 to 3. A drive's partition layout is defined in the partition table, which is located in the first sector of the hard disk drive.

To overcome the limit of four partitions per hard disk, it is possible to use a special partition type named extended partition. Within an extended partition, multiple logical drives can be defined. As `BMFDISK` only supports extended partitions in a rudimentary way, we will not deal with extended partitions in more detail.

On a PC's first hard disk, a special flag determines which partition is active. The active partition is used for booting the PC. Only one partition can be active at the same time.

Track, Sector And Head Numbers

When using the standard BIOS Int 13h functions, the hard disk is seen in terms of heads, tracks, and sectors. To access a certain sector on the hard disk through system BIOS functions, you have to provide an 8-bit head number, a 10-bit track number and a 6-bit sector number. This way, you can access hard disks with up to 256 heads (numbered 0 to 255), up to 1024 tracks (numbered 0 to 1023) and up to 63 sectors per track (numbered 1 to 63)[1].

This addressing scheme imposes several limitations on high capacity hard disk drives:

- Common hard disks have only a few heads, so most of the 8-Bit head number addressing space is wasted.
- Large hard disks often have more than 1024 tracks, so the 10-Bit track number is not adequate to address all tracks, and hard disk space is wasted.
- Modern hard disks have a varying number of sectors per track, since the outer tracks provide more space than the inner ones. This does not fit into the BIOS addressing scheme.

To overcome these limitations, modern hard disk controllers provide internal mapping functions that translate the real hard disk geometry into a logical head/track/sector scheme. Also, the system setup of a modern PC BIOS allows you to select hard disk mapping schemes as well (STANDARD, LARGE, LBA, etc.).

When using standard BIOS Int 13h functions, `BMFDISK` always accepts the mapped hard disk geometry that is presented by the BIOS. When specifying head/track/sector values on the `BMFDISK` command line, you must always use these mapped values. A hard disk's mapped geometry can be obtained easily by using the `-g` command.

1 Due to an old BIOS bug, the number of sectors per track is only 63 and not, as one might expect, 64.

Be careful when dealing with hard disk mapping schemes in the PC's System Setup. When a hard disk is already partitioned and has data on it, never change its mapping scheme, or you will render your hard disk inaccessible and also risk losing your data!

Also, be cautious when using the hard disk autotyping feature that most BIOSes offer. In some cases, autotyping may exceed predefined timeout values and the hard disk may be identified with a different geometry. This can have the same effect as changing the mapping scheme, leading to data loss.

Using all these tricks, you can address hard disks that have a maximum capacity of 1024 x 256 x 63 = 16515072 sectors. With a sector being 512 bytes in size, the maximum addressable hard disk size is 16515072 x 512 bytes = 8455716864 bytes, which is approximately 7.8 GBytes.

Therefore, using standard Int 13h functions, with all tricks, you can only address the first 7.8 GByte of every hard disk; all space beyond this limit is not addressable!

Block Numbers

Modern operating systems do not use the PC BIOS functions to access the hard disk. Instead, a special operating system driver communicates directly with the hard disk controller. This allows programs to see the hard disk as a linear vector of blocks that can be accessed by specifying a single block number[1]. The hard disk controller internally maps this block number to physical head/track/sector information.

To overcome the limitations of the standard BIOS Int 13h functions (C/H/S computation, mapping schemes, 7.8 GByte limit), a new set of functions called the Int 13h Extensions have been introduced in modern BIOSes. Using these functions, it is possible to address a hard disk in terms of block numbers.

As block numbers are 64 bit wide, a hard disk can have up to 18446744073709551616 blocks. With a sector being 512 bytes in size, the maximum addressable hard disk size is 18446744073709551616 x 512 bytes = 9444732965739290427392 bytes, which is approximately 8796093022208 GBytes or 8192 EBytes (Exabytes)[2].

1 This addressing mode is known as Large Block Addressing (LBA).

2 Although hard disk space is rapidly increasing these days, this should be sufficient for the next years ;-)

BMFDISK uses block numbers when directly reading from and writing to hard disk sectors (-r, -w and -z commands). For these commands, BMFDISK can use standard or extended BIOS Int 13h functions. When using standard Int 13h functions, BMFDISK translates the given block numbers to track/head/sector values.

At this time, BMFDISK only supports 32-bit block numbers, which reduces the maximum addressable hard disk size to 4294967296 blocks. With a sector being 512 bytes in size, the maximum addressable hard disk size is 4294967296 x 512 bytes = 2199023255552 bytes, which is approximately 2048 GBytes.

Block numbering starts at 0 (not 1). The highest available block number is the total number of blocks minus 1 (see the -p command).

Partition IDs

A hard disk can be logically subdivided into partitions, where each partition can hold a separate filesystem. Although the BIOS does not have any idea what a partition is, all major operating systems have a common understanding of partitions.

The very first block of a hard disk is called the master boot record (MBR) or master boot sector. It contains the partition table as well as the master boot loader.

Within the partition table, every primary partition has an associated hexadecimal partition ID that defines its partition type, i.e. whether it is a DOS, NTFS, UNIX, NetWare or other partition. A list of common partition IDs is shown in the following table:

ID	Partition type
00	None (partition not used)
01	FAT12 (12-Bit FAT, < 10 MByte)
04	FAT16 (16-Bit FAT, < 32 MByte)
05	Extended partition (container for logical drives), may be FAT, HPFS or NTFS
06	FAT16 (16-Bit FAT, > 32 MByte)
07	used for both NTFS (Windows NT) and HPFS (OS/2)
0B	FAT32 (< 2047 GByte)
0C	FAT32 (same as 0B, but uses LBA addressing through BIOS Int 13h extensions)
0E	FAT16 (same as 06, but uses LBA addressing through BIOS Int 13h extensions)
0F	Extended (same as 05, but uses LBA addressing through BIOS Int 13h extensions)
63	UNIX (GNU HURD)

ID	Partition type
64,65	Novell NetWare
82	Linux swap
83	Linux native

Commands, Options And Parameters

The first command line option that is given to `BMFDISK` is called the command; it determines the major function that `BMFDISK` performs. The various commands require different numbers of command parameters which are listed in detail in the command descriptions.

Following the command and its parameters, optional command line arguments may be specified, these are called options. Like commands, options can also have parameters.

To prevent accidental alteration of hard disk blocks, the execution of all commands that write to hard disk blocks is delayed for 10 seconds while displaying a warning message and a countdown timer. Within this time, you can prevent the command execution by pressing any key.

In unattended setup batch scripts, you may want to bypass this security check and time delay. To do so, apply the `-f` option to these commands.

Before discussing every command, its parameters and the allowed options for this command, we want to give you an overview of all available commands and options in the following tables:

BMFDISK Commands

The following commands are available with BMFDISK:

Command	Explanation
`-?`	display BMFDISK usage
`-a`	add a +/- value to part. ID (ERRORLEVEL: new part. ID)
`-b`	write master boot record
`-c`	check for existence (ERRORLEVEL: no = 0, yes = 1)
`-d`	clear partition boot sector
`-g`	display disk geometry
`-G`	same as -g but does not display error message if fails
`-i`	search for partition
`-m`	make a partition
`-o`	set partition ID

Command	Explanation
-p	display partition table
-P	display partition table using environment variable format
-q	quick format a partition (write boot sector and FATs/DIR)
-r	read from the disk and write to a file
-t	set partition to active
-w	read from a file and write to the disk
-z	write zero (sectors with content 0) to the disk

BMFDISK Options And Parameters

The following options and parameters can be used with BMFDISK:

Option / Parameter	Explanation
b#	a decimal number specifying an absolute block
blks	a decimal number specifying a number of blocks
drv	a hexadecimal drive number
part	a hard disk partition number (0, 1, 2 or 3)
id	a hexadecimal partition ID
-s *b#*	specifies the starting block number
-e *b#*	specifies the ending block number
-n *blks*	specifies the number of blocks
-l *kbs*	limit throughput to *kbs* kilobytes per second
-f	force immediate execution, don't delay to allow user abort
-v	display more information upon command execution

Create, Delete, Modify Partitions (-m)

The -m command is used to create, delete, or modify a partition. There are various different forms of the -m command available, and all of them only modify the partition table. The partition's boot sector (PBR) and all the information within the partition itself is not altered.

To prevent accidental partitioning, the execution of the -m command is delayed for 10 seconds so that you can interrupt it by hitting a key before partitioning actually begins. When using the -m command from within DOS batch files, you can override the execution delay by appending the -f option.

There are several different ways to specify partition sizes:

- in partition slot 0, create a single large active primary FAT16 partition that spans the entire hard disk (max. 2 GBytes).
- provide partition size in Megabytes
- provide partition size as a percentage of the total hard disk space
- provide partition size in (512-Byte-)Blocks
- create a partition that uses the remaining unpartitioned hard disk space
- delete a partition entry (set all values to 0)
- create a partition by specifying all the geometry-dependent parameters

You must reboot the PC before the system BIOS recognizes changes to the partition table! If you want to quick format a FAT16 partition using the -q option, note that you must do this directly after creating the partition and before rebooting.

Create Active Maximum Sized FAT16 Partition

```
bmfdisk -m drv [-f]
```

This creates a maximum-sized (2 GBytes) active FAT16 partition in partition slot 0 of drive *drv*. If the hard disk's total capacity is equal to or less than 2 GBytes, then the created partition takes up the entire hard disk space.

On a 3 GByte hard disk, the following example creates a 2 GByte FAT16 partition and leaves the remaining 1 GByte unpartitioned (available for other partitions):

```
rem executed on a 3GByte hard disk
bmfdisk -m 80
```

On a 1 GByte hard disk, the same command creates a 1 GByte FAT16 partition that spans the entire hard disk and leaves no space for other partitions:

```
rem executed on a 1GByte hard disk
bmfdisk -m 80
```

Create Partition, Specify Size In Megabytes

```
bmfdisk -m drv part a id sizem [-f]
```

The following example creates an active primary DOS partition on the first hard disk that spans approx. 400 MByte.

```
rem create a 400 MB primary active DOS partition on first drive
bmfdisk -m 80 0 Y 06 400m
```

Note that the number of megabytes must be immediately followed by the letter 'm'.

Create Partition, Specify Size As Percentage

```
bmfdisk -m drv part a id sizep [-f]
```

This command allows you to specify a partition's size as a percentage of the total hard disk space. Note that the percentage value always refers to the total hard disk space, not the remaining unpartitioned space!

The following example creates a partition that takes up 40 percent of the total hard disk space:

```
rem create a partition that takes up 40% of total hard disk space
bmfdisk -m 80 0 Y 06 40p
```

Note that the number that gives the percentage must be immediately followed by the letter 'p'.

Create Partition, Specify Size In Blocks

```
bmfdisk -m drv part a id sizeb [-f]
```

It is also possible to specify a partition's size in 512-byte-blocks. Note that BMFDISK will round the partition size so that it ends on a full cylinder.

The following example creates a partition that spans 921600 512-byte-blocks (equals 450 MBytes).

```
rem create a partition that spans 921600 blocks
bmfdisk -m 80 0 Y 06 921600b
```

Note that the number of blocks must be immediately followed by the letter 'b'.

Create Partition, Use Rest Of Available Space

```
bmfdisk -m drv part a id r [-f]
```

It is also possible to create a partition that takes up all the currently unpartitioned space. Note that the unpartitioned space must be located in a single area that follows all previously created partitions.

The following example creates an inactive primary DOS partition on the first hard disk in the second partition slot that spans the rest of the available hard disk space:

```
rem create a DOS partition that takes up the remaining hard disk space
bmfdisk -m 80 1 N 06 r
```

Delete Partition Entry

```
bmfdisk -m drv part a id c [-f]
```

Deleting a partition entry means that all fields in the corresponding partition table slot are set to the value 0. The data within the partition itself (including the partition boot sector) is not modified.

The following example deletes partition entry 2 on the second hard disk:

```
rem delete partition 2 on drive 81
bmfdisk -m 80 2 N 0 c
```

You can also delete a partition entry by using the detailed form and setting all fields to the value 0. However, using this shorthand is somewhat easier.

Create Partition, Detailed Form

```
bmfdisk -m drv part a id t s h sec t s h len [-s b#] [-f]
```

The detailed form of the `-m` command allows to completely control all the values in a partition entry and uses the same argument format for entering partition data as the `-p` command uses for displaying it.

If you want to exactly define the start and end sectors of a partition, use the "full-size" form. Be aware that you have to be familiar with your hard disk's geometry (cylinders, heads, sectors). This way, you can even modify extended partitions.

```
rem create a 200 MB primary active DOS partition on first drive
bmfdisk -m 80 0 Y 06 0 1 1 63 101 63 63 411201 -f

rem delete second partition on first drive
bmfdisk -m 80 1 N 0 0 0 0 0 0 0 0 0 -f
```

When using the `-m` command from within DOS batch files, you can override the execution delay by appending the `-f` option.

The -m command can also be used to access the partition table of an extended partition. By using the -s option, you can create extended partitions and logical drives within them by specifying the offset to the extended partition table.

```
rem the following commands affect the standard partition table

rem create primary partition 200 MB on second drive
bmfdisk -m 81 0 N 06 0 1 1 63 101 63 63 411201 -f

rem create extended partition 300 MB on second drive
bmfdisk -m 81 1 N 05 102 1 0 411264 254 63 63 616896 -f


rem the following commands affect the extended partition table

rem create logical drive 100MB
bmfdisk -m 81 0 N 06 102 1 1 63 152 63 63 205569 -s 411264 -f

rem chain to next partition table in extended partition
bmfdisk -m 81 1 N 05 153 1 0 205632 254 63 63 411264 -s 411264 -f

rem create logical drive 200MB
bmfdisk -m 81 0 N 06 153 1 1 63 254 63 63 411201 -s 616896 -f
```

Display Partition Table Information

Partition Table Overview (-p)

```
bmfdisk -p drv [-s b#]
```

The -p command displays the partition table of a hard disk. You must specify the hard disk's drive number on the command line.

The -p command uses the same output format for displaying partition data as the -m command uses for entering it:

```
bmfdisk -p 80

         --------Start--------- ----End-----
# Act Id  Trk Sec  Hd    Sector  Trk Sec  Hd    Length
------------------------------------------------------
0   Y 06    0   1   1        63   64  63 254   1044162
1   N 05   65   1   0   1044225  388  63 254   5205060
2   N 07  389   1   0   6249285  519  63 254   2104515
3   N 07  520   1   0   8353800  547  63 254    449820
```

By using the -s option, you can display the partition table of an extended partition.

Detailed Partition Information (-P)

```
bmfdisk -P drv part [-s b#]
```

The `-P` command can be used to store partition related information in DOS environment variables. This way, you can refer to the starting track, ending sector or length in sectors of a partition by using the corresponding environment variable. The `-P` command output is meant to be redirected into a DOS batch file which is then executed to import the environment variables as shown in the following example:

```
# examine partition 0 of the first hard disk
# and store output in the SETPART.BAT file
bmfdisk -P 80 0 > setpart.bat

# execute the SETPART.BAT file to import the environment variables
call setpart.bat
```

The batch file *setpart.bat* contains SET commands to define the following environment variables:

Environment Variable	Explanation
P_NUM	Partition number (0, 1, 2 or 3)
P_ACT	Partition active (Y or N)
P_ID	Partition ID (hex)
P_STA_T	Starting track (decimal)
P_STA_S	Starting sector (decimal)
P_STA_H	Starting head (decimal)
P_START	Absolute starting block (decimal)
P_END_T	Ending track (decimal)
P_END_S	Ending sector (decimal)
P_END_H	Ending head (decimal)
P_LENGTH	Number of blocks in partition (decimal)

The generated batch file also contains commands to delete the environment variables. To do so, call the same batch file with the parameter 'clean' as in the following example:

```
# remove partition specific environment variables
call setpart.bat clean
```

Check and Manipulate Partition IDs

Check Partition ID (-c)

```
bmfdisk -c drv part id
```

The `-c` command checks if the partition ID of partition number *part* on drive number *drv* matches the value *id*. If the partition ID values match, the DOS ERRORLEVEL environment variable is set to 1. If they don't match, ERRORLEVEL is set to 0. This can be used in DOS batch files to determine whether a certain partition exists or not.

```
rem check if partition slot 0 on first hard disk
rem already contains a FAT16 partition
bmfdisk -c 80 0 06
if ERRORLEVEL 1 goto EXISTS

rem DOS partition does not exist, create it
bmfdisk -m 80 0 Y 06 800m -f

:EXISTS
```

Locate Partition (-i)

```
bmfdisk -i drv id
```

The `-i` command is used to locate the first partition whose ID value matches the given hexadecimal *id*. The partition slot number in which the partition ID was found is returned in the DOS ERRORLEVEL.

The following example shows how to locate the first partition with ID value 07:

```
rem locate first partition with ID value 07
bmfdisk -i 80 07
if ERRORLEVEL 4 goto NOT_FOUND
if ERRORLEVEL 3 goto PART_3
if ERRORLEVEL 2 goto PART_2
if ERRORLEVEL 1 goto PART_1
if ERRORLEVEL 0 goto PART_0
```

Set Partition ID (-o)

```
bmfdisk -o drv part val [-f]
```

The `-o` command sets the ID field of partition number *part* on drive *drv* to the hexadecimal value *val* without changing the partition's geometry data:

```
rem set ID value of last partition on first hard disk to hex F1
bmfdisk -o 80 3 F1
```

Increment/Decrement Partition ID (-a)

```
bmfdisk -a drv part val [-f]
```

The `-a` command increments the ID field of partition number *`part`* on drive *`drv`* by the value *`val`*. You may also decrement the partition ID by specifying a negative value. This is interesting when (mis)using a partition entry as status indicator (e.g. when you want to retain status information over a PC reboot).

```
rem increment ID value of last partition on first hard disk by 2
bmfdisk -a 80 3 2

rem decrement ID value of last partition on second hard disk by 1
bmfdisk -a 81 3 -1
```

The result of the operation (the new partition ID value) is also returned in the DOS ERRORLEVEL variable. To determine the ID of a certain partition, you can use the `-a` command to add the value 0 and check ERRORLEVEL

```
rem query ID value of first partition on first hard disk
rem result is stored in ERRORLEVEL
bmfdisk -a 80 0 0

rem check for NTFS partition
if ERRORLEVEL 7 goto NTFS

rem check for primary DOS partition
if ERRORLEVEL 6 goto PRI_DOS

rem check for extended partition
if ERRORLEVEL 5 goto EXTEND
```

Activate Partition (-t)

```
bmfdisk -t drv part [-f]
```

The `-t` command sets the specified partition to active. At the same time, the other three primary partitions are reset to inactive. The PC BIOS will boot from the active partition.

```
rem set second partition of first hard disk to active
bmfdisk -t 80 1
```

Hard Disk Geometry Information

Display Hard Disk Geometry (-g)

```
bmfdisk -g drv
```

The `-g` command queries the BIOS for information about the hard disk size and geometry. It displays the logical geometry information derived from the standard BIOS Int 13 interface and also, if available, the physical geometry information derived from the extended BIOS Int 13 interface.

If the hard disk given by *drv* does not exist, an error is displayed.

To explain the information that the `-g` command provides, consider the following example which was created for a 3 GByte hard disk:

```
Enhanced Disk Drive Specification: Version=0x01, Extensions=0x0003
BIOS int 13h/08h: C=784, H=128, S=63, Size=3161088KB, Blocks=6322176
BIOS int 13h/48h: C=6282, H=16, S=63, Size=3166128KB, Blocks=6332256
```

The first line tells us that the BIOS supports the extended Int 13h interface, so that physical hard disk information is available.

The second line displays the logical geometry information that was obtained from the standard BIOS Int 13h interface (using function 08h):

- The disk has 784 cylinders (or tracks), 128 heads, and 63 sectors per track.
- This gives a total number of 784 x 128 x 63 = 6322176 blocks.
- With each block holding 512 bytes, the total hard disk size is 6322176 x 512 bytes which gives 3236954112 bytes or 3161088 KBytes or 3087 MBytes.

Remember that this is the mapped logical geometry as seen by the BIOS.

The third line displays the physical geometry information that was obtained from the extended BIOS Int 13h interface (using function 48h):

- The disk has 6282 cylinders (or tracks), 16 heads, and 63 sectors per track.
- This gives a total number of 6282 x 16 x 63 = 6332256 blocks.
- With each block holding 512 bytes, the total hard disk size is 6332256 x 512 bytes which gives 3242115072 bytes or 3166128 KBytes or 3092 MBytes.

The differences between logical and physical geometry become more drastic if the hard disk size exceeds the “7.8 GByte barrier” as in the following example of a 12 GB hard disk:

```
Enhanced Disk Drive Specification: Version=0x21, Extensions=0x0005
BIOS int 13h/08h: C=1024, H=255, S=63, Size=8225280KB, Blocks=16450560
BIOS int 13h/48h: C=1582, H=255, S=63, Size=12707415KB, Blocks=25414830
```

Here, the standard BIOS Int 13h interface has reached its limit, so it can only address the first 16450560 blocks, which is approximately 7.8 GBytes.

However, the extended BIOS Int 13h interface can address the full number of 25414830 blocks, which is approximately 12 GBytes.

You can use the `-g` command to determine the highest available block number that can be used with the `-e` or other options that specify block ranges. As block counting starts at 0, the highest available block number is the total number of blocks minus 1.

Display Hard Disk Geometry (-G)

```
bmfdisk -G drv
```

The `-G` command is identical to the `-g` command, except that it does not return an error message if the hard disk given by *drv* does not exist. This is useful to search for installed disks in the system.

```
rem look for installed disks and report them in a log file
bmfdisk -G 80 >  disks.log
bmfdisk -G 81 >> disks.log
bmfdisk -G 82 >> disks.log
bmfdisk -G 83 >> disks.log
```

Write Master Boot Record (-b)

```
bmfdisk -b <drv> [-f]
```

The first sector of a hard disk is called the master boot sector or master boot record (MBR). It contains the hard disk's partition table and, on bootable hard disks only, operating system independent executable code that is able to determine the active partition and transfer control to the partition's boot sector.

The `-b` command writes a standard operating system independent hard disk bootstrap loader to the hard disk that is specified by drive number *drv*. The hard disk's partition table is not modified by this operation.

```
rem write master boot record to first hard disk
bmfdisk -b 80
```

As with all other commands that write to the hard disk, the `-b` command displays a warning message and a 10 second countdown timer that allows you to abort the allows you to abort by keypress. For use in unattended batch files, you can specify the `-f` option to circumvent the warning message:

```
bmfdisk -b 80 -f
```

Quick Formatting Partitions

Quick Format FAT16 Partition (-q)

```
bmfdisk -q drv part filesys [-f]
```

Before you can use a partition under DOS, you have to format it using the DOS FORMAT program. This can be slow because FORMAT also checks for bad sectors within the DOS partition. Using the -q command, BMFDISK can quickly format a DOS partition by only writing the partition boot sector and clearing the file allocation table and root directory sectors.

To prevent accidental formatting, the execution of the -q command is delayed for 10 seconds so that you can interrupt it by hitting a key before formatting actually begins. When using the -q command from within DOS batch files, you can override the execution delay by appending the -f option.

```
rem quick format partition 0 on drive 80 with FAT16 file system
bmfdisk -q 80 0 fat16 -f
```

As opposed to the DOS FORMAT program, the BMFDISK quick formatting routine does not require that you reboot the PC after creating a partition. However, when using the BMFDISK quick formatting routine, you must reboot your PC after formatting.

Example:

```
rem create a primary FAT16 partition on drive 80
bmfdisk -m 80 0 y 06 500m -f

rem quick format the just-created partition with the FAT16 filesystem
bmfdisk -q 80 0 fat16 -f

rem before DOS can access the new partition, we must reboot
reboot
```

Clear First Partition Sector (-d)

```
bmfdisk -d drv part [-f]
```

The -d command is used to clear the very first sector of the given partition, the so-called partition boot record (PBR).

When using BMFDISK to partition and quick format hard disks, you should never need to use the -d command.

However, when you create a partition with BMFDISK, but use a different program to format it (e. g. DOS FORMAT.COM), you may encounter problems in rare cases. In these cases, use the -d command before formatting.

Directly Read/Write Disk Blocks

Read Disk Blocks (-r)

```
bmfdisk -r drv file [-s b#] [-e b#] [-v] [-l kbs]
bmfdisk -r drv file [-s b#] [-n blks] [-v] [-l kbs]
```

The `-r` command reads blocks from the hard disk specified by drive number *`drv`* and writes them to the file *`file`*. This can be used to create a hard disk image file for fast operating system installation on multiple identical PC's. Also, you can create a backup of the partition table, the contents of a single partition or even the entire hard disk contents for emergency recovery.

The file *`file`* must not be located on the same hard disk that is specified by drive number *`drv`*. It can be located on either a different hard disk or on a network drive. Make sure that enough space is on that drive because hard disk image files can get very large.

By default, the `-r` command reads the entire hard disk from the first to last block.

```
rem backup the entire hard disk contents to file n:\hdbackup.dsk
bmfdisk -r 80 n:\hdbackup.dsk
```

If you only want to read a certain block range, you can specify the starting and ending block numbers by using the `-s` and `-e` options.

```
rem backup the partition sector to file n:\partsec.dsk
bmfdisk -r 80 n:\partsec.dsk -s 0 -e 0

rem this does the same as above
bmfdisk -r 80 n:\partsec.dsk -e 0
```

Instead of specifying a starting and ending block number, you can also specify a starting block number and the number of sectors by using the `-s` and `-n` options.

```
rem backup the partition sector to file n:\partsec.dsk
bmfdisk -r 80 n:\partsec.dsk -s 0 -n 1

rem this does the same as above
bmfdisk -r 80 n:\partsec.dsk -n 1
```

The following example first uses the `-p` command to determine the size and location of the active DOS partition and then uses the `-r` command to create a backup of this partition.

```
rem determine the size and location of the active DOS partition
bmfdisk -p 80

          --------Start--------- ----End-----
# Act Id  Trk Sec  Hd    Sector  Trk Sec  Hd    Length
-------------------------------------------------------
0   Y 06    0   1   1        63   64  63 254   1044162
1   N 05   65   1   0   1044225  388  63 254   5205060
2   N 07  389   1   0   6249285  519  63 254   2104515
3   N 07  520   1   0   8353800  547  63 254    449820

rem now copy the DOS partition (ID 06)
bmfdisk -r 80 n:\dospart.dsk -s 63 -n 1044162
```

Information about the status of the copy process and throughput can be displayed by adding the `-v` option to the command line.

```
rem backup the entire hard disk contents to file n:\hdbackup.dsk
rem display information about copy status and throughput
bmfdisk -r 80 n:\hdbackup.dsk -v
```

Copying the contents of a large partition or an entire hard disk to a file that is located on a network drive puts a constantly high load on your network. Doing this on multiple PCs simultaneously may load your network so heavily that overall performance drops substantially.

To avoid this, use the `-l` option to limit the throughput of `BMFDISK`. The *`kbs`* parameter following the `-l` option (approximately) specifies the maximum throughput in kilobytes per second.

```
rem backup the entire hard disk contents to file n:\hdbackup.dsk
rem display information about copy status and throughput
rem limit throughput to max. 100 kBytes per second
bmfdisk -r 80 n:\hdbackup.dsk -v -l 100
```

Write Disk Blocks (-w)

```
bmfdisk -w drv file [-s b#] [-e b#] [-v] [-l kbs] [-f]
bmfdisk -w drv file [-s b#] [-n blks] [-v] [-l kbs] [-f]
```

The `-w` command reads the hard disk image file *`file`* and writes its contents block by block to the hard disk specified by drive number *`drv`*. This can be used to speed-up an operating system installation by writing a pre-configured image file to the hard disk. Also, you can restore the partition table, the contents of a single partition, or even the entire hard disk contents from a backup file.

```
rem restore the entire hard disk from file n:\hdbackup.dsk
bmfdisk -w 80 n:\hdbackup.dsk
```

Command line options and parameters are the same as with the `-r` command.

You can specify starting and ending block numbers:

```
rem restore the partition sector from file n:\partsec.dsk
bmfdisk -w 80 n:\partsec.dsk -s 0 -e 0

rem this does the same as above
bmfdisk -w 80 n:\partsec.dsk -e 0
```

You can also specify the starting block number and number of sectors.

```
rem create a 200 MB primary active DOS partition on first drive
bmfdisk -m 80 0 Y 06 0 1 1 63 64 63 254 1044162 -f

rem restore contents of DOS partition from file n:\dospart.dsk
bmfdisk -w 80 n:\dospart.dsk -s 63 -n 1044162
```

Note that the `-v` option, for displaying status information, and the `-l` option, for limiting throughput, are available. See the `-r` command for usage details.

Clear/Erase Disk Blocks (-z)

```
bmfdisk -z drv [-s b#] [-e b#] [-v] [-f]
bmfdisk -z drv [-s b#] [-n blks] [-v] [-f]
```

This command is somewhat equivalent to the `-w` command since it writes blocks to the hard disk, but the block data contents is not read from a file. Instead, the `-z` command writes 'zero blocks', i.e. each of the 512 data bytes within a block is set to the value 0.

```
rem clear entire hard disk
bmfdisk -z 80
```

When using the `-m` command to remove a partition, you only remove the partition entry information in the partition table, but not the data that is stored within the blocks that were located within the partition. If you want to make sure that a specified disk area or even the whole disk is entirely cleared, use the `-z` command.

Command line options and parameters are the same as with the `-r` and `-w` commands. The `-l` option is not available since no network traffic is generated.

The following example removes a partition entry from the partition table and also clears all data blocks that were located within this partition:

```
rem display the original partition table
bmfdisk -p 80

         --------Start--------- ----End-----
# Act Id  Trk Sec  Hd    Sector  Trk Sec  Hd    Length
-----------------------------------------------------
0   Y 06    0   1   1        63   64  63 254   1044162
1   N 05   65   1   0   1044225  388  63 254   5205060
2   N 07  389   1   0   6249285  519  63 254   2104515
3   N 07  520   1   0   8353800  547  63 254    449820

rem remove the DOS partition entry from partition table
bmfdisk -m 80 0 N 00 c -f

rem clear partition area
bmfdisk -z 80 -s 63 -n 1044162 -f
```

The BMPCSCAN Program

BMPCSCAN is a DOS program designed to detect and identify the PCI and PnP devices that are installed in your PC. BMPCSCAN can create detailed hardware report lists or store device information in environment variables. It is possible to control the amount of information that BMPCSCAN displays. Also, you can instruct BMPCSCAN to detect only devices of a specific type. BMPCSCAN was designed to run from within a boot image and will only work when the PC has been booted using the BootManage® TCP/IP BOOT-PROM or the BootManage® PXE Toolkit.

Installing BMPCSCAN

BMPCSCAN is a DOS program that does not need to be installed. To use BMPCSCAN, simply copy the file *bmpcscan.exe* into your boot image. When you want to get verbose information about PCI and/or PnP devices, you also need to copy the files *pcicode.dat* and *pnpcode.dat*.

BMPCSCAN Command Line Options

BMPCSCAN is entirely controlled by command line options. The command line syntax is as follows:

```
bmpcscan [pci|pnp] [-s] [-t type] [-f fname] [-v vlevel]
```

The following table shows all available BMPCSCAN options:

Option	Description
pci	Scan for PCI devices.
pnp	Scan for embedded Plug-and-Play devices.
-s	Use environment variable output format. If this option is not present, use report output format instead.
-t *type*	Only display information about devices of the given type
-f *fname*	Full pathname of database file (*pcicode.dat* / *pnpcode.dat*)
-v *vlevel*	Verbosity level (0 ... 2)

Selecting The Output Format

BMPCSCAN provides two different types of output:

- a report format that is used for informational purposes, e.g. when you want to keep a record of the hardware that is installed in your PC.
- an "environment variable" format that is used to assist in modular automated installation environments.

When the -s switch is present on the command line, BMPCSCAN uses the environment variable output format. When the switch -s is not present, the report format is used instead.

The Report Format

When the -s switch is not present on the command line, BMPCSCAN formats its output in a report style which is intended to be easily human-readable. You can redirect the report format output into a file to keep a record of the computer's installed PCI and embedded Plug-and-Play devices.

Depending on the verbosity level (selected with the -v option), BMPCSCAN produces a different amount of information. For verbosity levels higher than 0, BMPCSCAN needs to read information from the database file (*pcicode.dat* in PCI mode, *pnpcode.dat* in PnP mode).

A PCI device report in verbosity level 0 produces one line of output for every detected PCI device like in the following example:

```
bmpcscan pci -v 0

Bus  Device BaseClass SubClass ProgIf VendID DevID SubsysID RevID
00   20     02        00       00     8086   1229  30008086 08
01   00     03        00       00     1002   474d  00041002 65
00   18     04        01       00     1073   000c  000c1073 03
00   00     06        00       00     8086   7190  00000000 03
00   38     06        01       00     8086   7110  00000000 02
00   08     06        04       00     8086   7191  00000000 03
00   3b     06        80       00     8086   7113  00000000 02
```

A Plug-and-Play device report in verbosity level 0 also produces one line of output for every detected Plug-and-Play device:

```
bmpcscan pnp -v 0

Name     Handle   Size  BaseClass  SubClass  ProgIf
PNP0000  00       37    08         00        00
PNP0200  01       69    08         01        01
PNP0100  02       29    08         02        02
PNP0B00  03       29    08         03        03
PNP0303  04       37    09         00        00
PNP0800  05       26    08         80        80
PNP0C04  06       29    0b         01        01
PNP0C01  07       54    05         00        00
PNP0C02  08       66    08         80        80
PNP0A03  09       66    06         04        04
PNP1F03  0a       26    09         02        02
PNP0501  0b       69    07         00        00
PNP0700  0c       48    01         02        02
PNP0400  0d       60    07         01        01
PNP0501  0e       69    07         00        00
```

In most cases, when creating a hardware report, you may want to have more verbose information about the installed devices. For this purpose, BMPCSCAN keeps detailed information about devices in the database files *pcicode.dat* and *pnpcode.dat*. When using a verbosity level that is higher than 0, BMPCSCAN looks up additional information for every detected device in the corresponding database file and displays it as part of the report.

The following excerpt from a sample PCI device report in verbosity level 1 shows additional type information for two PCI devices:

```
bmpcscan pci -v 1

-------------------------------------------------------------------
Bus  Device BaseClass SubClass ProgIf VendID DevID SubsysID RevID
00   20     02        00       00     8086   1229  30008086 08

 Type     : General Ethernet Controller


-------------------------------------------------------------------
Bus  Device BaseClass SubClass ProgIf VendID DevID SubsysID RevID
01   00     03        00       00     1002   474d  00041002 65

 Type     : Generic VGA compatible
```

If you need all available details about the detected PCI devices, use verbosity level 2 as in the following example:

```
bmpcscan pci -v 2

------------------------------------------------------------------
Bus  Device BaseClass SubClass ProgIf VendID DevID SubsysID RevID
00   20     02        00       00     8086   1229  30008086 08

 Company  : INTEL ( INTEL CORP. )
 Device   : Intel 8255x-based PCI Ethernet Adapter (10/100)
 Chipset  : 82557
 Type     : General Ethernet Controller


------------------------------------------------------------------
Bus  Device BaseClass SubClass ProgIf VendID DevID SubsysID RevID
01   00     03        00       00     1002   474d  00041002 65

 Company  : ATI ( ATI TECHNOLOGIES INC )
 Device   : ATI Technologies RAGE XL AGP 2X
 Type     : Generic VGA compatible
```

When scanning for Plug-and-Play devices, verbosity level 1 already displays all available information:

```
bmpcscan pnp -v 1

------------------------------------------------------------------
Name     Handle   Size  BaseClass  SubClass  ProgIf
PNP0000  00       37    08         00        00

 PnP Device    : PNP0000 - AT Interrupt Controller
 BaseType      : 8 - System Peripherals
 SubType       : 0 - Programmable Interrupt Controller (8259 Compatible)
 InterfaceType : 0 - Generic 8259 PIC

------------------------------------------------------------------
Name     Handle   Size  BaseClass  SubClass  ProgIf
PNP0200  01       69    08         01        01

 PnP Device    : PNP0200 - AT DMA Controller
 BaseType      : 8 - System Peripherals
 SubType       : 1 - DMA Controller (8237 Compatible)
 InterfaceType : 0 - Generic DMA Controller
```

When no verbosity level is given in report format, BMPCSCAN displays the maximum amount of information.

The Environment Variable Format

When the `-s` option is given on the command line, `BMPCSCAN` formats its output in a format that makes it easy to store information about detected devices in environment variables.

Every output line is of the form

```
set variable=value
```

To integrate device information into your environment, simply redirect the output of `BMPCSCAN` into a batch file and execute this batch file as in the following example:

```
bmpcscan pci -s > pcivars.bat
call pcivars.bat
```

Make sure that you have enough environment space available. If you get the error message "Out of environment space" when the *pcivars.bat* file is executed, increase the amount of environmment space by using the `/E` switch of the DOS command processor, `COMMAND.COM`

Having this information available from within other batch files and scripts, it is easily possible to integrate new hardware in fully automated operating system installations.

For PCI devices, you can control the amount of output information with the verbosity level (selected with the `-v` option). For PnP devices, the verbosity level has no effect.

To illustrate this, let us have a look at some sample output. First, we want to get the PCI vendor and device IDs of all PCI devices in the system:

```
bmpcscan pci -s -v 0

SET PCI_NET0=80861229
SET PCI_VGA0=1002474d
SET PCI_MMDAUD0=1073000c
SET PCI_BRIDGEHOST0=80867190
SET PCI_BRIDGEISA0=80867110
SET PCI_BRIDGEPCI0=80867191
SET PCI_BRIDGE0=80867113
```

For every detected PCI device, `BMPCSCAN` outputs a single environment variable. The variable name reflects the PCI device type and number, and the variable value is the concatenated hexadecimal PCI vendor and device ID. Device numbering

starts at 0, so the first PCI video device is named PCI_VGA0, the second PCI_VGA1, and so on.

In some cases, you may need to get additional addition such as the PCI device's subsystem ID and revision level. Also, you may want to have the PCI vendor and device IDs in separate environment variables. For this purpose, use verbosity level 2 as shown in the next example:

```
bmpcscan pci -s -v 1

SET PCI_NET0v=8086
SET PCI_NET0d=1229
SET PCI_NET0s=30008086
SET PCI_NET0r=08
SET PCI_VGA0v=1002
SET PCI_VGA0d=474d
SET PCI_VGA0s=00041002
SET PCI_VGA0r=65
SET PCI_MMDAUD0v=1073
SET PCI_MMDAUD0d=000c
SET PCI_MMDAUD0s=000c1073
SET PCI_MMDAUD0r=03
SET PCI_BRIDGEHOST0v=8086
SET PCI_BRIDGEHOST0d=7190
SET PCI_BRIDGEHOST0s=00000000
SET PCI_BRIDGEHOST0r=03
SET PCI_BRIDGEISA0v=8086
SET PCI_BRIDGEISA0d=7110
SET PCI_BRIDGEISA0s=00000000
SET PCI_BRIDGEISA0r=02
SET PCI_BRIDGEPCI0v=8086
SET PCI_BRIDGEPCI0d=7191
SET PCI_BRIDGEPCI0s=00000000
SET PCI_BRIDGEPCI0r=03
SET PCI_BRIDGE0v=8086
SET PCI_BRIDGE0d=7113
SET PCI_BRIDGE0s=00000000
SET PCI_BRIDGE0r=02
```

Here, BMPCSCAN outputs four lines for every detected PCI device. Separate values identify the device's PCI vendor ID, device ID, subsystem ID and revision number.

For Plug-and-Play devices, the verbosity level has no effect. Here, device information always looks as in the following example:

```
bmpcscan pnp -s

SET PNP_SYSTEM0=PNP0000
SET PNP_SYSTEM1=PNP0200
SET PNP_SYSTEM2=PNP0100
SET PNP_SYSTEM3=PNP0B00
SET PNP_INPUT0=PNP0303
SET PNP_SYSTEM4=PNP0800
SET PNP_SYSTEM5=PNP0C04
SET PNP_MEM0=PNP0C01
SET PNP_SYSTEM6=PNP0C02
SET PNP_BRIDGEPCI0=PNP0A03
SET PNP_INPUT1=PNP1F03
SET PNP_COM0=PNP0501
SET PNP_MSDFD0=PNP0700
SET PNP_LPT0=PNP0400
SET PNP_COM1=PNP0501
```

Setting The Verbosity Level

The `-v` option is used to control the amount of information that is generated by `BMPCSCAN`. Both output formats (report and environment variable format) are affected by the `-v` option. Three verbosity levels (0, 1, and 2) are available.

For verbosity levels higher than 0, `BMPCSCAN` needs to read information from the database file (*pcicode.dat* in PCI mode, *pnpcode.dat* in PnP mode).

If you do not specify the `-v` option, `BMPCSCAN` uses default verbosity level which varies depending on the output format.

Specifying The Database File Location

With the `-f` option, you can specify the path and filename of the database file that holds the verbose device information.

If you do not specify the `-f` option, `BMPCSCAN` tries to use the default file in the same directory from which `BMPCSCAN` itself was executed. When scanning PCI devices, the filename *pcicode.dat* is used. When scanning PnP devices, the filename *pnpcode.dat* is used instead.

Examples:

```
bmpcscan pci -f w:\util\pcicode.dat

bmpcscan pnp -f x:\bootprom\pnpcode.dat
```

Selecting Device Types

Using the `-t` option, you can instruct `BMPCSCAN` to detect only devices of the given type. To report only Plug-and-Play network devices in environment variable format, use:

```
bmpcscan pnp -t NET -s
```

If you want to create a detailed report containing all PCI video display devices, use:

```
bmpcscan pci -t VGA -v 2
```

It is only possible to specify a single device type, you cannot select multiple device types on the `BMPCSCAN` command line.

The following list contains all device type keywords that can be used with the `-t` option:

keyword	device type
VGA	VGA/AGP Display controller
NET	Network controller
MMD	Multimedia device
MEM	Memory controller
MSC	Mass storage controller
MSDFD	Floppy disk controller
MMDVID	Multimedia video device
MMDAUD	Multimedia audio device
MSCIDE	IDE controller
MSCSCSI	SCSI bus controller
MSCIPIBC	IPI bus controller
BRIDGE	Bridge device
BRIDGEHOST	Host bridge
BRIDGEISA	ISA bridge
BRIDGEEISA	EISA bridge
BRIDGEMC	MC bridge
BRIDGEPCI	PCI-to-PCI bridge
BRIDGEPCMCIA	PCMCIA bridge
UNKNOWN	Unknown Device
SYSTEM	System Devices
COM	Communications Device
LPT	Parrallel Port Device
INPUT	Input Device

The BPMENU Boot Loader

BPMENU is a menu script interpreter. It can be downloaded and invoked by the TCP/IP BOOT-PROM. The menu prompts the user for a RAM disk image that is then downloaded. Using BPMENU, the user can choose between different configurations and servers.

TCP/IP BOOT-PROMs earlier than version 1.47 cannot run BPMENU. In order to make use of BPMENU, the TCP/IP BOOT-PROM must be of version 1.47 or higher!

Installing BPMENU

The BPMENU program and a menu script file are held on the server. The BPMENU program is then specified as the bootfile in the */etc/bootptab* file of the BOOTP daemon. Instead of downloading the RAM disk image, BPMENU is downloaded by the TCP/IP BOOT-PROM.

```
diskless:\
    :tc=pcnfs:ha=00004b0719e1:ip=193.141.47.198:\
    :hd=/tftpboot:bf=bpmenu:
```

After a successful download, control is given to the BPMENU program, which now resides in the diskless Personal Computer's memory. A menu script file is then downloaded by the BPMENU program. BPMENU searches for the menu script file by adding the extension *.M* (dot and capital M) to its name.

For example, if BPMENU is downloaded from */tftpboot/bpmenu*, then BPMENU uses the filename */tftpboot/bpmenu.M* on the same server.

Optional menu scripts can be accessed by renaming or linking the BPMENU program to a different filename.

After downloading the menu script, its contents is interpreted and BPMENU waits for keyboard input. A sample BPMENU screen is shown in *Figure 17-1*.

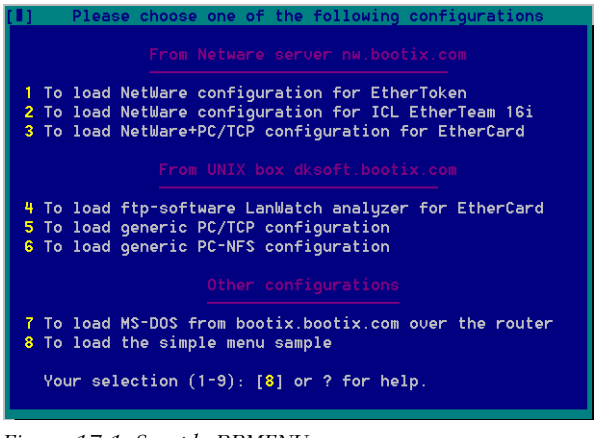


Figure 17-1. Sample BPMENU screen

The BPMENU Script Language

The BPMENU menu script language has a simple line-by-line command syntax. All commands start with a dot '.' in the first column, a three-character command given by capital letters, and a space character if an argument follows. All lines which do not conform to this specification or that have an invalid argument syntax are ignored and treated as comment lines. However, in our examples all comment lines start with the character "#":

```
# clear screen
.CLS

# the selections
.WLN This is a TCP/IP BOOT-PROM menu script, please select:
.WLN
.WLN 1 - will download a RAM disk image into Extended Memory
.KEY 1 192.109.23.193:/tftpboot/image.X
.WLN 2 - will download a RAM disk image into Base Memory
.KEY 2 192.109.23.193:/tftpboot/image

# prompt for input and wait for keypress
.WRT Your choice:
```

Some commands take an optional argument. The argument starts after the space following the command and extends until the end the of line. Thus, appended spaces are also interpreted. To delimit these appended spaces, terminate the line with a backslash '\'. In this case, the backslash is ignored by BPMENU.

After the menu script has been interpreted, BPMENU waits for input from the keyboard. The action taken for a particular keystroke is specified in the menu script.

To create, edit and display BPMENU scripts, you may want to use the BPSHELL program as described in *"Display BPMENU Script"* on page 153.

Script Language Commands

The BPMENU script language provides the following commands:

ATT

```
.ATT [attribute_val]
```

Sets the current character attribute to the value of the optional decimal argument *attribute_val*. If *attribute_val* is not given, the value is toggled back to that one used before the last change. Possible values for *attribute_val* are 0 to 255. See *"Video Adapter Character Attributes"* on page 209.

CLS

```
.CLS [attribute_val]
```

Fills the screen with the character attribute *attribute_val* and sets the cursor to position x=0, y=0. If no argument is given, the character attribute defaults to the value 7, which means white characters on black background. Possible values for *attribute_val* are 0 to 255. See *"Video Adapter Character Attributes"* on page 209.

DEF

```
.DEF key [timeout_val]
```

Sets a time-out key which can be one of [a-zA-Z0-9?]. Note that no national keyboard mapping is done. If no key is pressed within timeout_val seconds, the specified key is used as the user's input. If no *timeout_val* is given, 15 seconds is the default.

KEY

```
.KEY key [[gip_addr/]ip_addr]:filename
```

Defines which file is to be downloaded when the key given by the argument *key* is pressed. If filename is tagged by '*.M*', this file is downloaded and used as a new

menu script. If the filename is tagged by '.*X*', the file is loaded into Extended Memory.

If no *ip_addr* is given or the IP address is 0.0.0.0, `BPMENU` uses the former IP address, that is the address from which the last menu script was downloaded. For booting over a router or gateway, its address can be specified by *gip_addr*.

In any case, the separator character ':' is mandatory.

POS

```
.POS [x] [y]
```

Positions the cursor to position *x*, *y*. The upper left corner is 0, 0. If argument *y* is missing, the cursor is positioned to column *x* within the current row. If the argument *x* is also missing, the cursor is placed to the upper left corner.

PWD

```
.PWD key cpasswd
```

Associates a password with *key*. If the user presses the key that is given by the argument *key*, a password must be given in order to load the configuration. The encrypted password *cpasswd* can be generated using the `BPSHELL` program as described in *"Encode A Password"* on page 151.

WLN

```
.WLN [text]
```

Writes the string given by argument *text* to the current cursor position using the current character attributes and moves the cursor to the next line. If no argument *text* is given, the cursor is just moved to the next line. Within the text, a backslash '\' can be used to mark the end of the string if the string ends with space. The terminating backslash will not be printed.

WRT

```
.WRT text
```

Similar to `.WLN` but does not move the cursor to the next line after writing the string given by argument text to the screen. Within the text, a backslash '\' can be used to mark the end of the string if the string ends with space. The terminating backslash will not be printed.

The maximum size of a menu script is 8192 bytes. The maximum number of key commands allowed is 16. If the user presses the RETURN key, the default key is used.

Video Adapter Character Attributes

bit encoding								function	values
7	6	5	4	3	2	1	0		
x								blink	0 = no blink, 1 = blink
	x	x	x					background color	see color table
				x				intensity	0 = normal, 1 = high intensity
					x	x	x	foreground color	see color table

For monochrome video adapters, the possible colors are 000 for black and 111 for white. Only for the foreground color, there is a special value 001 available which causes characters to be underlined.

For color (CGA/EGA/VGA) video adapters, the following colors are available for both background and foreground:

bit encoding	color
000	black
001	blue
010	green
011	cyan
100	red
101	magenta
110	brown
111	white

Examples:

white on black: 00000111 (bin) = 7 (dec)

white on blue: 00011111 (bin) = 31 (dec)

The BPBOOT Boot Loader

BPBOOT is a powerful multi-purpose bootstrap loader that is downloaded by the TCP/IP BOOT-PROM instead of a boot image. BPBOOT checks for "magic keywords" that are embedded in the BOOTP/DHCP reply information. Based on these keywords and also on local hard disk status information, BPBOOT can perform commands before the operating system itself is loaded. Being able to dynamically decide whether to boot from a boot image file or from the local hard disk, BPBOOT takes an essential part in completely automated operating system installations.

Introducing BPBOOT

The TCP/IP BOOT-PROM can not only download a boot image file and install this as a virtual boot disk, it can also download and execute application programs that have been developed to interact with the TCP/IP BOOT-PROM code.

Like BPMENU, BPBOOT is such an application program that can be downloaded by the TCP/IP BOOT-PROM in place of a boot image file. BPBOOT empowers the network administrator to setup automated conditional boot environments.

BPBOOT is configured by "magic keywords" that it takes from custom options within the BOOTP/DHCP reply. These keywords instruct BPBOOT to query for information and to execute commands depending upon the query results.

This is especially useful in completely automated operating system installations. Microsoft operating systems like Windows 95/98/ME/NT/2000 require that the PC is rebooted multiple times during the installation process. The TCP/IP BOOT-PROM, on it's own, will download the same boot image every time the PC starts up.

With BPBOOT, you can implement a fully automated operating system installation like the following:

First PC boot

BPBOOT decides to load a boot image from the network. Within the boot image, the BMFDISK utility is used to create partitions on the hard disk and to quick format FAT16 partitions. Then, the PC is automatically rebooted.

Second PC boot

Again, BPBOOT decides to load a boot image from the network. Within the boot image, a network drive is mapped to the installation server. Using this network drive, the operating system setup program can copy all required filed files from the network server to the PC's local hard disk. After that, the client PC is automatically rebooted.

Third PC boot and all following reboots

This time, BPBOOT decides to boot from the local hard disk in order to continue the operating system installation. The operating system installation routine will perform additional reboots, and BPBOOT will take care that they all occur from the local hard disk.

Triggering a new operating system installation

To trigger a new operating system installation, all that has to be done is to change a magic keyword in the client PC's boot configuration on the BOOTP/DHCP server. When the PC boots the next time, the installation process will start over.

Installing BPBOOT

The BPBOOT boot loader replaces the boot image file on the server and is downloaded by the TCP/IP BOOT-PROM. Therefore you can treat BPBOOT like a boot image file.

Copy the file *bpboot* from the TCP/IP BOOT-PROM Utility Disk into the directory on your server where all TFTP and boot image files are located. Make sure that the file is world-readable.

In your boot server configuration file, specify the bootstrap loader as the boot image.

The following sample assumes that the file *bpboot* has been copied into the server's */tftpboot* directory:

```
# excerpt from a sample bootptab entry using BPBOOT

   :hd=/tftpboot:bf=bpboot:
```

BPBOOT's Magic Keywords

The BPBOOT boot loader is controlled by "magic" keywords that appear as custom options in the BOOTP/DHCP reply information. The following keywords are available:

Keyword	Explanation
BpSrV	TFTP server IP address
BpRoU	Router/Gateway IP address
BpOpT	Download global and/or individual custom options from ASCII files
BpDiS	Allow user to request network boot
BpDbG	Enable diagnostic information
BpBoOt	Condition string (implemented as binary option)

These keywords can be located in any option field of the BOOTP/DHCP reply. If you want to use multiple keywords, you can use either one BOOTP/DHCP option per keyword or concatenate multiple keywords in a single option by separating them with a semicolon.

The following excerpt from a *bootptab* file shows how to use one option per keyword:

```
:T128="BpSrV=172.16.0.1":\
:T129="BpRoU=172.16.0.254":
```

The same can be achieved by concatenation:

```
:T128="BpSrV=172.16.0.1;BpRoU=172.16.0.254":
```

BpSrV - TFTP Server IP Address

In order to download a boot image, the BPBOOT boot loader needs to know the IP address of the TFTP server that holds the boot image file. Some BOOTP/DHCP servers do not set their own IP address in the BOOTP/DHCP reply. If, after starting the client, the BPBOOT bootstrap loader comes-up with the error message *BPBOOT-E10: TFTP server IP address not set*, you must make an additional entry in your BOOTP/DHCP configuration file similar to the following:

```
# TCP/IP BOOT-PROM test client
bpclient:\
        :hn:vm=rfc1048:ht=ethernet:\
        :ha=0000c0094ff9:ip=172.16.10.1:\
        :hd="c:/tftpboot":bf=bpboot:\
        :T128="BpSrV=172.16.0.1":
```

Here, 172.16.0.1 is the IP address of the TFTP server. Note that the TFTP server can reside on the same server as the BOOTP/DHCP server. The option number

(128 in this example) can be any number greater than or equal to 128. BPBOOT will automatically search all user options for the BpSrV keyword.

BpRoU - Router IP Address

If the TFTP server can only be reached via a router or gateway, then you can use the keyword BpRoU to specify the router's IP address.

It is very likely that you will use BpRoU together with BpSrV. You can concatenate both options using a semicolon ';' as in the following example:

```
:T140="BpSrV=172.16.0.1;BpRoU=172.16.0.254":
```

BpOpT - Options Files

Some BOOTP/DHCP servers are limited in the amount of custom option information they can send. To overcome these limitations, the BPBOOT bootstrap loader is designed to transfer text based *options files* from the TFTP server to the client via TFTP and include its option values in the client's BOOTP/DHCP buffer space. Then, using the BPUTIL program, the client can retrieve these values for various uses as the boot process proceeds. The format of an options file is as follows:

```
# Sample options file
#
# CD-ROM driver
T128=CDROM.SYS
# DNS domainname
T129=bootmanage.com
# Installation server
T130=172.16.0.1
# Windows NT serial number
T150=123-1234567
#
# BpBoOt condition string (hexadecimal encoded)
B160=4270426f4f74001122334455
```

It is possible to specify both ASCII and hexadecimal options. For an ASCII option, each line contains the letter T, a three-digit decimal number specifying the custom option number, an equal sign, and an ASCII string defining the contents of the custom option.

For a hexadecimal option, each line contains the letter B, a three-digit decimal number specifying the custom option number, an equal sign, and a string of hexadecimal characters. Two hecadecimal characters, respectively, represent one byte of the value that is placed in the option field.[1]

1 Hexadecimal options are especially useful to specify BpBoOt condition strings within options files.

Lines starting with a hash '#' character are treated as comments.

Please note that the option numbers are arbitrary. For custom options, you can use any number between 128 and 254, inclusive.

The `BpOpT` keyword instructs the `BPBOOT` boot loader to download an options file. As a parameter, `BpOpT` accepts a two digit hexadecimal number which specifies the action to be taken:

Keyword	Explanation
`BpOpT=01`	download only global options
`BpOpT=02`	download only individual options
`BpOpT=03`	download both global and individual options

Global Options (BpOpT=01)

If `BpOpT` is set to 01, then the `BPBOOT` bootstrap loader will download a file named *bpboot.opt* from the same directory as where the *bpboot* file is stored. As such, these options can be used by all clients, hence, as *global options*. Note that if you have renamed `bpboot` to e.g. `nt4inst`, it will look for the file *nt4inst.opt* instead!

After downloading the *bpboot.opt* file, `BPBOOT` incorporates these additional custom options with the in-memory BOOTP/DHCP reply information. If a certain option is present in both the BOOTP/DHCP server and the global options file, then the value in the global options file takes precedence. All other custom options will be merged.

You can build client groups by renaming the `BPBOOT` bootstrap loader. For example, renaming *bpboot* to *sales* causes the bootstrap loader to use the global option file *sales.opt*. In this case, use *sales* as the bootfile name in the BOOTP/DHCP configuration file.

Individual Options (BpOpT=02)

If BpOpT is set to 02, then the `BPBOOT` bootstrap loader will download an option file named <*MACaddr*>*.opt*, where <*MACaddr*> represents the last 8 digits of the client's MAC (hardware) address. The format of this individual option file is the same as the global option file (*bpboot.opt*). Hence, a client with the MAC address 00.00.c0.12.54.ef would download an option file named *c01254ef.opt*. This can be used to provide clients with individual custom options. If a certain option is present in both the BOOTP/DHCP server and the individual options file, then the value in the individual options file takes precedence. All other custom options will be merged.

Global And Individual Options (BpOpT=03)

If BpOpT is set to 03, then the BPBOOT bootstrap loader will **first** download the global option file *bpboot.opt* and **then** the individual option file <*MACaddr*>.*opt*. If a certain option is present in the BOOTP/DHCP server and/or the global and/or the individual options file, then the following precedence rules apply:

- global options take precedence over BOOTP/DHCP Server options.
- individual options take precedence over global options
- individual options also take precedence over BOOTP/DHCP options.

The following table illustrates this.

BOOTP/DHCP Server	global options file	individual options file	effective result
		T160="individual"	**T160="individual"**
	T160="global"		**T160="global"**
	T160="global"	T160="individual"	**T160="individual"**
T160="server"			**T160="server"**
T160="server"		T160="individual"	**T160="individual"**
T160="server"	T160="global"		**T160="global"**
T160="server"	T160="global"	T160="individual"	**T160="individual"**

It is possible to combine the BpOpT and BpBoOt keywords to build advanced configurations. For detailed information, see *"BpBoOt - Multi-Purpose Conditions"* on page 219.

BpDiS - Interactive Network Boot

To allow end-users to force a network boot, the keyword BpDiS is provided. If BPBOOT finds the keyword BpDiS in the BOOTP/DHCP reply, it displays a message on the client's screen and allows the user to request a network boot by pressing a key within a given time. If the user presses the key, BPBOOT will boot from the network and set the custom option T254 to the value BpKeY. If the user does not press the key, then BPBOOT will continue as if the BpDiS keyword was not present.

The first digit of the two digit argument following the BpDiS keyword defines the message to be displayed, and the key to be pressed, and also the filename of the boot image file to be downloaded. Supported values are:

first digit	message displayed	key(s) to press	boot image filename
0x	Press <SPACE> to start installation services	SPACE	*filename*.X
1x	Press <F10> to start installation services	F10	*filename*.X
2x	Press <Alt><F10> to start installation services	Alt F10	*filename*.X
4x	Please wait	SPACE	*filename*.X
5x	Please wait	F10	*filename*.X
6x	Please wait	Alt F10	*filename*.X
8x	Press <SPACE> to start installation services	SPACE	*filename*0
9x	Press <F10> to start installation services	F10	*filename*0
Ax	Press <Alt><F10> to start installation services	Alt F10	*filename*0
Cx	Please wait	SPACE	*filename*0
Dx	Please wait	F10	*filename*0
Ex	Please wait	Alt F10	*filename*0

The second digit following the BpDiS configuration option specifies the time duration in seconds for which the BPBOOT bootstrap loader displays the message. Using an argument of less than 1 for the time duration disables the BpDiS option.

Example 1

In this example, we demonstrate how to let the user choose between network and local boot. Let us assume that you have the following entries in your bootptab file:

```
pc014:\
      ...
      :bf=bpboot:\
      ...
      :T150="BpDiS=03":
```

At PC startup, the BootManage TCP/IP BOOT-PROM downloads and executes the BPBOOT boot loader (because the bootfile's name is specified as *bpboot* in the bf option). When BPBOOT is executed, it scans the BOOTP REPLY for magic keywords and finds BpDiS=03 in custom option 150. The value 03 instructs BPBOOT to display the message *Press <SPACE> to start installation services* for 3 seconds. If the user presses the space bar within this time, BPBOOT will download the boot image file *bpboot.X*[1] and boot the PC from it. If the user does not press the space

1 The filename bpboot.X is constructed from the boot loader's own filename and the extension which is implicitly specified by the first digit of the BpBoOt magic keyword's argument (see table).

bar, then the BPBOOT bootstrap loader will continue with the standard boot process.

Example 2

In this example, we demonstrate how to hide the boot choice from the user, so only initiated users will know that they can trigger a network boot. Let us assume that you have the following entries in your bootptab file:

```
train17:\
      ...
      :bf=nt4inst:\
      ...
      :T145="BpDiS=51":
```

Here, the boot loader's filename has been renamed from *bpboot* to *nt4inst*. At PC startup, the BootManage TCP/IP BOOT-PROM downloads and executes the BPBOOT boot loader (the filename has changed, but it still contains the BPBOOT code). When BPBOOT is executed, it finds BpDiS=51 in custom option 145. The value 51 instructs BPBOOT to display the message *Please wait* for one second. If the user presses the F10 key within this time, BPBOOT will download the boot image file *nt4inst.X*[1] and boot the PC from it. If the user does not press the space bar, then the BPBOOT bootstrap loader will continue with the standard boot process.

Example 3

In our last example, we show a more complex configuration in which, at user's keypress, the menu loader BPMENU will be loaded instead of a boot image. Let us assume that you have the following entries in your *bootptab* file:

```
testpc3:\
      ...
      :bf=bpboot:\
      ...
      :T170="BpDiS=A5":
```

At PC startup, the BootManage TCP/IP BOOT-PROM downloads and executes the BPBOOT boot loader (bf=bpboot). When BPBOOT is executed, it finds BpDiS=A5 in custom option 170. The value A5 instructs BPBOOT to display the message *Press <Alt><F10> to start installation services* for 5 seconds. If the user presses the Alt and F10 keys within this time, BPBOOT will download and execute the file *bpboot0* which is the renamed menu loader BPMENU. Thus, we have found a means of presenting a boot menu to a user, but only if the user explicitly requests is. If the user does not press Alt F10, then the BPBOOT bootstrap loader will continue with the standard boot process.

---

1 Again, the filename nt4inst.X is constructed from the boot loader's own filename (which has been renamed here to nt4inst) and the extension given in the table.

It is possible to combine the BpDiS and BpBoOt keywords to build advanced configurations. If the user does not press the key(s) indicated by BpDiS, the BPBOOT boostrap loader will scan the BOOTP REPLY for a BpBoOt condition string. If it finds one, it will analyze it as documented in *"BpBoOt - Multi-Purpose Conditions"* on page 219.

BpDbG - Diagnostic Information

The `BpDbG` keyword instructs the `BPBOOT` boot loader to display diagnostic information. As a parameter, `BpDbG` accepts a two digit hexadecimal number which defines the amount of diagnostic information to be displayed:

Keyword	Explanation
`BpDbG=00`	Disable diagnostic information
`BpDbG=01`	Diagnostic level 1 (lowest)
`BpDbG=02`	Diagnostic level 2
`BpDbG=03`	Diagnostic level 3
`BpDbG=04`	Diagnostic level 4 (highest)

BpBoOt - Multi-Purpose Conditions

The `BPBOOT` program is remotely controlled by a condition string that is located within a custom option of the BOOTP/DHCP reply. Any custom option field can be used for this condition. The `BPBOOT` program detects its configuration by a magic ID which is the hexadecimal representation of the ASCII string BpBoOt. `BPBOOT` will scan any vendor field to check if it contains that magic ID at its beginning. If it finds one, it will use the information following the magic ID as a condition.

As opposed to the other magic keywords, BpBoOt requires to be specified as hexadecimal, not as ASCII string. So, the string BpBoOt is represented as the hexadecimal values 42, 70, 42, 6f, 4f, 74, which are followed directly by the (also hexadecimal) condition bytes.

The following excerpt from a bootptab file illustrates the usage of a BpBoOt condition string:

```
# this is a sample bootptab entry using BPBOOT:
pc123:\
     :hn:vm=rfc1048:ht=ethernet:ms=1024:ha=0020aff9cb77:\
     :sm=255.255.255.240:gw=195.4.136.199:ds=195.4.136.199:\
     :ip=195.4.136.198:\
     #     B p B o O t <cond> \
     :T160=4270426f4f74000000:\
     :hd="c:/tftpboot":bf=bpboot:
```

This example passes the six hex digits 000000 to BPBOOT as a condition. The meaning of these digits is explained later in this chapter.

You can also pass multiple 6-digit-groups to BPBOOT. The length must always be a multiple of 6, otherwise BPBOOT will ignore the condition:

```
pc123:\
     :hn:vm=rfc1048:ht=ethernet:ms=1024:ha=0020aff9cb77:\
     :sm=255.255.255.240:gw=195.4.136.199:ds=195.4.136.199:\
     :ip=195.4.136.198:\
     #     B p B o O t <cond><cond> \
     :T160=4270426f4f74001122334455:\
     :hd="c:/tftpboot":bf=bpboot:
```

Here, two groups of conditions are passed, the first is 001122 and the second one is 334455. BPBOOT starts interpreting the condition from left to right. If the condition is not true, it will continue execution at the next group. If the condition is true, it executes the command of the condition and continues execution at the next group. However, if the command tells BPBOOT to boot from a device, reboot or lock the PC, then the next condition is not evaluated.

If none of the conditions is true, then BPBOOT executes a "default condition" which is 000000. You can think of the default condition as if a "hidden" condition of 000000 is always appended to your condition string. For detailed information, please refer to *"The Hidden Default Condition"* on page 224.

Condition String Syntax

The BPBOOT condition string is subdivided into the following parts:

- the 12 hex digit magic ID 4270426f4f74 (the ASCII code representation of the string BpBoOt)
- one or more 6 hex digit conditions that are evaluated one at a time, from left to right

The following example shows a sample condition string that only contains of a single condition:

```
rem a condition string containing a single condition
:T160=4270426f4f74000001:
      \__________/\____/
           |         |
       magic ID    condition
```

The following example shows a sample condition string that contains three conditions:

```
rem a condition string containing three conditions
:T160=4270426f4f7413001413f11413f300:
      \__________/\____/\____/\____/
           |         |     |     |
       magic ID    cond1 cond2 cond3
```

Each 6 hex digit condition is subdivided into the following parts:

- a one hex digit query (Q)
- a one hex digit query option (QO)
- a two hex digit argument (ARG)
- a one hex digit command (C)
- a one hex digit command option (CO)

The following sample identifies the elements of a condition:

```
                    ARG
                    ||
:T160=4270426f4f74000001:
                   ||  ||
                   |QO |CO
                   Q   C
```

Query (Q)

The query is a 4-Bit value represented by one hex digit. Possible values are:

Q	Query
0	Always
1	If partition ID of partition number QO is equal to ARG
2	If partition QO is marked active
3	If partition ID of partition QO is not equal to ARG
4	If partition ID of partition QO is higher than ARG
5	If partition ID of partition QO is lower than ARG
6	No operation

Query Option (QO)

The query option is a 4-Bit value represented by one hex digit. Possible values are:

QO	Query Option
0	Partition number 0
1	Partition number 1
2	Partition number 2
3	Partition number 3
4	Any partition

Argument (ARG)

The argument is a general purpose 8-Bit value represented by two hex digits. It can hold any value between 00 and FF.

Command (C)

The command is a 4-Bit value represented by one hex digit. Possible values are:

C	Command
0	Boot from the hard disk. CO indicates the partition to boot from.
1	Boot from the network. CO indicates the boot image to boot from. Magic keywords such as BpOpt are evaluated.
2	Boot using the standard system BIOS bootstrap.
3	Increment the partition ID of partition QO by one.
4	Set the partition ID of partition QO to the value ARG.
5	Reboot system.
6	Lock system.
7	Request boot-time user authentication and boot from the hard disk. CO indicates the partition to boot from. See *"Boot-Time User Authentication"* on page 226.
8	Request boot-time user authentication and boot from the network. CO indicates the boot image to boot from. See *"Boot-Time User Authentication"* on page 226.
9	Request boot-time user authentication and boot using the standard system BIOS bootstrap. See *"Boot-Time User Authentication"* on page 226.
A	Increment the partition ID of partition QO by one and boot from the hard disk.
B	Increment the partition ID of partition QO by the value given in ARG.
C	Boot from the hard disk. CO indicates the partition to boot from. Magic keywords such as BpOpt are evaluated.
E	Similar to command '1', but use a mapped boot image file name built from the original filename and the client's MAC address.

Command Option (CO)

The command option is a 4-Bit value represented by one hex digit. Its meaning depends upon the value of the command (C).

If the command indicates a hard disk boot (C = '0'), the command option has the following meaning:

CO	Command Option for hard disk boot (C = '0')
0	active partition

If the command indicates a network boot (C = '1'), the command option has the following meaning:

CO	Command Option for network boot (C = '1')		Comment
0	*filename*.X	(append .X to filename)	Use standard TFTP to download the boot image file *filename?.X*, install it as a RAM disk in extended memory and boot from it.
1	*filename*1.X	(append 1.X to filename)	
2	*filename*2.X	(append 2.X to filename)	
3	*filename*3.X	(append 3.X to filename)	
4	*filename*.PX	(append .PX to filename)	Use large block TFTP to download the boot image file *filename?.PX*, install it as a RAM disk in extended memory and boot from it.
5	*filename*1.PX	(append 1.PX to filename)	
6	*filename*2.PX	(append 2.PX to filename)	
7	*filename*3.PX	(append 3.PX to filename)	
8	*filename*0	(append 0 to filename)	Use standard TFTP to download the boot loader *filename?* to conventional memory and transfer execution control to it.
9	*filename*1	(append 1 to filename)	
A	*filename*2	(append 2 to filename)	
B	*filename*3	(append 3 to filename)	
C	*filename*.X	MTFTP IP address + 1	Increment the Multicast IP address that has been used to download BPBOOT, and use it to download the boot image *filename?.X*.
D	*filename*1.X	MTFTP IP address + 2	
E	*filename*2.X	MTFTP IP address + 3	
F	*filename*3.X	MTFTP IP address + 4	

In this table, *filename* stands for the file name that has been used for the BPBOOT bootstrap loader itself. By default, this is *bpboot*. To support multiple boot configurations, you can rename *bpboot* to any filename you like.

Using the CO values 8, 9, A or B allows BPBOOT to download other boot loader programs like the BPMENU menu interpreter.

Using the CO values C, D, E or F allows BPBOOT to download boot image files by Multicast TFTP when BPBOOT itself has also been downloaded by Multicast TFTP. For detailed information, see *"Using BPBOOT With Multicast TFTP"* on page 227.

If the command indicates a network boot using a mapped filename (C = 'E'), the command option has the following meaning:

CO	Command Option for network boot using mapped filename (C = 'E')	Comment
0	MAP<*pathname*><*MACADDR*>.X	Use standard TFTP to download the boot image file <*MACADDR*>*?.X*, install it as a RAM disk in extended memory and boot from it.
1	MAP<*pathname*><*MACADDR*>1.X	
2	MAP<*pathname*><*MACADDR*>2.X	
3	MAP<*pathname*><*MACADDR*>3.X	
4	MAP<*pathname*><*MACADDR*>.PX	Use large block TFTP to download the boot image file <*MACADDR*>*?.PX*, install it as a RAM disk in extended memory and boot from it.
5	MAP<*pathname*><*MACADDR*>1.X	
6	MAP<*pathname*><*MACADDR*>2.X	
7	MAP<*pathname*><*MACADDR*>3.X	
8	MAP<*pathname*><*MACADDR*>0	Use standard TFTP to download the boot loader <*MACADDR*>*?* to conventional memory and transfer execution control to it.
9	MAP<*pathname*><*MACADDR*>1	
A	MAP<*pathname*><*MACADDR*>2	
B	MAP<*pathname*><*MACADDR*>3	

This makes it possible to define a common primary boot image file (the BPBOOT boot loader) for all clients which, in turn, requests an individual secondary boot image file for every client. This individual secondary boot image file is built using the client's MAC address.

Used in combination with the BootManage TFTP Server's filename mapping function, it is possible to build 'boot image groups' of PC clients. For more information, see *"Using Mapped Filenames"* on page 120.

The Hidden Default Condition

There is always one hidden condition appended to the condition string. BPBOOT evaluates the condition string from left to right. The hidden condition is 000000 which is always true and says *boot from active partition on hard disk*.

If BPBOOT doesn't find a valid condition string within a custom option of the BOOTP/DHCP reply, it will also evaluate the hidden condition. So, if BPBOOT seems to ignore your condition string and instead always boots from the hard disk, double-check the following:

- Is the custom option containing the condition string included in the BOOTP/DHCP reply? You may use the -s option of the BPUTIL or BMUTIL32 utility programs for this.
- Is the custom option correctly set up?

Sample Condition Strings

So far, we have been very theoretical. To show you the power of conditional booting, here are some sample configurations that use `BPBOOT`.

Example 1

In this example, the only condition used is 000000.

```
:T160=4270426f4f74000000:
      \_________/\____/
           |        |
       magic ID    cond1
```

BPBOOT evaluates this condition in the following way:

Condition	Corresponding Action
000000	Always boot from the active partition of the local hard disk

Example 2

Here we have two conditions, 10f300 and 000010.

```
:T160=4270426f4f7410f300000010:
      \_________/\____/\____/
           |        |     |
       magic ID    cond1 cond2
```

BPBOOT evaluates these conditions in the following way

Condition	Corresponding Action
10F300	If the first partition of the local hard disk has the partition ID value f3, then boot from the active partition of the local hard disk.
000010	Boot from the network using the boot image file *bpboot.X*.

Example 3

Here, the only condition is 50F210.

```
:T160=4270426f4f7450F210:
      \_________/\____/
           |        |
       magic ID    cond1
```

BPBOOT evaluates this condition in the following way:

Condition	Corresponding Action
50F210	Use the partition ID field of partition number 0 as a counter and boot from the network using the boot image file *bpboot.X* until this counter reaches the value F2. Then, boot from the active partition of the local hard disk.

Boot-Time User Authentication

BPBOOT allows to restrict access to the client PC at boot time by requesting a username and password. If the user provides a valid username and password, BPBOOT continues the boot process, otherwise it locks the client PC.

Upon successful login, BPBOOT sets the custom option T150 to the username and T151 to the (clear-text) password. This allows to use these values e.g. from within a batch file at a later time in the PC boot process, so the user does not need to enter them multiple times.

```
rem Excerpt from an AUTOEXEC.BAT demonstrating login
net logon #@T150*######### #@T151*######### /y /savepw:no
```

Boot-time user authentication only works with a special enhanced UNIX based TFTP server. Note that the password is **NOT** sent in clear text over the network.

Using BPBOOT With BMFDISK

In order to decide whether to boot the PC from the network or from the local hard disk, BPBOOT checks on status information that it stores in the partition table of the PC's local hard disk. Although BPBOOT is also able to modify this status information, normally BPBOOT only reads the status information while a different program modifies it.

From within the boot image file *bpboot.X*, you can query, set, increase and decrease the value in a partition ID field using the BMFDISK utility program.

This can be used to perform completely unattended operating system installations, including hard disk partitioning. By (mis)using the partition ID field of an unused partition as a counter or status flag, you can retain information across a PC reboot when the local hard disk isn't even partitioned or formatted.

Advanced Configuration

Combining BpBoOt And BpOpT

When the BPBOOT bootstrap loader does not find a BpBoOt condition string in the BOOTP/DHCP reply information, it looks for the BpOpT keyword. If the BpOpT keyword is present, the BPBOOT bootstrap loader downloads the global and/or individual options file(s). Then, the BPBOOT bootstrap loader looks for the BpBoOt condition string within the options file(s).

This technique allows to use individual BpBoOt condition strings with DHCP servers that can not provide individual options to clients.

Using BPBOOT With Multicast TFTP

When the TCP/IP BOOT-PROM has BPBOOT by using Multicast TFTP, BPBOOT, in turn, can be instructed to download a boot image also by Multicast TFTP. To do this, BPBOOT must know about the Multicast IP address that the Multicast TFTP server uses when sending the boot image by Multicast.

For this purpose, use the command option (CO) values 8, 9, A and B for network boot as described in *"Command Option (CO)"* on page 223.

BPBOOT first determines the Multicast IP address that has been used to download BPBOOT itself. Next, BPBOOT increments this value by 1, 2, 3 or 4 according to the command option CO. Then, BPBOOT uses the calculated valued to download the corresponding boot image by Multicast.

You can only make use of Multicast TFTP if you have a Multicast-enabled TFTP server such as the BootManage® TFTP Server. For more information about TFTP Multicast, see *"Multicast TFTP Operation"* on page 119 and also *"Multicast TFTP"* on page 127.

The BMDRV Device Driver

BMDRV is a Windows NT/2000 device driver that works in conjunction with the BootManage bootstrap loaders BPBOOT and PXBOOT[1]. BMDRV captures the BOOTP/DHCP reply information that was obtained by the BootManage bootstrap loader and makes it available under Windows NT/2000. The BMUTIL32 application/service can then patch this information into system configuration files and also into the Windows NT/2000 Registry. BMDRV can be configured to display a warning message and/or lock the PC if it has not been booted under control of the BootManage bootstrap loader.

Introducing BMDRV

The BootManage bootstrap loader is capable of controlling the bootstrap process of a PC that runs the Windows NT or Windows 2000 operating system. Since the BootManage bootstrap loader only runs in real mode, its operation is stopped when the Windows NT/2000 kernel is executed.

BMDRV is a Windows NT/2000 device driver which captures the BOOTP/DHCP reply information that was obtained by the BootManage bootstrap loader. This information can then be used by the BMUTIL32 application/service to patch system configuration files and/or the Windows NT/2000 Registry.

As a device driver, BMDRV is loaded and initialized very early in the startup phase of Windows NT/2000, and you can configure it to display a warning message and/or lock the system if BMDRV detects that the PC has not been booted under control of the BootManage bootstrap loader.

1 The BMDRV device driver is included in the TCP/IP BOOT-PROM and PXE Toolkit distributions. For the TCP/IP BOOT-PROM distribution, the bootstrap loader is BPBOOT. For the PXE Toolkit distribution, the bootstrap loader is PXBOOT.

Installing BMDRV

As a Windows NT/2000 device driver, BMDRV comes with an INF file that handles all installation activities. You can use the *BMDRV.INF* file to install or uninstall the BMDRV device driver. To support automated installations, you can also use the *BMDRV.INF* file to integrate the BMDRV device driver in an automated deployment of Windows NT/2000.

Manual Installation

To install the BMDRV device driver, insert the BootManage distribution disk or CD-ROM and navigate to the *BMDRV* folder. Within this folder, you find the following files:

- *BMDRV.SYS* the device driver program file
- *BMDRV.INF* control file for driver installation and deinstallation
- *INSTALL.CMD* command file for driver installation
- *UNINSTAL.CMD* command file for driver deinstallation
- *CMDLINES.TXT* sample *cmdlines.txt* entry for automated deployment

To install the BMDRV device driver, execute the *INSTALL.CMD* batch file and reboot the PC. After rebooting, the BMDRV device driver is listed in the Windows NT/2000 device manager as "BootManage BMDRV Driver".

To deinstall the BMDRV device driver, execute the *UNINSTAL.CMD* batch file and reboot the PC.

To install or deinstall the BMDRV device driver manually, you must be logged on as a user who has permissions to add/remove device drivers, e.g. the local or domain administrator.

The *CMDLINES.TXT file* contains information about how to add the BMDRV device driver to an automated Windows NT/2000 deployment which is discussed next.

Automated Installation

You can integrate the BMDRV device driver in an automated deployment of Windows NT/2000, so that BMDRV is automatically installed together with the operating system. The following description assumes that you are familiar with the Microsoft mechanisms of integrating third-party components in a Windows NT/

2000 automated deployment environment. If you are not familiar with these mechanisms, please refer to the Microsoft Windows NT 4.0 Resource Kit documentation.

First, make sure that your Windows NT/2000 distribution share contains a *OEM* directory, and within this, a subdirectory named *C*. In the *C* directory, create a subdirectory named *bootix* and copy the *BPDRV.SYS* and *BPDRV.INF* files to this subdirectory.

Then, in the *OEM* directory, open the file *cmdlines.txt* with a text editor and add the following lines:

```
[Commands]

; install BootManage BMDRV Device Driver
"rundll32 syssetup,SetupInfObjectInstallAction BMDRV_Install 128 c:\bootix\↵
bmdrv.inf"

; cleanup installation source directory
"cmd /c rmdir /s /q c:\bootix"
```

Note that, in the above example, the line starting with "rundll32..." had to be split in two lines due to space limitations. In the *cmdlines.txt* file, this has to be a single line. A sample *cmdlines.txt* file is located in the *BMDRV* folder of the BootManage distribution disk or CD-ROM.

Last, make sure that the Windows NT/2000 *unattend.txt* file contains an entry to include OEM components in the unattended setup and install a client PC to verify that the BMDRV device driver is automatically deployed together with the operating system.

Additional Requirements For Windows 2000

If you encounter the problem that the BMDRV driver fails to load on your Windows 2000 PC, you may have to add the switch /NOGUIBOOT to the Windows 2000 *BOOT.INI* file which is located in the root of the boot drive. Following, you see a sample *BOOT.INI* file that contains the /NOGUIBOOT switch.

```
[boot loader]
timeout=30
default=multi(0)disk(0)rdisk(0)partition(1)\WINNT

[operating systems]
multi(0)disk(0)rdisk(0)partition(1)\WINNT="Windows 2000" /NOGUIBOOT
```

Configuring BMDRV

You can configure how the BMDRV device driver behaves when it detects that the PC has not been started under control of the BootManage bootstrap loader[1]. This

is especially useful in environments where you want to enforce that the PC can only be started under control of the BootManage boot loader.

For this purpose, we recommend that the *BMDRV.INF* file be modified with a text editor before it is used to install the `BMDRV` device driver. See *"Editing The BMDRV.INF File"* on page 233 about how to do this.

If the `BMDRV` device driver is already installed, you may also use a graphical Registry editing tool like `REGEDIT` or `REGEDT32` to perform the changes directly in the Windows NT/2000 Registry. These changes will be in effect on the next reboot.

Be extremely careful when modifying device driver information in the Windows NT Registry. Incorrect entries can cause your system to crash or lock up and be rendered unusable.

Configuration Parameters

Configuration information for the `BMDRV` device driver is located in the Windows NT/2000 Registry under the key

```
HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\BMDRV\Parameters
```

Table 19-1 shows the available configuration parameters under this Registry key.

Table 19-1. Registry configuration parameters for the BMDRV device driver

Value Name	Value Type	Description
DisplayColor	REG_BINARY	Color code for failure and lock messages
DisplayMessages	REG_BINARY	Enable/Disable failure and lock messages
FailureMessage	REG_SZ	Message to be displayed on failure
LockMessage	REG_SZ	Message to be displayed on system lock
LockSystemOnFailure	REG_BINARY	Determine if the system is locked on failure

DisplayColor

This REG_BINARY value defines the color attribute used for displaying messages. Some sample (hexadecimal) values are:

4F: high intensity white characters on red background
84: flashing red characters on black background
1F: high intensity white characters on blue background

1 This can happen if a BOOTP/DHCP/TFTP timeout occurred or if the network boot process was aborted by the user.

A complete table containing all possible color values is listed in *"Color Codes"* on page 235.

DisplayMessages

This REG_BINARY value defines whether or not BMDRV displays failure or lock messages. If set to the value 00, BMDRV does not display messages when it detects that the PC hadn't been started under control of the BootManage bootstrap loader. If set to a value other than 00, BMDRV displays messages.

FailureMessage

This REG_SZ value holds the message that is displayed when BMDRV detects that the PC has not been started under control of the BootManage bootstrap loader and LockSystemOnFailure is set to 00. You must set DisplayMessages to 01 to enable displaying messages.

LockMessage

This REG_SZ value holds the message that is displayed when BMDRV detects that the PC has not been started under control of the BootManage bootstrap loader and LockSystemOnFailure is set to 01. You have to set DisplayMessages to 01 to enable displaying messages.

LockSystemOnFailure

This REG_BINARY value defines whether or not BMDRV locks the PC in case of failure. If set to a value other than 00, BMDRV locks the system when it detects that the PC has not been started under control of the BootManage bootstrap loader. If set to 00, BMDRV does not lock the system.

Editing The BMDRV.INF File

Open the *BMDRV.INF* file with a text editor and locate the [Strings] section at the bottom of the file. There, you see variables that have the same name as the above mentioned configuration parameters. Change the values of these variables as desired, and use the modified *BMDRV.INF* file to install the BMDRV driver. Following, you see sample entries in the [Strings] section of the *BMDRV.INF* file:

```
[Strings]
LockMessage         = "The system is locked by the BMDRV driver"
FailureMessage      = "The BootManage bootstrap loader failed to boot"
LockSystemOnFailure = 0
DisplayMessages     = 1
DisplayColor        = 0x4f
```

Be careful when editing the *BMDRV.INF* file. After installing the BMDRV driver with a modified *BMFDRV.INF* file, use Registry Editor to verify that the values have been integrated correctly into the Windows NT/2000 Registry.

Sample Configurations

The following three configuration samples demonstrate how you can use the BMDRV driver to control the Windows NT/2000 startup process. All mentioned actions are only taken when BMDRV detects that the system wasn't started under control of the BootManage bootstrap loader.

Example 1

Don't display any messages and don't lock the system:

```
LockSystemOnFailure = REG_BINARY:00
DisplayMessages     = REG_BINARY:00
```

Windows NT/2000 comes up and allows you to log in. However, you will see a pop-up message saying that a device driver failed to load. The Event Viewer contains a record saying that BMDRV failed to load, and all patching options of the BMUTIL32 program do not work.

Example 2

Display a warning message but don't lock the system. For displaying the message, use high intensity white characters on red background:

```
LockSystemOnFailure = REG_BINARY:00
DisplayMessages     = REG_BINARY:01
DisplayColor        = REG_BINARY:4F
FailureMessage      = REG_SZ:"The BootManage bootstrap loader failed to boot"
```

The message "The BootManage bootstrap loader failed to boot" is displayed at the top of the screen in high intensity white characters on red background, but Windows NT/2000 comes up and allows you to log in. As in the previous example, the pop-up message appears, and BMUTIL32 patching does not work.

Example 3

Lock the system and display a message saying that the system is locked. For displaying the messages, use flashing red characters on black background:

```
LockSystemOnFailure = REG_BINARY:01
DisplayMessages     = REG_BINARY:01
DisplayColor        = REG_BINARY:84
LockMessage         = REG_SZ:"The system is locked by the BMDRV driver"
```

The message "The system is locked by the BMDRV driver" is displayed at the top of the screen in flashing red characters on black background. `BMDRV` locks the system, so no further Windows NT/2000 components are loaded. The only thing you can do is shut down or reboot the PC.

Security Concerns

To prevent users from tampering with the `BMDRV` device driver, make sure that the *BMDRV.SYS* file has appropriate file access rights so that users cannot delete or rename it. This is only possible if the underlying filesystem is NTFS. The *BMDRV.SYS* file is located in the *%SystemRoot%\System32\Drivers* directory.

Also, make sure that the Registry entries for `BMDRV` have appropriate access rights so that users cannot delete or rename them. The `BMDRV` Registry entries are located in the Registry key:

```
HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\BMDRV
```

Color Codes

Here, we show all the possible color codes that can be used to display `BMDRV` messages during PC startup.

bit encoding								function	values
7	6	5	4	3	2	1	0		
x								blink	0 = no blink, 1 = blink
	x	x	x					background color	see color table
				x				intensity	0 = normal, 1 = high intensity
					x	x	x	foreground color	see color table

For monochrome video adapters, the possible colors are 000 for black and 111 for white. Only for the foreground color, there is a special value 001 available which causes characters to be underlined.

For color (CGA/EGA/VGA) video adapters, the following colors are available for both background and foreground:

bit encoding	color
000	black
001	blue
010	green
011	cyan
100	red

bit encoding	color
101	magenta
110	brown
111	white

The BMUTIL32 Program

BMUTIL32 is a multi-purpose application program that runs on Windows NT and Windows 2000. Its main use is to query the BMDRV device driver for BOOTP/DHCP reply information and to patch this information into configuration files and/or into the Windows NT/2000 Registry. In addition, BMUTIL32 provides much of the functionality found in the BPUTIL/PXUTIL program described earlier in this manual. BMUTIL32 can also be installed as a Windows NT/2000 service to perform automatic patching during the earliest stages of the Windows NT/2000 boot process.

Introducing BMUTIL32

BMUTIL32 can be understood as the Windows NT/2000 version of the BPUTIL/PXUTIL program, which allows one to insert information from an ASCII file into the Windows NT/2000 Registry. BMUTIL32 can be used as a Windows NT/2000 service and, as such, is started at a very early stage in the system boot process. Also, it can be used as a command line program to make changes in the Windows NT/2000 Registry after the system has booted or after the user logs on.

BMUTIL32 relies on the BMDRV device driver to retrieve the BOOTP/DHCP information that has been obtained earlier by the BootManage bootstrap loader. All BOOTP/DHCP information passed to the client can be retrieved by BMUTIL32 and inserted into the Windows NT/2000 Registry.

In addition to the Windows NT/2000 Registry, BMUTIL32 can be used to patch BOOTP/DHCP information into ASCII and binary files.

Make sure that the BMDRV device driver is installed and running. Without BMDRV, BMUTIL32 can not access the BOOTP/DHCP information. BMUTIL32 can be used to modify the Windows NT/2000 Registry without BMDRV but, in this case, it cannot include BOOTP/DHCP information.

Installing BMUTIL32

You can use BMUTIL32 in one of two different ways:

- As a command line tool from the Windows NT/2000 command prompt, called interactively or from within a batch (*.BAT*) file, or Windows NT/2000 command (*.CMD*) file.
- As a Windows NT/2000 service which starts automatically at system startup.

Using BMUTIL32 As Command Line Tool

If you plan to use BMUTIL32 as a command line tool, you only need to copy the BMUTIL32 program into your Windows NT/2000 *%SystemRoot%\system32* directory.

Installing BMUTIL32 As Windows NT/2000 Service

You can install BMUTIL32 as a Windows NT/2000 service. This enables BMUTIL32 to run in a very early stage of the Windows NT/2000 system boot and patch information into the Registry before the Windows NT/2000 components that rely on these Registry keys are started. Also, being installed as a service allows BMUTIL32 to run within the LocalSystem security context, so BMUTIL32 has full access rights to Registry keys and NTFS files.

In *"BMUTIL32 As Windows NT/2000 Service"* on page 243, you will find detailed information about how to install BMUTIL32 as a Windows NT/2000 Service.

BMUTIL32 Commands

BMUTIL32 offers a variety of command line parameters which we will refer to as "commands". Quite a lot of these commands provide similar functionality as the BPUTIL or PXUTIL utility programs. BPUTIL is part of the TCP/IP BOOT-PROM distribution, whereas PXUTIL is included in the PXE Toolkit distribution. Whenever a BMUTIL32 command is similar to a BPUTIL/PXUTIL command in the following description, we will ask you to refer to the corresponding BPUTIL/PXUTIL command description that is located earlier in this manual.

In the TCP/IP BOOT-PROM Manual, please refer to the BPUTIL chapter. In the PXE Toolkit Manual, please refer to the PXUTIL chapter instead.

Command Overview

The following commands are available with BMUTIL32:

Command	Explanation
-?	Show BMUTIL32 usage
-a	patch BOOTP/DHCP reply information into ASCII file
-b	patch BOOTP/DHCP reply information into binary file
-c	show BootManage bootstrap loader configuration information
-C	check if the BMDRV device driver is running
-d	as -r but shows debugging information and does not modify the Registry
-D	as -R but shows debugging information and does not modify the Registry
-e	disables access to the BMDRV device driver
-f	flushes the Windows NT/2000 in-memory Registry copy to disk
-install	install BMUTIL32 as Windows NT/2000 service
-p	patch BOOTP/DHCP reply information into ASCII file
-parms	show BMUTIL32 Windows NT/2000 service startup parameters
-r	insert information from ASCII file into the Registry
-R	insert patched (DHCP/BOOTP) information into the Registry
-remove	remove BMUTIL32 Windows NT/2000 service
-run	start the BMUTIL32 Windows NT/2000 service
-s	show all or selected BOOTP/DHCP reply information
-S	show only the BOOTP/DHCP reply information that is available
-stop	stop the BMUTIL32 Windows NT/2000 service

Displaying BMUTIL32 Command Line Syntax

The -? command

```
bmutil32 -?
```

This command shows the copyright information and command line syntax of the BMUTIL32 program.

Patching ASCII And Binary Files

The -a, -p And -b Commands

```
bmutil32 -a text_patch_file [text_patch_output_file]
bmutil32 -p text_patch_file [text_patch_output_file]
bmutil32 -b tag binary_patch_file
```

The -a and -p commands are used to patch BOOTP/DHCP reply information into ASCII files, whereas the -b command patches binary files. The syntax of all these commands is identical to the corresponding options of the BPUTIL/PXUTIL program. For a detailed description, please refer to the BPUTIL/PXUTIL chapter.

Patching The Windows NT/2000 Registry

The -r Command

```
bmutil32 -r registry_file
```

The -r command inserts information from a specially formatted ASCII file into the Windows NT/2000 Registry. This ASCII file contains keys, value names and values. It can also contain comment lines starting with '#' or ';'.

```
C:\> bmutil32 -r c:\bootix\sample.bmr
```

This example instructs BMUTIL32 to read the file *sample.bmr* from the directory *c:\bootix* and insert its contents into the Registry. Below is the contents of a very simple *sample.bmr* file:

```
[HKLM\SOFTWARE\bootix\BootManage\TestKey]
"IPAddress"=REG_MULTI_SZ:"10.0.1.1"
```

This example shows how to set the value name IPAddress under the key HKLM\SOFTWARE\bootix\BootManage\TestKey to the value 10.0.1.1.

For the Windows NT/2000 Registry's root keys, BMUTIL32 supports abbreviations. For example, you can either use a root key's full name HKEY_LOCAL_MACHINE or its abbreviated form HKLM.

If you want to test the operations that the -r command performs without actually altering the Registry, use the -d command as described in *"The -d Command"* on page 241.

For a detailed description of the configuration file syntax, see *"Configuration File Syntax"* on page 246.

The -R Command

```
bmutil32 -R registry_patch_file
```

The -R command has the same functionality as the -r command, but before inserting information into the Registry, BMUTIL32 performs an in-memory patch with the BOOTP/DHCP reply information. The configuration file itself is not patched. The -R command allows you to use BOOTP/DHCP tags within the configuration file. For a complete list of all available tags, please refer to the BPUTIL/PXUTIL chapter.

Following, you see the contents of a sample configuration file named *sample.bmr* which contains two tags. BMUTIL32 does not require a special file extension, but as a convention, we use the extension *.bmr* (for BMUTIL32 Registry File) throughout this manual.

```
[HKLM\SOFTWARE\bootix\BootManage\TestKey\#@t130*#########\Parameters]
"IPAddress"=REG_MULTI_SZ:"#@yip*############"
```

The custom tag T130 contains the name of the network adapter used and the standard tag yip contains the IP address of the client. The contents of these tags is defined by the BOOTP/DHCP reply.

To patch and integrate the sample.bmr file into the Windows NT/2000 Registry, execute the command:

```
bmutil32 -R sample.bmr
```

The following example shows the contents of a larger `BMUTIL32` Registry File, where several keys and values are patched and inserted into the Registry:

```
[HKLM\SOFTWARE\bootix\BootManage\TestKey\#@t130*#########\Parameters]
"DefaultGateway"=REG_MULTI_SZ:"#@gw0*####################"
"EnableDHCP"=    REG_DWORD:0
"IPAddress"=     REG_MULTI_SZ:"#@yip*####################"
"SubnetMask"=    REG_MULTI_SZ:"#@smf*####################"

[HKLM\SOFTWARE\bootix\BootManage\TestKey\Parameters]
"Domain"=REG_SZ:"#@T155*################################################"
"Hostname"=REG_SZ:"#@chn*################"
"NameServer"=REG_SZ:"#@ds0*################"
"SearchList"=REG_SZ:"#@T155*##############################"

[HKLM\SOFTWARE\bootix\BootManage\TestKey\
\Parameters]
"ComputerName"=REG_SZ:"#@chn*################"
```

If you want to test the operations that the -R command performs without actually altering the Registry, use the -D command as described in *"The -D Command"* on page 241.

For a detailed description of the configuration file syntax, see *"Configuration File Syntax"* on page 246.

The -d Command

```
bmutil32 -d patch_file
```

The `-d` command is equal to the `-r` command but it does not change or insert values into the Windows NT/2000 Registry. The `-d` command can be used to debug and syntax-check the ASCII configuration file that contains the Windows NT/2000 Registry information.

The -D Command

```
bmutil32 -D registry_patch_file
```

The -D command is equal to the -R command but it does not change or insert values into the Windows NT/2000 Registry. The -D command can be used to debug and syntax-check the ASCII configuration file that contains the Windows NT/2000 Registry information.

The -f Command

```
bmutil32 -f
```

The -f command flushes (writes) the in-memory Windows NT/2000 Registry information to the hard disk. After the -f command has been used, the Windows NT/2000 Registry on the hard disk reflects the in-memory copy that Windows NT/2000 uses.

Display BOOTP/DHCP Information

The -s And -S Commands

```
bmutil32 -s [tag [tag ...]]
bmutil32 -S
```

The -s and -S commands are used to display BOOTP/DHCP reply information. The syntax of these commands identical to the corresponding options of the BPUTIL/PXUTIL program. For a detailed description, please refer to the BPUTIL/PXUTIL chapter.

Miscellaneous Commands

The -c Command

```
bmutil32 -c
```

The -c command is used to display TCP/IP BOOT-PROM or PXE related system information and is intended for debugging purposes only. The -c command is identical to the corresponding option of the BPUTIL/PXUTIL program. For a detailed description, please refer to the BPUTIL/PXUTIL chapter.

The -C Command

```
bmutil32 -C
```

The -C command is used to check if the BMDRV device driver is running and the BOOTP/DHCP reply information is available. The -C command uses the ERRORLEVEL to return this information. If the PC client was successfully booted under control of the BootManage bootstrap loader and the BMDRV device driver is run-

ning, then ERRORLEVEL is set to the value 0. If the BMDRV device driver failed to load (due to a network timeout or user abort), then the returned ERRORLEVEL value is 1.

The following command file illustrates the use of the -C command:

```
bmutil32 -C
if ERRORLEVEL 1 goto FAILURE
echo The BMDRV device driver is running,
goto END

:FAILURE
echo The BMDRV device driver failed to load.

:END
```

The -e Command

```
bmutil32 -e
```

The -e command disables access to the BMDRV device driver. Once the -e command is executed, the BOOTP/DHCP reply information can no longer be accessed. This is useful if BMUTIL32 is installed as Windows NT/2000 Service, and the BOOTP/DHCP reply information contains sensitive data that should not be visible to users. For more information, see *"Using Multiple Commands On The Command Line"* on page 245.

BMUTIL32 As Windows NT/2000 Service

BMUTIL32 is capable of being installed as a Windows NT/2000 service, so that it can make changes in the Windows NT/2000 Registry at a very early stage of the Windows NT/2000 bootstrap process. These changes can include patched information from the BOOTP/DHCP reply, which allows one to remotely reconfigure Windows NT/2000 computers simply by changing values in the BOOTP/DHCP configuration file on the boot server. Upon the next reboot, the BMUTIL32 service migrates these new values into the Windows NT/2000 Registry.

The -install Command

```
bmutil32 -install commandline
```

The -install command is used to install BMUTIL32 as a Windows NT/2000 service. On the command line, you must specify the command that BMUTIL32 should execute upon Windows NT/2000 system startup.

```
bmutil32 -install -R c:\bootix\startup.bmr
```

In this example, on every Windows NT/2000 system startup BMUTIL32 reads the file *c:\bootix\startup.bmr* into memory, performs an in-memory patch of all tags it

finds, and then migrates the information into the Windows NT/2000 Registry. The file *startup.bmr* itself is not changed.

You can specify multiple commands that should be executed by the BMUTIL32 service upon Windows NT/2000 system startup.

```
bmutil32 -install -R startup.bmr -a sample.txt
```

The following example instructs the BMUTIL32 service to execute the following two commands upon Windows NT/2000 system startup:

- migrate the information from the file *startup.bmr* into the Windows NT/2000 Registry
- patch the file *sample.txt* with the BOOTP/DHCP reply information

For a complete description of the configuration file syntax, see *"Configuration File Syntax"* on page 246. For a complete list of all available tags, please refer to the BPUTIL/PXUTIL chapter.

The -parms Command

```
bmutil32 -parms
```

The -parms command shows the commands that have been passed to BMUTIL32 with the -install command. These are the commands that BMUTIL32 executes upon Windows NT/2000 system startup. Before you can use the -parms command, you must install the BMUTIL32 service using the -install command.

The -remove Command

```
bmutil32 -remove
```

The -remove command removes the BMUTIL32 service.

The -run Command

```
bmutil32 -run
```

The -run command starts the BMUTIL32 service. This is the same as clicking a service's "Start" button from the Services control in the Windows NT/2000 control panel or executing "net start BMUTIL32" from the command line. Before you can use the -run command, you have to install the BMUTIL32 service using the -install command.

The -stop Command

```
bmutil32 -stop
```

The `-stop` command stops the `BMUTIL32` service. This is the same as clicking a service's "Stop" button from the Services control in the Windows NT/2000 control panel or executing "net stop BMUTIL32" from the command line. Before you can use the `-stop` command, you have to install the `BMUTIL32` service using the `-install` command.

Normally, the `-stop` command is unnecessary because `BMUTIL32` automatically stops after completing its work.

Using Multiple Commands On The Command Line

You can concatenate multiple commands on the `BMUTIL32` command line. Instead of calling `BMUTIL32` multiple times to perform different tasks, it is possible to combine these tasks and call `BMUTIL32` only once. The following example illustrates this.

Let us assume that you want `BMUTIL32` to perform three different tasks:

- patch the BOOTP/DHCP reply information into the file *c:\bootix\patch_me.txt*
- patch and integrate the contents of the file *c:\bootix\reg_me.bmr* into the Registry
- disable access to the BMDRV driver so the BOOTP/DHCP reply information is no longer available

Of course, you can call `BMUTIL32` three times, either interactively or from within a batch (*.BAT*) or command (*.CMD*) file:

```
bmutil32 -a c:\bootix\patch_me.txt
bmutil32 -R c:\bootix\reg_me.bmr
bmutil32 -e
```

However, the same tasks can be done in a single call to `BMUTIL32`:

```
bmutil32 -a c:\bootix\patch_me.txt -R c:\bootix\reg_me.bmr -e
```

This is especially useful when `BMUTIL32` is installed as a Windows NT/2000 Service, in which case it will only accept a single command line:

```
bmutil32 -install -a c:\bootix\patch_me.txt -R c:\bootix\reg_me.bmr -e
```

Security Concerns

To prevent users from tampering with the `BMUTIL32` service or its configuration files, make sure that the *BMUTIL32.EXE* file has appropriate file access rights so that users cannot delete or rename it. This is only possible if the underlying filesystem is NTFS. The *BMUTIL32.EXE* file is located in the *%SystemRoot%\System32*

directory. Also, check the file access rights for all *.BMR* files and for all text and binary files that are patched by `BMUTIL32`.

Make sure that the Registry entries for `BMUTIL32` have appropriate access rights so that users cannot delete or rename them. The `BMUTIL32` Registry entries are located in the Registry key:

```
HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\BMUTIL32
```

Configuration File Syntax

Here, we describe `BMUTIL32` Registry File structure that is used with the `BMUTIL32 -R` and `-r` commands to insert information into the Windows NT/2000 Registry. Although any file names and extensions are accepted, we encourage you to use the file extension *.bmr* for these files.

Comments

You can insert comment lines into the configuration file for readability and documentation purposes. A comment line begins with the character '#' or ';', placed in the first column of the line.

```
; this is a comment which is continued
; in the following line

# this is also a comment
```

Appending Lines

If a line gets longer than the editor or screen can display, the line can be split over multiple lines by adding the backslash '\' character at the end of that line.

```
; these are actually two lines\
but they are treated as one

; another example
[HKLM\SOFTWARE\bootix\
\BootManage\TestKey\Parameters]

; this is the same as
[HKLM\SOFTWARE\bootix\BootManage\TestKey\Parameters]
```

A backslash at the end of a line has the function to 'escape the invisible end-of-line character' and logically concatenate two lines. Make sure that the backslash is really the last character on the line, with no invisible characters (e.g. spaces) between the backslash and the end-of-line character.

Section Header

Each section in the configuration file needs a header which defines the key and subkey. The key and subkey start with an opening square bracket '[' character and end with a closing square bracket character ']'. After this header, multiple value names can follow, each on a seperate line.

```
; this is the section header defining registry key and subkey
[HKEY_LOCAL_MACHINE\SOFTWARE\bootix\BootManage\TestKey\Parameters]

; the following value names are all located within the above key
"IPAddress"  = "10.0.1.1"
"SubnetMask" = "255.0.0.0"
"Gateway"    = "10.0.0.254"
```

Root Key Abbreviations

For the Registry root keys, you can use either the full names or the standard abbreviations. Here is a list of all possible root keys and their corresponding abbreviations:

Abbreviation	Corresponding root key
HKLM	HKEY_LOCAL_MACHINE
HKCU	HKEY_CURRENT_USER
HKCR	HKEY_CLASSES_ROOT

Abbreviation	Corresponding root key
HKCC	HKEY_CURRENT_CONFIG
HKU	HKEY_USERS

The root keys HKEY_DYN_DATA and HKEY_PERFORMANCE_DATA are not supported because Windows NT/2000 permits read-only access to them.

Values

Values and their data follow a section header which defines the key and subkey under which the value is to be stored.

The default value type is REG_SZ, which stands for a string that is terminated by a 0 character. If no value type is given, then REG_SZ is chosen. In the following example, all value names will hold the same data:

```
[HKEY_LOCAL_MACHINE\SOFTWARE\bootix\BootManage\TestKey\Parameters]

"IPAddress1"="10.0.1.1"
"IPAddress2"=REG_SZ: "10.0.1.2"
"IPAddress3"= "10.0.1.3"
"IPAddress4" ="10.0.1.4"
"IPAddress5" = "10.0.1.5"
```

The value name is enclosed in double quotes and may be followed by spaces or tab characters. After the equal sign, the data for that key follows.

Supported Value Types

The following Registry value types are supported by the BMUTIL32 program:

REG_SZ

A REG_SZ value represents a string that is terminated by a 'null' character (SZ stands for StringZero). If no value type is given, BMUTIL32 assumes the REG_SZ value type as default. In a *.bmr* file, REG_SZ values can be specified in multiple ways as demonstrated in the following example:

```
; A string is enclosed in double quotes
"value1" = REG_SZ:"ABCDEFGH"

; Single characters are enclosed in single quotes
"value2" = REG_SZ:'A','B','C', 'D','E' , 'F' ,'G','H'

; Hex values for the characters A - H
"value3" = REG_SZ:41,42,43, 44,45 , 46 ,47,48

; Strings, characters and hex values can be mixed
"value4" = REG_SZ:'A',42,'C', 'D', 45 , "FGH"
```

REG_EXPAND_SZ

This is the same as `REG_SZ`, but can contain references to environment variables (e.g. `%PATH%` or `%SystemRoot%`) that are expanded before evaluation.

```
; A path that includes a system environment variable
"PathName" = REG_EXPAND_SZ:"%SystemRoot%\system32"
```

REG_MULTI_SZ

An array of null-terminated strings. The array itself is terminated by an additional 'null' character.

```
; A Multi-String entry containing three strings
"MultiString" = REG_MULTI_SZ:"first","second","third"
```

REG_BINARY

Binary data in hexadecimal format.

The data format must be a one-byte hexadecimal value represented as two hex digits, e.g. A8. If the value data is smaller than hex 10, it must be padded by a zero character to be 2 hex digits long, i.e. the value 1 must be written as 01. A REG_BINARY entry can hold multiple binary values.

```
; Some sample binary entries
"Value1" = REG_BINARY:01
"Value2" = REG_BINARY:10,a2,b5,7f,99
"Value3" = REG_BINARY:04, 13, AA, BF, 63, 8E
```

REG_DWORD

A 32-Bit number in little endian hexadecimal format. The most significant byte of a word is the high-order byte, and the most significant word of a longword is the high-order word.

The data format must be a 4-byte hexadecimal value represented as 8 hex digits, e.g. 12345678. Even if the value data is smaller than 4 bytes, it must be padded by zero characters to be 8 hex digits long, i.e. the value 1 must be written as 00000001. Otherwise, the data would be interpreted as a byte value.

```
; Some sample DWORD entries
"Maximum" = REG_DWORD:A23F0008
"Minimum" = REG_DWORD:00071AB2
"Average" = REG_DWORD:12345678
```

Advanced Techniques

Reusing TCP/IP BOOT-PROM Memory

You can use the memory area which is occupied by the program code in the TCP/IP BOOT-PROM for uploading programs. Use `EMM386` to do so:

```
device=emm386.exe i=cc00-cfff noems
```

This includes the memory range 0xcc00 to 0xcfff which is allocated by the TCP/IP BOOT-PROM on the network adapter for uploading DOS programs. The size and location of the ROM area depends on the network card and configuration.

Faster TFTP Transfer

The TCP/IP BOOT-PROM is able to adapt to non RFC standard TFTP packet sizes. If the TFTP server sends a larger packet than the RFC standard of 512 bytes, the TCP/IP BOOT-PROM will also work with the extended packet size.

Using larger packet sizes decreases the number of transmitted packets, which results in shorter boot time and reduced server load.

Packet sizes larger than 512 bytes will not work together with application which can not adapt to the packet size or reply on the standard RFC specification.

In order to use standard RFC (512 bytes) and non RFC standard TFTP servers in one network concurrently you can bind the non RFC standard TFTP daemon to UDP port 59 (reserved for private file transfer service). The standard RFC port for the TFTP daemon is 69 (reserved for TFTP file transfer service).

If the bootfilename is tagged by the capital letter P (e.g. */tftpboot/ramd.P*) or PX if the image is to be loaded in Extended Memory (e.g. */tftpboot/ramd.PX*), the TCP/IP BOOT-PROM first tries to connect to the TFTP daemon at port 59. If the TFTP request is not answered after 3 tries, the TCP/IP BOOT-PROM falls back to the standard RFC port 69.

Large Boot Images

If the RAM disk can not hold all the data you need to start your operating system and network software you can delete already loaded programs from the RAM disk and transfer new files using the `BPUTIL -T` option.

You may also want to create a larger RAM disk using the MS-DOS `RAMDRV` program and extract a compressed archive to this RAM disk.

Booting Across Routers

To find its network address and configuration, the TCP/IP BOOT-PROM uses the BOOTP protocol. This BOOTP request must be done by broadcasting over the network. Since routers do not route MAC (e.g. network hardware address) broadcasts by default, special measures must be taken if you want to enable BOOTP forwarding.

Some routers are able to do MAC routing or use IP helper addresses to handle BOOTP broadcasts. Otherwise, bridges must be used in order to transfer broadcasts between two network segments.

Troubleshooting

So you have a problem with the TCP/IP BOOT-PROM or with one of its associated programs? Don't panic! Setting up a powerful and complex remote boot / remote installation configuration implies that there are multiple things that can go wrong during the setup and testing phase. Fortunately, the TCP/IP BOOT-PROM and its associated programs provide various debugging functions that let you quickly isolate and resolve the problem.

Before contacting technical support, please check if your problem is listed below, and see if it can be solved using the given solution.

TCP/IP BOOT-PROM Is Not Activated

Problem: *When booting the client PC, the TCP/IP BOOT-PROM does not come up with its copyright message, and the PC boots from the local hard disk instead.*

Solution: Check with *"Installation"* on page 29 if you have correctly installed and activated the TCP/IP BOOT-PROM. Make sure that the PROM device is correctly inserted into the network adapter's socket and that the network adapter is configured to enable the PROM. For some PCs, you may also need to enable network boot in the System Setup.

BOOTP..... Displayed

Problem: *The TCP/IP BOOT-PROM displays its copyright message, but then only shows "BOOTP....."*

Solution: The TCP/IP BOOT-PROM does not receive a valid answer packet from a BOOTP/DHCP server. There may be multiple reasons for this behaviour.

- On a Combo network adapter, check that the TCP/IP BOOT-PROM uses the correct network connector. If the Network adapter is configured to use the

BNC connector, but you are actually using UTP, then packets are sent on the wrong port.

- Check that at least one BOOTP/DHCP server is running on the local network.
- In the BOOTP server's bootptab file, the record for the client PC is either missing or faulty. Enable the BOOTP server's debug mode to see if BOOTREQUEST packets are received and if BOOTREPLY packets are transmitted back.
- If all of the available IP addresses in a DHCP scope are already assigned to other PCs, the DHCP server will not offer an IP address.
- If there is a router between the client and the BOOTP server, check if the router is correctly configured to forward BOOTP packets. Routers normally use IP helper addresses to achieve this.

M41: BOOTP Timeout Occurred

Problem: *The TCP/IP BOOT-PROM displays the error message "M41: BOOTP timeout occurred"*

Solution: The TCP/IP BOOT-PROM repeatedly tried to receive a valid answer packet from a BOOTP/DHCP server and finally gave up trying. For possible solutions, see *"BOOTP..... Displayed"* on page 253.

M42: No Client Or Server IP

Problem: *The TCP/IP BOOT-PROM displays the error message "M42: no client or server IP"*

Solution: The TCP/IP BOOT-PROM has received a BOOTP/DHCP reply in which either the client IP address field (yip) or the server IP address field (sip) was not set. Make sure that you have valid client and server IP address entries in your BOOTP/DHCP server configuration.

M43: No Bootfilename

Problem: *The TCP/IP BOOT-PROM displays the error message "M43: no bootfilename"*

Solution: The TCP/IP BOOT-PROM has received a BOOTP/DHCP reply in which the bootfilename field was not set. Make sure that you have a bootfilename entry in your BOOTP/DHCP server configuration, and that the specified file exists.

M30: Can't ARP TFTP Address

Problem: *The TCP/IP BOOT-PROM displays the error message "M30: can't ARP TFTP address"*

Solution: The TCP/IP BOOT-PROM has received a correct BOOTP/DHCP reply in which the TFTP server's IP address is a different one than the BOOTP/DHCP server's IP address. When trying to resolve the MAC address of the TFTP server using the ARP protocol, the TCP/IP BOOT-PROM did not get an ARP reply. Check if the network path to the TFTP server is set up correctly. If you use a router, check if it is configured to allow access to the TFTP server.

TFTP..... Displayed

Problem: *The TCP/IP BOOT-PROM displays its copyright message, but then only shows "TFTP....."*

Solution: The TCP/IP BOOT-PROM has received a valid answer packet from a BOOTP/DHCP server, but cannot contact the TFTP server in order to download the boot image file. Note that the TFTP server can be a different one than the BOOTP/DHCP server. In order to determine the TFTP server's IP address, use the TCP/IP BOOT-PROM's verbose mode by pressing the key v at startup as described in *"Installation"* on page 29. Make sure that the client is able to contact the TFTP server, especially when working across subnets.

M32: TFTP Open Timeout Occurred

Problem: *The TCP/IP BOOT-PROM displays the error message "M32: TFTP open timeout occurred"*

Solution: The TCP/IP BOOT-PROM repeatedly tried to contact the TFTP server and finally gave up trying. For possible solutions, see *"TFTP..... Displayed"* on page 255.

T??: Access Violation

Problem: *The TCP/IP BOOT-PROM displays the error message "T??: access violation"*

Solution: The TFTP server is not able to access the requested boot image file. In order to determine the full path and filename of the boot image file, use the TCP/IP BOOT-PROM's verbose mode by pressing the key v at startup as described in *"Installation"* on page 29. Make sure that the

specified file exists in the TFTP server's file system and that the TFTP server has the permission to access this file. Also, check the TFTP server's log for possible access errors.

PC Hangs While Downloading A Boot Image

Problem: *The PC downloads a boot image and hangs somewhere between 0x200 and 0x260 KByte.*

Solution: The filename of the RAM disk image does not end with the capital letter *X*. Therefore, the TCP/IP BOOT-PROM loads the boot image into Base Memory (below 1 MB) and overwrites internal BIOS structures. Add *X* to the filename of the boot image (e.g. */tftpboot/ramd.X*), so that the boot image is loaded into Extended Memory.

E6E: Can Not Start From RAM Disk

Problem: *The PROM did not find the bootable ID in the bootsector of the RAM disk.*

Solution: Check if the diskette from which the RAM disk image was made is bootable. Verify that the RAM disk image was transferred binary, e.g. FTP will strip off the 8th bit if ASCII mode is enabled. If you use DOS versions earlier than 4.0 which did not have the bootable ID, make an 18 sector RAM disk image.

RAM Disk Is Destroyed After Loading HIMEM.SYS

Problem: *The HIMEM.SYS driver from DOS 6.2 by default performs a destructive extended memory test that destroys the TCP/IP BOOT-PROM RAM disk.*

Solution: Disable `HIMEM`'s destructive extended memory test function by using the `/testmem:off` option.

```
device=a:\dos\himem.sys /testmem:off
```

Programs In High Memory Are No Longer Accessible

Problem: *Using EMM386 with the RAM or NOEMS options causes that the configuration of the High Memory (memory between 640 KByte and 1 MByte) gets lost after the BPUTIL program has restored the memory allocated by the RAM disk. It shows that it is not possible to use MEM /C or LOADHIGH after the BPUTIL -R command has been executed. Programs*

which are already loaded into High Memory are not influenced by this problem.

Solution: In the *CONFIG.SYS* file, load the *BPUTIL.SYS* driver twice: with the `-f` option before *HIMEM.SYS* and with the `-x` option after *EMM386.EXE*.

```
device=bputil.sys -f
device=himem.sys /testmem:off
device=emm386.exe noems
device=bputil.sys -x
```

EMM386 Hangs With DMA Based NIC's

Problem: *On some systems EMM386 claims that the number of DMA handles are exceeded and halts the system. This only appears when the RAM disk image is loaded into Extended Memory.*

Solution: Increase the number of DMA handles via the `/D=nnn` option of `EMM386`. If this does not help, check with the manufacturer of your network controllers device-driver. This problem is not caused by the TCP/IP BOOT-PROM. It is caused by the memory transfer between Extended and Base Memory. Some DMA based network drivers can not handle this.

Diskless Windows 3.x Can Not Find EMM386

Problem: *When starting MS-Windows it can not find EMM386.*

Solution: Include the `/Y` option if `EMM386` is used on a diskless system. This helps programs that switch into Protected Mode to find *EMM386.EXE* since it has been started from the RAM disk drive which does no longer exist after `BPUTIL -R` is executed. A sample *CONFIG.SYS* file entry may look like the following:

```
device=emm386.exe /y=c:\dos\emm386.exe
```

IBM PC Hangs After Loading EMM386

Problem: *Some IBM PCs successfully download the boot image, but when the EMM386 driver is loaded, the system hangs.*

Solution: This is caused by a bug in the IBM PC's BIOS. A special TCP/IP BOOT-PROM version is available from bootix that contains a workaround for this problem. If you encounter that EMM386 hangs an IBM PC, first obtain the latest BIOS. If this doesn't help, please contact bootix technical support.

PC Hangs When Copying Files Into Boot Image

Problem: *The boot image file has been successfully downloaded and installed as RAM disk drive A:. When you copy additional files to this RAM disk drive, the PC hangs.*

Solution: When creating a boot image file, `BMIMAGE` restricts the size of the boot image file to the amount that is actually needed by the files contained in the boot image. Also, the TCP/IP BOOT-PROM only installs a RAM disk of the boot image's size. When copying additional files to the RAM disk, make sure that there is enough free space in it. To reserve free space in a boot image with `BMIMAGE`, see *"The -P Option"* on page 139.

Windows 3.0 Requests Location Of WINA20.386

Problem: *Running MS-DOS with Microsoft Windows 3.0 requests to specify the location of the WINA20.386 file.*

Solution: This can be done by adding the following lines to your *CONFIG.SYS* and *SYSTEM.INI* files:

CONFIG.SYS:

```
switches=/w
```

SYSTEM.INI:

```
[386Enh]
device=c:\dos\wina20.386
```

Diskless Windows Prevents Formatting Drive A:

Problem: *If the PC is booted using the TCP/IP BOOT-PROM, the MS-Windows diskette format menu does not work on the boot drive A:.*

Solution: MS-Windows does not allow to format the drive it was booted from. As the TCP/IP BOOT-PROM installs the boot image RAM disk as drive A:, Windows believes that it was booted from floppy disk. To change the boot drive with `BPUTIL`, see *"The -o Option"* on page 172. Note that you can always use the MS-DOS `FORMAT` program to format a diskette in drive A:.

Some Custom Tags Are Not Present

Problem: *Some BOOTP servers limit the amount of space that is provided for custom tags/options in the reply packet. If you find that custom information*

you have included in your bootptab file is not available to the TCP/IP BOOT-PROM, this is likely caused by limitations of the BOOTP server.

Solution: If the BOOTP server definitely limits the amount of custom tag/option information, there are several possible solutions to this problem:

- If possible, upgrade your BOOTP server to a version that supports extended custom tag/option information.
- Try to limit the amount of custom tag/option information that is used in your configuration.
- Use the BPBOOT bootstrap loader's capability to include custom tag/option information in options files. For more information, see *"BpOpT - Options Files"* on page 214.

Error Messages

When the TCP/IP BOOT-PROM encounters a problem, it will display one of the error messages that are described in this chapter. TCP/IP BOOT-PROM error messages consist of an error code (e. g. M30) and a verbose message that describes the problem. Some Ethernet controllers have only a small amount of ROM space available (e. g. controllers with an 8 KByte ROM socket). On these controllers, only the error code is displayed, and you must look up the corresponding verbose error message in the following table.

M10 ARP canceled by keystroke
The ARP request has been canceled because the user pressed a key.

M11 ARP time-out occurred
The ARP request has not been answered.

M20 Can't copy memory
The System BIOS (int 15h, service 87h) was not able to copy data.

M21 can't write to memory
There is not enough memory to store all MTFTP packets.

M22 can't write to memory
The RAM memory is to low and reached the ROM range.

M30 can't ARP tftp address
The TFTP server did not answer the ARP request.

M31 TFTP open canceled by keystroke
The TFTP open has been canceled because the user pressed a key.

M32 TFTP open time-out occurred
The TFTP server did not answer or the TFTP daemon is disabled.

M33 unknown TFTP opcode
The TFTP server sends unknown packets.

M34 TFTP read canceled by keystroke
The TFTP read has been canceled because the user pressed a key.

M35 TFTP read time-out occurred
The TFTP server did not answer even that the open request was granted.

M36 TFTP write canceled by keystroke
The TFTP write has been canceled because the user pressed a key.

M37 TFTP write time-out occurred
The TFTP server did not answer even that the open request was granted.

M38 can't open TFTP connection
The TFTP server did not answer or is disabled.

M39 can't read from TFTP connection
The TFTP server did not answer even that the open request was granted.

M40 BOOTP canceled by keystroke
M40 DHCP canceled by keystroke
The DHCP/BOOTP request has been canceled because the user pressed a key.

M41 BOOTP time-out occurred
M41 DHCP time-out occurred
The DHCP/BOOTP server did not answer, is disabled or has no entry for this client.

M42 no client or server IP
The BOOTP request was answered but did not contain a client or server IP.

M43 no bootfilename
The BOOTP request was answered but did not contain a filename because the file does not exist or is unreadable.

M50 FFRAME canceled by keystroke
The RPL FIND FRAME command has been canceled because the user pressed a key.

M51 FFRAME timeout occurred
The RPL FIND FRAME command has not been answered.

M52 SFREQUEST canceled by keystroke
The RPL SEND FILE REQUEST command has been canceled because the user pressed a key.

M53 SFREQUEST time-out occurred
The RPL SEND FILE REQUEST command was not answered even that the FILE FRAME command was granted.

M6f System is locked! Press <Ctrl>-<Alt>-<Del> to reboot.
The diskless boot failed and the system is now locked.

M70 canceled by keystroke
The checksum procedure has been canceled because the user pressed a key.

M80 RARP canceled by keystroke
The RARP request has been canceled because the user pressed a key.

M81 RARP time-out occurred
The RARP request has not been answered.

M82 no client IP
The RARP request has been answered but did not contain a client IP.

M90 can't init controller for multicast
Failed to re-initialize the network controller to receive multicast packets.

M91 MTFTP open canceled by keystroke
The MTFTP open has been canceled because the user pressed a key.

M92 MTFTP open time-out occurred
The MTFTP server did not answer or the TFTP daemon is disabled.

M93 unknown MTFTP opcode
The MTFTP server sends unknown packets.

M94 MTFTP read canceled by keystroke
The MTFTP read has been canceled because the user pressed a key.

M95 MTFTP read time-out occurred
The MTFTP server did not answer even that the open request was granted.

M96 can't ARP mtftp address
The MTFTP server did not answer the ARP request.

M98 can't open MTFTP connection
The MTFTP server did not answer or is disabled.

M99 can't read from MTFTP connection
The MTFTP server did not answer even that the open request was granted.

M9a too many MTFTP packets
The MTFTP server is sending more packets as can be stored.

E60 ROM disabled via setup
The network controller setup disables the ROM.

E61 Keystroke BOOT-PROM aborted
The TCP/IP BOOT-PROM has been aborted because the user pressed a key.

E62 can't init controller (???)
The network controller can not be initialized.

E63 can't init controller (i/o port)
The network controller can not be found.

E64 can't init controller (shm)
The network controller's shared memory can not be set.

E65 can't init shm (using 0xd0000)
The network controller's shared memory is forced to 0xd0000.

E66 can't init controller (shm)
The network controller's shared memory can not be set.

E67 can't init controller (???)
The network controller can not be initialized.

E68 can't init controller (MC-ID)
The network controller can not be found in any MCA slot.

E69 can't init controller (shm)
The network controller's shared memory can not be set.

E6a ROM not licensed to this controller
The network controller's MAC address is not the same as the one the ROM is licensed to.

E6b can not find RPL server
The RPL server did not answer.

E6c can not find RARP server
The RARP server did not answer.

E6d can not find BOOTP server
The BOOTP server did not answer.

E6e can not start from RAM disk image
The RAM disk image does not contain a valid bootsector.

E71 invalid checksum
The RAM disk image does not have a valid checksum.

E72 unknown boot protocol
No boot protocol (DHCP/BOOTP/RARP/RPL) has been configured.

T?? <TFTPD error string>
A TFTP error occurred, ?? holds the TFTP error code.

Index

Symbols

A

B

D

E

F

H

I

L

M

N

O

P

Q

R

S

T

U

X

The **BootManage® TCP/IP BOOT-PROM** allows you to boot your PC over the network and remotely configure anything imaginable.

Remotely upgrade the system BIOS, partition the hard drive, install or boot the operating system, restore a corrupted boot sector - all of these operations and more are possible.

Remote Boot Means Remote Installation

Using the **BootManage® TCP/IP BOOT-PROM**, you can remotely boot your PC and devise your own configuration and installation scripts to manage tens to thousands of PC's. What's more, the **BootManage® TCP/IP BOOT-PROM** allows you to do this much more economically than any other product in existence.

Operating Systems and Protocols

Support is available for a broad variety of operating systems. These include Windows 95/98/ME/NT/2000, Windows 3.x, DOS, and even Linux.

The **BootManage® TCP/IP BOOT-PROM** relies on the standard, time-tested Internet protocols DHCP, BOOTP, and TFTP. BOOTP and TFTP server programs are included for a variety of platforms. The bootix versions of these server programs are optimized for greater throughput and scaleability, but the user is welcome to use the server programs which may already be included with their system.

Maintain Boot Image Files

The **BootManage® TCP/IP BOOT-PROM** provides both interactive and batch tools to create, restore and maintain your remotely booted system images.

Install 1000 PC's With One Boot Image

Use BOOTP/DHCP dynamic configuration variables to remotely install thousands of uniquely identified PC's with just a single boot image.

bootix Technology GmbH
Geranienstrasse 19
D-41466 Neuss

Phone ++49 (0)2131 7486-0
Fax ++49 (0)2131 7486-26

Internet http://www.bootix.com

E-Mail info@bootix.com