

BootManage[®] PXE Toolkit

User Manual



BootManage[®] PXE Toolkit

User Manual

Copyright © 1999 bootix Technology GmbH, Geranienstrasse 19, D-41466 Neuss

Copyright:

All rights reserved. No part of this work covered by copyright may be reproduced in any form or by any means - graphic, electronic or mechanical, including photocopying, recording, taping, or storage in an information retrieval system - without prior written permission of the copyright owner.

Trademarks:

BootManage is a registered trademark of bootix Technology GmbH, D-41466 Neuss. All other company and product names are trademarks of the company or manufacturer, respectively.

Colophon:

This document was produced using FrameMaker 5.5 on Windows 95 and NT workstations. Page proofs were reviewed on the screen using GhostView 2.3 and printed to paper on a 600 dpi PostScript Level II printer, and final output was run directly to film. The PDF file for online reading was created using Adobe Acrobat 3.0.

Revision history:

February 1999, first version

April 1999, minor changes

Project team: Ralf Büttner, Larry Epstein, Dirk Köppen

Printed in Germany.

Table of Contents

<i>Overview.....</i>	<i>11</i>
What Is WfM?	11
What Is PXE?	11
How Does PXE Work?	12
What Is the BootManage® PXE Toolkit?	13
PXBOOT	13
PXSHELL	13
PXDISK	13
PXFDISK	14
PXUTIL	14
BOOTPD32	14
TFTPD32	14
 <i>Getting Started.....</i>	 <i>15</i>
Configuring DHCP and BOOTP Servers	15
The Network Bootstrap Program	16
Creating Boot Image Files	16
Utilities for DOS Boot Image Files	17
BOOTP and TFTP Servers	17
 <i>PXBOOT.....</i>	 <i>19</i>
Utilizing PXBOOT	19
Booting the Client From a Boot Image	20
Controlling PXBOOT	20
PxSrV - Setting the TFTP Server's IP Address	21

PxRoU - Specifying a Router	21
PxOpT - Downloading Additional Options	21
PxInS - Unattended Installations	23
PxDiS - Interactive Network Boot	25
PxDbG - Displaying Diagnostic Information	26

PXSHELL..... 27

Installing PXSHELL	27
Creating Boot Images	27
Restoring Boot Images	30
PXSHELL Menu Options	32

PXDISK 35

Installing PXDISK	35
Common PXDISK Operations	36
Creating a DOS Boot Image	36
The PXDISK Environment Variable	37
Inserting and Extracting Files	37
PXDISK Commandline Options	38
The -C Option	38
The -d Option	38
The -D Option	39
The -E Option	39
The -F Option	39
The -I Option	40
The -i Option	40
The -M Option	40
The -O Option	41
The -o Option	41
The -P Option	41
The -T Option	41
The -v option	41

PXFDISK..... 43

Introducing PXFDISK	43
---------------------------	----

Installing PXFDISK	44
Technical Information	44
PXFDISK Options and Parameters	46
PXFDISK Commands	47
Partition Table Commands	47
The -m Command	47
The -p Command	49
The -c Command	50
The -o Command	50
The -a Command	50
The -t Command	51
The -g Command	51
The -G Command	52
Writing a Master Boot Record	52
The -b Command	52
Formatting a DOS Partition	52
The -q Command	52
Accessing Disk Sectors	53
The -r Command	53
The -w Command	54
The -z Command	55
 PXUTIL.....	57
PXUTIL Command-line Options	57
Patching Reply Information Into Files	58
The -a Option	58
The -b Option	61
The -s Option	62
The -S Option	64
The -p Option	64
Miscellaneous PXUTIL Options	64
The -e and -E Options	64
The -o Option	64
The -y Option	64

<i>BOOTPD32</i>	65
BOOTP Versus DHCP Servers	65
Features of BOOTPD32	65
Installing BOOTPD32	66
Using BOOTPD32 with PXE Clients	66
BOOTPD32 Commandline Options	67
Multiple Network Interfaces	68
Subnetting	68
Using an Arguments File	68
New Features	68
Increasing the BOOTP Reply Size	68
Delaying BOOTP Replies	69
Changing the Boot Server's IP Address	69
 <i>TFTPD32</i>	 71
TFTPD32 Features	71
Installing TFTPD32	71
Installation As a Service	72
Using TFTPD32 with PXE Clients	72
TFTPD32 Commandline Options	73
The -c Option	73
The -d Option	73
The -h Option	74
The -i Option	74
The -k Option	74
The -l Option	74
The -m Option	74
The -p Option	75
The -r Option	75
The -s Option	75
The -u Option	76
The -v Option	76
The -w Option	76
The -x Option	76

<i>Working With the Microsoft DHCP Server.....</i>	77
DHCP Versus BOOTP Servers	77
Installing the Microsoft DHCP Server	77
Configuring the Microsoft DHCP Server	79
Starting DHCP Manager	79
Creating and Activating a DHCP Scope	79
The Client Class Identifier Option	79
The Bootfile Name Option	81
Implementing Custom Options	82
TFTP Server IP Address	84
Implementing Multiple Custom Options	85
 <i>Working With the Microsoft Network Client</i>	
<i>Administrator.....</i>	87
Starting the Network Client Administrator	87
Adding Windows NT Entries	89
Creating an Installation Diskette	89
Replace the Real Mode Network Card Driver	91
Test the Installation Floppy Disk	91
 <i>A Sample Windows NT Installation.....</i>	93
Environment	93
Installation Overview	94
What Microsoft Provides	94
What the BootManage® PXE Toolkit Provides	94
How the Installation Process is Sequenced	94
Step 1: Prepare the Installation Server	95
Step 2: Add Files to the Installation Server	95
Step 3: Create a Network Client Boot Diskette	96
Step 4: Add the BootManage® PXE Toolkit Files	96
Root directory	96
Directory BIN	97
Directory NET	97

Step 5: Modify the Configuration Files	97
A:\CONFIG.SYS	97
A:\AUTOEXEC.BAT	98
A:\INSTALL.BAT	98
A:\NET\PROTOCOL.INI	101
A:\NET\SYSTEM.INI	101
Step 6: The unattend.txt File	102
Step 7: Create the Boot Image File	103
Step 8: Install BOOTPD32 and TFTP32	104
Step 9: Start BOOTPD32 and TFTP32	105
Step 10: Install the PXE Client	106
Administrator Initiated Reinstallation	106
User Initiated Reinstallation	106
Suggestions	107
Installing Multiple Clients	107
Clients With Different Network Cards	107
Installing Windows NT 4.0 Server Clients	107
Installing Windows 95 and Windows 98 Clients	107
Using Different Transport Protocols	107
Using Different Installation Servers	108
Installing From an NFS Server	108

Overview

What Is WfM?

In 1996, Intel Corporation launched an initiative called *Wired For Management (WfM)*. Unlike previous attempts to create standards aimed at lowering PC total cost of ownership, much of WfM is built into the PC's hardware. As a result, WfM's base capabilities can be found in nearly every new PC manufactured today. These capabilities include the Preboot Execution Environment (PXE), Remote Wakeup, and the Advanced Configuration and Power Interface (ACPI).

The BootManage[®] PXE Toolkit was created to take advantage of WfM's PXE capability and empower its use.

PXE is a powerful feature; it allows administration applications to capture control of the PC before the PC boots from its local hard disk. By preempting the boot process, PC configuration operations, which heretofore required a person to be present at the PC, can now be performed remotely. Imagine being able to install operating systems on blank hard disks, upgrade system BIOS's, remove viruses from infected hard disks, or recover from catastrophic system failures without visiting the PC. All this is possible with PXE, and the BootManage[®] PXE Toolkit enables you to quickly build your own PXE applications without ever writing a single line of code!

Further details concerning WfM and PXE can be found on the Intel Corporation website, <http://www.intel.com>.

What Is PXE?

The Preboot Execution Environment (PXE) embodies three technologies that establish a common and consistent set of pre-boot services within the boot firmware of a PC:

- A standard method of initiating the pre-boot firmware to invoke PXE on the PC client machine.

- A uniform protocol that allows the PC to request the allocation of a network address and subsequently request the download of a Network Bootstrap Program (NBP) from a network boot server.
- A standard set of Application Programming Interfaces (API's) exposed by PXE which can be employed by the NBP or system BIOS.

The PXE program code resides in the PC's firmware: Either in the system BIOS or in a separate option ROM. For PC's with a network controller that is integrated on the motherboard, the PXE code is usually found in the system BIOS.

For PC's that use a separate network adapter, the PXE code generally resides in a non-volatile option ROM found on the network adapter. This option ROM, or "Boot Prom", can be a PROM, an EPROM, or a field-programmable "flash" EPROM.

In the vast majority of cases, network adapters include an empty socket designed to hold a Boot Prom but do not ship with the Boot Prom itself. To enable organizations to preserve their network adapter investment while enabling them to take advantage of PXE, bootitx provides Boot Proms for over 80 different Ethernet and Token Ring adapters.

How Does PXE Work?

At PC startup, the PXE code takes control of the PC's bootstrap procedure and tries to obtain TCP/IP configuration information for this PC from a DHCP or BOOTP server. This configuration information is comprised of a number of options that are defined in a configuration database somewhere on the DHCP or BOOTP server. Numerous options are available for the many settings with which a PC client might need to identify itself on a network. Examples include:

- the PC client's IP address
- the PC client's subnet mask
- the PC client's name
- the IP address(es) of one or more routers the client should use
- the name of a Network Bootstrap Program (NBP) the client should execute

Upon retrieval, the PXE code stores the PC's configuration information for later use by other utilities and then tries to download an NBP file (specified as an option) from a TFTP server (specified as another option).

After having downloaded the NBP file, the PXE code transfers program control to the NBP code. It is now up to the NBP to do something useful in order to boot the PC. To allow the NBP to communicate with the network adapter and to access the PC's configuration information, the PXE code implements a preboot application programming interface (Preboot API).



The Network Bootstrap Program (NBP) itself is **not** part of PXE. However, the BootManage® PXE Toolkit contains a special-purpose NBP called PXBOOT.

What Is the BootManage® PXE Toolkit?

The BootManage® PXE Toolkit is designed to work in conjunction with PXE clients. It contains a versatile Network Bootstrap Program as well as various programs which assist you in remotely booting and remotely configuring PC's.

The BootManage® PXE Toolkit consists of the following programs:

PXBOOT

PXBOOT is a Network Bootstrap Program (NBP) that is designed to be downloaded and executed by a PXE client. PXBOOT operates as a versatile and remotely configurable boot loader which allows one to

- load a boot image file on a PXE client, install the file's contents as bootable RAM disk and boot the PXE client from this RAM disk
- retrieve user variables from a server for use during the PXE client boot
- configure the hard disk or control the network boot process of a PXE client

PXSHELL

an interactive, menu-driven utility for

- creating a boot image from a floppy disk
- restoring a boot image to a floppy disk

PXDISK

PXDISK is a utility for maintaining boot image files. Using PXDISK, one can

- create and restore boot images from directory structures on a hard disk, rather than floppy disk
- directly access and modify the files within a DOS boot image
- perform various other operations on DOS boot images

PXFDISK

a command line utility for

- creating and deleting partitions on PXE client's hard disk
- changing and checking partition table entries
- quick formatting DOS FAT partitions
- create and restore hard disk images from PXE client's hard disk to server

PXUTIL

a DOS device driver and command line utility for

- retrieving and viewing DHCP/BOOTP options
- patching various DHCP/BOOTP options into configuration files
- rebooting the PXE client

BOOTPD32

a 32-Bit BOOTP server for Windows 95, 98, and NT which

- is capable of being run as a standard application or Windows NT service
- supports custom tags and large BOOTP replies
- permits operation on multiple network interfaces
- operates on a single text based configuration file (bootptab)

TFTPD32

a 32-Bit TFTP server for Windows 95, 98, and NT which

- is capable of being run as standard application or Windows NT service
- supports large block TFTP and multicast TFTP for faster performance
- is highly optimized for downloading bootable RAM disk files

Getting Started

Setting up a site with PXE clients for remote booting or remote system deployment services involves configuring multiple components. At least one DHCP or BOOTP server must be installed, as well as one or more TFTP servers; Boot image files must be created; And, batch files containing the scripts for the tasks to be performed during remote booting, such as unattended installation of an operating system, need to be prepared.

It is important to understand that the PXE code in a PXE client PC handles the retrieval of PC configuration parameters and also handles download and execution of a Network Bootstrap Program (NBP). The PXE code also provides an application programming interface (Preboot API) to the NBP, so that the NBP can access the configuration parameters retrieved earlier by the PXE code. Also, the NBP can use the Preboot API to access the network adapter driver and, for example, download a file from a TFTP server.

As the name *Preboot Execution Environment* says, PXE specifies only the environment for the NBP, and not the NBP code itself. The PXE code relies on the NBP to actually do something useful in order to remote boot or remotely manage the PC.

Configuring DHCP and BOOTP Servers

When a PXE client starts up, it sends an extended DHCP broadcast message on the local area network in order to obtain its configuration parameters and the name and location of a Network Bootstrap Program. Although obtaining client configuration parameters can involve multiple packet exchanges between the PXE client and multiple configuration servers*, a single DHCP or BOOTP server will do the job.

To become familiar with PXE and the BootManage® PXE Toolkit, the easiest way is to start by setting up both a DHCP (or BOOTP) and a TFTP server on the same

* Discussing the details of the initial PXE protocol is beyond the scope of this manual. For detailed information, please refer to the PXE specification document.

machine. You can always add or change DHCP/BOOTP/TFTP servers at any time without breaking the general concept.

A PXE client only recognizes DHCP or BOOTP packets which contain the *Client Class Identifier* (Option 60) set to the value *PXEClient*. Thus, DHCP servers can also serve up IP addresses to non-PXE clients without disturbing the operation of PXE clients.

The Network Bootstrap Program

The BootManage[®] PXE Toolkit contains the program PXBOOT which acts as a versatile Network Bootstrap Program (NBP, also called *boot loader*) for PXE clients. Configure your DHCP or BOOTP server to use PXBOOT as the boot filename and copy the file PXBOOT to your TFTP server. At startup, PXE clients will download and execute PXBOOT.

When PXBOOT gets control of the PXE client, it uses custom options contained within the DHCP/BOOTP reply information to determine what it should do. For example, PXBOOT can analyze and modify the local hard disk's partition table as part of a completely automated and unattended operating system deployment mechanism.

By simply changing these custom options for a specific PXE client on the DHCP or BOOTP server, you can control whether the PXE client boots “normally”, uses a diskless emergency boot image, or has its system completely re-installed.

Also, PXBOOT provides the capability to let a user choose between local and network boot (if the administrator enables this feature for that PXE client).

By being able to download additional options from text based files, PXBOOT overcomes the limitations of some DHCP servers (small DHCP reply size, option type restrictions, etc.). Both global and per-client option files are provided.

Creating Boot Image Files

The PXBOOT boot loader can download a boot image file from a TFTP server, install the contents of this file as a bootable RAM disk in the PXE client's memory, and boot the client from this RAM disk. Basically, a boot image file is a mirror image of a bootable floppy diskette.

The BootManage[®] PXE Toolkit provides the utilities PXSHELL and PXDISK to create and maintain boot image files. PXSHELL is an interactive menu-driven utility, while PXDISK is commandline driven and, therefore, highly useful within batch files.



Do not specify the name of a boot image as the boot file name option in the DHCP or BOOTP server configuration. Doing so will cause the PXE client to hang.

This is because a PXE client cannot directly download a boot image but, instead, must use the PXBOOT boot loader to load the boot image.

Users of the TCP/IP BOOT-PROM* will note that this is different from the procedure with which they are accustomed.

Utilities for DOS Boot Image Files

The BootManage[®] PXE Toolkit provides two utility programs designed to be used within DOS boot image files.

The first utility, PXFDISK, is mainly a commandline-driven hard disk manipulation tool which provides partitioning and formatting, extensive partition table manipulation and creation of a master boot record. This is useful for preparing a PXE client's hard disk for the unattended installation of an operating system. Moreover, PXFDISK can directly access physical disk sectors. With this feature, one can backup the entire contents of a hard disk to a file on the network server and also restore the PXE client's hard disk from this file.

The second utility, PXUTIL, is a multi-purpose utility which is also driven by commandline options. PXUTIL can access the DHCP/BOOTP reply information (obtained earlier by the PXE code) and patch this information into text and binary files. This is very useful when a single boot image file is shared by multiple clients. Although this is its most powerful option, PXUTIL also provides some other options like rebooting the PC.

BOOTP and TFTP Servers

As mentioned earlier, a PXE client generally requires two network services processes in order to operate:

- a DHCP or BOOTP server which supports custom vendor options
- a TFTP server

* As opposed to a PXE client, the TCP/IP BOOT-PROM is capable of directly downloading and booting from a boot image. To get more information about the TCP/IP BOOT-PROM and related products, point your web browser to <http://www.bootmanage.com> or dial ++49 2131 74860.

As long as the servers are standards-conformant, they can run on virtually any operating system. For sites which are already running DHCP/BOOTP/TFTP servers, these will probably be sufficient for serving their PXE clients.

The BootManage[®] PXE Toolkit contains both BOOTP and TFTP server programs for the Microsoft Windows 95, Windows 98, and Windows NT (Workstation or Server) operating systems. Both servers can be run as standard applications on all mentioned operating systems. On Windows NT, the BOOTP and TFTP servers can also be installed as a Windows NT service. Also note that the TFTP server provides extended functionality such as large frames and multicast operation to enhance performance and scalability.

PXBOOT

PXBOOT is a Network Bootstrap Program (NBP) designed to be downloaded and executed by a PXE client at boot time. Controlled by “magic” keywords transmitted in the BOOTP/DHCP reply data, PXBOOT performs many functions. Using PXBOOT, one can:

- Download a remote boot image and boot the PXE client from it.
- Retrieve TFTP server and router IP addresses from custom BOOTP/DHCP options.
- Retrieve global and individual custom options from text files.
- Automatically launch unattended operating system installations or reinstallations.

Utilizing PXBOOT

The PXBOOT Network Bootstrap Program can be found on the BootManage[®] PXE Toolkit Utility Diskette. Since it is downloaded to the PXE client during boot time by TFTP, it should be copied to the server’s directory where all files for TFTP transfer are stored. Usually, this directory is named */tftpboot* or *c:\tftpboot*.

In the BOOTP or DHCP server’s configuration, set PXBOOT to be the boot file name of the PXE client so that the PXE client will download and execute PXBOOT at startup. If you are using the BOOTPD32 server, the appropriate entry in the *bootptab* configuration file might look like this:

```
# PXE test client
pxeclient:\
    :hn:vm=rfc1048:ht=ethernet:\
    :ha=0000c0094ff9:ip=172.16.80.1:\
    :hd="c:/tftpboot":bf=pxboot:\
    :T60="PXEClient":
```

If you are using Microsoft’s Windows NT DHCP Server, see “*The Bootfile Name Option*” on page 81.

Booting the Client From a Boot Image

Since PXBOOT's job is to download a boot image and then boot from it, a boot image is obviously needed. Such a boot image can be created using the PXSHELL or PXDISK programs. Please see "The PXSHELL Program" on page 27 or "The PXDISK Program" on page 21 for complete details on how to create and manage boot images.

After the boot image is created, it should be renamed to *pxboot.X*. Please make sure there is a capital 'X' at the end of the filename. Now your TFTP server directory should look like this:

pxboot (the boot loader, about 10 KBytes)
pxboot.X (the boot image, about 1.4 MBytes)



To determine the boot image file name, pxboot uses its own filename as the root and appends ".X" to it. If you want to use different boot image filenames for different clients, simply make a copy of the file *pxboot* and rename it as you wish. For example, when renamed *nt4inst*, pxboot will look for the boot image *nt4inst.X*. When renamed *w95inst*, pxboot will look for the boot image *w95inst.X*.

Controlling PXBOOT

PXBOOT is controlled by "magic" keywords that appear as custom options in the BOOTP/DHCP reply information. The following keywords are available:

Keyword	Explanation
PxSrV	TFTP server IP address
PxRoU	Router/Gateway IP address
PxOpT	download global and/or individual user variables from ASCII files
PxInS	enable procedures for unattended installation
PxDiS	allow user to request network boot
PxDbG	display diagnostic information

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="PxSrV=172.16.0.1":\  
:T129="PxRoU=172.16.0.254":
```

The same can be achieved by concatenation:

```
:T128="PxSrV=172.16.0.1;PxRoU=172.16.0.254":
```

With the Microsoft DHCP server, you can only use the concatenation method as described in *“Implementing Multiple Custom Options”* on page 85.

PxSrV - Setting the TFTP Server's IP Address

In order to download a boot image, the PXBOOT 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 PXE client, the PXBOOT bootstrap loader comes-up with the error message *PXBOOT-E10: TFTP server IP address not set*, you must make an additional entry in your BOOTP/DHCP configuration file similar to the following:

```
# PXE test client  
pxeclient:\  
:hn:vm=rfc1048:ht=ethernet:\  
:ha=0000c0094ff9:ip=172.16.80.1:\  
:hd="c:/tftpboot":bf=pxboot:\  
:T60="PXEClient":\  
:T128="PxSrV=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 tag number (T128 in this example) can be any number greater than or equal to 128. PXBOOT will automatically search all user options for the `PxSrV` keyword.

PxRoU - Specifying a Router

If the TFTP server can only be reached via a router or gateway, then you can use the keyword `PxRoU` to specify the router's IP address.

It is very likely that you will use `PxRoU` together with `PxSrV`. You can concatenate both options using a semicolon ';' as in the following example:

```
:T140="PxSrV=172.16.0.1;PxRoU=172.16.0.254":
```

PxOpt - Downloading Additional Options

Some BOOTP/DHCP servers are limited in the amount of vendor option information they can send. To overcome these limitations, the PXBOOT bootstrap loader is

designed to transfer an ASCII *options file* from the TFTP server to the PXE client via TFTP and include its option values in the PXE client's BOOTP/DHCP buffer space. Then, using the PXUTIL program, the PXE client can retrieve these values for various uses as the boot process proceeds. The format of an options file is as follows:

```
# 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
```

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. Lines starting with a hash '#' character are treated as comments.

Please note that the option numbers are arbitrary. That is, they can be assigned and used by PXE application developers however they see fit as long as their value is greater than or equal to 128.

The PxOpT keyword instructs the PXBOOT boot loader to download an options file. As a parameter, PxOpT accepts a two digit hexadecimal number which specifies the action to be taken:

Keyword	Explanation
PxOpT=01	download only global options
PxOpT=02	download only individual options
PxOpT=03	download both global and individual options

Global Options (PxOpT=01)

If PxOpT is set to 01, then the PXBOOT bootstrap loader will download a file named *pxboot.opt* from the same directory as where the *pxboot* and *pxboot.X* files are stored. As such, these options can be used by all clients, hence, as *global options*. Note that if you have renamed *pxboot* to e.g. *nt4inst*, it will look for the file *nt4inst.opt* instead!

After downloading the *pxboot.opt* file, PXBOOT incorporates these additional custom options with the in-memory BOOTP/DHCP reply information.



You can build client groups by renaming the PXBOOT bootstrap loader. For example, renaming *pxboot* to *sales* causes the bootstrap loader to use the boot image *sales.X* and the global option file *sales.opt*. In this case, use *sales* as the bootfile name in the BOOTP/DHCP configuration file.

Individual Options (PxOpT=02)

If PxOpT is set to 02, then the PXBOOT bootstrap loader will download an option file named *<MACAddr>.opt*, where *<MACAddr>* represents the last 8 digits of the PXE client's MAC (hardware) address. The format of this individual option file is the same as the global option file (*pxboot.opt*). Hence, a PXE client with the MAC address 00.00.c0.12.54.ef would download an option file named *00c01254ef.opt*. This can be used to provide PXE clients with individual custom options.

Global And Individual Options (PxOpT=03)

If PxOpT is set to 03, then the PXBOOT bootstrap loader will **first** download the global option file *pxboot.opt* and **then** the individual option file *<MACAddr>.opt*. If a certain option is present in both *pxboot.opt* and *<MACAddr>.opt*, then the value in *<MACAddr>.opt* takes precedence. All other custom options will be merged.

PxInS - Unattended Installations

If the PxInS keyword is not used, then the PXBOOT bootstrap loader will always start the PXE client from the network, i.e. PXBOOT will download the boot image *pxboot.X* and start the PXE client from it.

If the PxInS keyword is used, then PXBOOT will check the ID value of a certain partition of the local hard disk. Depending on this ID value, PXBOOT will either boot the PXE client from the network or from the local hard disk.



Using this technique, state information can be saved on the client from one boot to the next boot. This is invaluable for constructing unattended operating system installation applications that require multiple reboots as the system installation progresses through its various stages.

To allow remote-initiated reinstallation of the PXE client, the PxInS keyword takes three arguments:

```

    w is either 0 (use yy value) or 1 (use zz value)
    |
    | x is the partition number (0, 1, 2 or 3)
    ||
PxInS=wx,yy,zz
    || ||
    || zz is the value to check against partition ID if w = 1
    ||
    yy is value to check against partition ID if w = 0
```

If the first argument to PxInS is set to **0X**, where X is the partition number (0, 1, 2 or 3) to be checked, PXBOOT checks the ID value of the corresponding hard disk partition against the **second** argument of PxInS (YY).

If the first argument to PxInS is set to **1X**, where X is the partition number (0, 1, 2 or 3) to be checked, PXBOOT checks the ID value of the corresponding hard disk partition against the **third** argument to PxInS (ZZ).

If the value of the second/third argument matches the actual partition ID value, then PXBOOT boots the PXE client from the hard disk. **In all other cases**, PXBOOT boots the PXE client from the network boot image.

As this may not be easy to understand, consider some examples:

Example 1

```
PxInS=00,63,a3
```

In this example, w is 0, x is 0, yy is 63, and zz is a3. PXBOOT checks if partition 0 (x=0) has the ID value 63 (w=0, yy=63). If this is true, then PXBOOT boots from the hard disk. Otherwise, it boots from the network.

Example 2

```
PxInS=10,63,a3
```

In this example, w is 1, x is 0, yy is 63, and zz is a3. PXBOOT checks if partition 0 (x=0) has the ID value a3 (w=1, zz=a3). If this is true, then PXBOOT boots from the hard disk. Otherwise, it boots from the network.

Example 3

```
PxInS=03,e3,f3
```

In this example, w is 0, x is 3, yy is e3, and zz is f3. PXBOOT checks if partition 3 (x=3) has the ID value e3 (w=0, yy=e3). If this is true, then PXBOOT boots from the hard disk. Otherwise, it boots from the network.

Example 4

PxInS=13,e3,f3

In this example, w is 1, x is 3, yy is e3, and zz is f3. PXBOOT checks if partition 3 (x=3) has the ID value f3 (w=1, zz=f3). If this is true, then PXBOOT boots from the hard disk. Otherwise, it boots from the network.

You may have noticed that examples 1 and 2 work together, as do examples 3 and 4. By toggling the 'w' value between 0 and 1, you can remotely initiate the reinstallation of a PXE client.

In unattended installation environments, the PxInS keyword is used to load an installation boot image from the network, install an operating system on the PXE client, and then boot the newly-installed operating system on its local hard disk.

From within a DOS boot image, you can determine whether w is set to 0 or 1 by examining option 253:

Option 253 value	Explanation
PxInS0	First argument of PxInS is 0x
PxInS1	First argument of PxInS is 1x
PxInS2	First argument of PxInS is invalid

If the PxInS keyword is not present, PXBOOT will not modify option 253.

PxDiS - Interactive Network Boot

If the PxInS keyword is not used, then the PXBOOT bootstrap loader, by default, downloads the boot image and boots the PXE client from it. To allow end-users to force a network boot, the keyword PxDiS is provided. If PXBOOT finds the keyword PxDiS 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, PXBOOT will boot from the network. Additionally, PXBOOT will set option 254 to the value PxKeY.

If the user does not press the key, then PXBOOT will continue as if the PxDiS keyword was not present. Option 254 will not be modified.

The first digit of the two digit argument following `PxDiS` defines the message to be displayed and the key to be pressed. Supported values are:

first digit	message displayed	key(s) to be pressed
0x	Press <SPACE> to start installation services	SPACE
1x	Press <F10> to start installation services	F10
2x	Press <Alt><F10> to start installation services	Alt F10
4x	Please wait	SPACE
5x	Please wait	F10
6x	Please wait	Alt F10

The second digit following the `PxDiS` configuration option specifies the time duration in seconds for which the `PXBOOT` bootstrap loader displays the message. For example, `PxDiS=03` will display the message *Press <SPACE> to start installation services* for 3 seconds. If the user does not press the space bar within this time, then the `PXBOOT` bootstrap loader will continue with the standard boot process defined by the `PxInS` keyword.

Using an argument of less than 1 for the time duration disables the `PxDiS` option.

From within the boot image, you can determine whether or not the user has pressed a key by examining the value of option 254 as in the following example:

```
rem check if user pressed key
if _#@T254*##### == _PxKeY goto KEY_PRESSED
```

PxDbG - Displaying Diagnostic Information

The `PxDbG` keyword instructs the `PXBOOT` boot loader to display diagnostic information. As a parameter, `PxDbG` accepts a two digit hexadecimal number which specifies the amount of diagnostic information to be displayed:

Keyword	Explanation
<code>PxDbG=00</code>	does not display diagnostic information
<code>PxDbG=01</code>	displays all <code>Px???</code> keywords and their parameters
<code>PxDbG=02</code>	displays additional activities (e.g. TFTP download of *.opt files)
<code>PxDbG=03</code>	displays contents of downloaded *.opt files

PXShell

PXShell is a menu-based DOS program used to create boot image files from physical diskettes and restore boot image files to physical diskettes. Various diskette types and sizes up to 2.88 MBytes are supported.

PXShell operates under the following operating systems:

- native DOS
- Windows 3.x, Windows for Workgroups 3.x (DOS Window)
- Windows 95 and 98 (DOS Window)
- Windows NT Workstation and Server (DOS Window)

To perform advanced operations on DOS boot image files or to automate DOS boot image manipulation through batch files, use the PXDISK program.

Installing PXShell

To install PXShell, simply copy the files *PXShell.EXE* and *PXShell.HLP* from the BootManage® PXE Toolkit Utility Diskette to your system's hard disk:

```
C:\> mkdir c:\pxetools
C:\> cd pxetools
C:\PXETOOLS> copy a:\pxshell.*
```

Creating Boot Images

The PXShell program is used to create 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.



Do not confuse the terms *boot image* and *boot loader*! A PXE ROM is not capable of directly handling boot images created with PXSHELL or PXDISK. Instead, a PXE ROM downloads and executes PXBOOT which is a boot loader, also called Network Bootstrap Program (NBP).

PXBOOT, in turn, downloads a boot image, installs it as a bootable RAMdisk in the client PC, and boots the client PC from it.

When configuring the *boot file option* for a PXE client on your BOOTP or DHCP server, do not enter the name of your boot image. Instead, specify the boot loader's name (PXBOOT).

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 PXSHELL menu, select BootImage → Create from diskette. The window shown in *Figure 4-1* can also be entered directly by invoking PXSHELL -C from the system prompt.

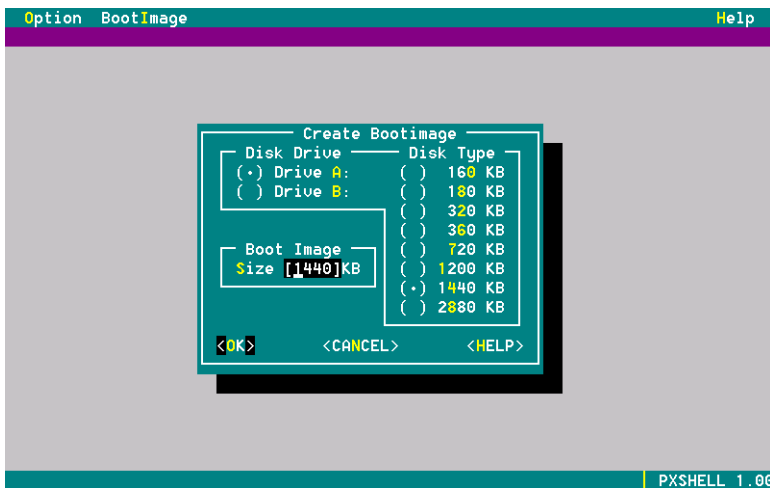


Figure 4-1. Create boot image with PXSHELL

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. PXSHELL 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, PXSHELL 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, PXSHELL 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 4-2*.

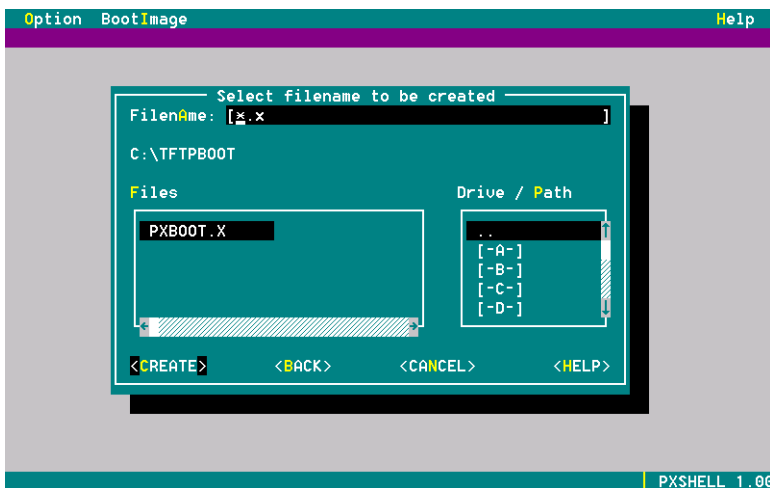


Figure 4-2. Select boot image filename

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 sub-window.

Now press <CREATE> to start the boot image creation procedure. You will see a progress bar like in *Figure 4-3* which indicates the progress of the boot image creation process.

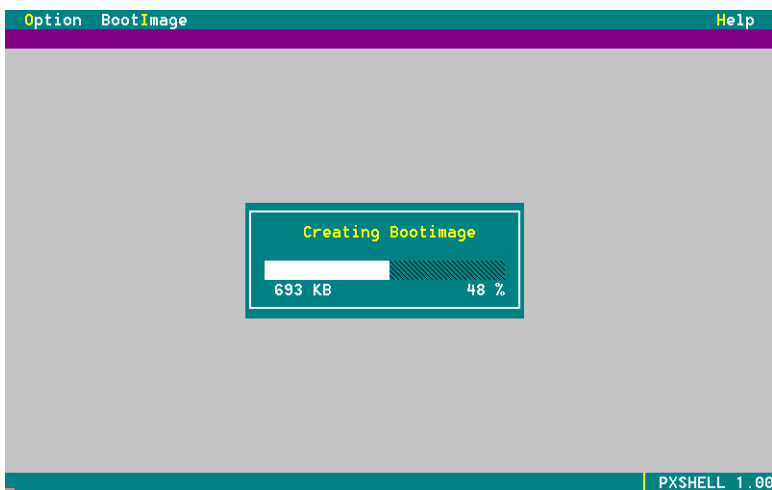


Figure 4-3. Boot image creation progress bar

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.

Restoring Boot Images

Once a boot image has been created, it must be stored on the TFTP server so that the PXE 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 `PXShell` to re-create the boot image on a diskette. Alternatively, `PXDisk` can be used to directly manipulate the individual files contained in the boot image. See chapter “*PXDisk*” on page 35 for more information.



Restoring a boot image to a diskette will overwrite all of the original contents of the diskette.

To restore a boot image, invoke the PXSHELL program and select the BootImage → Restore to diskette window. The window shown in *Figure 4-4* can be entered directly by typing `PXSHELL -R` from the system prompt.

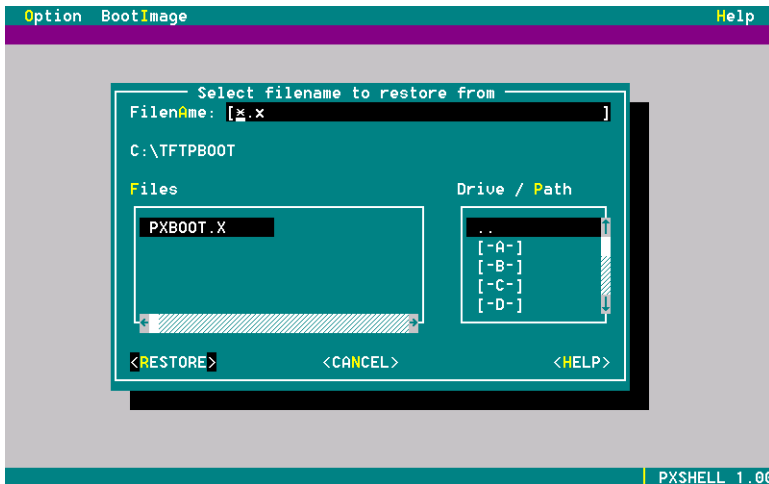


Figure 4-4. Restoring a boot image to diskette

Select the boot image you want to restore and select `<RESTORE>`. The Restore Bootimage dialog will now appear as seen in *Figure 4-5*.

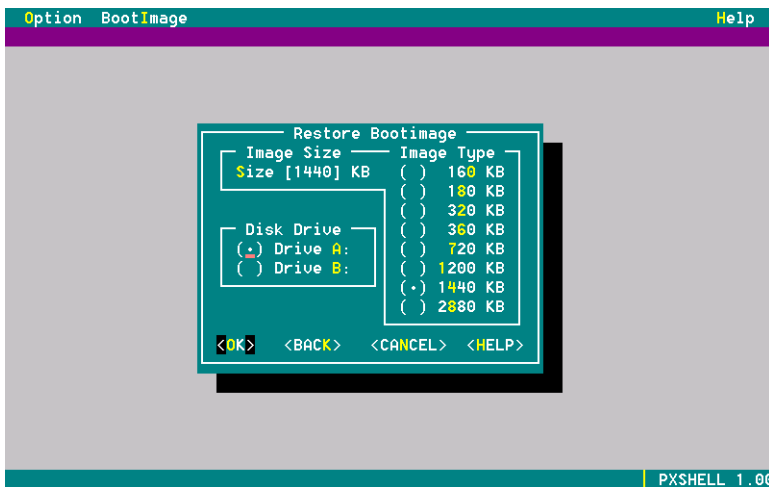


Figure 4-5. Restore boot image parameters

The Restore Bootimage dialog allows you to override the default parameters PXSHELL 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. PXSHELL 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 like in *Figure 4-6* which indicates the progress of the boot image restoration process.

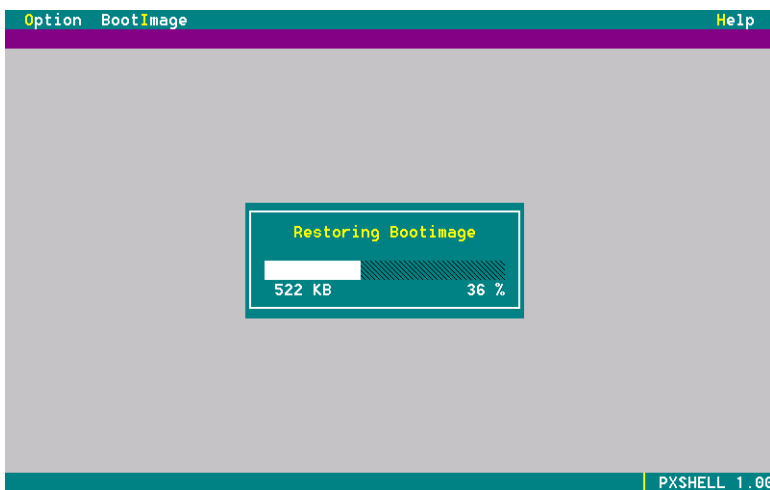


Figure 4-6. Boot image restoration progress bar

PXSHELL Menu Options

Various options can be set for the PXSHELL program. Upon selecting the Option menu, you will see a screen as shown in *Figure 4-7*.

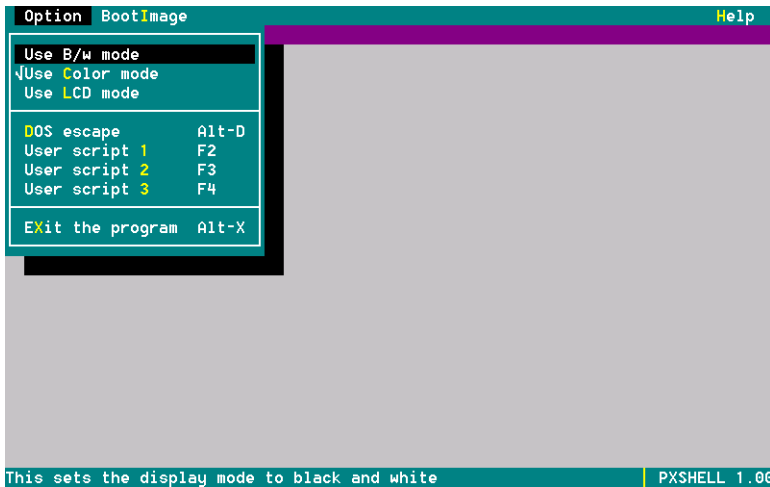


Figure 4-7. PXShell Option menu

In detail these options are:

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 *PXShell.CNF* and read each time the PXShell program is started. If *PXShell.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 PXShell and invokes a DOS command interpreter. Enter exit at the DOS prompt to return to PXShell.

User script 1, 2 and 3

User scripts are DOS batch files which can be called via a function key. PXShell searches for these batch files in the same directory where the PXShell 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 boot images to and from the server without leaving PXShell.

Exit the program

Exit PXShell and return to MS-DOS.

PXDISK

PXDISK is a DOS commandline program designed to create, restore, and modify DOS boot image files without requiring an actual diskette. PXDISK accomplishes this by directly reading and modifying DOS files within the boot image.

Using PXDISK, 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.

PXDISK operates under the following operating systems:

- native DOS
- Windows 3.x, Windows for Workgroups 3.x (DOS Window)
- Windows 95 and 98 (DOS Window)
- Windows NT Workstation and Server (DOS Window)

To interactively create and restore boot image files from boot diskettes, or to handle non-DOS boot images, use the PXSHELL program.

Installing PXDISK

To install PXDISK, simply copy the file *PXDISK.EXE* from the BootManage[®] PXE Toolkit Utility Diskette to your system's hard disk:

```
C:\> mkdir c:\pxetools
C:\> cd pxetools
C:\PXETOOLS> copy a:\pxdisk.exe
```

Common PXDISK Operations

The most common operations of PXDISK 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 PXDISK with the `-d` and `-F` options:

```
C:\PXETOOLS> pxdisk -d simple.X -F 1440,A:
```

This instructs PXDISK 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 PXDISK to display the files and directories contained in *simple.X*:

```
C:\PXETOOLS> pxdisk -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 by the PXBOOT boot loader in order to remotely boot your PXE client.



Do not confuse the terms *boot image* and *boot loader*! A PXE ROM is not capable of directly handling boot images created with PXSHELL or PXDISK. Instead, a PXE ROM downloads and executes PXBOOT which is a boot loader, also called Network Bootstrap Program (NBP).

PXBOOT, in turn, downloads a boot image, installs it as a bootable RAMdisk in the client PC, and boots the client PC from it.

When configuring the *boot file option* for a PXE client on your BOOTP or DHCP server, do not enter the name of your boot image. Instead, specify the boot loader's name (PXBOOT).

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:\PXETOOLS> pxdisk -d n:\tftpboot\pxboot.X -F 2880,simple.X
```

This creates a new DOS boot image named *pxboot.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 *pxboot.X* can hold up to 2.88 MBytes, whereas *simple.X* can only hold 1.44 MBytes.

The PXDISK Environment Variable

The PXDISK program uses the `-d` option to determine the name and location of the DOS boot image file. As an alternative, the environment variable PXDISK can be used for this purpose.

Whenever the PXDISK 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 PXDISK.

When the `-d` option is present, the PXDISK environment variable is ignored. The following example demonstrates this behaviour.

```
C:\PXETOOLS> set PXDISK=n:\tftpboot\pxboot.X
C:\PXETOOLS> pxdisk -D                      (displays the contents of pxboot.X)
C:\PXETOOLS> pxdisk -d simple.X -D          (displays the contents of simple.X)
```

Inserting and Extracting Files

To make our *pxboot.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 PXDISK 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:\PXETOOLS> pxdisk -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:\PXETOOLS> pxdisk -I \network\start.bat,m:\myfiles\start.bat
C:\PXETOOLS> pxdisk -I \autoexec.bat,c:\image\autoexec.bat
```

If the subdirectory *network* does not exist in the boot image, PXDISK 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:\PXETOOLS> pxdisk -O \autoexec.bat,c:\image\autoexec.bat
C:\PXETOOLS> edit c:\image\autoexec.bat
C:\PXETOOLS> pxdisk -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:\PXETOOLS> pxdisk -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:\PXETOOLS> pxdisk -o c:\image2
```

PXDISK Commandline Options

Here is an alphabetical reference to all PXDISK commandline options. **Please note that all of these options are case sensitive.**

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:

```
pxdisk -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:\ftpboot\ramd.X*.

If the *-d* option is not given, PXDISK takes the image that is defined by the environment variable *PXDISK*. If neither the *-d* option is given nor the *PXDISK* environment variable is defined, PXDISK 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:

```
pxdisk -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*:

```
pxdisk -d ramd.X -E autoexec.bat
```

This deletes the file *hosts* from the etc directory in the DOS boot image:

```
pxdisk -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 `PXDISK`. 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, `PXDISK` 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:

```
pxdisk -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:

```
pxdisk -d ramd.X -F 1440,f:\tftpbboot\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:

```
pxdisk -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:

```
pxdisk -d ramd.X -I \config.sys,config.txt
pxdisk -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*:

```
pxdisk -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:

```
pxdisk -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*:

```
pxdisk -d f:\tftpboot\ramd.X -i .
```

This copies all of the files from diskette drive **A:** into the DOS boot image *ramd.X*:

```
pxdisk -d ramd.X -i a:\
```

The -M Option

The *-M* option creates a new subdirectory in the DOS boot image.

These examples create a new subdirectory *dos* in the DOS boot image:

```
pxdisk -d ramd.X -M dos
pxdisk -d ramd.X -M \dos
pxdisk -d ramd.X -M /dos
```


This creates a new subdirectory *subdir* in the existing directory *dos*. All parent directories must already exist:

```
pxdisk -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:

```
pxdisk -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:

```
pxdisk -d ramd.X -T autoexec.bat
```

The following example displays the contents of the file *hosts* in the subdirectory *etc* on the screen:

```
pxdisk -d ramd.X -T \etc\hosts
```

The -v option

The *-v* option gives more technical feedback about PXDISK 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:

```
pxdisk -d ramd.X -F 1440,A: -v
```

PXFDISK

PXFDISK is a DOS program which can be used to create, remove, and modify partitions of any type on a PC's local hard disks. PXFDISK can also query for information about partitions and set the DOS ERRORLEVEL to allow conditional processing in batch files. PXFDISK provides options to write a master boot sector to the hard disk and quick format a FAT partition. In addition, PXFDISK can read and write directly to physical sectors on the hard disk. Since PXFDISK is entirely controlled by commandline options, it can be used in DOS batch files to automate partitioning, formatting, and low level disk operations.

Introducing PXFDISK

PXFDISK is a DOS program that uses BIOS system calls to directly access data blocks on a PC's local hard disk or floppy disk. Windows 95, 98 and NT do not allow this, so you must use PXFDISK under DOS.

PXFDISK provides the following functions:

- create, remove, and modify partition entries in a hard disk's partition table
- query information about partitions and return the result in the DOS ERRORLEVEL variable.
- write a master boot sector to a hard disk
- quick format a DOS 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

Installing PXFDISK

Copy the file *PXFDISK.EXE* from the BootManage[®] PXE Toolkit distribution disk to the place where you need it: A directory on your hard disk, to a floppy disk, or into a boot image.

Technical Information

As PXFDISK accesses the hard disk using low level system BIOS calls, it sees the hard disk in terms of drive, partition, and block numbers.

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.

DOS can create multiple logical drives on a System BIOS drive. Therefore, drive 80 may contain the DOS 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.

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

The PC's system BIOS sees a hard disk 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, up to 1024 tracks and up to 63 sectors per track. 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 specifying head/track/sector values on the PXFDISK commandline, you must always use these mapped values. You can easily query a hard disk's mapped geometry by using the `pxfdisk -g` command.

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 sectors that can be accessed by specifying a single block number. The hard disk controller internally maps this block number to head/track/sector information.

When directly reading from and writing to hard disk sectors, PXFDISK expects block numbers as arguments on the commandline.



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

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. An (incomplete) list of standard partition IDs is shown in the following table:

ID	Partition type
00	none (partition not used)
05	extended DOS
06	primary DOS FAT16 (16-Bit FAT, >32MByte)
07	used for both NTFS (Windows NT) and HPFS (OS/2)
0B	primary Win95 OSR2 FAT32 (32-Bit FAT)
63	UNIX (GNU HURD)
64, 65	Novell NetWare
82	Linux swap
83	Linux native

PXFDISK Options and Parameters

The following options and parameters can be used with PXFDISK:

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 to be <b#>
-e <b#>	specifies the ending block number to be <b#>
-n <blks>	specifies the number of blocks to be <blks>
-l <kbs>	limit throughput to <kbs> kilobytes per second

Option / Parameter	Explanation
-f	force immediate execution, don't delay to allow user abort
-v	display more information upon command execution

PXFDISK Commands

The following commands are available with PXFDISK:

Command	Explanation
-?	display PXFDISK 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)
-g	display disk geometry
-G	same as -g but does not display error message if fails
-m	make a partition
-o	set partition ID
-p	display partition table
-r	read from the disk and write to a file
-w	read from a file and write to the disk
-z	write zero (sectors with content 0) to the disk
-q	quick format a drive (write boot sector and FATs/DIR)
-t	set partition to active

Partition Table Commands

The following commands deal with creating, deleting, modifying, and querying information about partitions.

The -m Command

```
pxfdisk -m <drv>
pxfdisk -m <drv> <part> <a> <id> <size>m
pxfdisk -m <drv> <part 1-3> <a> <id> r[est]
pxfdisk -m <drv> <part> <a> <id> <t> <s> <h> <sec> <t> <s> <h> <len>
[-s <b#>] [-f]
```

The -m command is used to create, delete, or modify a partition. You need to provide the following parameters:

- drive number: <drv> = (80, 81, 82, 83...)
- partition number: <part> = (0, 1, 2 or 3)

- whether the partition is marked active or not: <a> = (Y or N)
- the partition data (in starting and ending track/sector/head notation and also in starting sector and length notation)

The -m command uses the same argument format for entering partition data as the -p command uses for displaying it.

There are four different ways to specify partition sizes:

- create a single large active primary FAT16 partition that spans the entire disk (max. 2 GBytes).
- provide partition size in Megabytes.
- use the remaining unpartitioned hard disk space to create a partition.
- specify all the geometry-dependent parameters to create a partition.

The first example creates an active primary FAT16 partition in the first partition slot that spans the entire hard disk:

```
rem create an active DOS partition on first drive
pxfdisk -m 80
```

If the hard disk is larger than 2 GB, the partition is set to span only the first 2 GB.

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
pxfdisk -m 80 0 Y 06 400m
```

The next 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 spanning the remaining hard disk space
pxfdisk -m 80 1 N 06 r
```

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
pxfdisk -m 80 0 Y 06 0 1 1 63 101 63 63 411201 -f

rem delete second partition on first drive
pxfdisk -m 80 1 N 0 0 0 0 0 0 0 0 0 -f
```




You must reboot the PC before the system BIOS recognizes changes to the partition table!

To prevent accidental creation or deletion of partitions, the execution of the `-m` command is delayed so that you can interrupt it by hitting CTRL-C before it actually modifies the hard disk's partition table.

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
pxfdisk -m 81 0 N 06 0 1 1 63 101 63 63 411201 -f
```

```
rem create extended partition 300 MB on second drive
pxfdisk -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
pxfdisk -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
pxfdisk -m 81 1 N 05 153 1 0 205632 254 63 63 411264 -s 411264 -f
```

```
rem create logical drive 200MB
pxfdisk -m 81 0 N 06 153 1 1 63 254 63 63 411201 -s 616896 -f
```

To delete a partition, simply set all partition information to zero.

```
rem delete second partition on first drive
pxfdisk -m 80 1 N 0 0 0 0 0 0 0 0 0
```

The -p Command

```
pxfdisk -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 commandline.

The `-p` command uses the same output format for displaying partition data as the `-m` command uses for entering it:

```
pxfdisk -p 80
```

#	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.

The `-c` Command

```
pxfdisk -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 primary DOS partition exists on first hard disk
pxfdisk -c 80 0 06
if ERRORLEVEL 1 goto EXISTS

rem DOS partition does not exist, create it
pxfdisk -m 80 0 Y 06 0 1 1 63 101 63 63 411201 -f

:EXISTS
```

The `-o` Command

```
pxfdisk -o <drv> <part> <val>
```

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
pxfdisk -o 80 3 F1
```

The `-a` Command

```
pxfdisk -a <drv> <part> <val>
```

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

ing 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
pxfdisk -a 80 3 2

rem decrement ID value of last partition on second hard disk by 1
pxfdisk -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
pxfdisk -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
```

The -t Command

```
pxfdisk -t <drv> <part>
```

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
pxfdisk -t 80 1
```

The -g Command

```
pxfdisk -g <drv>
```

The -g command displays the physical geometry of a hard disk as the system BIOS sees it. The following information is shown:

- drive number (as specified on the commandline)
- total number of tracks
- number of sectors per track
- number of heads
- total hard disk size in KBytes

- total number of 512-Byte blocks available on the hard disk

```
pxfdisk -g 80
Drive=0x80, Tracks=548, Sectors=63, Heads=255, Size=4401810 KB, Blocks=8803620
```

If the hard disk given by *<drv>* does not exist, an error is displayed.

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.

The -G Command

```
pxfdisk -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 search for installed disks and log into file
pxfdisk -G 80 > disks.log
pxfdisk -G 81 >> disks.log
pxfdisk -G 82 >> disks.log
pxfdisk -G 83 >> disks.log
```

Writing a Master Boot Record

The -b Command

```
pxfdisk -b <drv>
```

The *-b* command writes a standard hard disk bootstrap loader (also called master boot record) to the hard disk that is specified by drive number *<drv>*.

```
rem clear hard disk by erasing bootstrap and partition table
pxfdisk -z 80 -n 1 -f

rem create an active DOS partition
pxfdisk -m 80 0 Y 06 0 1 1 63 101 63 63 411201 -f

rem write master boot record
pxfdisk -b 80
```

Formatting a DOS Partition

The -q Command

```
pxfdisk -q <drv> [-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, PXFDISK can quickly format a DOS partition by only writing the hard disk boot sector, partition boot sector, file allocation table, and root directory entries.

To prevent accidental formatting, the execution of the `-q` command is delayed so that you can interrupt it by hitting CTRL-C 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 DOS format drive C: (FAT16 partition)
pxfdisk -q 80 -f
```

Accessing Disk Sectors

The `-r` Command

```
pxfdisk -r <drv> <file> [-s <b#>] [-e <b#>] [-v] [-l kbs]
pxfdisk -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:\hdbbackup.dsk
pxfdisk -r 80 n:\hdbbackup.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
pxfdisk -r 80 n:\partsec.dsk -s 0 -e 0
```

```
rem this does the same as above
pxfdisk -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
pxfdisk -r 80 n:\partsec.dsk -s 0 -n 1
```

```
rem this does the same as above
pxfdisk -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
pxfdisk -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)
pxfdisk -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
pxfdisk -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 kbs` option to limit the throughput of `PXFDISK`, where *kbs* is a value that (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
pxfdisk -r 80 n:\hdbackup.dsk -v -l 100
```

The `-w` Command

```
pxfdisk -w <drv> <file> [-s <b#>] [-e <b#>] [-v] [-l kbs] [-f]
pxfdisk -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
pxfdisk -w 80 n:\hdbackup.dsk
```

Commandline 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
pxfdisk -w 80 n:\partsec.dsk -s 0 -e 0

rem this does the same as above
pxfdisk -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
pxfdisk -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
pxfdisk -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.

The -z Command

```
pxfdisk -z <drv> [-s <b#>] [-e <b#>] [-v] [-f]
pxfdisk -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
pxfdisk -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.

Commandline 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
pxfdisk -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
pxfdisk -m 80 0 N 00 0 0 0 0 0 0 0 -f
```

```
rem clear partition area
pxfdisk -z 80 -s 63 -n 1044162 -f
```

PXUTIL

PXUTIL is a multi-purpose program that works in conjunction with the PXE PROM to:

- Patch information embedded in the DHCP reply into ASCII and binary files
- Display DHCP reply information
- Set the DOS boot drive
- Restart the PC client
- Perform other miscellaneous functions.

PXUTIL is available as a DOS .COM program-file that can be executed from within batch files.

PXUTIL Command-line Options

All commandline options use a dash '-' as the leading character, e.g. -a. As a general rule, only one option can be present on the command line. The only exception to this rule is the -y option, which is used to specify the DHCP reply type. The following table shows all of the PXUTIL options in alphabetical order:

command line option	function
-a <i>fnam</i>	Patch the DHCP reply into an ASCII file.
-b <i>tnum fnam</i>	Patch the DHCP reply into a binary file.
-e	Reboot the PC via System BIOS call.
-E	Reboot the PC via <Ctrl> <Alt> .
-o <i>drv</i>	Set the boot drive
-p <i>fnam</i> >	Same as -a but uses a space to separate IP addresses.
-s [<i>tag</i> [<i>tags</i>]]	Display the entirety or just a specified option from the DHCP reply.
-S	Like -s but shows only the DHCP options which have been set

command line option	function
-y	Specify DHCP reply type
-?	Display the PXUTIL options.

Patching Reply Information Into Files

“Patching” is a powerful method for modifying the contents of client configuration files while the PC client is booting. By using patching, an infinite number of clients can share the same configuration files yet be uniquely configured at boot time. Patching works by replacing the contents of a string, or “tag ID”, within a file with a value, or “field”, that is sent to the client via the DHCP reply.

Some BOOTP and DHCP servers limit the number of bytes that can be used by user-defined fields to as few as 64 bytes. This limitation comes from the original BOOTP protocol specification. PXUTIL will issue a warning if the BOOTP/DHCP vendor field is exhausted. PXE clients and PXUTIL can support collections of fields which are up to 1024 bytes in length if the DHCP server is able to support them.

The -a Option

PXUTIL -a patches information given by the DHCP server into ASCII files. The PXUTIL 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.

Optionally, you can use the -y option to define the PXE packet reply type. For example the file contents

```
# /etc/hosts Internet address database
#
127.0.0.1      localhost local
#@yip##### #@chn#####
```

shows the two tags #@yip and #@chn. When running PXUTIL -a on the file, the two tags will be replaced by strings embedded in the DHCP reply. The #@chn will be replaced by the hostname of the PXE client, and the tag #@yip will be replaced by the Internet address of the PXE client. 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 with PXUTIL -a a:\etc\hosts, the file 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\bosts*).

The following tags can be used with the PXUTIL program:.

tag	RFC951 BOOTP reply
#@bfn#	Bootfilename
#@caa#	Client hardware address
#@cha#	Client hardware address, separated by points
#@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 domain name server IP address.
#@ds1#	Second domain name server IP address.
#@dsf#	All 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.
#@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.

tag	RFC1048 BOOTP vendor field
#@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	misc. 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).
#@typ#	Type of controller (hex 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 or DHCP server which is used. The BOOTPD32 server that ships with the BootManage[®] PXE Toolkit supports all of the above tags.

The alignment of a string replacing a tag can also 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 the 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
```

PXUTIL does not change the size of a patched file. Therefore, all spaces are moved either to the end of the line or padded to the end of the field.

The -b Option

The `-b` option is 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 (a program, driver, encoded data, etc.) 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 a sample program is hard-coded within the executable file and has to be changed before it is loaded. First, determine if there are differences between the two program files using the MS-DOS FC command

```
C:\> fc /b program.1 program.2
Comparing files PROGRAM.1 and PROGRAM.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 PXUTIL 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 within a custom option in the BOOTP or DHCP server's configuration.

```
fileoffset 49 = 000031 (hex)
ASCII 12345678 = 31 32 33 34 35 36 37 38 (hex)
```

The corresponding entry in a *bootab* file would be:

```
diskless:\
:tc=site:ha=0000c0e23324:ip=193.141.47.198:\
:bf=client.X:T132=0000313132333435363738:
```

Then, the PXUTIL program is called from *AUTOEXEC.BAT* to patch the serial number into the program before it is loaded:

```
pxutil -b 132 a:\program.exe
...
program.exe
```

The -s Option

The reply packet given by the DHCP server can be displayed by using the *-s* option of the PXUTIL program. This option displays all of the fields available to the PXUTIL program and any additional option or user-defined fields.

A sample output follows:

```
A:\> pxutil -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 options you want to see:

```

A:\> pxutil -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:\> pxutil -s yip sip > tmp.bat
A:\> call tmp.bat
A:\> _

```

The `-s` option also displays binary options. If an option contains binary characters (less than ASCII 32 or greater than ASCII 127), the option is displayed as follows:

```
t132= [5] f0023489f1
```

Where `[5]` indicates that the length of the option is 5 bytes. The bytes themselves follow in hexadecimal notation.

The last two fields `dmp1` and `dmp2` of the `PXUTIL` output show a hex-dump of the DHCP reply.

The -S Option

The `-S` option lists the same options 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 the Microsoft Network Client for MS-DOS and Digital's Path-Works.

Miscellaneous PXUTIL Options

The -e and -E Options

The `-e` option reboots a standard PC by calling the System BIOS reboot routine. The `-E` option reboots a PC by inserting the `<Ctrl> <Alt> ` sequence into the keyboard buffer.

Both options can be used in an MS-DOS batch file if the installation of the network software failed.

The -o Option

This option is used to set the DOS boot drive. This will make DOS believe that the system has been booted from the drive you specify.

```
# make DOS believe that it has been booted from drive C:
pxutil -o c
```

The -y Option

This option is not meant to be used alone, but in conjunction with other options in order to specify the PXE packet reply type. If no packet type is specified, the default of 3 (BINL) is used.

```
pxutil -y 1 ...    (specify DHCP DISCOVER packet type)
pxutil -y 2 ...    (specify DHCP ACK packet type)
pxutil -y 3 ...    (specify BINL packet type)
```

You do not need to specify the `-y` option when you use a PXE Client with only a BOOTP or DHCP and a TFTP server. If the packet types mentioned above do not mean anything to you, simply do not use the `-y` option.

BOOTPD32

BOOTPD32 is a 32-Bit BOOTP server for the Microsoft Windows 95, Windows 98 and Windows NT (Workstation and Server) operating system. On Windows NT, BOOTPD32 can run as either a Win32 application or as a Windows NT service.

BOOTP Versus DHCP Servers

You can use either a BOOTP or DHCP server to provide configuration information to a PXE 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.

A PXE client will accept replies from both BOOTP and DHCP servers, as long as the PXE specific information is provided (see “*Getting Started*” on page 15). Only if you want to hand out IP addresses to clients dynamically, you must use DHCP.



On the Windows NT 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”.

Features of BOOTPD32

BOOTPD32 supports custom tags, which is required for PXE operation. Also, BOOTPD32 supports large BOOTP replies so that maximum use can be made of the advanced patching features described in the PXUTIL chapter. Additional features include simultaneous operation on multiple network interfaces and configuration through a single text based configuration file (bootptab). If you have worked with the *bootptab* file on UNIX before, you will find the format familiar.

Installing BOOTPD32

First, copy the file *bootpd32.exe* from the BootManage[®] PXE Toolkit Utility Diskette to the local hard disk of a Windows 95, Windows 98, Windows NT Workstation or Server machine.

```
C:\> copy a:\bootpd32.exe %WINDIR%           (on Windows 95/98)
C:\> copy a:\bootpd32.exe %SystemRoot%\system32 (on Windows NT)
```

Next, create a bootptab file (e. g. by using a text editor like notepad). By default, BOOTPD32 expects the *bootptab* file to be located in the *c:\etc* directory, but you can instruct BOOTPD32 to read this file from any other location.

```
C:\> mkdir c:\etc
C:\> cd etc
C:\> notepad bootptab
```

The BootManage[®] PXE Toolkit Utility Diskette contains sample bootptab files, so you do not need to type in a bootptab file manually.

BOOTPD32 will look up port numbers for the bootps and bootpc ports in the file *%SystemRoot%\system32\drivers\etc\services*. If these entries are 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 *-cmd* option and all other options that suit your needs, e. g.:

```
C:\> bootpd32 -cmd -i 195.4.136.167/255.255.255.224 -d -d -d -d
```

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

```
C:\> bootpd32 -install -i 195.4.136.167/255.255.255.224 -d -d -d -d
```

For a detailed list of all supported options, please see “*BOOTPD32 Commandline Options*” on page 67.

Using BOOTPD32 with PXE Clients

To use BOOTPD32 with PXE clients, a PXE client's record must contain the PXE client class identifier, or else the PXE client will ignore the BOOTP reply.

Also, you may have to set the IP address of the TFTP server which holds the file the PXE client should download and execute.

```
pxeclient:\
:(standard BOOTP tags):\
:T60="PXEClient":           (PXE client class identifier)
```

:T128="PxSrV=195.4.136.168": (TFTP server's IP address)

BOOTPD32 Commandline Options

The following arguments can be given to the BOOTPD32 server:

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

The following options are provided:

-i ipaddr[/subnetmask]:subnetbits [-b] [-c port] [-d] [-h] \ [-n] [-s] [-t timeout] [-u] [-z] [configfile [dumpfile]]

where:

Option	Description
-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.
-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.
-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.
-n	ignore all requests with non-zero source IP address
-s	Enable stand-alone mode (always true).
-t timeout	terminate BOOTPD32 after <i>timeout</i> 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 <i>c:\etc\bootptab</i>).
dumpfile	Filename which BOOTPD32 uses to dump the in-memory configuration to (default is <i>c:\etc\bootpd.dmp</i>).

Multiple Network Interfaces

If you want BOOTPD32 to operate on multiple network interfaces, specify multiple `-i` options on the command line, e.g.:

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

Subnetting

If you use subnetting, you must add subnet information to the `-i` option:

```
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)
```

Using an Arguments File

If the commandline 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.arg`:

```
# 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 the BOOTPD32 daemon using the `@` option:

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

New Features

The BOOTPD32 daemon adds the following features:

- New tag `ms` to increase the BOOTP reply size.
- Automatic detection of maximum usable BOOTP reply size.
- New tag `mw` to delay a BOOTP reply.
- New tag `sa` 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=rand.X:sa=193.141.47.193:
```

TFTPD32

TFTPD32 is a TFTP server that runs on Microsoft Windows 95, Windows 98 and Windows NT Workstation and Server machines. On Windows NT, TFTPD32 can be run as a Win32 application or installed as a Windows NT service.

The Windows NT Server includes a DHCP server, but it does not include a TFTP server. PXE clients require a TFTP server in order to download the Network Bootstrap Program (PXBOOT).

TFTPD32 Features

BootManage TFTPD32 adds several features which are not available in standard TFTP implementations (tftpd):

- Allows restricted file access for security.
- Does not spawn a new process for every TFTP transfer (less server load).
- Allows larger TFTP segment sizes (LTFTP) for higher throughput and fewer network transactions.
- Supports multicast file transfer (MTFTP) for faster concurrent booting of multiple clients.

Installing TFTPD32

First, copy the file *tftpd32.exe* from the BootManage[®] PXE Toolkit Utility Diskette to the local hard disk of a Windows 95, Windows 98, Windows NT Workstation or Server machine.

```
C:\> copy a:\tftpd32.exe %WINDIR%           (on Windows 95/98)
C:\> copy a:\tftpd32.exe %SystemRoot%\system32 (on Windows NT)
```

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 commandline using the `-cmd` option and any other options which suit your needs, e. g.:

```
C:\> tftpd32 -cmd -v 2
```

If you do not specify any options on the commandline, TFTPD32 will read the options from the text file `c:\etc\tftpd.cnf`. For a detailed list of all supported options, please see “*TFTPD32 Commandline Options*” on page 73.

Installation As a Service

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

```
C:\> tftpd32 -install
```

Please note that when using `-install`, you cannot specify any other options on the commandline. 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 system startup. Like any other service, you can start and stop TFTPD32 using the Windows NT Service Control Manager. In addition, you can control TFTPD32 from the commandline:

```
C:\> net start tftpd           (start TFTPD32 service)
C:\> net stop tftpd           (stop TFTPD32 service)
```

Using TFTPD32 with PXE Clients

To use TFTPD32 with PXE clients, there are no special requirements. Place all of the files that the PXE client must access on the TFTP server and make sure that these files have public read access. These files will most likely be:

- the bootstrap loader program, *pxboot* (will be downloaded by the PXE code)
- one or more network boot image files, e. g. *pxboot.X* (will be downloaded by the bootstrap loader PXBOOT).
- global and individual option files, e. g. *pxboot.opt*, *a03425f3.opt* (will also be downloaded by the bootstrap loader PXBOOT).

TFTPD32 Commandline 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)
```

The following options are provided:

```
[-c num] [-d [path]] [-h] [-i num] [-k num] [-l [file]] [-m file port] \
[-p num] [-r] [-s num port] [-u num] [-v level] [-w] [-x]
```

where:

argument	meaning
-c <i>num</i>	Number of concurrent TFTP transfers (max. 64)
-d [<i>path</i>]	Restrict TFTP access to directory <i>path</i> and its subdirectories.
-h	Enable read-ahead buffer.
-i <i>num</i>	Exit after <i>num</i> seconds of inactivity.
-k <i>num</i>	Set TFTP transmission keep-alive time.
-l [<i>file</i>]	Log all messages to a file.
-m <i>file port</i>	MTFTP configuration file and listen port.
-p <i>num</i>	Set TFTP listen port.
-r	Restrict file access to current directory.
-s <i>num port</i>	Set TFTP segment size and listen port.
-u <i>num</i>	Number of MTFTP unicasts to be sent (default: 4).
-v <i>level</i>	Set level of verbosity (0, 1, or 2).
-w	Enable writing to existing files on the TFTP server
-x	If -w is also set, allow creation of files on the TFTP server

The -c Option

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
```

The -d Option

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
```

The -h Option

This option enables read-ahead buffers for the TFTP file transfer. That is, TFTP32 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, TFTP32 does not read-ahead.

The -i Option

TFTP32 can be automatically terminated by using the `-i` option. If there is no TFTP activity for the number of seconds specified, TFTP32 terminates. The following example aborts TFTP32 after 45 seconds of inactivity:

```
tftpd32 -i 45
```

The -k Option

TFTP32 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.

The -l Option

All output of TFTP32 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
```

The -m Option

The `-m` option enables the TFTP32 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 TFTP32 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.

The -p Option

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
```

The -r Option

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
```

The -s Option

This option allows one to set the TFTP segment size. The standard TFTP protocol specifies a segment size of 512 bytes. By increasing the segment size, fewer network transactions are needed and, therefore, faster file transfer can be achieved. 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. Opening a TFTP transmission at the standard TFTP port 69 lets TFTPD32 send 512 byte packets.

Opening a TFTP transmission at the non-standard UDP port 59 allows TFTP32 to send up to 1408 bytes in TFTP packets.

The -u Option

The -u option is used in conjunction with multicast file transfer (MTFTP). The -u option specifies the number of unicast packets that TFTP32 sends to the MTFTP client after a transmission has been initiated.

The default is 4 packets and should not be changed.

The -v Option

The debugging and activity output of TFTP32 can be changed by the -v option. The default is 1, and increasing the number generates more output. 0 does not create any output.

The -w Option

Normally, TFTP32 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.

The -x Option

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

Working With the Microsoft DHCP Server

This chapter describes how to configure the Microsoft DHCP Server for use with PXE clients. The Microsoft DHCP Server ships with the Microsoft Windows NT 4.0 Server. It is assumed that the reader is already familiar with the Microsoft DHCP server and, therefore, only the particulars for working with PXE clients are covered here. For general information on how to install and configure the Microsoft DHCP server, please refer to the Microsoft documentation.

DHCP Versus BOOTP Servers

You can use either a BOOTP or DHCP server to provide configuration information to a PXE 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.

A PXE client will accept replies from both BOOTP and DHCP servers, as long as the PXE specific information is provided (see “*Getting Started*” on page 15). Only if you want to hand out IP addresses to clients dynamically, you must use DHCP.

Installing the 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 Servers should be listed in the Services tab as shown in *Figure 10-1*, and you are then asked to reboot the NT Server.

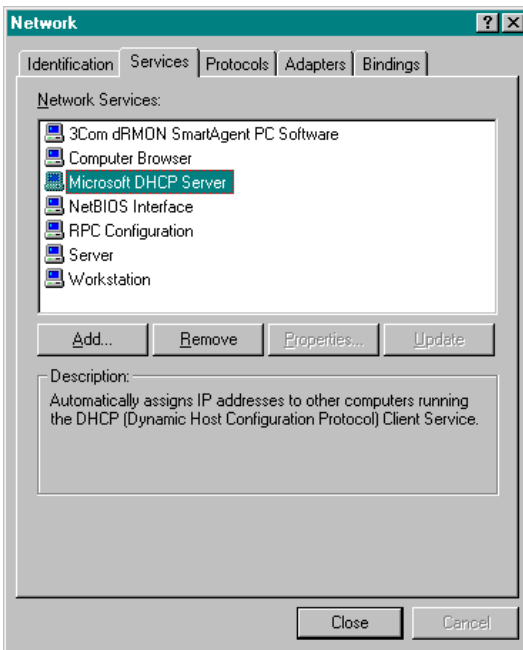


Figure 10-1. Installing the Microsoft DHCP Server

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 10-2*.

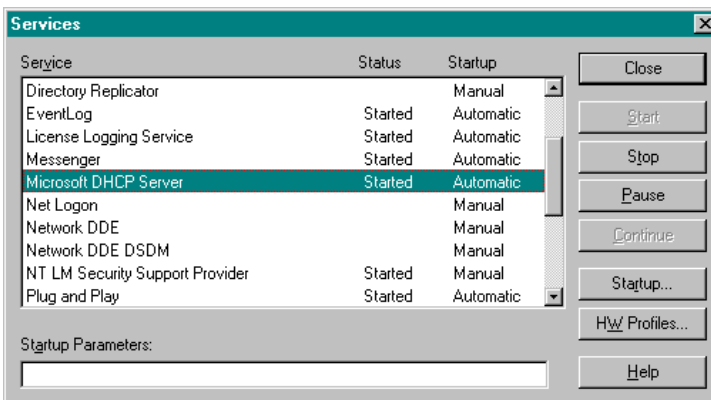


Figure 10-2. Services Control Panel

Configuring the Microsoft DHCP Server

The Microsoft DHCP Server can be configured through both a graphical interface and through a commandline 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 commandline program, DHCPCMD.EXE, does not ship with the Windows NT 4.0 Server, but can be found in the Microsoft Windows NT 4.0 Server Resource Kit. 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 10-3*.

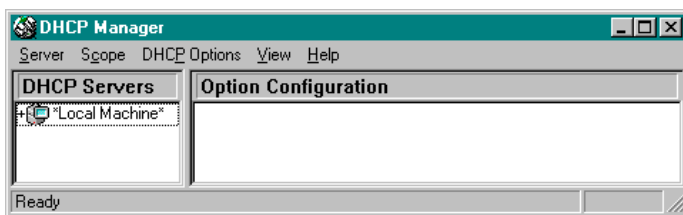


Figure 10-3. DHCP Manager

Creating and Activating a DHCP Scope

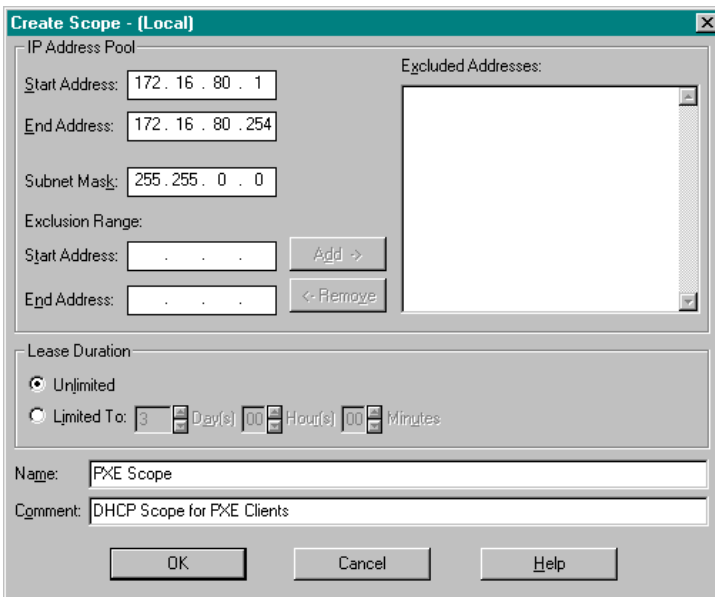
To create a DHCP scope, highlight Local Machine in the DHCP Servers pane and select Create from the Scope menu. In the Create Scope Dialog, enter the Scope information as displayed in *Figure 10-4* and then click OK.

A dialog box is displayed asking if you want to activate the just-created scope. Since there are more modifications to be done to this scope, click NO here.

The Client Class Identifier Option

Conforming to the PXE specification, the client class identifier option (060) must be set to the value PXEClient. To do this, we must first define the client class identifier option, add it to our DHCP scope, and then set it to the value PXEClient.

To define the client class identifier option, highlight the just-created scope in the DHCP Servers pane and then select DHCP Options → Defaults. In the opening dialog box, click New and complete the Add Option Type dialog as shown in *Figure 10-5*. Then, select OK to create the new option.



Create Scope - (Local)

IP Address Pool

Start Address: 172.16.80.1

End Address: 172.16.80.254

Subnet Mask: 255.255.0.0

Excluded Addresses:

Exclusion Range:

Start Address: . . . Add ->

End Address: . . . <- Remove

Lease Duration

☒ Unlimited

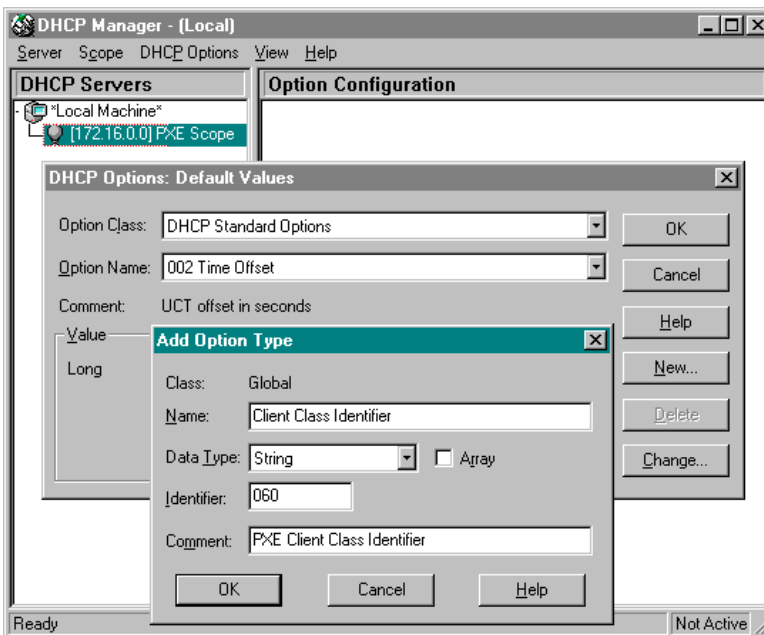
☐ Limited To: 3 Day(s) 00 Hour(s) 00 Minutes

Name: PXE Scope

Comment: DHCP Scope for PXE Clients

OK Cancel Help

Figure 10-4. Create DHCP Scope



DHCP Manager - (Local)

Server Scope DHCP Options View Help

DHCP Servers

Local Machine

(172.16.0.0) PXE Scope

Option Configuration

DHCP Options: Default Values

Option Class: DHCP Standard Options OK

Option Name: 002 Time Offset Cancel

Comment: UCT offset in seconds Help

Value: Add Option Type New...

Long: Delete

Class: Global Change...

Name: Client Class Identifier

Data Type: String ☐ Array

Identifier: 060

Comment: PXE Client Class Identifier

OK Cancel Help

Ready Not Active

Figure 10-5. Define Client Class Identifier option

Next, add the client class identifier to your scope by selecting Scope from the DHCP Options menu. In the Unused Options box, select 060 Client Class Identifier and click Add. Then, click Value and enter the string PXEClient in the text field, as shown in *Figure 10-6*.

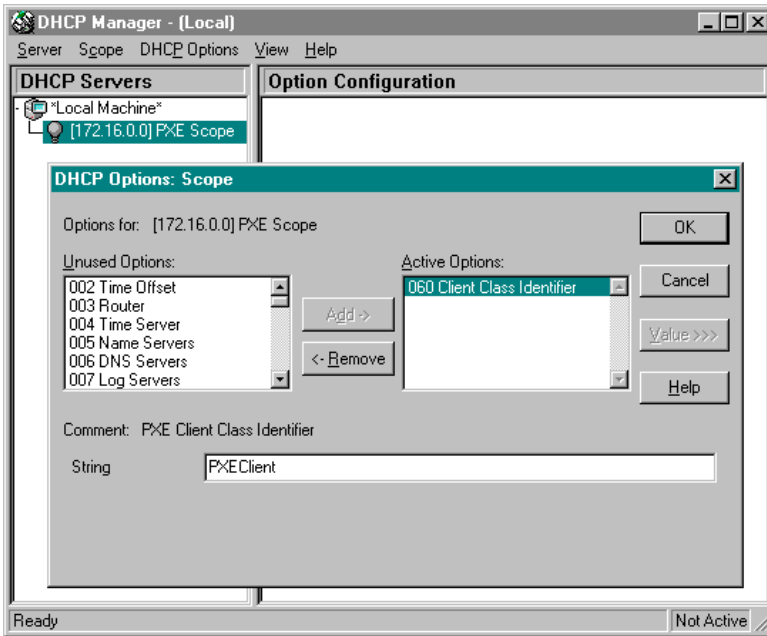


Figure 10-6. Add Client Class Identifier option to DHCP scope

After clicking OK, the client class identifier option and its value should appear in the Option Configuration pane.

The Bootfile Name Option

The PXE client must be told to download and execute the PXBOOT boot loader. To achieve this, the Bootfile Name option (067) is used. Proceeding in a manner similar to the Client Class Identifier option, set the Bootfile Name option to the value pxboot and add it to the DHCP scope (See *Figure 10-7*).

After adding the new option to your scope by clicking OK, the PXE client is set-up. If you boot the PXE client now, you may get an error message saying "PXBOOT-E10: TFTP server IP address not set" because the Microsoft DHCP server does not set its IP address in the DHCP reply. To set the TFTP server's IP address, a custom option must be created.

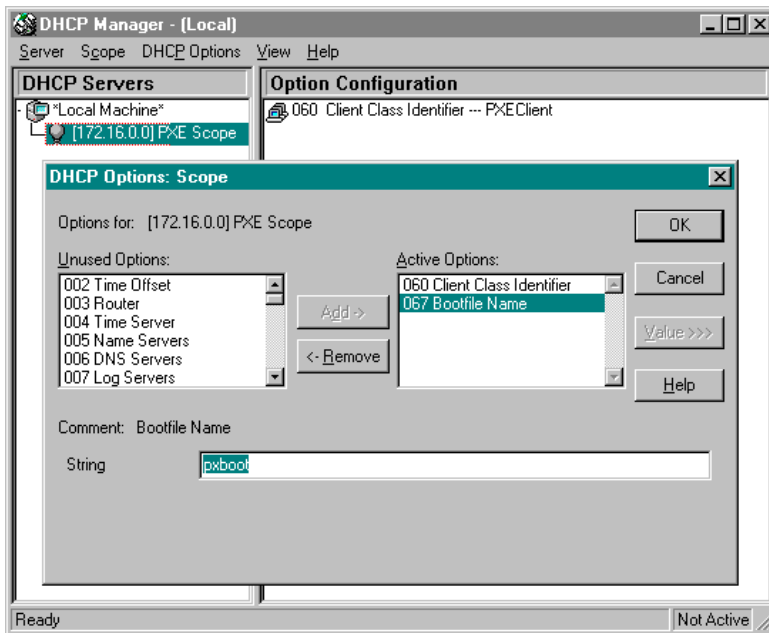


Figure 10-7. Add Bootfile Name option to DHCP scope

Implementing Custom Options

In contrast with BOOTPD32, the Microsoft DHCP server does not automatically send custom vendor options to PXE clients. Only one such field is sent and its name is Vendor Specific Info (043). Since custom options are used to pass control and patch information from the server to the PXE client, we describe a technique here to embed multiple custom options within the single Vendor Specific Info (043) option.

When trying to add and edit the Vendor Specific Info (043) option in the same manner as previously done with the Bootfile Name (067) option, you will notice that you cannot easily enter a string value, but have to enter byte values in decimal or hexadecimal (see *Figure 10-8*).

Therefore, if you want to use “PxInS=01” as the value of the Vendor Specific Info (043) option, you would have to lookup the ASCII code values of the characters that comprise the string “PxInS=01” and then enter them, one at a time, in the Microsoft DHCP Server’s Numeric Value Array Editor dialog.

Recognizing that this is extremely tedious, the BootManage[®] PXE Toolkit supports a mechanism which allows the direct entry of the option value as a string. Note that this method does not conform to the DHCP standard and when used in conjunction the Microsoft DHCP server and other PXE boot loaders, the field option

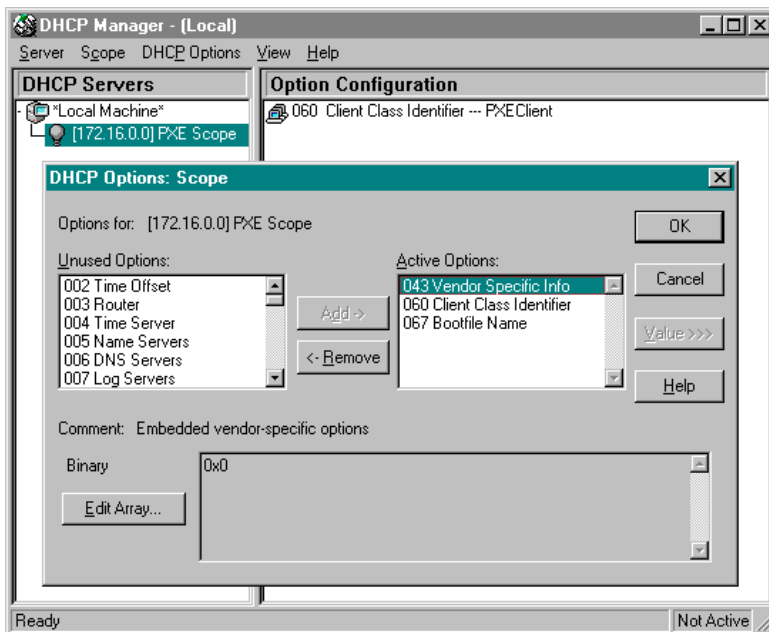


Figure 10-8. Vendor Specific Info (Original)

number and length still cannot be properly encapsulated. However, when used with the BootManager® PXE Toolkit, things will work as expected.

Close DHCP Manager and start the Microsoft REGEDIT program. Select the key HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\DHCPServer\Configuration\OptionInfo\043. Right-click the folder icon labeled 043 in REGEDIT's left window pane as shown in Figure 10-9 and select Delete from the pop-up context menu.

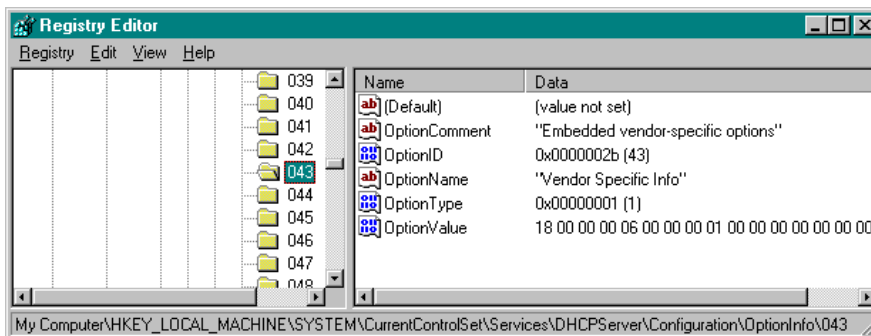


Figure 10-9. Deleting option 043 with Registry Editor

Make sure that the 043 key has been deleted and close REGEDIT. Now, start DHCP Manager and verify that the Vendor Specific Info (043) option is no longer

present. Use the same method as described earlier in “*The Client Class Identifier Option*” to add option 043. *Figure 10-10* shows the values to enter.

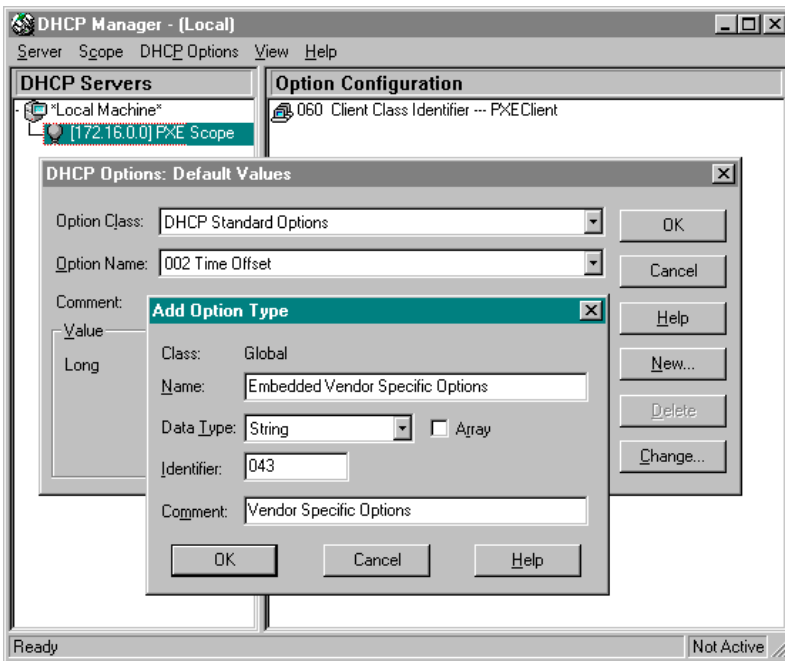


Figure 10-10. Add option 043 as string type

Now you can directly enter a string as the value of option 043. We will use this feature when telling the PXE client about the TFTP server's IP address (see *Figure 10-11*).

TFTP Server IP Address

The PXE client must know the IP address of the TFTP server it should use to download the PXBOOT boot loader. Unfortunately, the Microsoft DHCP Server does not set its own IP address in the DHCP reply. To set the TFTP server IP address, use the magic string `PxSrV=172.16.0.1` in the vendor option field* as shown in *Figure 10-11*. Note that the TFTP server can reside on a different machine than the DHCP server.

If you performed all of the steps in this chapter, your DHCP scope should now have three options configured, as seen in *Figure 10-12*.

These options are sufficient to allow a PXE client to locate and download the PXBOOT boot loader. Note that, in order to make use of the BootManage® PXE Toolkit's advanced features, you must provide additional custom options.

* In this example, 172.16.0.1 is the TFTP server's IP address

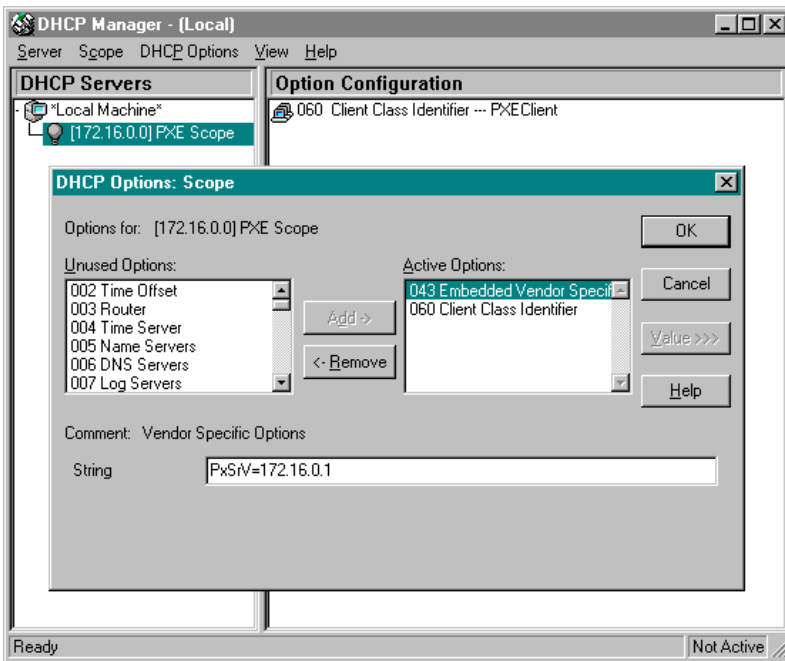


Figure 10-11. Set the TFTP server's IP address

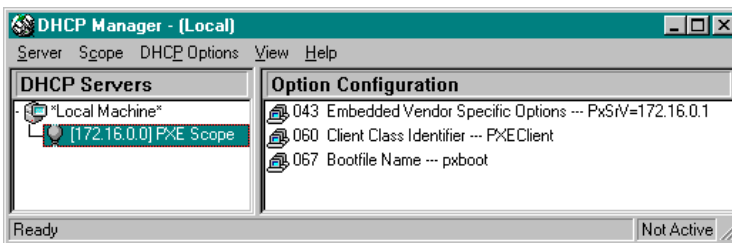


Figure 10-12. Basic options for PXE operation

Implementing Multiple Custom Options

ABOOTP server allows one to specify multiple custom options* for a client. The following excerpt from a bootptab (configuration) file shows that each custom value is described using its own dedicated option:

```
:T128="Pxsrv=172.16.0.1":           (TFTP server's IP address)
:T129="PxroU=172.16.0.254":         (Default IP Router)
```

* In BOOTP terminology, options are called 'tags'.

As mentioned above, the Microsoft DHCP Server only supports a single custom option (043) that can be sent to the PXE client. To overcome this limitation, one must concatenate multiple custom options to form a single custom option as shown in Figure 10-13. Within this concatenation string, individual options are separated by a semicolon.

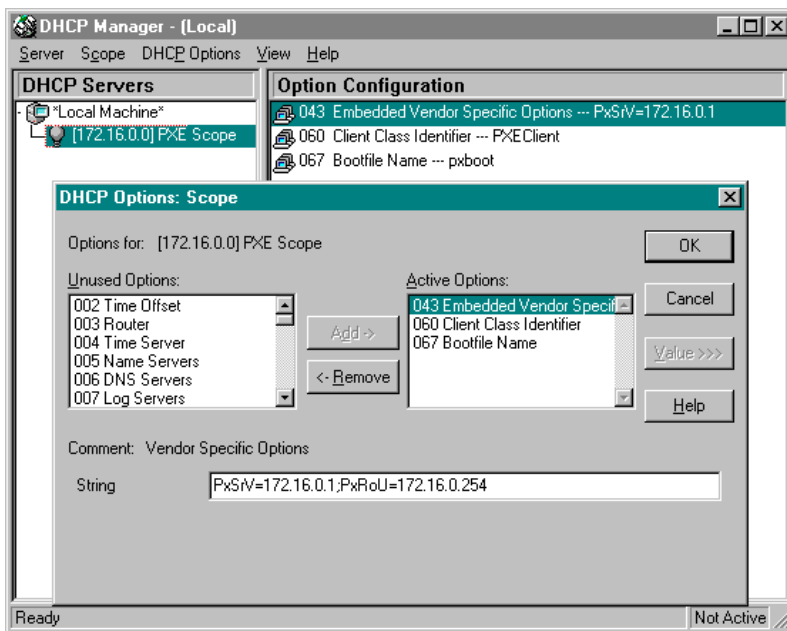


Figure 10-13. Concatenating multiple custom options

Working With the 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 4.0 Workstation 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. It also creates over-the-network installation diskettes that can be easily transferred to boot images through the BootManage[®] PXE Toolkit 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 4.0 Workstation, 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 the Network Client Administrator

On your Windows NT 4.0 Server, you can start the Network Client Administrator program by clicking Start → Programs → Administrative Tools (Common) → Network Client Administrator (see *Figure 11-1*).

Select Make Network Installation Startup Disk and then click Continue. Complete the next dialog box as shown in *Figure 11-2* 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 will create the directory *C:\CLIENTS* and copy 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 11-3*.

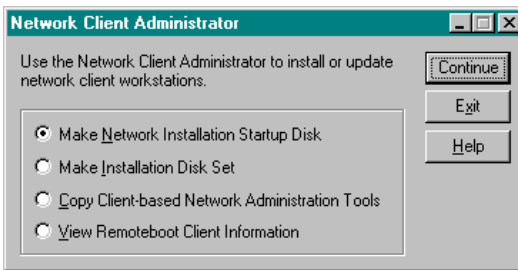


Figure 11-1. The Network Client Administrator

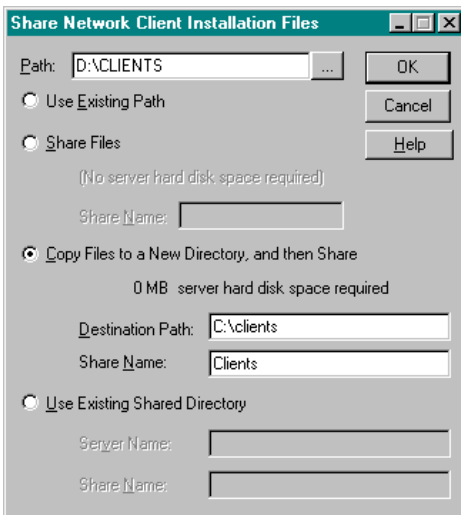


Figure 11-2. Share Network Client Installation Files

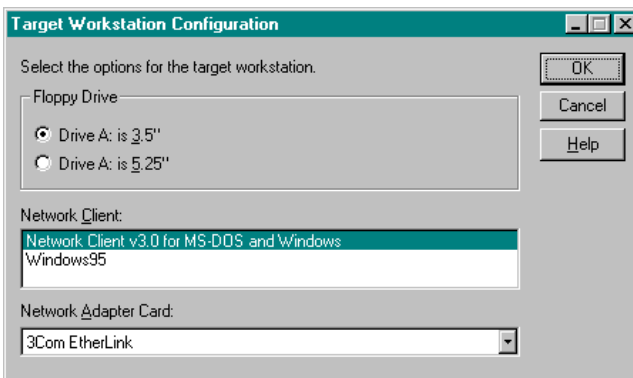


Figure 11-3. Target Workstation Configuration

Adding Windows NT Entries

As of this writing, the Network Client drop-down box 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 4.0 Workstation and Windows NT 4.0 Server available, too, select CANCEL and terminate the Network Client Administrator.

The file `\CLIENTS\SUPPORT\README.TXT` on the Windows NT 4.0 Server CD-ROM describes how to setup server-based installations of Windows NT 4.0 Workstation and Windows NT 4.0 Server. Here is an abbreviated description:

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

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

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

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

Creating an Installation Diskette

Again, start the Network Client Administrator and click through the screens (do not change any settings) until you reach the Target Workstation Configuration dialog. This time, it should look like *Figure 11-4*.

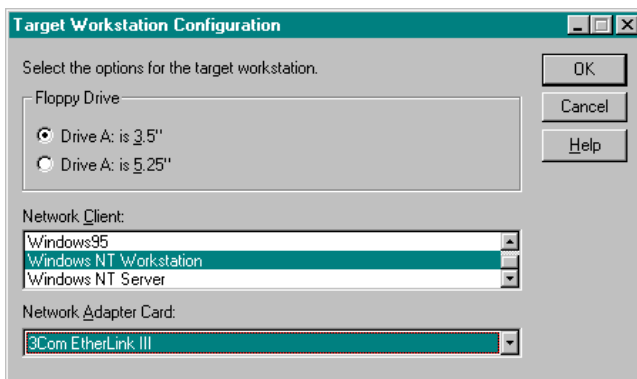


Figure 11-4. Target Workstation Configuration (Updated)

In the Network Client drop-down 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 PXE client is not listed here (e.g. Intel EtherExpress Pro/100B), 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 11-5*.

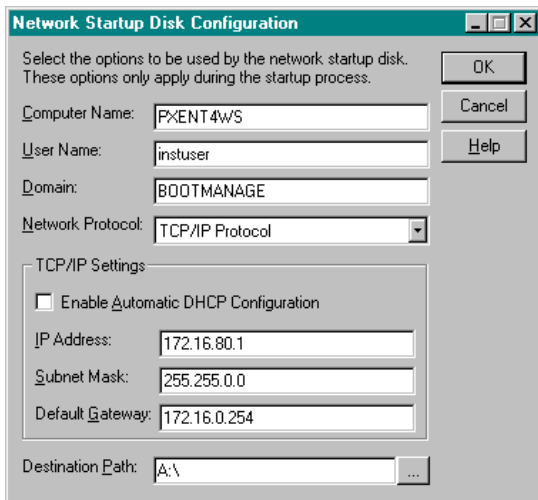


Figure 11-5. Network Startup Disk Configuration

The name of our PXE client computer is PXENT4WS, and it belongs to the work-group (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[®] PXE Toolkit's patching capabilities.*

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 PXE 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†:

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

* 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.

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

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*.
- 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 the 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 EtherExpress Pro/100B 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*.

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 “intuser” who will only need to have read access to the CLIENTS share.

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

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.

A Sample Windows NT Installation

This chapter shows how to perform a hands-free unattended installation of the Windows NT 4.0 Workstation OS (US version) on a PXE client PC using the Boot-Manage[®] PXE Toolkit. Note that this is a completely “bare metal” remote installation, beginning with partitioning the hard disk and never requiring a visit to the client machine. With minimal modifications, one can also use this description to install the Windows NT 4.0 Server on a PC equipped with PXE.

Environment

As an installation server, we use an Intel-based Windows NT 4.0 Server (US Version) with Service Pack 3 installed.

During the real-mode installation phase, the transport protocol is TCP/IP.

The PXE client PC is equipped with an Intel EtherExpress Pro/100B network card.

We assume that you are already familiar with:

- creating server-based installations for Windows NT 4.0 Workstation and Server machines by using the Network Client Administrator program that is included with the Windows NT 4.0 Server operating system.
- creating an answer file for unattended installation with the Windows NT Setup Manager tool that is included with the Microsoft Windows NT 4.0 Workstation Resource Kit.
- adding OEM components to the automated installation as described in the Microsoft Windows NT 4.0 Workstation Resource Kit.
- Configuring the Microsoft DHCP Server that is included with the Windows NT 4.0 Server Operating System.
- installing and configuring the BOOTPD32 and TFTP32 servers for use with PXE clients as described earlier in this manual.

Installation Overview

Before proceeding with the step-by-step installation instructions, please familiarize yourself with this overview 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 4.0 Workstation operating system over the network. This is described in the Microsoft Windows NT 4.0 Workstation 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 Microsoft Network Client (a real mode network client), connects to the installation server, and then copies all of the installation files to the local hard disk. The user is instructed to remove the boot diskette and the system reboots from the local hard disk, which continues the Windows NT installation process.

What the BootManage[®] PXE Toolkit Provides

The BootManage[®] PXE Toolkit 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 PXE code downloads and executes the PXBOOT bootstrap loader. By using the ID field of an unused partition entry as a status flag, PXBOOT checks the current phase of the automated installation and decides whether to boot from the installation boot image or from the local hard disk.

In phases 0, 1, and 2, PXBOOT downloads the installation boot image. Within the boot image, the *INSTALL.BAT* file contains a separate command section for each installation phase. Phase 0 partitions the local hard disk, phase 1 formats it, and

phase 2 connects to a network drive on the installation server and executes Microsoft's WINNT program. WINNT then copies all of the Windows NT installation files to the PC's local hard disk and reboots the PC in order to continue the operating system installation.

In phase 3, PXBOOT starts the PC from the local hard disk to complete the installation (Text mode setup, NTFS conversion, GUI mode setup).

The text-based configuration files *INSTALL.BAT*, *PROTOCOL.INI*, *SYSTEM.INI*, and *UNATTEND.TXT* 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 bootptab configuration file, if using BOOTPD32, or in the DHCP server configuration database, if using the Microsoft DHCP Server. The option values are patched into the text-based configuration files by the PXUTIL program after downloading the boot image.



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 4.0 Server is called NT4SERVER. It is member of the workgroup BOOTMANAGE and has the IP address 172.16.0.1 and netmask 255.255.0.0.

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 4.0 Server operating system. Follow the instructions in the chapter entitled "*Working With the Microsoft Network Client Administrator*" on page 87.

Step 2: Add Files to the Installation Server

You may want to add additional software packages to the automated installation, e.g. a Windows NT Service Pack or a set of standard applications which should be installed together with the operating system. 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 4.0 Workstation Resource Kit for instructions on how to do this.

Step 3: Create a Network Client Boot Diskette

As described in “*Working With the Microsoft Network Client Administrator*” on page 87, use the Microsoft Network Client Administrator to create a network boot diskette. **Do not proceed** without verifying that the diskette works: You must be able to boot the client from the diskette, log-on, and connect to the appropriate network drive on the installation server.



Since no BootManage[®] PXE Toolkit 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 4: Add the BootManage[®] PXE Toolkit Files

So far, we have entirely followed the Microsoft way of performing unattended installations over the network, and there was nothing specific to the BootManage[®] PXE Toolkit.

The next step is to create a boot image from the installation diskette, but before doing so we must copy the PXE Toolkit's real mode utilities to the diskette. In addition, the configuration files must also be modified.

On the installation diskette, create the directory *A:\BIN* and then copy the files *PXUTIL.COM*, *PXFDISK.EXE*, and *REBOOT.COM* from the BootManage[®] PXE Toolkit Utility Disk to this directory.

Copy the file *SMARTDRV.EXE* from an MS-DOS 6.22 system to the *A:\NET* directory.

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

Root directory

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

Directory BIN

PXFDISK	EXE
PXUTIL	COM
REBOOT	COM

Directory NET

IFSHLP	SYS
NDISHLP	SYS
HIMEM	SYS
NEMM	DOS
TCPDRV	DOS
E100B	DOS
PROTMAN	DOS
EMM386	EXE
SMARTDRV	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 5: Modify the Configuration Files

Use a text editor to modify the configuration files *A:\CONFIG.SYS*, *A:\AUTOEXEC.BAT*, *A:\INSTALL.BAT*, *A:\NET\PROTOCOL.INI*, and *A:\NET\SYSTEM.INI* so that they match following listings (except where you may have made site-specific customizations).

Samples of these files can be found on the BootManage[®] PXE Toolkit Utility Diskette, in the directory *A:\SAMPLES\PXENT4WS*, so entering these files manually is not necessary.

A:\CONFIG.SYS

In *CONFIG.SYS*, it is **critically important** to specify */testmem:off* for *HIMEM.SYS*. Otherwise, *HIMEM.SYS*'s RAM check will overwrite the boot image.

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

device=a:\net\himem.sys /testmem:off
device=a:\net\emm386.exe noems

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

A:\AUTOEXEC.BAT

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

rem patch the installation batch file
pxutil -a a:\install.bat

rem execute the (patched) installation batch file
install.bat
```

A:\INSTALL.BAT

```
rem check if user pressed key to request re-initialization
rem this is only relevant if enabled by the PxDiS option!
if not _#T254*##### == _PxKeY goto PHASE
cls
echo User requested reinstallation at boot time
goto REINSTALL

rem determine installation phase
:PHASE
pxfdisk -c 80 0 00
if ERRORLEVEL 1 goto PHASE0

pxfdisk -c 80 0 a1
if ERRORLEVEL 1 goto PHASE1
pxfdisk -c 80 0 61
if ERRORLEVEL 1 goto PHASE1

pxfdisk -c 80 0 a2
if ERRORLEVEL 1 goto PHASE2
pxfdisk -c 80 0 62
if ERRORLEVEL 1 goto PHASE2

pxfdisk -c 80 0 a3
if ERRORLEVEL 1 goto REINSTALL
pxfdisk -c 80 0 63
if ERRORLEVEL 1 goto REINSTALL

rem unknown phase - perform reinstallation
cls
echo Unknown Phase !
```

```
:REINSTALL
echo.
echo Press a key to clean the hard disk and restart installation.
pause

pxfdisk -m 80 0 N 00 0 0 0 0 0 0 0 0 -f
pxfdisk -m 80 1 N 00 0 0 0 0 0 0 0 0 -f
pxfdisk -m 80 2 N 00 0 0 0 0 0 0 0 0 -f
pxfdisk -m 80 3 N 00 0 0 0 0 0 0 0 0 -f
goto PHASE

rem Phase 0: Partition the hard disk
:PHASE0
cls
echo PHASE0: Create a DOS partition on the hard disk
echo.

rem create flag partition 0
pxfdisk -m 80 0 N 00 10m -f

rem create an active DOS partition 1
pxfdisk -m 80 1 y 06 #@T200*##m -f

rem clear partition 2 and 3
pxfdisk -m 80 2 N 00 0 0 0 0 0 0 0 0 -f
pxfdisk -m 80 3 N 00 0 0 0 0 0 0 0 0 -f

rem write master boot record
pxfdisk -b 80

rem check installation line
if _#@T253*##### == _PxInS1 goto PHASE0_2

rem set flag partition 0 to the value 61
pxfdisk -o 80 0 61
reboot

:PHASE0_2
rem set flag partition 0 to the value A1
pxfdisk -o 80 0 a1
reboot

:PHASE1
cls
echo PHASE1: Format DOS partition
echo.

rem PXFDISK quick format
pxfdisk -q 80 -f

rem flag that we are done with phase1
```

```
pxfdisk -a 80 0 1
goto PHASE

:PHASE2
cls
echo PHASE2: Connect to server and run NT installation
echo.

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

rem logon to installation server
net logon #@T162*##### /savepw:no /y
rem connect w: to general installation share
net use w: \\#@T160*#####\CLIENTS

rem copy unattend.txt file for our configuration
copy w:\winnt\config\#@T161*#####\unattend.txt a:\

rem patch unattend.txt file with DHCP/BOOTP information
pxutil -a a:\unattend.txt
mkdir c:\temp
copy a:\unattend.txt c:\temp\unattend.txt

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

rem cd to installation directory
w:
cd \winnt\netsetup

rem flag that we are done with PHASE2
pxfdisk -a 80 0 1

rem run Windows NT installation
winnt /b /s:w:\winnt\netsetup /u:c:\temp\unattend.txt

rem !UNREACHED! as WINNT will reboot after install
reboot
```

```
:end
```

A:\NET\PROTOCOL.INI

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 PXE clients.

```
[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$NDISLHP

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

A:\NET\SYSTEM.INI

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

```
[network]
filesharing=no
printsharing=no
autologon=yes
computername=#@chn#####
lanroot=A:\NET
username=install
workgroup=#@Tl65#####
reconnect=no
dospophotkey=N
lmlogon=0
logondomain=
preferredredir=full
autostart=full
maxconnections=8

[network drivers]
netcard=e100b.dos
transport=tcpdrv.dos,nemm.dos
```

```
devdir=A:\NET
LoadRMDrivers=yes
```

Step 6: 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 4.0 Workstation 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 *A:\SAMPLES\PXENT4WS* on the BootManage[®] PXE Toolkit Utility Disk, there is a sample *unattend.txt* file which already contains the options that we use in this configuration example.

```
[OEM_Ads]
Banner = "BootManage PXE Toolkit Setup For Windows NT"

[Unattended]
OemSkipEula = yes
OemPreinstall = yes
NoWaitAfterTextMode = 1
NoWaitAfterGUIMode = 1
FileSystem = #@T201*#####
ExtendOEMPartition = 0
ConfirmHardware = no
NtUpgrade = no
Win31Upgrade = no
TargetPath = WINNT
OverwriteOemFilesOnUpgrade = no
KeyboardLayout = "US"

[UserData]
FullName =      "#@T202*#####"
OrgName =       "#@T163*#####"
ComputerName =  "#@chn*#####"
ProductId =     "#@T164*#####"

[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 = "#@Tl65*#####"

[ProtocolsSection]
TC = TParamSection

[TParamSection]
DHCP = no
IPAddress = #@yip*#####
Subnet = #@smf*#####
;Gateway = #@gw0*#####
;DNSServer = #@dsf*#####
;DNSName = #@chn*#####.#@T40*#####

[ServicesSection]

```

Copy this *unattend.txt* file to the *c:\clients\winnt\config\pxent4ws* directory of your installation server and modify it to suit your needs.

Step 7: Create the Boot Image File

Copy the file *pxdisk.exe* from the BootManage[®] PXE Toolkit Utility Disk to the Windows NT Server:

```
C:\> copy a:\pxdisk.exe %SystemRoot%\system32
```

On the installation server, create the directory *c:\tftpboot* and copy the file *pxboot* from the BootManage[®] PXE Toolkit Utility Disk to this directory:

```

C:\> mkdir c:\tftpboot
C:\> cd tftpboot
C:\tftpboot> copy a:\pxboot

```

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

```

C:\tftpboot> pxdisk -d pxboot.X -F 2880,a:
C:\tftpboot> pxdisk -d pxboot.X -i a:\

```

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

```
C:\tftpboot> pxdisk -d pxboot.X -D
```

Step 8: Install BOOTPD32 and TFTP32

On the installation server, create the directory *C:\ETC*. From the BootManage[®] PXE Toolkit Utility Diskette, copy the files *BOOTPD32.EXE* and *TFTP32.EXE* and also the sample *bootptab* file to the *ETC* directory:

```
C:\tftpboot> cd \  
C:\> mkdir etc  
C:\> cd etc  
C:\etc> copy a:\bootpd32.exe  
C:\etc> copy a:\tftpd32.exe  
C:\etc> copy a:\samples\pxent4ws\bootptab
```

Open the *bootptab* file with a text editor and replace the sample address 00.60.97.a7.ed.ae with the hardware (ethernet or MAC) address of your network adapter.

Here is an example of the *bootptab* file.

```
# global parameters of the local network  
  
bootmanage:\  
:hn:vm=rfc1048:ht=ethernet:ms=1024:\  
# :gw=172.16.0.254:\  
# :ds=172.16.0.10:\  
# DNS domain name (not used in this example)  
# :T140="bootmanage.com":\  
:sm=255.255.0.0:  
  
# common entries for all PXE NT4 Workstation clients  
  
pxe-nt4ws-common:\  
:tc=bootmanage:\  
:hd="c:/tftpboot":bf=pxboot:\  
# PXE Client Class Identifier  
:T60="PXEClient":\  
# Installation server  
:T160="NT4SERVER":\  
# Configuration group (must be 8.3 filename conform)  
:T161="pxent4ws":\  
# Installation dummy user and password  
:T162="instuser instpass":\  
# Windows NT licensing information - Company  
:T163="ACME Corporation":\  
# Windows NT licensing information - License key  
:T164="000-1234567":\  
:T165="BOOTMANAGE":  
  
# entries for NT4 Workstation PXE clients  
  
pxent4ws:\
```



```

:tc=pxe-nt4ws-common:\
:ha=00.a0.c9.42.da.23:ip=172.16.80.1:\
# Hard disk partition size in Mbytes
:T200="400":\
# Filesystem type, either ConvertNTFS (NTFS) or LeaveAlone (FAT16)
:T201="ConvertNTFS":\
# Windows NT licensing information - User name
:T202="BootManage PXE Toolkit User":\
# Enable user initiated reinstallation
:T204="PxDis=03":\
# Enable administrator initiated reinstallation
:T203="PxInS=00,63,a3":

```

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.

Here are some further details to provide a better understanding of our sample configuration:

We use two template entries (bootmanage and pxe-nt4ws-common) and one host entry (pxent4ws). The template entries act as macros and are transferred to other entries by the *tc* option.

Most of the user variables or options are used in the *UNATTEND.TXT* file. See the *UNATTEND.TXT* file to learn how these values are used.

Other options are used in *\INSTALL.BAT*, *\NET\PROTOCOL.INI*, and *\NET\SYSTEM.INI* in the boot image file.

Individual options hold information that is specific to the client. Multiple clients can be installed by adding a similar entry for each client.

Step 9: Start BOOTPD32 and TFTP32

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 Windows NT DHCP server is disabled.

Start BOOTPD32 from the command line:

```
C:\etc> bootpd32 -run -d -d -d -d -i 172.16.0.1/255.255.0.0 -t 0
```

In our case, 172.16.0.1 is the NT server's IP address and 255.255.0.0 is the NT server's subnet mask.

Start TFTP32 from the command line:

```
C:\etc> tftpd32 -cmd -v 2
```

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 10: Install the PXE Client

Turn on the PXE client PC and wait for the PXE code to come-up with its message. The PXE code should download the file *pxboot*, which is the BootManage® PXE Toolkit boot loader. PXBOOT, in turn, should download the boot image file *pxboot.X* and start the automated installation.

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

Administrator Initiated Reinstallation

As the administrator, you can easily schedule a reinstallation of the PXE client which will be executed at the next boot. To do this, simply change the “installation line” digit of the PxDiS keyword in the bootptab from 0 to 1:

```
# before modification (installation line 0)
:T203="PxInS=00,63,a3":

# after modification (installation line 1)
:T203="PxInS=10,63,a3":
```

After the PXE client has been reinstalled, you can again schedule a reinstallation by toggling the installation line back to its original value.

User Initiated Reinstallation

In our example, the user is allowed to request a complete reinstallation of the PXE client at boot time. Driven by the PxDiS=03 keyword, the PXBOOT 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, PXBOOT downloads the boot image (instead of booting from the hard disk) and completely reinstalls Windows NT on the PXE client.

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

```
...
:T202="BootManage PXE Toolkit User":\
# :T204="PxDiS=03":\
:T203="PxInS=00,63,a3":
```

Suggestions

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 4.0 Workstation client machines by simply adding an entry for each client in the *bootptab* file.

Clients With Different Network Cards

You can use the same boot image to install multiple clients that have different network cards. 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 4.0 Server Clients

In addition to installing Windows NT 4.0 Workstation clients, you can also automatically install Windows NT 4.0 Server, Windows 95, and Windows 98 clients. You may want to use an additional tag that points to a subdirectory of the installation server's CLIENTS share and is used as an operating system type selector tag.

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.



Harness the Power of Your WfM PC's

The **P**reboot **eX**ecution **E**nvironment (PXE), hardwired into your Wired for Management (WfM) enabled PC, allows you to boot your PC over the network and remotely configure anything imaginable.

Remotely upgrade the system BIOS, partition the hard drive, install the operating system, restore a corrupted boot sector - all of these operations and more are possible with the PXE capability in your PC.

The only thing missing are the tools that allow you to build the applications to perform this magic.

Remote Boot Means Remote Installation

Using the **BootManage**® PXE Toolkit, you can remotely boot your PXE PC and devise your own configuration and installation scripts to manage tens to thousands of PC's. What's more, the **BootManage**® PXE Toolkit 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 NT, Windows 95 and 98, Windows 3.x, DOS, and even Linux.

The **BootManage**® PXE Toolkit relies on standard, time-tested Internet protocols including DHCP, BOOTP, and TFTP. The Toolkit includes BOOTP and TFTP server programs for a variety of platforms. The Toolkit's versions of these server programs are optimized for greater throughput and scalability, but the user is welcome to use the server programs which may already be included with their system, such as Windows NT's DHCP Server.

Maintain Boot Image Files

The **BootManage**® PXE Toolkit 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 the Toolkit's 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>
<http://www.bootmanage.com>

E-Mail info@bootix.com
info@bootmanage.com