



*PXE Engineering*

Intel Architecture Labs

---

# **PXE Product Development Kit Instructions**

**Version 2.2**

**March 20, 1998**

Intel Corporation assumes no responsibility for errors or omissions in this guide. Nor does Intel make any commitment to update the information contained herein.

- \* Other product and corporate names may be trademarks of other companies and re used only for explanation and to the owners' benefit, without intent to infringe.

Copyright © 1998 Intel Corporation. All rights reserved.  
Intel Corporation, 5200 N.E. Elam Young Parkway, Hillsboro, OR 97124-6497

# Contents

<b>1. LICENSE</b>	<b>1</b>
<b>2. INTRODUCTION</b>	<b>3</b>
2.1 INTENDED AUDIENCE	3
2.2 EQUIPMENT NEEDED	3
2.3 WFM TEST COMPLIANCE: DISCLAIMER	4
2.4 RELATED DOCUMENTS	4
<b>3. PDK INSTALLATION AND SETUP</b>	<b>5</b>
3.1 OVERVIEW	5
3.2 SETTING UP THE SERVER(S)	5
3.3 SETTING UP THE CLIENT WORKSTATION(S)	8
3.4 CONFIGURING THE TEST SERVER SERVICES	8
<b>4. LSA2: INTEL PXE</b>	<b>9</b>
4.1 INSTALLATION	9
4.2 SETUP AND OPERATION	10
4.3 LSA2 STATUS AND ERROR MESSAGES	11
<b>5. PXE SERVICES</b>	<b>15</b>
5.1 OVERVIEW	15
5.2 PXE PROXYDHCP SERVICE	15
5.3 MULTICAST TRIVIAL FTP (MTFTP) SERVICE	15
5.4 PROTOCOL SPECIFICATION	15
<b>6. SERVICES AND REGISTRY CONFIGURATION</b>	<b>21</b>
6.1 OVERVIEW	21
6.2 PXE NT SERVICES CONFIGURATION	21
6.3 LOCATING BSTRAP BOOTSERVER FILES	23
6.4 LOCATING BOOTSERVER FILES OTHER THAN BSTRAP	23
6.5 BOOTSERVER DIRECTORY INCLUDED IN THE PDK	24
6.6 CREATING NEW DOS BOOTFILES	24
6.7 ADDING MENU OPTIONS	25
6.8 REGISTRY ENTRIES	26
<b>7. TESTING PXE</b>	<b>29</b>
7.1 TESTS PROVIDED BY PDK	29
7.2 TEST LOGS	30
7.3 INITIATING THE TESTS	32
<b>8. REVISION HISTORY</b>	<b>33</b>
8.1 PXE BOOT ROM	33
8.2 PXE SERVICES	37
<b>9. BIOS SUPPORT</b>	<b>38</b>
9.1 OVERVIEW	38
9.2 GUID	38
9.3 REMOTE WAKE UP SOURCE	44
9.4 BOOTSTRAPS	45
9.5 MEMORY MANAGEMENT	48
<b>10. THIRD PARTY DESIGN SUPPORT</b>	<b>50</b>
<b>11. LSA2 OPERATION AND TROUBLESHOOTING FAQs</b>	<b>51</b>

---

# 1. License

---

## INTEL END USER SOFTWARE LICENSE AGREEMENT

**IMPORTANT: READ BEFORE COPYING, INSTALLING OR USING.**

**BY USING THIS SOFTWARE, YOU ARE AGREEING TO BE BOUND BY THE TERMS OF THIS AGREEMENT. DO NOT USE THIS SOFTWARE UNTIL YOU HAVE CAREFULLY READ AND AGREED TO THE FOLLOWING TERMS AND CONDITIONS. IF YOU DO NOT AGREE TO THE TERMS OF THIS AGREEMENT, PROMPTLY RETURN THE SOFTWARE FILES AND ANY ACCOMPANYING ITEMS AND DO NOT COPY, INSTALL OR USE THE SOFTWARE.**

**IF YOU USE THIS SOFTWARE, YOU WILL BE BOUND BY THE TERMS OF THIS AGREEMENT AS FOLLOWS:**

**LICENSE:** Intel Corporation ("Intel") grants you a non-exclusive, royalty-free, copyright license to the enclosed software program and materials in object code format, ("the Software") subject to the terms herein. No other license is granted except as expressly provided herein. You will not use, copy, modify, rent, distribute, sell nor transfer the Software nor any portion thereof except as expressly provided in this Agreement.

**YOU MAY:**

1. Install a copy of the Software on one or more computers for internal use only and for the sole purpose of testing PC systems for adherence to the Wired for Management Baseline Specification;
2. Copy the Software solely for backup or archival purposes and for internal transfer to another user in your company who agrees to the terms of this Agreement.

**YOU WILL NOT:**

1. Sublicense the Software;
2. Reverse engineer, decompile or disassemble the Software;
3. Copy the Software, in whole or in part, except as specifically provided for in this Agreement;
4. Distribute or transfer the Software outside your company.

**TRANSFER:** You may transfer a copy of the Software to another person in your company (i.e., the company which was originally granted access to this Software by Intel) on the condition that a copy of this Agreement accompanies the transferred copy and the receiving party agrees to the terms of this Agreement.

**OWNERSHIP AND COPYRIGHT OF SOFTWARE:** Title to the Software and all copies thereof remain with Intel or its vendors. The Software is copyrighted and is protected by United States and international copyright laws. You will not remove the copyright notice from the Software. You agree to prevent any unauthorized copying of the Software. Except as otherwise expressly provided, Intel does not grant any express or implied right to You under Intel patents, copyrights, trademarks or trade secret information.

**WARRANTY DISCLAIMER:** The Software is provided "AS IS" without warranty of any kind.

**INTEL MAKES NO WARRANTIES OF ANY KIND, WHETHER EXPRESS, IMPLIED OR STATUTORY, AND EXPRESSLY DISCLAIMS WARRANTIES OF MERCHANTABILITY, NONINFRINGEMENT OF INTELLECTUAL PROPERTY RIGHTS, AND FITNESS FOR ANY PARTICULAR PURPOSE. INTEL WILL NOT PROVIDE ANY SUPPORT, ASSISTANCE, INSTALLATION, TRAINING OR OTHER SERVICES. INTEL WILL NOT PROVIDE ANY UPDATES, ENHANCEMENTS OR EXTENSIONS.**

Intel further does not warrant or assume any responsibility for the accuracy or completeness of the information, text, graphics, links or other items contained in the software. Intel may make changes to the Software or to any products described therein at any time without notice.

**LIMITATION OF LIABILITY: IN NO EVENT SHALL INTEL OR ITS VENDORS BE LIABLE FOR ANY INDIRECT, INCIDENTAL, SPECULATIVE, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES OF ANY KIND, INCLUDING BUT NOT LIMITED TO LOSS OF PROFITS, LOSS OF USE, LOSS OF DATA OR INTERRUPTION OF BUSINESS, WHETHER UNDER THIS AGREEMENT OR OTHERWISE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.**

**TERMINATION OF THIS LICENSE:** Intel may terminate this Agreement at any time if you are in breach of any of its terms and conditions. Upon termination, you will immediately destroy the Software or return all copies of the Software and documentation to Intel at Intel's sole discretion.

**U.S. GOVERNMENT RESTRICTED RIGHTS:** The Software and documentation were developed at private expense and are provided with RESTRICTED RIGHTS. Use, duplication or disclosure by the Government is subject to restrictions as set forth in FAR52.227-14 and DFAR252.227-7013 et seq. or its successor. Use of the Software by the Government constitutes acknowledgment of Intel's proprietary right to them.

**EXPORT LAWS:** You will not export/re-export the Software in violation of the laws, regulations, orders or other restrictions of the U.S. Export Administration Regulations.

**APPLICABLE LAW:** This Agreement is governed by the laws of the United States and the State of Delaware, including patent and copyright laws, without regard to its conflict of law provisions.

## 2. Introduction

This PDK includes a test environment and a series of tests to provide functional and stress testing of PXE Boot ROMs and the PXE APIs. It also includes services necessary for the test server.

### 2.1 Intended Audience

This PDK is intended for

- Developer's implementing PXE
  - Boot ROM OEMs
  - Network Interface OEMs
  - Possibly PC System OEMs
- OEMs who provide non-PXE elements which WfM requires for PXE support
  - PC System OEMs
  - BIOS OEMs
- OEMs who are developing "Bootservers" to inter-operate with PXE.

### 2.2 Equipment Needed

The following table provides a guide to what a particular OEM would be testing for compliance, and what other compliant elements would be required to successfully complete the tests. For example, a "Network Interface OEM" would be testing their implementation of PXE for their NIC. In addition to their "test unit" (the PXE enabled NIC), they would also need a WfM compliant PC system in which to install their NIC for the tests.

OEMs ↓	Equipment Needed					
	WfM compliant PC HW, BIOS, and PXE	WfM compliant PC HW and BIOS	WfM compliant PXE enabled NIC	WfM compliant PC HW	WfM compliant BIOS	WfM PXE PDK test setup
<b>BIOS OEMs</b>	NA	NA	Needed in addition to test unit	Needed in addition to test unit	<b>Test Unit</b>	Needed in addition to test unit
<b>Boot ROM OEMs</b>	NA	Needed in addition to test unit	<b>Test Unit</b>	NA	NA	Needed in addition to test unit
<b>Network Interface OEMs</b>	NA	Needed in addition to test unit	<b>Test Unit</b>	NA	NA	Needed in addition to test unit
<b>PC System OEMs</b>	<b>Test Unit</b> (if network interface in PC)	<b>Test Unit</b> (if no network interface in PC)	Needed in addition to test unit (if no network interface in PC)	NA	NA	Needed in addition to test unit
<b>Bootservers OEMs</b>	Needed (if network interface in PC)	Needed (if no network interface in PC)	Needed (if no network interface in PC)	NA	NA	Needed

In addition to the equipment listed above, at least one test server and a test network are required. To insure proper operation in a wide variety of environments it is strongly recommended that a second round of testing is done that includes routers in the test network.

## 2.3 WfM Test Compliance: Disclaimer

Successfully executing the tests in this PDK neither guarantees overall Wired for Management test compliance nor the functional integrity of the unit under test. While the tests do provide a good indication of PXE WfM compliance, they do not define compliance. Definitive test requirements are found in "Wired for Management Baseline Testing". For the latest version of "Wired for Management Baseline Testing" please go to:

<ftp://download.intel.com/ial/wfm/wfmpro.pdf>

## 2.4 Related Documents

### 2.4.1 Wired for Management

**Wired for Management (WfM) Baseline Version 1.1a.**

<http://www.intel.com/managedpc/spec.htm>

<http://developer.intel.com/ial/wfm/design/index.htm>

**PXE PDK**

<http://www.intel.com/ial/wfm/tools/pxe/index.htm>

**PXE Powerpoint Presentation**

<http://www.intel.com/ial/wfm/class/index.htm>

### 2.4.2 BIOS Specifications

**BIOS Boot Specification**

Version 1.01

January 11, 1996

<http://www.phoenix.com/techs/specs.html>

**System Management BIOS Reference Specification**

Version 2.1

June 16, 1997

<http://www.phoenix.com/techs/smbios/htframe.html>

**POST Memory Manager Specification**

Version 1.01

November 21, 1997

**Plug and Play BIOS Specification**

Version 1.0A

May 5, 1994

<http://www.phoenix.com/techs/specs.html>

### 2.4.3 Other PC System Documents

**PC 98 System Design Guide, v1.0**

<http://developer.intel.com/design/pc98/>

**Network PC Design Guidelines, v1.0b**

<http://developer.intel.com/design/netpc/netovr.htm>

---

## 3. PDK Installation and Setup

---

### 3.1 Overview

This section explains how to install the PDK and create a PXE test environment.

To install the PDK you will set up one or more Windows NT servers, execute the setup program, and create image (boot) files.

In addition, you will set up at least one client PC containing a PXE option ROM. (Although multicast TFTP can be used with a single client, to observe the full effect of master/slave multicast TFTP you need multiple clients.) The PDK contains binaries and programming utilities to install PXE onto an Intel EPRO100B NIC for the client.

**Note:** The test environment requires the use of a DHCP service. You should set up a test network apart from your main network to avoid conflicting with production DHCP servers.

### 3.2 Setting up the Server(s)

The PXE test environment includes one or more Windows NT servers running Microsoft DHCP, PXE ProxyDHCP, and the PXE MTFTPD services. (While it is not absolutely required, we suggest you use a dedicated DHCP server and install ProxyDHCP and MTFTPD on a second server.)

The installed services provide the following capabilities:

- The DHCP service is responsible for allocating dynamic IP addresses.
- ProxyDHCP service supplements the DHCP service in two ways:
  1. by providing DHCP options that cannot be programmed into the DHCP service
  2. by handling the discovery process used by the booting client to locate a bootserver to provide boot files.
- MTFTPD service provides both multicast and unicast TFTP services for downloading the boot files..

#### 3.2.1 Setting up NT Server

- Install Windows NT Server (3.51 or later) on a one or more PCs.
  - Install TCP/IP
  - Install the Microsoft DHCP Server as part of the Windows NT installation on one server.
    - Assign a static IP address to the server
    - Create a valid scope on the server.
    - If you are using the same host to provide the DHCP service and to be the test server, you must add the DHCP Class Identifier option, tag value 60, to the DHCP service. This option must be set to "PXEClient". Do not add this tag if the DHCP service is provided on a host separate from the test server.
  - You will probably want to test that the network and DHCP are working properly by bringing up a DHCP client on the network (such as an NT Workstation with DHCP enabled).
  - Ensure the "GUEST" account is enabled.
  - Copy the self-extracting executable PXEPDK.EXE to a temporary ("`<temp>`" in the following listing) directory on the test server.
- ➔ Extract the setup files by running PXEPDK.EXE.





	--ifshlp.sys	-Source files for APITest.1
	--lmhosts	-Source files for APITest.1
	--ndis.dos	-Source files for APITest.1
	--ndishlp.sys	-Source files for APITest.1
	--nemm.dos	-Source files for APITest.1
	--net.exe	-Source files for APITest.1
	--net.msg	-Source files for APITest.1
	--netbind.com	-Source files for APITest.1
	--netstart.bat	-Source files for APITest.1
	--networks	-Source files for APITest.1
	--nmtsr.exe	-Source files for APITest.1
	--protman.dos	-Source files for APITest.1
	--protman.exe	-Source files for APITest.1
	--protocol	-Source files for APITest.1
	--protocol.ini	-Source files for APITest.1
	--system.ini	-Source files for APITest.1
	--tcpdrv.dos	-Source files for APITest.1
	--tcptsr.exe	-Source files for APITest.1
	--tcputils.ini	-Source files for APITest.1
	--tinyrfc.exe	-Source files for APITest.1
	--umb.com	-Source files for APITest.1
	--wfwsys.cfg	-Source files for APITest.1
	\---DOSUNDI	-Directory for DOSUNDI Service bootfiles
	--dosundi.0	-Layer 0 bootfile for DOSUNDI
	--mkdos.bat	-Batch file for creating DOSUNDI.1 bootfile
	\---MKIMAGE	-Source files for DOSUNDI.1
	--autoexec.bat	-Source files for DOSUNDI.1
	--config.sys	-Source files for DOSUNDI.1
	\---NET	-Source files for DOSUNDI.1
	--csagent2.exe	-Source files for DOSUNDI.1
	--emsbfr.exe	-Source files for DOSUNDI.1
	--ifshlp.sys	-Source files for DOSUNDI.1
	--lmhosts	-Source files for DOSUNDI.1
	--ndis.dos	-Source files for DOSUNDI.1
	--ndishlp.sys	-Source files for DOSUNDI.1
	--nemm.dos	-Source files for DOSUNDI.1
	--net.exe	-Source files for DOSUNDI.1
	--net.msg	-Source files for DOSUNDI.1
	--netbind.com	-Source files for DOSUNDI.1
	--netstart.bat	-Source files for DOSUNDI.1
	--networks	-Source files for DOSUNDI.1
	--nmtsr.exe	-Source files for DOSUNDI.1
	--protman.dos	-Source files for DOSUNDI.1
	--protman.exe	-Source files for DOSUNDI.1
	--protocol	-Source files for DOSUNDI.1
	--protocol.ini	-Source files for DOSUNDI.1
	--system.ini	-Source files for DOSUNDI.1
	--tcpdrv.dos	-Source files for DOSUNDI.1
	--tcptsr.exe	-Source files for DOSUNDI.1
	--tcputils.ini	-Source files for DOSUNDI.1
	--tinyrfc.exe	-Source files for DOSUNDI.1
	--umb.com	-Source files for DOSUNDI.1
	--wfwsys.cfg	-Source files for DOSUNDI.1
	\---TESTLOG	-Directory for test logs
	--file1	-Test files for NDIS testing
	--file2	-Test files for NDIS testing
	--file3	-Test files for NDIS testing
	--file4	-Test files for NDIS testing
	--file5	-Test files for NDIS testing
	--file6	-Test files for NDIS testing
	--file7	-Test files for NDIS testing
	--file8	-Test files for NDIS testing
	--file9	-Test files for NDIS testing
	--file10	-Test files for NDIS testing
	--file11	-Test files for NDIS testing
	--testfile.exe	-Executable to create test files FILE1-11

### 3.2.3 Create the boot files: APITEST.1 and DOSUNDI.1

Now you need to create APITEST.1 and DOSUNDI.1 boot files (these are not provided with the PDK). To create these files:

- ➔ Create two 1.44MB MSDOS 6.22 boot floppies using FORMAT /S or SYS.COM from a DOS machine.
- ➔ Label one diskette "APITest" and the other diskette "DOSUNDI".
- ➔ On the APITEST disk add the following DOS files in the root directory:
  - HIMEM.SYS
  - RAMDRIVE.SYS
  - MORE.EXE
  - FC.EXE.
- ➔ On the DOSUNDI disk add the following DOS files in the root directory:
  - MORE.EXE.
- ➔ To create the APITest.1 bootfile, insert the diskette labeled "APITest" in the drive and run the batch file :  
*<install directory>\PDK\system\images\x86pc\undi\APITest\mktest.bat*
- ➔ To create the DOSUNDI.1 bootfile, insert the diskette labeled "DOSUNDI" in the drive and run the batch file :  
*<install directory>\PDK\system\images\x86pc\undi\DOSUNDI\mkdos.bat*

When the mktest.bat and mkdos.bat files execute they will copy additional files to the diskette. Then a sector by sector image of the diskette is read into a file (either APITest.1 or DOSUNDI.1) and the file is placed into the appropriate directory (either APITest or DOSUNDI). This creates the boot files that are downloaded and booted by the client

If you want to change the contents of the images, modify the files in the APITest or DOSUNDI directories and run the batch files again. As an example, you may want to automatically log in to a server and run a test suite of your own by adding commands to the AUTOEXEC.BAT file in the DOSUNDI subdirectory.

**Caution!!:** File size, order of operation, resulting memory footprint, etc are all critical to correct operation. Be aware of these restrictions and change image content with care and only if necessary.

## 3.3 Setting up the Client Workstation(s)

Please see section 4 below.

## 3.4 Configuring the Test Server Services

Please see sections 5 and 6 below

---

## 4. LSA2: Intel PXE

---

LSA2 (LANDesk Service Agent) is Intel's implementation of PXE.

Included in this PDK is a binary (e100b.nic) and a flash programming utility (FUTIL.EXE) to allow installing LSA onto an Intel EtherExpress Pro/100B (E100B) network interface card (NIC).

Also included is a binary (E100B.LD ) that may be integrated into the BIOS of platforms which contain the Intel EtherExpress Pro/100B (E100B) network interface on the motherboard. Use of this binary requires building the BIOS with the binary and then updating the BIOS of the system under test. This operation will normally only be done by the BIOS engineering department of the motherboard manufacturer.

These binaries are provided to allow creating a known good PXE NIC for verifying the test environment or for use in testing the BIOS or a non\_lan\_on\_motherboard PC.

Note, per the PDK license you may not redistribute these binaries. They are provided for test purposes only.

### 4.1 Installation

You can program the PXE Boot PROM code, e100b.nic, into the FLASH memory on an E100B card using the command: **futil e100b.nic**

You can incorporate the BIOS-based PXE Boot PROM code (E100B.LD) into your client two ways. The first way is to program the image into the FLASH memory on an E100B card using the command: **futil e100b.ld**

#### FUTIL.EXE documentation.

(This following is also displayed by typing: FUTIL /?)

```
[FUTIL ver 2.16] - Intel PCI NIC FLASH Update Utility
Copyright (C) 1995,1996 Intel Corporation. All rights reserved.
```

```
Usage: FUTIL options command/filename
```

```
Options:
```

```
-A20=#
    This option forces the use of a specific A20 gate service.
    If this option is not specified, the A20 gate service will
    be determined through software.
    A20 services: 1=XMS, 2=Int15, 3=Port92

-bus=# -dev=# -func=#
    Specify PCI bus/device/function numbers of Intel PCI
    network adapter to be programmed. If not specified,
    -bus, -dev and -func will default to all detected
    Intel PCI network adapters.
```

```
Commands:
```

```
-erase
    Erases the contents of the FLASH memory.
```

```
Filename
    Program a raw binary image into the FLASH memory.
```

```
Exit codes:
```

```
0 := All FLASH operations completed successfully.
1 := Bad command line parameter.
2 := No supported Intel PCI network adapters detected.
3 := No supported FLASH devices detected.
4 := FLASH operation failed
5 := Image file is missing or corrupted.
```

The second way is to incorporate the BIOS-based PXE Boot PROM code (E100B.LD) into your client is to include the BIOS-based image (E100B.LD) as an option ROM image in your system BIOS image (as you would include a video option ROM image). You must use your own BIOS programming utility to program this BIOS image into the motherboard's FLASH memory.

## 4.2 Setup and Operation

### 4.2.1 CMOS Setup

After installing LSA onto the NIC or motherboard you will need to insure the NIC is the primary boot device. The mechanism for doing this will vary from system to system, but generally you will need to enter setup during POST, locate the boot order setting, and make the NIC the first boot device.

The ability to make NIC the first boot device depends on underlying BIOS support. WfM compliance requires BIOS Boot Spec support. If this is present then setting the boot order should not be a problem. If you are testing without a compliant platform you may still be able to reorder the NIC to the first boot device in POST setup. Failing this, LSA provides a proprietary mechanism that may allow setting the NIC to first in boot order. This mechanism is described below.

#### **<Ctrl+T>      *Bootstrap interrupt 18h/19h toggle***

When a NIC based LSA2 is being initialized in a BIOS that does not support Plug and Play BIOS Boot Specification function 60h (Get Version and Installation Check), the user needs to decide whether to use bootstrap interrupt 18h (in some newer BIOS's that support network boot devices) or 19h (in legacy Bioses).

During option ROM initialization LSA2 will display one of these messages:

- PXE-M04: Hooking bootstrap interrupt 18h
- or PXE-M04: Hooking bootstrap interrupt 19h

While this message is being displayed, the user can press **<Ctrl+T>** to toggle between 18h and 19h. Which interrupt is to be used will be stored in bit 0Eh of the Vendor Extension word (3Bh) in the EEPROM on the NIC.

The LSA2 boot ROM may not work properly if the bootstrap interrupt is set incorrectly. Possible problems are:

1. LSA2 will not boot. You will see the PXE-M04 message, but will not see the LSA2 copyright message. This will happen if interrupt 18h is used in a legacy BIOS that has a bootable floppy or hard disk installed.
2. LSA2 always boots first, even though 'Network' is not the first item in the BIOS startup list. This can happen if interrupt 19h is used in a newer interrupt 18h BIOS.

#### **<Esc> or <Ctrl+C>      *Cancel network boot***

Once the NIC with LSA2 has been selected as the network boot device, the user can cancel the network boot at any time by pressing **<Esc>** or **<Ctrl+C>**. When one of these keys is pressed, LSA2 will reset the NIC, remove itself from RAM and return control to the BIOS.

### 4.2.2 Time-outs and Retries

#### 4.2.2.1 *DHCP*

Four retries using 4, 8, 16 then 32 seconds, for a total of 60 seconds.

#### 4.2.2.2 *BINL*

Three retries using 4, 8 then 16 seconds, for a total of 28 seconds.

#### 4.2.2.3 TFTP Open

Six retries, each using 4 seconds.  
The first three retries use IP port specified by caller.  
The last three retries use default TFTP IP port (69).

#### 4.2.2.4 MTFTP Open

Initial time-out sent by DHCP/Proxy server (defaults to 4 seconds).  
Six retries, each using initial time-out.  
After six retries, boot ROM will try TFTP open.

#### 4.2.2.5 TFTP/MTFTP Read

Initial time-out delay is .25 seconds.  
Up to eight packets can be lost in one session.  
Each lost packet will cause a time-out and increase the time-out delay by 1 second.

### 4.3 LSA2 Status and Error Messages

All LSA2 message are prefixed with 'PXE-'. This prefix is used by to identify messages from PXE (Preboot eXecution Environment) compatible boot devices.

LSA2 displays five types of messages:

1. Version and copyright messages.
2. Network communication status:
3. TFTP Messages: **PXE-Txx:**
4. Error messages: **PXE-Exx:**
5. Status messages: **PXE-Mxx:**

#### 4.3.1 Version and Copyright Messages

*Intel LANDesk (R) Service Agent, version x.xxx Copyright (C) 1997 Intel Corporation. All rights reserved.*

This message is displayed once by each copy of LSA2 that boots.

#### 4.3.2 Network Communication Status

DHCP...

Is displayed when DHCP Discover packet is sent out. Each '.' represents a transmitted packet. LSA2 will retry four times, the first timeout is four seconds, each retry will double the timeout value. LSA2 will wait a maximum of one minute before giving up on the DHCP/BOOTP server.

MTFTP...

Displayed during MTFTP Open requests. Each '.' represents a transmitted packet. LSA2 will retry MTFTP a total of eight times, four with the supplied server and client ports and then four more with the default server and client ports. Each timeout is four seconds. If MTFTP fails to open, LSA2 will switch to TFTP.

TFTP...

Displayed during TFTP Open requests. TFTP also retries a total of eight times, like MTFTP.

#### 4.3.3 TFTP Messages

PXE-T01: File not found  
Requested file was not found on the TFTP server.

PXE-T02: Access violation  
TFTP server does not have access writes to requested file.

PXE-T08: No multicast address  
Requested file does not have a multicast address assigned to it.

#### 4.3.4 LSA2 Error Messages

- PXE-E00: There is no free memory between 480K and 640K.  
LSA2 requires a 64 Kbyte block of base memory. It looks for a block of zero-filled memory between 480K and 640K. If it cannot find one, it will not boot.
- PXE-E01: PCI Vendor and Device IDs do not match!  
This message should never be seen in a production BIOS. The only time this message can be displayed is if the PCI BIOS tries to initialize the LSA2 ROM with invalid PCI bus/device/function numbers in AX. Refer to the PCI BIOS Specification for more information about the interaction between the PCI BIOS and PCI option ROMs.
- PXE-E03: Extended memory copy of LSA2 has been corrupted.  
This message is only displayed by LoM (LAN-on-Motherboard) versions of LSA2. During option ROM initialization, LSA2 is copied into upper memory (between C8000h and F0000h) by the BIOS. LSA2 then allocates a block of extended memory and copies itself into the allocated memory. LSA2 will then reduce the amount of used upper memory to 2 Kbytes (it started out using about 29 Kbytes). The image of LSA2 in extended memory can become corrupted if the BIOS, or other option ROMs, use the extended memory allocated by LSA2.  
  
LSA2 uses two methods to allocate extended memory. If the BIOS supports POST Memory Manager, LSA2 will use it. If the BIOS does not support POST Memory Manager, LSA2 will chain BIOS interrupt 15h and reduce the amount of extended memory reported by service AH=88h (Get Extended Memory Size).
- PXE-E04: Error reading PCI configuration space.  
This message is displayed on a PCI NIC (not LOM) when the PCI configuration space cannot be read using the PCI BIOS services. This message can be caused by a failing PCI controller on the NIC.
- PXE-E05: EEPROM checksum error.  
The message is displayed on when the EEPROM checksum is not valid. This message can occur if the system is turned off when the NIC EEPROM is being updated.
- PXE-E10: ARP canceled by keystroke.  
This message is displayed when <Esc> or <Ctrl+C> is pressed during ARP negotiation.
- PXE-E11: ARP timeout.  
This message is displayed when LSA2 cannot complete ARP negotiation.
- PXE-E20: BIOS extended memory copy failed.  
The BIOS returned an error during an extended memory copy. This should never happen.
- PXE-E31: TFTP open canceled by keystroke.  
This message is displayed when <Esc> or <Ctrl+C> is pressed during TFTP open negotiation.
- PXE-E32: TFTP open timeout. Make sure TFTP server is running.  
This message is displayed if TFTP open negotiation cannot be completed.
- PXE-E34: TFTP read canceled by keystroke.  
This message is displayed when <Esc> or <Ctrl+C> is pressed during the TFTP read process.
- PXE-E35: TFTP read timeout. Make sure TFTP server is running.  
This message is displayed if the TFTP connection is lost.
- PXE-E38: TFTP cannot open connection. Check network cable.  
This message is displayed if the network boot PROM could not transmit the TFTP open request packet.

- PXE-E39: TFTP cannot read from connection. Check network cable.  
This message is displayed if the network boot PROM could not read a TFTP packet or transmit an ACK packet.
- PXE-E3B: TFTP error - file not found.  
The requested file is not on the server. Check the pathname of the requested file. Make sure it is typed in correctly.
- PXE-E3C: TFTP error - access violation.  
The TFTP service does not have permission to read the requested file. Check the permissions on the file and the directory. Make sure the TFTP service has permission to read the requested file.
- PXE-E40: BOOTP canceled by keystroke.  
BOOTP negotiation has been canceled by <Esc> or <Ctrl+C>.
- PXE-E43: BOOTP cannot find bootfile name.  
LSA2 will only display this message if there is no ProxyDHCP reply and the first BOOTP reply does not have the bootfile field filled in.
- PXE-E50: DHCP canceled by keystroke.  
DHCP negotiation has been canceled by <Esc> or <Ctrl+C>.
- PXE-E51: DHCP timeout. Make sure DHCP server is running.  
This message is displayed if DHCP negotiation cannot be completed.
- PXE-E53: DHCP cannot find bootfile name.  
LSA2 will only display this message if there is no ProxyDHCP replay and the first DHCP reply does not have the bootfile field filled in.
- PXE-E62: Network media (wire) test failure, check cable.  
This message is displayed when the network connection is first initialized, and there is no active network cable connected.
- PXE-E91: MTFTP open canceled by keystroke.  
This message is displayed when <Esc> or <Ctrl+C> is pressed during MTFTP open negotiation.
- PXE-E92: MTFTP open timeout.  
This message is displayed if MTFTP open negotiation cannot be completed.
- PXE-E93: MTFTP unknown opcode.  
This message should never be seen. If this message is displayed, the TFTP server sent a packet with an unsupported TFTP opcode.
- PXE-E94: MTFTP read canceled by keystroke.  
This message is displayed when <Esc> or <Ctrl+C> is pressed during the MTFTP read process.
- PXE-E95: MTFTP read timeout.  
This message is displayed if the MTFTP connection is lost.
- PXE-E98: MTFTP cannot open connection.  
This message is displayed if the network boot PROM could not transmit the MTFTP open request packet.
- PXE-E99: MTFTP cannot read from connection.  
This message is displayed if the network boot PROM could not read a MTFTP packet or transmit an ACK packet.
- PXE-E9A: MTFTP too many packages. Reboot and try again.  
This message should never be displayed. This message is caused by increasing the size of a file on the TFTP server, while there is an MTFTP slave that is waiting to re-open the changed file.



PXE-E9B: MTFTP error - file not found.  
The requested file is not on the server. Check the pathname of the requested file. Make sure it is typed in correctly.

PXE-E9C: MTFTP error - access violation.  
The MTFTP service does not have permission to read the requested file. Check the permissions on the file and the directory. Make sure the MTFTP service has permission to read the requested file.

#### **4.3.5 LSA2 Status Messages**

PXE-M01: No extended memory services. PROM image will stay in UMB.  
This message should never be seen. LSA2 developers should be notified.

PXE-M04: Hooking bootstrap interrupt 18h.  
While this message is displayed, the user can toggle between bootstrap interrupts 18h and 19h, by pressing <Ctrl+T>. This feature is only available on LSA2 NICs.

PXE-M0F: Exiting LANDesk Service Agent.  
This message is seen when a network boot is not performed. This is the last message displayed by LSA2 before control is returned to the system BIOS.

PXE-M60: NIC driver cannot initialize NIC for multicast.  
This message is displayed if the NIC driver cannot initialize the NIC for multicast. This message is displayed if the NIC hardware does not support multicast addresses or the NIC driver is not supporting multicast addresses in software.

---

## 5. PXE Services

---

### 5.1 Overview

The PDK includes two services: ProxyDHCP and Multicast Trivial FTP (MTFTPD).

ProxyDHCP responds to DHCP packets sent by the client. ProxyDHCP provides the client with the information needed to download bootfiles. This information is provided in the form of DHCP options.

MTFTPD provides multicast and unicast TFTP capability. The client uses this service to download bootfiles.

### 5.2 PXE ProxyDHCP Service

Depending on how it is configured, ProxyDHCP responds to:

- DHCP Discover packets unicast, multicast, or broadcast by the client to port 67
- Or to DHCP Discover packets unicast or multicast to port 4011
- Or to DHCP Request packets unicast to port 4011

Flags stored in the registry determine the listening ports and transmission methods used by ProxyDHCP and the client. These are described in the section on configuration.

In addition to those DHCP options required by the DHCP protocol, the packet ProxyDHCP returns to the client contains three categories of options:

1. DHCP options the client can use to identify the server responding,
2. DHCP options the client needs to download the bootfile
3. DHCP options the bootfile will use once it begins executing.

The client specifies the service it is trying to discover through the vendor option PXE\_BOOT\_ITEM (value 71). This option contains a service number and the number of the bootfile the client is requesting. The BSTRAP service provides the initial bootfile BSTRAP.0.

If the client packet received by ProxyDHCP does not contain the PXE\_BOOT\_ITEM option it is assumed the client is requesting the BSTRAP.0 bootfile.

These options are described in the section 5.4 below.

### 5.3 Multicast Trivial FTP (MTFTP) Service

MTFTPD provides multicast and unicast file transfers. MTFTPD is based on the Trivial File Transfer Protocol.

When MTFTPD receives a request for a multicast file transfer that is not currently in progress, it:

- locates the multicast IP transfer address for the file in the registry;
- appends the directory from the registry value BASE\_DIR to the relative path requested, and
- then transmits the file on the multicast IP address.

### 5.4 Protocol Specification

#### 5.4.1 Relationship to DHCP

The client may receive options from both a DHCP and a ProxyDHCP service. The client will always give preference to options received from the DHCP server over ProxyDHCP responses.

#### 5.4.2 Expanded DHCP Options

The following table contains all of the expanded DHCP options used in the PXE Bootserver protocol.

PXE DHCP Options (Full List)					
Tag Name	Tag Number	Length Field	Type(1)	Data Field	
Client machine identifier (UUID/GUID). DHCP_PLATFORMID	97	17	0=UUID	Type 0 = UUID(16)	Required <b>Note #1</b>
Client network interface identifier. DHCP_NICIF	94	3-13	1 = UNDI 2 = PCI  3 = PnP	Type 1 = Major ver(1), Minor ver(1) Type 2 = Vendor ID(2), Device ID(2), Class Code(3), Rev(1), Sub- Vendor ID(2), Sub-Device ID(2) Type 3 = EISA Device ID(4), Class Code(3) <i>Byte order is defined by PCI and PnP specs.</i>	Required
Client system architecture. DHCP_SYSARCH	93	2	0 = IA x86 PC(1) 1 = NEC/PC98(1) 2 = etc.(1)		Required
Class Identifier DHCP_CLASS	60	9	<b>“PXEClient”</b> (This field must not be null terminated.)		Required
DHCP_VENDOR	43	Varies	<i>Encapsulated options below. Multiple DHCP_VENDOR options can be used.</i>		Required
PXE_MTFTP_IP	1	4	<b>Multicast IP Address</b> Multicast IP address of bootfile.		<b>Note #2</b>
PXE_MTFTP_CPORT	2	2	<b>UDP Port Number INTEL ORDER</b> UDP port that client should monitor for MTFTP responses.		
PXE_MTFTP_SPORT	3	2	<b>UDP Port Number INTEL ORDER</b> UDP port that MTFTP servers are using to listen for MTFTP open requests.		
PXE_MTFTP_TMOUT	4	1	<b>Open Timeout</b> Number of seconds a client will listen for activity before trying to start a new MTFTP transfer.		
PXE_MTFTP_DELAY	5	1	<b>Reopen Timeout</b> Number of seconds a client will listen before trying to restart a MTFTP transfer.		
PXE_DISCOVERY_CONTROL	6	1	<b>(Bit field. Bit 0 is least significant bit.)</b> bit 0 = If set, disable broadcast discovery. bit 1 = If set, disable multicast discovery. bit 2 = If set, only use/accept servers in PXE_BOOT_SERVER. bit 3-7 = Must be 0. If this tag is not supplied, all bits assumed to be 0.		<b>Note #3</b>
Discovery_MCast_Addr	7	4	<b>Multicast IP-addr</b> Bootserver discovery multicast IP address. Bootservers capable of multicast discovery must listen on this multicast address.		

PXE DHCP Options (Full List)					
Tag Name	Tag Number	Length Field	Type(2)	Data Field	
PXE_BOOT_SERVERS	8	varies	<b>Bootserver type(2)</b> Type 0 = PXE bootstrap server Type 1 = Microsoft Windows NT bootserver Type 2 = Intel LCM bootserver Type 3 = DOS/UNDI bootserver Type 4 through 32767 = reserved Type 32768 through 65534 = vendor use Type 65535 = PXE API Test server	<b>IPcnt(1), IP-addr-list(IPcnt*4), type(2)....</b> IPcnt cannot be 0  Bootservers must not respond to discovery requests of types that they do not support.	<b>Note #3</b>
PXE_BOOT_MENU	9	varies	<b>Bootserver type(2)</b> Type 0 = reserved for local boot	<b>desclen(1), "description", Bootserver type(2)....</b>  Boot "order" is implicit in the menu order.  "desclen" is length of "description" "desclen" cannot be 0.	<b>Note #4</b>
PXE_MENU_PROMPT	10	varies	<b>timeout(1), "prompt"</b> The timeout is the number of seconds to wait before auto-selecting the first boot menu item. The prompt is displayed followed by the number of seconds remaining before the first item in the boot menu is auto-selected. If <F8> is pressed, the menu will be displayed. If this option is not provided, the menu must be displayed without prompt and timeout. If the timeout is 0, the first item in the menu must only be auto-selected. If the timeout is 255, the menu and prompt must be displayed without auto-selecting or timeout.		
<i>Loader Options</i>	<i>64-127</i>	<i>varies</i>	<i>(bootserver specific)</i>		
PXE_BOOT_ITEM	71	3	<b>Bootserver type(2), layer(1)</b> Layer 0 = First file of selected bootserver type. If this tag is missing, type 0 and layer 0 is assumed		<b>Note #5</b>

PXE DHCP Options (from Client WS to ProxyDHCP)					
Tag Name	Tag Number	Length Field	Type(1)	Data Field	
Client machine identifier (UUID/GUID). DHCP_PLATFORMID	97	17	0=UUID	Type 0 = UUID(16)	Required <b>Note #1</b>
Client network interface identifier. DHCP_NICIF	94	3-13	1 = UNDI 2 = PCI  3 = PnP	Type 1 = Major ver(1), Minor ver(1) Type 2 = Vendor ID(2), Device ID(2), Class Code(3), Rev(1), Sub-Vendor ID(2), Sub-Device ID(2) Type 3 = EISA Device ID(4), Class Code(3) <i>Byte order is defined by PCI and PnP specs.</i>	Required
Client system architecture. DHCP_SYSARCH	93	2	0 = IA x86 PC(1) 1 = NEC/PC98(1) 2 = etc.(1)		Required
Class Identifier DHCP_CLASS	60	9	<b>“PXEClient”</b> (This field must not be null terminated.)		Required
DHCP_VENDOR	43	Varies	<i>Encapsulated options below. Multiple DHCP_VENDOR options can be used.</i>		Required
<i>Loader Options</i>	<i>64-127</i>	<i>varies</i>	<i>(bootserver specific)</i>		
PXE_BOOT_ITEM	71	3	<b>Bootserver type(2), layer(1)</b> Layer 0 = First layer of selected bootserver type. If this tag is missing, type 0 and layer 0 is assumed		<b>Note #5</b>

PXE DHCP Options (Returned to Client from ProxyDHCP)					
Tag Name	Tag Number	Length Field	Type(1)	Data Field	
Client machine identifier (UUID/GUID). DHCP_PLATFORMID	97	17	0=UUID	Type 0 = UUID(16)	Required <b>Note #1</b>
Class Identifier DHCP_CLASS	60	9	<b>“PXEClient”</b> (This field must not be null terminated.)		Required
DHCP_VENDOR	43	Varies	<i>Encapsulated options below. Multiple DHCP_VENDOR options can be used.</i>		Required
PXE_MTFTP_IP	1	4	<b>Multicast IP Address</b> Multicast IP address of bootfile.		<b>Note #2</b>
PXE_MTFTP_CPORT	2	2	<b>UDP Port Number INTEL ORDER</b> UDP port that client should monitor for MTFTP responses.		
PXE_MTFTP_SPORT	3	2	<b>UDP Port Number INTEL ORDER</b> UDP port that MTFTP servers are using to listen for MTFTP open requests.		
PXE_MTFTP_TMOUT	4	1	<b>Open Timeout</b> Number of seconds a client will listen for activity before trying to start a new MTFTP transfer.		
PXE_MTFTP_DELAY	5	1	<b>Reopen Timeout</b> Number of seconds a client will listen before trying to restart a MTFTP transfer.		

PXE DHCP Options (Returned to Client from ProxyDHCP)				
Tag Name	Tag Number	Length Field	Type(2)	Data Field
PXE_DISCOVERY_CONTROL	6	1	<b>(Bit field. Bit 0 is least significant bit.)</b> bit 0 = If set, disable broadcast discovery. bit 1 = If set, disable multicast discovery. bit 2 = If set, only use/accept servers in PXE_BOOT_SERVER. bit 3-7 = Must be 0. If this tag is not supplied, all bits assumed to be 0.	<b>Note #3</b>
Discovery_MCast_Addr	7	4	<b>Multicast IP-addr</b> Clients use this address to discover boot servers if PXE_DISCOVERY_CONTROL enables multicast discovery.	
PXE_BOOT_SERVERS	8	varies	<b>Bootserver type(2)</b> Type 0 = PXE bootstrap server Type 1 = Microsoft Windows NT bootserver Type 2 = Intel LCM bootserver Type 3 = DOS/UNDI bootserver??? Type 4 through 32767 = reserved Type 32768 through 65534 = vendor use Type 65535 = PXE API Test server	<b>Note #3</b> IPcnt(1), IP-addr-list(IPcnt*4), type(2).... IPcnt cannot be 0.  Bootservers must not respond to discovery requests of types that they do not support.
PXE_BOOT_MENU	9	varies	<b>Bootserver type(2)</b> Type 0 = reserved for local boot	<b>Note #4</b> <b>desclen(1), "description", Bootserver type(2)....</b>  Boot "order" is implicit in the menu order.  "desclen" is length of "description" "desclen" cannot be 0.
PXE_MENU_PROMPT	10	varies	<b>timeout(1), "prompt"</b> The timeout is the number of seconds to wait before auto-selecting the first boot menu item. The prompt is displayed followed by the number of seconds remaining before the first item in the boot menu is auto-selected. If <F8> is pressed, the menu will be displayed. If this option is not provided, the menu must be displayed without prompt and timeout. If the timeout is 0, the first item in the menu must only be auto-selected. If the timeout is 255, the menu and prompt must be displayed without auto-selecting or timeout.	
<i>Loader Options</i>	<i>64-127</i>	<i>varies</i>	<i>(bootserver specific)</i>	
PXE_BOOT_ITEM	71	3	<b>Bootserver type(2), layer(1)</b> Layer 0 = First file of selected bootserver type. If this tag is missing, type 0 and layer 0 is assumed.	<b>Note #5</b>

PXE DHCP Options (Returned from ProxyDHCP) used by bootserver				
Tag Name	Tag Number	Length Field	Type(1)	Data Field
Class Identifier DHCP_CLASS	60	9	"PXEClient" (This field does not need to be null terminated.)	Required
DHCP_VENDOR	43	Varies	<i>Encapsulated options below. Multiple DHCP_VENDOR options can be used.</i>	Required
Discovery_MCast_Addr	7	4	<b>Multicast IP-addr</b> Bootserver discovery multicast IP address. Bootservers capable of multicast discovery must listen on this multicast address.	

**Note #1**

See GUID programming notes in section 9.2.2 below

**Note #2**

These options define the client/server port numbers and open/re-open timeouts that must be used in MTFTP open/read requests.

**Note #3**

These options control the type of discovery mechanisms used by clients. Clients must use discovery methods in this order:

1. **Multicast:** If the client supports multicast discovery and multicast discovery is enabled and a multicast discovery IP address is available, (PXE\_DISCOVERY\_CONTROL, Opt43 tag #6)
2. **Broadcast:** If broadcast discovery is enabled, (PXE\_DISCOVERY\_CONTROL, Opt43 tag #6 and DISCOVERY\_MCAST\_ADDR, Opt 43 tag #7))
3. **Unicast:** If a *bootserver* list is available, (PXE\_BOOT\_SERVERS, Opt 43 tag #8), or back to the DHCP server if a *bootserver* list is not available.

**Note #4**

These options define the information displayed by the client, if any, during a network boot.

**Note #5**

This option is required to discover bootservers. Only the client may change the type field; either the client or the server may change the layer field. Layer 0 always indicates the first bootfile for a particular bootserver type. Only bootservers capable of providing the boot file requested in the PXE\_BOOT\_ITEM will respond.

---

## 6. Services and Registry Configuration

---

### 6.1 Overview

This section discusses the registry values used to control the functions of PXE Bootservers.

### 6.2 PXE NT Services Configuration

#### 6.2.1 ProxyDHCP

The listening ports used by the ProxyDHCP service are controlled by the registry flag UseDHCPPort.

- If this flag is set to 1, ProxyDHCP will open a TCP socket on port 67 and 4011.
- If this flag is set to 0, ProxyDHCP will open a TCP socket on port 4011 only.

If the ProxyDHCP service is running on the same host as the DHCP service, UseDHCPPort should be set to 0 because port 67 is used by the DHCP service. In this case, ProxyDHCP will process request packets sent to port 4011.

#### 6.2.2 Microsoft DHCP Service

If the ProxyDHCP service is running on the same host as the DHCP service, you must add the DHCP Class Identifier option, tag value 60, to the DHCP service. This option must be set to "PXEClient". When a PXE Client receives this class option by itself from the DHCP service, the client will immediately unicast a request to the ProxyDHCP server on port 4011 to complete the PXE DHCP configuration.

The purpose of defining option 60 to the DHCP service is so the client will receive the option as part of the DHCP Offer packet. The content of option 60 is defined to be "PXEClient" and this is what informs the client that PXE services are available on the same host as the DHCP service.

Specifically the client should receive the following byte sequence in the options section of the DHCP Offer (all values shown in hex):

**0x38 0x09 0x50 0x58 0x45 0x43 0x6c 0x69 0x65 0x6e 0x74**

The first byte is the option number 60.

The second byte is the length of the option data, 9 bytes.

The remaining bytes are the ASCII codes for the characters "PXEClient".

##### 6.2.2.1 DEFINING OPTION 60

To make the DHCP service send this sequence of bytes you must define option 60 to the DHCP server as a byte array and then define the value of the data portion of the option. The DHCP will attach the option number (0x38) and the data length (0x09).

- Start the DHCP Manager  
**Start->Programs->AdministrativeTools (Common)->DHCP Manager**
- In the panel labeled "**DHCP Servers**" double-click "**Local Machine**" to display the DHCP scopes you have previously defined.
- Single-click on any one of the scopes to highlight it.
- With a scope highlighted, click on the menu **DHCP Options->Default**.
- Select the "**New...**" button to display the "**Add Option Type**" dialog box.
- In the "**Name**" field enter the text "**Class ID**".
- Make sure the "**Data Type**" is set to "**Byte**" then click the "**Array**" checkbox so it is set.
- In the "**Identifier**" field enter "**60**"



- Click **OK**.

The dialog box titled “**DHCP Options: Default Values**” should be displayed. Click the drop-down arrow on the “**Option Name**” drop-down list. Scroll to the entry for option 060 and click it. The drop-down list should close and the “**Option Name**” box will have the text “**060 Class ID**” displayed.

Select the button “**Edit Array...**” to display the “**NumericValue Array Editor**”.

Enter the following bytes one at a time in the edit box labeled “**New Value:**”. The bytes are entered in decimal notation. After entering each value click the “**Add->**” button to add the byte to the list displayed in the “**Current Values**” list box. Make sure the values are displayed in the correct order. The values should be listed from top to bottom. Use the up and down arrows to the right of the “**Current Values**” list box to adjust the position of a value if needed.

**80 88 69 67 108 105 101 110 116**

After all of the bytes have been entered, the last byte might be a **0x00** that was added by the DHCP service. Delete this byte so you only have the 9 bytes that you entered.

Click **OK** to return to the “**DHCP Options: Default Values**” dialog. Review the values displayed in the “**Value**” group using the scroll bar to move up and down in the list. If you need to make any corrections select the “**Edit Array**” button and make the corrections needed.

Click **OK** to return to the DHCP Manager.

#### **6.2.2.2     *ASSIGNING THE OPTION TO THE SCOPES***

Now you need to assign the option to one or more scopes. The easiest way to do this is to assign the option globally.

With one of the scopes highlighted select the menu **DHCP Options->Global** to display the “**DHCP Options: Global**” dialog box.

Scroll the “**Unused Options**” list to locate the option “**060 Class ID**”. Highlight the option then click the “**Add->**” button to add the option to the list of “**Active Options**”. Click **OK**.

#### **6.2.2.3     *TESTING THE OPTION***

The option is now assigned to all of your scopes. At this point you should test that the option is being correctly sent to the client by performing a client boot while a packet sniffer captures the packets. Analyze the DHCP Offer packet to be sure the option bytes are present.

Depending on the configuration of your routers, you may also have to define a value for the Router option. This will ensure the client receives a default gateway IP address.

#### **6.2.3     Multicast Trivial FTP Service(MTFTPD)**

MTFTPD uses two registry values to enable it’s features: MCAST\_ENABLE, and MTFTP\_HIGH\_PERFORMANCE.

The registry value MCAST\_ENABLE enables multicasting. If multicasting is disabled MTFTPD will only respond to unicast file transmission requests. You may need to disable multicast if your routers cannot be configured to accept multicast transmissions.

The MTFTPD registry value MTFTP\_HIGH\_PERFORMANCE, determines whether files are transmitted in large or small packets.

The TFTP service provides two options for determining packet sizes; one for multi-cast and one for unicast. Under multi-cast the server will send files using a packet size of 1456 bytes if the MTFTPD registry

value MTFTP\_HIGH\_PERFORMANCE is set to 1. If MTFTP\_HIGH\_PERFORMANCE is set to 0 512 byte packets are used. Note that this size applies to all clients. Therefore, all of your clients must be capable of handling the larger packet size to use in increased performance. Otherwise, set MTFTP\_HIGH\_PERFORMANCE zero.

For unicast TFTP, the client can negotiate a packet size from 512 bytes to 2560 bytes. The client must use the TFTP block size option as specified in RFC 1783.

**NOTE: Multicast TFTP is not fully supported by Windows NT® Server 3.51. If you install the PDK onto a Windows NT® 3.51 server you will not be able to use multicast file transfer across routers. If you are not crossing a router, then Windows NT® 3.51 will work properly. Windows NT® 4.0 works properly in both cases and does not exhibit the problem just described.**

### 6.3 Locating BStrap Bootserver Files

There is only one file in the BStrap Bootserver, BSTRAP.0. The only information needed to form a path to BSTRAP.0 is the client system architecture. The client system architecture type is a two byte field contained in the DHCP\_SYSARCH option tag (value 93). The location of the BSTRAP.0 file is stored in the registry key with the same name as the client system architecture.

The registry value PROC\_ARCH contains a list of client system architecture values and the related text that is used for the architecture specified registry key. In the PDK, for an Intel architecture, value 0, you will see the text is "x86pc". The relative path to the BSTRAP.0 file is then located in the registry key ProxyDHCP\X86PC.

This key contains a value named Imagefile\_Name that contains the lowest valid file layer number, the highest valid file layer number, and the base name for the bootfile... In the PDK data in Imagefile\_Name is: BSTRAP

To determine the relative path for the BStrap bootfile the base filename, BSTRAP is appended to the text for the client system architecture. Then the file layer number, 0, is appended to the file name. This produces the relative path: X86PC\BSTRAP.0

The full path is formed by appending the relative path to the path stored in the MTFTPD value, BASE\_DIR. In the PDK this would form the path:

```
<install directory>\PDK\SYSTEM\IMAGES\X86PC\BSTRAP.0
```

### 6.4 Locating Bootserver Files other than BStrap

For bootservers other than BStrap, four pieces of data are required to form a full path to the NIC specific bootfiles: the architecture type, the NIC type, the bootserver type and the file layer number.

The architecture type is a two byte field contained in the DHCP\_SYSARCH option tag (value 93). This value is converted to text that matches both the sub-directory name and the sub-key name through the registry value "PROC\_ARCH"

The NIC type is a variable length option with the tag DHCP\_NICIF (value 94). The data in this option is translated into text that represents the sub-directory and the sub-key. Currently there are three possible NIC types: UNDI, PCI, and PnP. Sample text for each of them are:

```
UNDI-02-00
PCI-8086-0200-XXXXXX-XX-XXXX-XXXX
PNP-XXXX-XXXXXX
```

ProxyDHCP will search the registry for the key whose name has the longest match with the above examples. This permits you to use shorter versions of the names for bootfiles that are not dependent on the full designation for the NIC. For example, all of these are valid registry keys and directories for UNDI:

```
UNDI
UNDI-02
UNDI-03
UNDI-03-00
```

If all of these existed in the registry then, for an UNDI interface with major version 3 and minor version 0 the "UNDI-03-00" key/path would be selected. For a major version 2 and minor version 0, the "UNDI-02" key/path would be used.

The bootserver type and file layer number come from the option PXE\_BOOT\_ITEM (value 71). These are translated into text using the registry value *Service\_Types*. *Service\_Types* contains a list of service type numbers and service names. Only those *Service\_Types* supported by this host are listed. The service names match the sub-keys and the directory path where the bootfiles are stored. The sub-keys contain a value named *Imagefile\_Name* that contains the base name for the file. The file layer number is appended to the base name to form a full file name.

For instance, an UNDI compliant boot PROM on an X86 computer requesting file layer number 0 (APITest.0) of bootserver type 65535 (APITest) would result in retrieving the file name from the registry value:

```
PXE\Bootserver\X86PC\UNDI\APITest\Imagefile_Name.
```

This registry value contains the base name for the bootfile, the lowest valid file layer number and the highest valid file layer number.

In the PDK, the relative path to this file would be: X86PC\UNDI\APITest\APITest.0

The absolute path to the bootfile is formed by appending the relative path to the path contained in the MTFTPD value, BASE\_DIR.

## 6.5 Bootserver Directory Included in the PDK

The PDK will install "bootservers" for APITest and DOSUNDI. This means that clients will be able to choose to remote boot the PXE WfM tests or DOS. The "images" directory contains an architecture directory for each supported processor architecture, such as X86PC.

The architecture subdirectory contains a NIC directory for each NIC that has a bootserver defined.

Typical NIC subdirectory values:

```
\UNDI
\PCI-8086-1229-020000-01
\PNP-8130-020000
```

Each NIC Directory contains a subdirectory for each supported bootserver. These subdirectories contain the bootfiles for that architecture/NIC/bootserver combination.

**Table 1 Sample Directory Structure**

```
<install directory>\PDK\SYSTEM\IMAGES
    \X86PC
    BSTRAP.0
        \UNDI
            \APITest
                APITEST.0
                APITEST.1
            \DOSUNDI
                DOSUNDI.0
                DOSUNDI.1
```

## 6.6 Creating New DOS Bootfiles

The utility MKIMAGE.EXE included in the PDK, is used to create a DOS bootfile (such as APITest.1). To use this utility, you need to create a 1.44 MB MS-DOS\* 6.22 boot diskette (using the DOS sys command or format /s). You can leave the diskette as it is (containing only the MSDOS.SYS, IO.SYS, and COMMAND.COM files), or add AUTOEXEC.BAT and executables, etc. Whatever you put on the disk becomes part of the image.

Place this diskette in drive A: of a computer and run MKIMAGE.EXE from another drive. This writes an image of the A: diskette into the current directory in a file named TEST.BIN. Rename this file to match your bootserver name and place it in the appropriate as described above.

## 6.7 Adding Menu Options

To add menu options for additional bootfiles to download:

- Create new bootfiles in the appropriate directories
- Define the bootserver in the registry
- Define the bootfile in the registry
- Add the bootfiles to the MTFTPD list of files
- Add a menu option for the bootserver.

Section 3.2.3 above describes how to build new bootfiles. Once these bootfiles are created you must store them in a directory that reflects the client architecture, NIC type, and bootserver name.

To define the bootserver in the registry you need to add the bootserver to the list of `Service_Types` contained in the registry value "`Service_Types`". `Service_Types` is a `REG_MULTI_SZ` value where each string represents the bootserver number and the name of a bootserver. The bootserver name entered here must match the sub-directory name and the name of the registry sub-key.

Next create a definition for the bootfile in the registry. To do this, create a sub-key under the appropriate architecture type and NIC type. Give this sub-key the same name you used when you defined the bootserver in the `Service_Types` registry value. Within this sub-key create a value named *Imagefile\_Name* of type `REG_MULTI_SZ`. The first string contains the lowest valid file layer number, the second string contains the highest valid file layer number and the third string contains the base name for the bootfiles.

Under the MTFTPD key there is a sub-key named FILES. Each of the values under FILES represents a file that MTFTPD can transmit using multicast transmission. Create a new value for each of the files you have included in the bootserver. The name of the values is the relative path for the file starting with the architecture type and ending with the full filename. The values are of type `REG_SZ`. The data for the values is the multicast IP address for the file. These addresses are assigned dynamically by the MTFTPD service. After adding files to this list you MUST restart the MTFTPD service.

Menus are specific to a particular client architecture and NIC type. This ensures clients sees only menu options valid for their workstations. Locate the MENU value under the appropriate architecture and NIC type sub-keys and add your menu entry to this list.

A menu entry consists of the number for the bootserver type and the text you want displayed to the user.

## 6.8 Registry Entries

<b>Registry Keys for PXE Services</b>	
<b>Key Name:</b> SOFTWARE\Intel\PXE <b>Class Name:</b> <NO CLASS> <b>Last Write Time:</b> 11/24/97 - 1:12 PM	This is the root registry key for all PXE Services.
<b>Value 0</b> Name: DomainName Type: REG_SZ Data:	Domain or workgroup name for this server.
<b>Value 1</b> Name: IsDomain Type: REG_DWORD Data: 0	Indicates whether the DomainName reflects a Workgroup or a Domain.
<b>Value 2</b> Name: ServerName Type: REG_SZ Data:	Computer name of this server.
<b>MTFTPD</b>	
<b>Key Name:</b> SOFTWARE\Intel\PXE\MTFTPD <b>Class Name:</b> <NO CLASS> <b>Last Write Time:</b> 11/11/97 - 10:40 AM	Top Level for MTFTPD service. Contains one sub-key, FILES.
<b>Value 0</b> Name: BASE_DIR Type: REG_SZ Data: <install directory>\PDK\SYSTEM\IMAGES	The base directory where bootfiles are located. This path is prepended to the partial path that the client requests.
<b>Value 1</b> Name: MCAST_CLNT_PORT Type: REG_SZ Data: 1758	The client port for MTFTPD
<b>Value 2</b> Name: MCAST_ENABLE Type: REG_DWORD Data: 0x1	Determines whether multicast is enabled. 1 => Multicast Enabled 2 => Multicast Disabled
<b>Value 3</b> Name: MCAST_OPN_TO Type: REG_SZ Data: 1	Multicast open timeout.
<b>Value 4</b> Name: MCAST_REOPN_TO Type: REG_SZ Data: 2	Multicast reopen timeout.
<b>Value 5</b> Name: MCAST_SRVR_PORT Type: REG_SZ Data: 1759	The port where MTFTPD listens for file transfer requests.
<b>Value 6</b> Name: MCAST_START_ADDRESS Type: REG_SZ Data: 224.1.1.1	The starting multicast IP address of the range of addresses assigned to the files.
<b>Value 7</b> Name: MTFTP_HIGH_PERFORMANCE Type: REG_DWORD Data: 0x1	Enables the use of large packets in file transfers. 1 => Enabled 0 => Disabled
<b>Key Name:</b> SOFTWARE\Intel\PXE\MTFTPD\FILES <b>Class Name:</b> <NO CLASS> <b>Last Write Time:</b> 11/11/97 - 10:41 AM	Contains a set of registry values for each file to be sent by MTFTPD. The data for each value is the multicast IP address for the file.
<b>Value 0</b> Name: X86PC\BSTRAP.0 Type: REG_SZ Data: 224.1.1.1	BStrap.0 file. The IP address for this file MUST be the same as the value in Layer_0_IP.
<b>Value 1</b> Name: x86pc\undi\APITest\APITest.0 Type: REG_SZ Data: 224.1.1.2	First bootfile for the API Test.
<b>Value 2</b> Name: x86pc\undi\APITest\APITest.1 Type: REG_SZ Data: 224.1.1.3	Second bootfile for the API Test.

Registry Keys for PXE Services	
Value 3 Name: x86pc\undi\DOSUNDI\DOSUndi.0 Type: REG_SZ Data: 224.1.1.4	First bootfile for the DOS boot.
Value 4 Name: x86pc\undi\DOSUNDI\DOSUndi.1 Type: REG_SZ Data: 224.1.1.5	Second bootfile for the DOS boot.
ProxyDHCP	
Key Name: SOFTWARE\Intel\PXE\ProxyDHCP Class Name: <NO CLASS> Last Write Time: 11/24/97 - 3:04 PM	Top level for the ProxyDHCP service. Contains sub-keys for each PC architecture supported.
Value 0 Name: Discovery_BCast_Disabled Type: REG_DWORD Data: 0	Controls whether client is able to broadcast Discovery requests. 0 => Broadcasting Enabled 1 => Broadcasting Disabled
Value 1 Name: Discovery_List Type: REG_MULTI_SZ Data:	A explicit list of servers authorized to provide bootfiles to the client. The format for each entry is: s,n,IP <sub>1</sub> ...IP <sub>n</sub> s => bootserver type n => number of addresses in this list IP => IP address of server Each list is a separate string.
Value 2 Name: Discovery_MCast_Addr Type: REG_SZ Data: 224.0.1.2	The multicast address for sending multicast discovery packets.
Value 3 Name: Discovery_MCast_Disabled Type: REG_DWORD Data: 0	Disable multicasting discovery packets. 1 => multicasting disabled 0 => multicasting enabled
Value 4 Name: Discovery_Server_List_Only Type: REG_DWORD Data: 0	Enable server list. If enabled client is to only accept bootfiles from the list of servers contained in the Discovery_List registry value. 0 => Disable Discovery_List 1 => Enable Discovery_List
Value 5 Name: Discovery_Srvr_IP Type: REG_SZ Data: 128.128.0.2	Unicast IP address of the discovery server that provides BStrap.0.
Value 6 Name: PROC_ARCH Type: REG_MULTI_SZ Data: 0,X86PC 1,NECPC9800	Mapping of Client System Architecture types to sub-Key Names.
Value 7 Name: Prompt Type: REG_SZ Data: 10,Select a boot option	Text and display time for the prompt displayed after the client menu.
Value 8 Name: Service_Types Type: REG_MULTI_SZ Data: 0,BStrapSrv 1,WinNT 2,Intel LCM 3,DOSUNDI 65535,APITest	Mapping of BootServer Types to sub-key and directory names.
Value 9 Name: TestOn Type: REG_DWORD Data: 1	Output packet analysis for client testing.
Value 10 Name: TestPath Type: REG_SZ Data: <Install Dir>\PDK\TestLog	Directory where test log sub-directories are created when doing client testing.
Value 11 Name: UseDHCPPort Type: REG_DWORD Data: 1	Controls whether the PXE Bootserver listens on port 67. This should be turned off if PXE Bootserver shares the server with a DHCP service.

<b>Registry Keys for PXE Services</b>	
<b>Key Name:</b> SOFTWARE\Intel\pxe\ProxyDHCP\X86PC <b>Class Name:</b> <NO CLASS> <b>Last Write Time:</b> 11/7/97 - 1:10 PM	Key to contain data unique to X86PCs.
<b>Value 0</b> Name: Imagefile_Name Type: REG_MULTI_SZ Data: 0 0 BStrap	Relative path to the BStrap.0.
<b>Value 1</b> Name: Vendor_DLL Type: REG_SZ Data:	<RESERVED> Will point to a DLL that can provide any vendor options that are unique to this bootserver.
<b>Value 2</b> Name: Vendor_Options Type: REG_BINARY Data:	<RESERVED> Will provide the ability to specify vendor options unique to this bootserver directly in the registry.
<b>Key Name:</b> SOFTWARE\Intel\pxe\ProxyDHCP\X86PC\UNDI <b>Class Name:</b> <NO CLASS> <b>Last Write Time:</b> 11/24/97 - 3:03 PM	Key to contain data unique to X86Pcs with NICs that have UNDI interfaces.
<b>Value 0</b> Name: MENU Type: REG_MULTI_SZ Data: 65535,PXE API Test 3,DOS UNDI 0,Local Boot	Menu list displayed by BStrap.0 on the client. Each entry is the bootserver number followed by the menu item text, separated by a comma. Each menu item is a separate string.
<b>Key Name:</b> SOFTWARE\Intel\pxe\ProxyDHCP\X86PC\UNDI\APITest <b>Class Name:</b> <NO CLASS> <b>Last Write Time:</b> 11/24/97 - 8:46 AM	Contains data for telling the client how to download the bootfiles for the API Test Bootserver.
<b>Value 0</b> Name: Imagefile_Name Type: REG_MULTI_SZ Data: 0 1 APITest	Root name of the bootfile along with the low and high valid index numbers.
<b>Value 1</b> Name: Vendor_DLL Type: REG_SZ Data:	<RESERVED> Will point to a DLL that can provide any vendor options that are unique to this bootserver.
<b>Value 2</b> Name: Vendor_Options Type: REG_BINARY Data:	<RESERVED> Will provide the ability to specify vendor options unique to this bootserver directly in the registry.
<b>Key Name:</b> SOFTWARE\Intel\pxe\ProxyDHCP\X86PC\UNDI\DOSUNDI <b>Class Name:</b> <NO CLASS> <b>Last Write Time:</b> 11/24/97 - 8:46 AM	Contains data for telling the client how to download the bootfiles for the DOS UNDI Bootserver.
<b>Value 0</b> Name: Imagefile_Name Type: REG_MULTI_SZ Data: 0 1 DOSUNDI	Root name of the bootfile.
<b>Value 1</b> Name: Vendor_DLL Type: REG_SZ Data:	<RESERVED> Will point to a DLL that can provide any vendor options that are unique to this bootserver.
<b>Value 2</b> Name: Vendor_Options Type: REG_BINARY Data:	<RESERVED> Will provide the ability to specify vendor options unique to this bootserver directly in the registry.

---

## 7. Testing PXE

---

### 7.1 Tests Provided by PDK

Three different types of tests are provided in the this PDK:

1. Packet Analysis
2. UNDI Stress Test
3. PXE API Tests

#### 7.1.1 Packet Analysis

DHCP packet analysis consists of determining if the packet correctly contains the following options:

Description	Option Number
Client machine identifier (GUID)	97
Client system architecture option	93
Client network interface identifier	94
Correct DHCP Msg type	53
Class Identifier set to "PXEClient"	60

If any of these options are missing or the DHCP Message type and class identifier contain incorrect values the client is considered to fail the compliance test.

#### 7.1.2 UNDI Stress Test

To test the UNDI API with the universal NDIS driver, we copy files with different sizes (maximum file size is 5MB) from the server's mapped drive to the RAMDisk and then compare the 2 sets of files.

#### 7.1.3 PXE API Test

The "Wired for Management Baseline, Version 1.1a" document specifies a list of API calls to be provided by the PXE through a common entry that can be obtained through interrupt 1A. The entry procedure for the API calls takes a parameter block in ES:DI and function number for the call in BX. It returns a success or failure status in AX and sets the status in parameter block accordingly. All other register contents must be retained to their original values.

This PXETEST program attempts to verify all the API calls for their existence. It also verifies the contents of all the registers if modified by the call. It prints the results to a log file. It first tests the APIs in the following sequence: BINL\_INFO, UDP, TFTP and UNDI. Since the PXETEST program shuts down the connection between the UNDI layer and the UDP/IP protocol stack in the boot ROM, the TFTP and UDP APIs will not be available if the PXETEST program is run for the second time.

**NOTE:** This PXETEST program tests a new call GET\_IFACE\_INFO which is not a part of the "Wired for Management Baseline, Version 1.1a" document. This API call is not required (is optional) if the UNDI is implemented for Ethernet. This is needed for non Ethernet implementations of the UNDI for the universal NDIS to work.



The Syntax of the call is as follows:

```
/* definitions */  
  
#define PXENV_UNDI_GET_IFACE_INFO 0x0013  
  
typedef struct s_PXENV_UNDI_GET_IFACE_INFO {  
    UINT16 Status; /* OUT: See PXENV_STATUS_xxx constants */  
    UINT8 IfaceType[16]; /* OUT: Type name of MAC, AsciiZ format */  
    /* This is used by the Universal NDIS */  
    /* Driver to fill the driver type in it's */  
    /* MAC Service specific characteristic table */  
    /* refer to NDIS2 specification for available */  
    /* (or supported) type strings */  
    /* OUT: bits/sec */  
    UINT32 LinkSpeed; /* OUT: as defined in NDIS Spec 2.0X */  
    UINT32 ServiceFlags; /* OUT: will be filled with 0s till defined */  
    UINT32 Reserved[4];  
} t_PXENV_UNDI_GET_IFACE_INFO;
```

/\* API Description \*/

Opcode: PXENV\_UNDI\_GET\_IFACE\_INFO

input: ES:DI points to a t\_PXENV\_UNDI\_GET\_IFACE\_INFO parameter structure that has been initialized by the caller.

output: PXENV\_EXIT\_SUCCESS or PXENV\_EXIT\_FAILURE will be returned in AX, with the Carry Flag set accordingly. The status field in the parameter structure will be set to one of the values represented by the PXENV\_STATUS\_xxx constants. If the PXENV\_EXIT\_SUCCESS is returned the parameter structure will contain the interface specific information.

Description: This call, if successful, provides the interface specific information necessary for the universal NDIS driver to report the correct network interface type supported by UNDI. If this call is not supported the NDIS driver will report the interface type as Ethernet (DIX+802.3) and link speed as 10 Mbit.

## 7.2 Test Logs

During the API Test we create a ramdrive on the client machine with 8MB of extended memory. All the client test output files will be created on this virtual A drive.

In addition, if the universal NDIS driver successfully loads, the <install directory>\PDK\TESTLOG share of the bootserver is mapped to a drive by the client and the test log files are copied into the client's sub-directory.

In addition, the ProxyDHCP service on the test server will place the results of the Packet Analysis test in log files in the <install directory>\PDK\TESTLOG client sub-directory.

The test logs are cumulative. Therefore the client's subdirectory should be deleted before performing a new test boot.

In summary:

- The test logs are located in the sub-directory: <install directory>\PDK\testlog.
- The log files for a particular client machine are in the sub-directory with the last 8 characters of the client's MAC address.

This sub-directory will contain the files:

```
TESTSUM.TXT  
BINL1.TXT  
BINL2.TXT  
BINL3.TXT  
NDISTEST.TXT  
DHCPPKT.TXT
```

TESTSUM.TXT will contain a summary of the results of the PXE API test and the Packet Analysis tests. The packet analysis summary is first, followed by the summary results of the PXE API test.

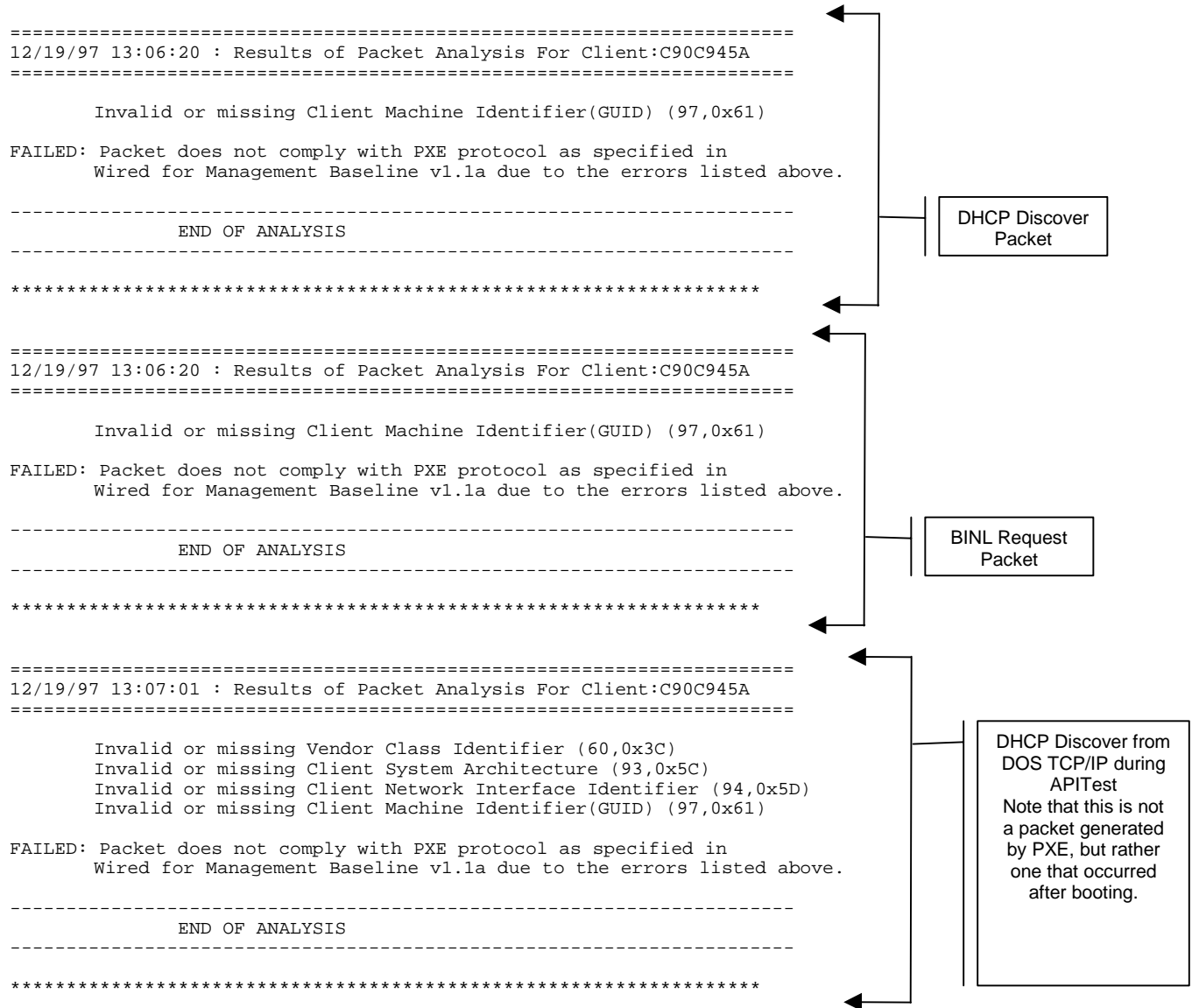
The files BINL1.TXT, BINL2.TXT, and BINL3.TXT contain hex dumps of the three packets PXE cached.

NDISTEXT.TXT contains the detailed output from performing a series of file transfers to stress test the NDIS interface.

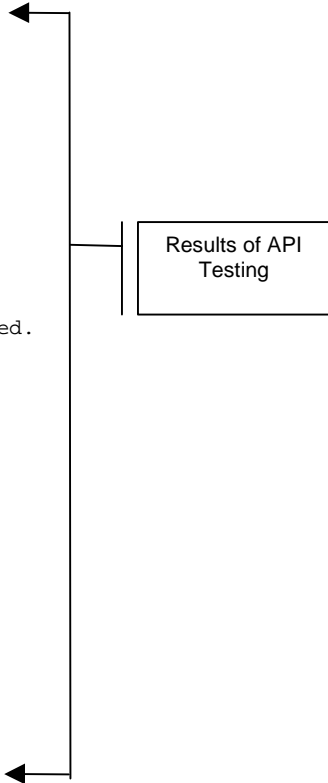
DHCPPKT.TXT contains a hex dump of the DHCP\_Discover or BINL\_Request packet and a detailed analysis of the PXE options found or missing. DHCPPKT.TXT may contain multiple packet analysis depending on how many DHCP transactions were performed.

Following are example contents of the TESTSUM.TXT file. This example illustrates the results of a client that sends a DHCP Discover when it first boots and then follows up with a BINL Request. This client has failed the compatibility test because it does not include a GUID in the DHCP or BINL packet. (The third packet analysis is from the DOS TCP/IP stack that is loaded during the API Test. This was a DHCP transaction that occurred after the client booted.)

The text after the packet analysis indicates the client passed all of the APITests.



```
GET_DHCP_DISCOVER: Passed..
GET_DHCP_ACK: Passed..
GET_BINL_REPLY: Passed..
UDP_OPEN: Passed..
UDP_WRITE: Passed..
UDP_READ: Passed..
UDP_CLOSE: Passed..
TFTP_GET_FILE_SIZE: Passed..
TFTP_OPEN: Passed..
TFTP_READ: Passed..
TFTP_CLOSE: Passed..
TFTP_READ_FILE: Passed..
UNDI_SHUTDOWN: Passed..
UNDI_INITIATE_DIAGS: Call is present but not supported.
                    This is an optional call.
UNDI_INIT: Passed..
UNDI_GET_STAT: Passed..
UNDI_CLEAR_STAT: Passed..
UNDI_OPEN: Passed..
UNDI_GET_INFO: Passed..
UNDI_GET_NIC_TYPE: Passed..
UNDI_GET_IFACE_INFO: Passed..
UNDI_SET_STATION_ADDR: Passed..
UNDI_GET_MCAST_ADDR: Passed..
UNDI_SET_MCAST_ADDR: Passed..
UNDI_SET_PACKET_FILTER: Passed..
UNDI_TRANSMIT: Passed..
UNDI_FORCE_INTERRUPT: Passed..
UNDI_RESET_NIC: Passed..
UNDI_CLOSE: Passed..
UNDI_SHUTDOWN: Passed..
```



Results of API  
Testing

### 7.3 Initiating the Tests

Packet Analysis testing is enabled by setting the test server registry flags *TestOn* and *TestPath* (**Key Name:** SOFTWARE\Intel\PXE\ProxyDHCP). *TestPath* must point to a subdirectory that is shared with the name "TESTLOG" (i.e. <install directory>\PDK\TESTLOG).

PXE API Test and UNDI Stress Test are initiated by booting the PXE Test from the test server.

## 8. Revision History

### 8.1 PXE Boot ROM

PXE PDK Release Date	PXE PDK Ver #	LSA Code Ver#	Wired for Management Baseline	LSA Code Changes
<u>3/20/98</u>	V2.2	<u>v.99f</u>	V1.1a	<b>Bug Fixes:</b> <ul style="list-style-type: none"> <li>Corrected bug in TFTP open w/ options. The ROM was not responding to the TFTP OACK with a TFPT ACK of package zero.</li> </ul>
<u>2/11/98</u>	V2.1a	<u>v.99e</u>	V1.1a	<b>Bug Fixes:</b> <ul style="list-style-type: none"> <li>Disabled all debug code in option ROM initialization code. This code was checking for an enabled game port and removing the option ROM image if one was found.</li> </ul>
<u>1/30/98</u>	V2.1	<u>v.99d</u>	V1.1a	<b>Bug Fixes:</b> <ul style="list-style-type: none"> <li>The procedure that was written to read the UUID out of the System Management BIOS System Information (type 1) table, was never being called. Code added to correctly locate and identify the System Management BIOS structure table entry point..</li> </ul>
<u>12/22/97</u>	V2.0	<u>v.99c</u>	V1.1a	<b>Enhancements:</b> <p><b>Loader</b></p> <ul style="list-style-type: none"> <li>Source split into multiple modules for stub design. Removed all Intel specific hardware references from stub version of code.</li> </ul> <p><b>Base code</b></p> <ul style="list-style-type: none"> <li>Added the following DHCP options to the parameter request list: Time Offset (2), Time Server (4), Domain Name Server (6), Host Name (12) and Domain Name (15).</li> <li>Replaced BINL... and BOOTP... messages with DHCP.... Each dot represents a transmitted packet.</li> <li>Removed "Transferring control to bootstrap" message. It was not internationalizable.</li> <li>Added code in udpwrite() to check if the destination IP is a multicast address before sending an ARP request out on it. For multicast UNDI is called to convert multicast IP address to MAC address.</li> <li>Added code to check second gateway, if a different one was given in the ProxyDHCP Offer or BINL Reply than was given in the DHCP Ack. Fixes problem where the client would not find MTFTPD if the client was between two routers with DHCP and Discovery servers were past one router and ProxyDHCP was past the other and if the routers were not configured to communicate with each other.</li> </ul> <p><b>Bug Fixes:</b></p> <ul style="list-style-type: none"> <li>Adjusted test to see if ROM should request a BINL packet. The original test did not take into account a bootfile sent in a Proxy packet. This is a non-fatal bug.</li> </ul>

PXE PDK Release Date	PXE PDK Ver #	LSA Code Ver#	Wired for Management Baseline	LSA Code Changes
<u>10/13/97</u>	V1.4	<u>v.99b</u>	V1.1a	<p><b>Bug Fixes: Functional Problems</b></p> <p><b>UNDI Driver</b></p> <ul style="list-style-type: none"> <li>Added code to check the AutoNegotiation Complete flag while detecting the PHY. This fixes a problem with 82558 on a 100 Mbit network with the WakeOnLan bit set.</li> </ul> <p><b>Loader</b></p> <ul style="list-style-type: none"> <li>Removed debug and startup messages during option ROM initialization to conform with PC98 specification.</li> <li>Removed 3 second debug timing delays at the end of option ROM initialization.</li> <li>LSA2 not working while using Soft-ICE during debugging. Added code to LOM version that will reduce the reported size of extended memory (Int 15h, AH-88h) by 64KB. This happens only in systems that do not support POST Memory Manager.</li> </ul> <p><b>Base Code</b></p> <ul style="list-style-type: none"> <li>Added code to insure a true DHCP Offer including PXE extensions is consistently preferred to a ProxyDHCP reply.</li> <li>Corrected use of registers in protected-mode TFTP read file API. Bug was found in-house using new protected-mode API test suite.</li> <li>Fixed bug in MTFTP restart. If a client missed the first packet of an MTFTP restart, they would wait until all network traffic to that multicast IP addressed stopped, before trying to restart. Clients will now start listening in on network connections if another client becomes master.</li> <li>Fixed mismatched #defines/equates in header files. Fixes pressing &lt;Esc&gt; in LOM version of LSA2.</li> </ul>

PXE PDK Release Date	PXE PDK Ver #	LSA Code Ver#	Wired for Management Baseline	LSA Code Changes
<u>9/5/97</u>	<u>V1.3</u>	<u>v.98i</u>	<u>V1.1a</u>	<p><b>Bug Fixes: Functional Problems</b></p> <p><b>UNDI Driver</b></p> <ul style="list-style-type: none"> <li>Added code to shutdown NIC if media test fails. Interrupt vector table was not being restored upon exit due to media test fail.</li> </ul> <p><b>Loader</b></p> <ul style="list-style-type: none"> <li>LOM version of loader no longer corrupts \$PMM memory tables when copying itself into the allocated memory.</li> <li>Corrected stack bug in loader. Systems that did not support Int 15h AX=E820h would hang.</li> </ul> <p><b>Base Code</b></p> <ul style="list-style-type: none"> <li>Added code to use gateway IP address from DHCP ACK packet if the gateway IP address in the BINL REPLY packet is zero.</li> <li>MTFTP now quits if it gets a "file not found" error. It used to try to TFTP the missing file.</li> <li>'ciaddr' field in DHCP request message is no longer filled in. This allows LSA2 to work with the Netware DHCP server.</li> </ul> <p><b>Bug Fixes: Due to non-compliance w/ Network PC DG spec.</b></p> <p><b>Base Code</b></p> <ul style="list-style-type: none"> <li>The CLIENT_ARCHITECTURE field in DHCP DISCOVER and BINL REQUEST packets was changed from one byte, to two bytes, in network order per the Network PC DG spec..</li> <li>New functionality per Wired for Management Baseline v1.1a</li> </ul> <p><b>UNDI Driver</b></p> <ul style="list-style-type: none"> <li>Added GET_NIC_TYPE call to UNDI so OS setup programs can determine the type of underlying NIC.</li> <li>Below changes are not required for PXE compliance</li> <li>Proposed Network PC DG new functionality</li> </ul> <p><b>UNDI Driver</b></p> <ul style="list-style-type: none"> <li>Added GET_IFACE_INFO call to UNDI API so drivers can get information about H/W interface; specifically, so universal drivers can determine network media type (Ethernet, Token Ring, etc.) At the moment, this proposed change is only important for implementations for non-Ethernet network adapters.</li> </ul> <p><b>Bug Fixes: Functional Problems in non-compliant legacy systems</b></p> <p><b>Loader</b></p> <ul style="list-style-type: none"> <li>Fixed hang on HP Vectra machines when user did not press &lt;Space&gt;. These BIOSes required the option ROM to return control by restoring the interrupt 19h bootstrap and jumping to the 'PC/AT Compatible' BIOS location of F000:E6F2h.</li> </ul> <p><b>Enhancements:</b></p> <p><b>Loader</b></p> <ul style="list-style-type: none"> <li>LSA2 loader now only allocates memory using \$PMM, or by reducing the size of extended memory reported by Int 15h AH=88h. This is to remove a conflict between our code and Soft-ICE.</li> </ul> <p><b>Base Code</b></p> <ul style="list-style-type: none"> <li>Reduced stored DHCP packets to minimum size (548 bytes). Reduces size of run-time memory by ~3 Kbytes.</li> <li>Added code to read UUID from BIOS_INFORMATION structure as documented in the "System Management BIOS Reference Specification v2.1</li> </ul>

PXE PDK Release Date	PXE PDK Ver #	LSA Code Ver#	Wired for Management Baseline	LSA Code Changes
<u>7/31/97</u>	<u>V1.2</u>	<u>v.98a</u>	<u>V1.1a</u>	<ul style="list-style-type: none"> <li>Added option negotiation for TSIZE.</li> <li>Added new TFTP API call to get file size using TSIZE.</li> <li>Number of retries in MTFTP and TFTP reduced from 8 to 4.</li> <li>t_jmpbuf structure used in LSA2 library (setjmp() and longjmp() calls) did not have enough fields. The structure checksum was overwriting the CPU flags field at the end of the t_jmpbuf structure.</li> <li>The s_lsa structure in the LSA2 option ROM initialization loader did not have storage reserved for the MLID initialized data field.</li> <li>LSA2 did not boot in legacy systems that only supported interrupt 19h bootstrap.</li> <li>TFTP/UDP packets were not being received in protected-mode on some NICs.</li> <li>TFTP read API was not resetting the buffer size if the TFTP read failed.</li> <li>Now recognizes PXE DHCP and ProxyDHCP packets that contain a bootfile. BINL communication is skipped.</li> <li>Increases the 'seconds' field in successive DHCP packets. This enables some DHCP relay agents to forward the DHCP packets.</li> </ul>
<u>6/20/97</u>	<u>V1.1</u>	<u>v.97</u>	<u>V1.0a</u>	<ul style="list-style-type: none"> <li>Added code to locate 'best' location for extended memory copy of LSA2 using known BIOS extended memory size services. Int 15h w/ AX=E820h, AX=E801h, AH=C7h, AH=8Ah, AX=DA88h &amp; AH=88h.</li> <li>Removed all debug output when in protected-mode. Fixes protection fault in protected-mode.</li> <li>Added TFTP/UDP open/read clean-up code when an error occurs.</li> <li>Moved all variables used in XMIT ISR into the base-code data segment. Fixes protection fault in protected-mode.</li> <li>Commented out POST Memory Manager allocations because of internal stack bug in loader.</li> <li>Corrected protected-mode segment register assignments.</li> <li>Added MODE_SWITCH API call so caller can change to/from protected-mode.</li> <li>Rewrite of core BOOTP/BINL packet processing to support SuperDHCP and SuperProxy servers.</li> <li>Moved PXE Entry Point structure from base-code to MLID.</li> <li>Added code to read and restore packet timer to fix ARP timeout bug.</li> <li>Added checks for zero length segments when loader is allocating and copying base-code and MLID.</li> <li>Added code to loader to handle MLID initialized data allocation and copying.</li> <li>New fields were added to the TFTP_OPEN and TFTP_READ API parameter structures.</li> <li>Completed multicast TFTP support.</li> <li>Added IP address filters in UPD_READ so different TFTP servers could send different files with the same multicast IP address.</li> <li>Corrected UNDI transmit block pointer type flag.</li> <li>Added code to compute physical address of TFTP buffer for protected-mode receiving.</li> <li>Removed unused error messages to save space.</li> </ul>
<u>5/2/97</u>	<u>V1.0</u>	<u>v.92</u>	<u>V1.0</u>	First distribution.

## 8.2 PXE Services

<u>PXE PDK Release Date</u>	<u>PXE PDK Version</u>	<u>Service Code Changes</u>
<u>12/11/97</u>	<u>V2.0</u>	<b>MTFTP</b> <ul style="list-style-type: none"> <li>• N/A</li> </ul>
		<b>PXE Bootservers</b> <ul style="list-style-type: none"> <li>• Merged ProxyDHCP and BINL into PXE Bootservers.</li> <li>• Added Bootserver function to dynamical discover bootfiles.</li> </ul>
<u>10/8/97</u>	<u>V1.4</u>	<b>MTFTP</b> <ul style="list-style-type: none"> <li>• Added check for multicast OPEN requests which will not allow multiple clients to be masters at the same time for the same file.</li> <li>• Corrected error with compiler optimization that resulted in a "slave" client not being able to become a "master" client.</li> </ul>
		<b>BINL</b> <ul style="list-style-type: none"> <li>• N/A</li> </ul>
		<b>ProxyDHCP</b> <ul style="list-style-type: none"> <li>• N/A</li> </ul>
<u>9/5/97</u>	<u>V1.3</u>	<b>MTFTP</b> <ul style="list-style-type: none"> <li>• Increased the Time-To-Live field for Multicast packets to allow them to pass through routers.</li> </ul>
		<b>BINL</b> <ul style="list-style-type: none"> <li>• N/A</li> </ul>
		<b>ProxyDHCP</b> <ul style="list-style-type: none"> <li>• N/A</li> </ul>
<u>7/31/97</u>	<u>V1.2</u>	<b>MTFTP</b> <ul style="list-style-type: none"> <li>• Performance improvements.</li> <li>• Negotiation added for file size and buffer size.</li> </ul>
		<b>BINL</b> <ul style="list-style-type: none"> <li>• Improved packet parsing to recognize illegal packets, such as LSA2 V0.94, and ignore them</li> <li>• Corrected number of bytes sent for packet size to include terminating 0XFF flag</li> </ul>
		<b>ProxyDHCP</b> <ul style="list-style-type: none"> <li>• Initial Release.</li> </ul>



---

## 9. BIOS Support

---

### 9.1 Overview

This section discusses BIOS support required for PXE compliance and how it is used by PXE boot devices (ROMs) and PXE Network Boot Programs (NBPs).

There are four major BIOS components that are used by PXE ROMs and NBPs. These components are: GUID/UUID (Globally/Universally Unique ID), Wake-up Source, Bootstraps and Memory Management.

**GUID:** A GUID/UUID provides a way to uniquely identify each compliant machine on a network.

**Wake-up Source:** Being able to determine the wake-up source gives NBPs the ability to choose an alternate boot path for a machine. (e.g.: A machine turned on by the power switch can boot from the local hard disk. A machine turned on by a remote wake-up packet on the network can boot from network.)

**Bootstraps:** The "BIOS Bootstrap Specification, Version 1.01" is required by the "Wired for Management Baseline, Version 1.1a". Legacy bootstrap support (hooking interrupt 18h or 19h) can be implemented, in addition to PnP/BBS.

**Memory Management:** At present, the "POST Memory Manager Specification, Version 1.0" is not required by the "Wired for Management Baseline, Version 1.1a". It is more reliable and robust method of allocating and managing extended memory than chaining interrupt 15h. If it is available, it will be used by PXE ROMs.

### 9.2 GUID

#### 9.2.1 Detection

Three methods of GUID detection should be supported by PXE compliant ROMs:

- Reading SMBIOS structures through the PnP function interface
- Reading table-based SMBIOS structures
- Reading table-based SYSID structures

##### 9.2.1.1 Reading SMBIOS structures through the PnP function interface:

Using this method, PXE ROMs can make repeated calls to function 51h (Get SMBIOS Structure) until System Information structure (table 1) is found.

#### Format of System Information structure:

Offset	Spec Version	Name	Length	Value	Description
00h	2.0+	Type	BYTE	1	Component ID Information Indicator
01h	2.0+	Length	BYTE	08h or 19h	Length dependent on version supported, 08h for 2.0 or 19h for 2.1.
02h	2.0+	Handle	WORD	Varies	
04h	2.0+	Manufacturer	BYTE	STRING	Number of Null terminated string
05h	2.0+	Product Name	BYTE	STRING	Number of Null terminated string
06h	2.0+	Version	BYTE	STRING	Number of Null terminated string
07h	2.0+	Serial Number	BYTE	STRING	Number of Null terminated string
08h	2.1+	GUID	16 BYTEs	Varies	Globally Unique ID number. If the value is all FFh, the ID is not currently present in the system, but is settable. If the value is all 00h, the ID is not present in the system.

Offset	Spec Version	Name	Length	Value	Description
18h	2.1+	Wake-up Type	BYTE	ENUM	Identifies the event that caused the system to power up.

### 9.2.1.2 Reading table-based SMBIOS structures:

The SMBIOS Entry Point structure, described below, can be located by application software by searching for the anchor-string on paragraph (16-byte) boundaries within the physical memory address range 000F0000h to 000FFFFFh. This entry point encapsulates an intermediate anchor string which is used by some existing DMI browsers.

**Note:** While the SMBIOS Major and Minor Versions (offsets 06h and 07h) currently duplicate the information present in the SMBIOS BCD Revision (offset 1Dh), they provide a path for future growth in this specification. The BCD Revision, for example, provides only a single digit for each of the major and minor version numbers.

#### Format of SMBIOS Entry Point structure:

Offset	Name	Length	Description
00h	Anchor String	4 BYTES	_SM_, specified as four ASCII characters (5F 53 4D 5F).
04h	Entry Point Structure Checksum	BYTE	Checksum of the Entry Point Structure ( <i>EPS</i> ). This value, when added to all other bytes in the <i>EPS</i> , will result in the value 00h (using 8-bit addition calculations). Values in the <i>EPS</i> are summed starting at offset 00h, for <i>Entry Point Length</i> bytes.
05h	Entry Point Length	BYTE	Length of the Entry Point Structure, starting with the Anchor String field, in bytes, currently 1Eh.
06h	SMBIOS Major Version	BYTE	Identifies the major version of this specification implemented in the table structures, e.g. the value will be 0Ah for revision 10.22 and 02h for revision 2.1.
07h	SMBIOS Minor Version	BYTE	Identifies the minor version of this specification implemented in the table structures, e.g. the value will be 16h for revision 10.22 and 01h for revision 2.1.
08h	Maximum Structure Size	WORD	Size of the largest SMBIOS structure, in bytes. This is the value returned as <i>StructureSize</i> from the <i>Get SMBIOS Information</i> function.
0Ah	Entry Point Revision	BYTE	Identifies the <i>EPS</i> revision implemented in this structure. Since only one revision is currently defined, the value is set to 0. All other values are reserved for assignment via this specification.
0Bh - 0Fh	Reserved	5 BYTES	Reserved for future assignment by this specification, must be set to all 00h.
10h	Intermediate anchor string	5 BYTES	_DMI_, specified as five ASCII characters (5F 44 4D 49 5F). <b>Note:</b> This field is paragraph-aligned, to allow legacy DMI browsers to find this entry point within the SMBIOS Entry Point Structure.
15h	Intermediate Checksum	BYTE	Checksum of Intermediate Entry Point Structure ( <i>IEPS</i> ). This value, when added to all other bytes in the <i>IEPS</i> , will result in the value 00h (using 8-bit addition calculations). Values in the <i>IEPS</i> are summed starting at offset 10h, for 0Fh bytes.
16h	Structure Table Length	WORD	Total length of SMBIOS Structure Table, pointed to by the Structure Table Address, in bytes.

Offset	Name	Length	Description
18h	Structure Table Address	DWORD	The 32-bit physical starting address of the read-only SMBIOS Structure Table, which can start at any 32-bit address. This area contains all of the SMBIOS structures fully packed together. These structures can then be parsed to produce exactly the same format as that returned from an <i>Get SMBIOS Structure</i> function call.
1Ch	Number of SMBIOS Structures	WORD	Total number of structures present in the SMBIOS Structure Table. This is the value returned as <i>NumStructures</i> from the <i>Get SMBIOS Information</i> function.
1Dh	SMBIOS BCD Revision	BYTE	Indicates compliance with a revision of this specification. It is a BCD value where the upper nibble indicates the major version and the lower nibble the minor version. For revision 2.1, the returned value is 21h. If the value is 00h, only the Major and Minor Versions in offsets 6 and 7 of the Entry Point Structure provide the version information.

### 9.2.1.3 Reading table-based SYSID structures

The SYSID Entry Point structure, described below, can be located by application software by searching for the anchor-string on paragraph (16-byte) boundaries within the physical memory address range 000E0000h to 000FFFFFh.

The UUID BIOS structure can be found by walking the list of SYSID BIOS structures in the SYSID BIOS Structure Table.

#### Format of SYSID Entry Point structure:

Element	Length	Description
Header/Type	7 Bytes	<u>_SYSID_</u>
Checksum	1 Byte	Checksum of the SYSID BIOS Entry Point Structure
Length	2 Bytes	Total length of SYSID BIOS Structure Table (Set to 011h).
SYSID BIOS Structure Table Address	4 Bytes	32 bit physical address of the beginning of the SYSID BIOS Structure Table. <b><i>This value is BYTE Aligned!!</i></b>
Number of SYSID BIOS Structures	2 Bytes	Total number of structures within the SYSID BIOS Structure Table.
SYSID BIOS Revision	1 Byte	Revision of the SYSID BIOS Extensions (Set to 00h).

#### Format of the SYSID BIOS structures:

Element	Length	Description
Header/Type	6 Bytes	<u>_????_</u>
Checksum	1 Byte	Checksum of the SYSID BIOS Structure
Length	2 Bytes	Total length of SYSID BIOS Structure
Variable Data Portion	?? Bytes	Depends on SYSID BIOS Structure Header/Type Field

#### Format of the UUID BIOS structure:

Element	Length	Description
Header/Type	6 Bytes	<u>_UUID_</u>
Checksum	1 Byte	Checksum of the UUID BIOS Structure
Length	2 Bytes	Total length of UUID BIOS Structure (Set to 0019h).
Variable Data Portion	16 Bytes	Actual UUID data (Initially set all bytes to 0FFh).

## 9.2.2 Programming

### 9.2.2.1 Purpose:

There are two GUID programming methods that can be supported by PXE bootstrap programs. (Note: BIOS support for writing GUIDs is not required for PXE compliance. However, it is straightforward, particularly if SMBIOS 2.1 is supported, and undeniably useful. BIOS vendors are encouraged to provide support for this capability.)

- Writing SMBIOS structures through the PnP function interface.
- Writing SYSID BIOS structures through the PnP function interface.

### 9.2.2.2 Writing SMBIOS structures through the PnP function interface

This method of writing and storing GUIDs is accomplished by making a SMBIOS call utilizing the Set DMBIOS Structure (52h) function.

The format of the Set SMBIOS Structure function is as follows:

```
short FAR (*entryPoint)(Function, dmiDataBuffer, dmiWorkBuffer, Control, dmiSelector, BiosSelector)
short Function; /* PnP BIOS Function 52h */
unsigned char FAR *dmiDataBuffer; /* Pointer to buffer containing new/change data */
unsigned char FAR *dmiWorkBuffer; /* Pointer to work buffer area for the BIOS */
unsigned char Control; /* Conditions for performing operation */
unsigned short dmiSelector; /* SMBIOS data read/write selector */
unsigned short BiosSelector; /* PnP BIOS readable/writeable selector */
```

The *dmiDataBuffer* parameter references a structure of the following format:

Offset	Field	Length	Description
00h	<i>Command</i>	BYTE	Identifies the structure-setting operation to be performed, one of: 00h A single byte of information is to be changed in the structure identified by StructureHeader 01h A word (two bytes) of information is to be changed in the structure identified by StructureHeader 02h A double-word (four bytes) of information is to be changed in the structure identified by StructureHeader 03h The structure identified by StructureHeader is to be added to the SMBIOS structure pool 04h The structure identified by StructureHeader is to be deleted from the SMBIOS structure pool 05h A string's value is to be changed in the structure identified by StructureHeader. 06h-0FFh Reserved for future assignment by this specification.
01h	<i>FieldOffset</i>	BYTE	For a structure change Command, identifies the starting offset within the changed structure's fixed data of the to-be-changed item. For a string-value change Command, identifies the offset within the structure's fixed data associated with the string's "number". This field is ignored for all other Commands.
02h	<i>ChangeMask</i>	DWORD	For a structure-change Command, identifies the ANDing mask to be applied to the existing structure data prior to applying the ChangeValue. The number of significant bytes within this area is defined by the Command. This field is ignored for all other Commands.

Offset	Field	Length	Description
06h	<i>ChangeValue</i>	DWORD	For a structure-change Command, identifies the data value to be ORed with the existing structure data – after applying the ChangeMask. The number of significant bytes within this area is defined by the Command. This field is ignored for all other Commands.
0Ah	<i>DataLength</i>	WORD	For a structure-add Command, identifies the full length of the to-be-added structure. The length includes the structure header, the fixed-length portion of the structure, and any string data which accompanies the added structure – including all null-terminators. For a string-value change Command, identifies the length of the string data (including the null-terminator); if the length is 1 (indicating that only the null-terminator is provided), the current string's data is deleted so long as the string's data-access rights are met. This field is ignored for all other Commands.
0Ch	<i>StructureHeader</i>	4 BYTES	Contains the structure header of the structure to be added, changed, or deleted.
10h	<i>StructureData</i>	Var	For a structure-add Command, contains the data to be associated with the SMBIOS Structure identified by the StructureHeader. For a string-value change Command, contains the string's data (the number of characters is identified by DataLength). This field is ignored for all other Commands.

The *dmiWorkBuffer* parameter references a work buffer for use by the BIOS in performing the request; the contents of the buffer are destroyed by the BIOS' processing. This work buffer must be read/write and sized to hold the entire SMBIOS structure pool, based on the information returned by Get SMBIOS Information (50h) function plus the size of any structure to be added by the request. For SMBIOS v2.0 implementations, the pool size is specified by the maximum of (*StructureSize \* NumStructures*) and (when *dmiStorageBase* is non-zero) *dmiStorageSize*; for v2.1 and later implementations, the pool size is specified by *dmiStorageSize*.

The *Control* flag provides a mechanism for indicating to the BIOS whether the set request is to take effect immediately, or if this is a check to validate the to-be-updated data.

*Control* is defined as:

Bit	0	0 = Do not set the specified structure, but validate its parameters. 1 = Set the structure immediately.
Bits	1:7	Reserved, must be 0.

If bit 0 of *Control* is 0, then the *dmiDataBuffer* values are checked for validity. If any are not valid, then the function returns DMI\_BAD\_PARAMETER; if any read-only field is modified, the function returns DMI\_READ\_ONLY. Validity checking is useful to determine if the BIOS supports setting a structure field to a particular value – or whether the BIOS supports writing to a specific structure field. For example, it may be useful for an OEM to determine beforehand whether the OEM's BIOS supports a "Reboot to Diagnostics Now" setting in an OEM-defined structure.

The protected-mode read/write selector *dmiSelector* has base equal to *dmiStorageBase* and a limit of at least *dmiStorageSize*, so long as the *dmiStorageBase* returned from Get SMBIOS Information (50h) function was non-zero.

The *BiosSelector* parameter enables the system BIOS, if necessary, to update system variables that are contained in the system BIOS memory space. If this function is called from protected mode, the caller must create a data segment descriptor using the 16-bit Protected Mode data segment base address specified in the Plug and Play Installation Check data structure, a limit of 64KB, and the descriptor must be read/write capable. If this function is called from real mode, *BiosSelector* should be set to the Real

mode 16-bit data segment address as specified in the Plug and Play Installation Check Structure. Refer to section 4.4 of the Plug and Play BIOS Specification revision 1.0a for more information on the Plug and Play Installation Check Structure and the elements that make up the structure.

This function is available in real mode and 16-bit protected mode.

**Returns:**

If successful - DMI\_SUCCESS

If an error occurred, the Error Code will be returned in AX. The FLAGS and all other registers will be preserved.

**Errors:**

- DMI\_BAD\_PARAMETER                    A parameter contains an invalid or unsupported value.
- DMI\_READ\_ONLY                        A parameter is read-only and differs from the present value – an attempt was made to modify a read-only value.
- DMI\_ADD\_STRUCTURE\_FAILED            The desired structure could not be added due to insufficient storage space.
- DMI\_INVALID\_HANDLE                 For an add (03h) Command, the structure handle present in the StructureHeader already exists or, for a change (00h to 02h and 05h) or delete (04h) Command, the structure handle does not exist.

**Example:**

The following example illustrates how the 'C' style call interface could be made from an assembly language module:

```

push    BiosSelector
push    dmiSelector
push    Control
push    segment/selector of dmiWorkBuffer    ;pointer to BIOS temporary buffer
push    offset of dmiWorkBuffer
push    segment/selector of dmiDataBuffer    ; pointer to structure
push    offset of dmiDataBuffer
push    SET_DMI_STRUCTURE                    ; Function number, 52h
call    FAR PTR entryPoint
add     sp, 16                               ; clean stack
cmp     ax, DMI_SUCCESS                      ; Successful?
jne     error                                 ; No, go handle error

```

**9.2.2.3 Writing SYSID BIOS structures through the PnP function interface**

This method of writing and storing these SYSIDs is accomplished by making a DMI BIOS call utilizing the DMI Control (54h) Function.

The format of the DMI control Function is as follows:

- Function*                    54h
- Sub Function*               4007h
- Data*                        See *Data* Structure below. Must have read \ write access.
- Control*                     Bit 0 set to 1 (perform operation immediately).
- dmiSelector*                Provided from Function 50h call.
- BiosSelector*               Provided from Function 50h call.

**Format of the *Data* parameter passed:**

Offset	Name	Length	Value	Description
00h	<i>type</i>	BYTE	Varies	Type of data to be written. See type definition table below.
01h	<i>length</i>	BYTE	Varies	Length in bytes of the data to be written.
02h	<i>idData</i>	DWORD	Varies	FAR * to actual data to be written.
06h	<i>LpDmiWorkBuffer</i>	DWORD	Varies	FAR * to a readable / writeable buffer at least the size of MinGPNVRWSize.
0Ah	<i>SecurityKey</i>	8 bytes	Varies	Security Key.

**Type** Specifies the type of Data (ID) to be written.

Type Description	type	length	idDATA	SecurityKey
Write UUID	00h	Must be 16d bytes	FAR * to a 16d byte buffer containing the UUID	Level 3 or above. The security key is NOT required if the current UUID is blank.
Write 1394 ID	01h	Must be 8d bytes	FAR * to a 8d byte buffer containing the 1394 Unique ID.	Level 3 or above. The security key is NOT required if the current 1394ID is blank.

**IdData** FAR pointer to a buffer containing the actual data to be written

**LpDmiWorkBuffer** FAR \* to a readable / writeable buffer at least the size of *MinGPNVRWSize*. *MinGPNVRWSize* value can be obtained by making a Get GPNV Information (Function 55h) call.

**SecurityKey** An 8 byte security key that meets or exceeds level 3 security (System Administrator level or above).

#### 9.2.2.3.1 Return Codes:

Upon a successful call the BIOS shall return DMI\_SUCCESS to the caller. If an error was made during the call, the BIOS shall return DMI\_BAD\_PARAMETER. If this control sub-function is not supported, the BIOS shall return DMI\_INVALID\_SUBFUNCTION to the caller.

If the caller does not provide a valid security key and the ID has already been written, the BIOS shall return a DMI\_READ\_ONLY error to the caller.

## 9.3 Remote Wake Up Source

### 9.3.1 Detection

Three methods of Remote Wake-Up source detection should be implemented by PXE compliant bootstrap programs.

- Reading SMBIOS structures through the PnP function interface (See section 9.2.1.1 above)
- Reading table-based SMBIOS structures (See section 9.2.1.2 above)
- Int 15h API

#### 9.3.1.1 Int 15h API

One of the ways a PXE compliant bootstrap program can detect the wake-up source is through an Int 15h API. This API is described below.

Determine source of system wake-up:

```

Enter:
    AX := 2307h
    BX := 5755h
    Int 15h
Exit:
    CF == 1 || AH != 0 Failure. Wake-up source detection not supported by BIOS.

    CF == 0 && AH == 0 Success. Wake-up source detection supported by BIOS.
    (CL & 7) == 6 Power switch.
    (CL & 7) == 5 LAN
    (CL & 7) == 4 COM1 ring
    (CL & 7) == 3 Timer
    Other values for (CL & 7) are reserved.
    All other register contents must be preserved.
  
```

NOTE: The information returned by this call may be destroyed by reading it. If you do not want to destroy information, you should try to write it back (see 9.3.2.1 below).

### 9.3.2 Programming

One method of wake-up source programming should be implemented by PXE compliant bootstrap programs.

- Int 15h API

#### 9.3.2.1 Int 15h API

If the determined source of system wake-up API call performs a destructive read, the BIOS must also implement the API call write source of system wake-up. This API is described below.

Write source of system wake-up:

```
Enter:
    AX := 2308h
    BX := 5755h
    CL (bits 2-0) := 6   Power switch.
    CL (bits 2-0) := 5   LAN
    CL (bits 2-0) := 4   COM1 ring
    CL (bits 2-0) := 3   Timer
    CL (bits 7-3) := 0
    Int 15h
Exit:
    CF == 1 || AH != 0   Failure.   Write wake-up source not supported by
                                BIOS.
    CF == 0 && AH == 0   Success.   Write wake-up source detection supported
                                by BIOS.
```

All other register contents must be preserved.

## 9.4 Bootstraps

Three common option ROM bootstrap mechanisms are used in PXE compliant boot ROMs:

- Plug and Play/BIOS Boot Specification (PnP/BBS)
- Int 18h
- Int 19h

PnP/BBS is required to be supported in the BIOS by the “Wired for Management Baseline, Version 1.1a”. If you are designing your PXE option ROM to work in legacy BIOS’s, you will also need to support bootstrap interrupts 18h and 19h.

NOTE: Bootstrap interrupts 18h and 19h are mutually exclusive.

### 9.4.1 Plug and Play/BIOS Boot Specification (PnP/BBS)

#### 9.4.1.1 How to tell if your option ROM is running in a PnP/BBS compatible BIOS

If your option ROM is going to be included in a BIOS image and you know that BIOS supports PnP/BBS, then you can write your option ROM initialization and boot code assuming PnP/BBS will be supported.

If you are writing an option ROM that is going to be included in a NIC that can be placed in any BIOS, you can detect the presence of PnP/BBS by issuing PnP functions *Get Version and Installation Check (60h)* and *Get Boot First (65h)*. If either of these calls are successful, you are in a PnP/BBS compatible BIOS. At this time, these calls are not required by the PnP/BBS, and your option ROM could be running in a PnP/BBS BIOS and have no knowledge of this.

Until the PnP/BBS gets updated so there is a definitive detection mechanism, your option ROM cannot detect if it is running in a PnP/BBS BIOS.



### 9.4.1.2 What is needed to make a NIC option ROM PnP/BBS compatible

The information is covered in detail in the “Plug and Play/BIOS Boot Specification, Version 1.01” and the “Plug and Play BIOS Specification, Version 1.0A”. This section highlights what is necessary to make an option ROM PnP/BBS compatible.

#### PnP Option ROM Header (PnP/ORH)

Offset	Size	Value	Description
00h	BYTE	55h	Signature byte 1.
01h	BYTE	AAh	Signature byte 2.
02h	BYTE	Varies	Option ROM length in 512-byte blocks.
03h	4 BYTES	Varies	Initialization entry point.
07h	17 BYTES	Varies	Reserved.
18h	WORD	Varies	Offset to PCI data structure.
1Ah	WORD	Varies	Offset to expansion header structure.

#### PnP Expansion Header (PnP/EH)

Offset	Size	Value	Description
00h	BYTE	'\$'	Signature byte 1.
01h	BYTE	'P'	Signature byte 2.
02h	BYTE	'n'	Signature byte 3.
03h	BYTE	'P'	Signature byte 4.
04h	BYTE	01h	Structure revision.
05h	BYTE	Varies	Length (in 16 byte increments).
06h	WORD	Varies	Offset of next header (0000h if none).
08h	BYTE	00h	Reserved.
09h	BYTE	Varies	Checksum.
0Ah	DWORD	Varies	Device identifier.
0Eh	WORD	Varies	Pointer to manufacturer string (Optional).
10h	WORD	Varies	Pointer to product name string (Optional).
12h	3 BYTES	Varies	Device type code.
15h	BYTE	Varies	Device indicators.
16h	WORD	Varies	Boot Connection Vector (BCV), 0000h if none.
18h	WORD	Varies	Disconnect Vector (DV), 0000h if none.
1Ah	WORD	Varies	Bootstrap Entry Vector (BEV), 0000h if none.
1Ch	WORD	0000h	Reserved.
1Eh	WORD	Varies	Static resource information vector (0000h if none).

#### 9.4.1.2.1 PnP/BBS Initialization Code Status

PnP/BBS option ROM initialization code needs to return status to the BIOS that an Initial Program Load (IPL) device has been attached. See the table below for AX register contents. All undefined bits must be set to zero.

AX Bit	Description
8	1 = IPL Device supports INT 13h Block Device format
7	1 = Output Device supports INT 10h Character Output
6	1 = Input Device supports INT 9h Character Input
5:4	00 = No IPL device attached 01 = Unknown whether or not an IPL device is attached 10 = IPL device attached (RPL devices have a connection). 11 = Reserved
3:2	00 = No Display device attached 01 = Unknown whether or not a Display device is attached 10 = Display device attached 11 = Reserved
1:0	00 = No Input device attached 01 = Unknown whether or not an Input device is attached 10 = Input device attached 11 = Reserved

#### 9.4.1.2.2 Boot Entry Vector (BEV)

The BEV replaces the bootstrap interrupt 18h, or 19h, code. If this option ROM is selected as the boot device, the BEV will be entered via a far-call from the system BIOS. If the BEV decides that it does not want to, or cannot, boot it needs to clean up and return control to the system BIOS via a far-return.

### 9.4.2 Int 18h

Historically, in legacy BIOSes, Int 18h was used to invoke the built-in ROM BASIC. This was to be done by any option ROM, or bootstrap program, that was invoked by Int 19h. It was also done by the system BIOS, if there were no option ROMs or bootstrap programs to run.

Before the Plug and Play/BIOS Boot Specification was defined, some BIOS vendors added to their BIOSes the ability to select which devices could be boot devices, and in what order to try to boot them.

At first, you could select from Floppy and Hard disk. Then, CD-ROMs were added to the list. Finally, Network was added. The limitation here, was that you could only have one of each type of bootable device. If you had two bootable network cards, there was no way to define the order that they should be tried; or even which one should be tried.

#### 9.4.2.1 How the BIOS detects network boot devices

BIOSes that support Int 18h boot order detected network boot devices by checking to see if the option ROM would hook Int 18h during initialization. If an option ROM hooks Int 18h, and Network was the first bootable device in the boot order list, the option ROM that hooked Int 18h would be called by the system BIOS. There was no checking to see if the option ROM really was a network device.

### 9.4.3 Int 19h (Legacy)

If your option ROM is being designed to work in legacy BIOSes, it will need to hook bootstrap interrupt 19h during option ROM initialization, before returning control to the system BIOS. If there is more than one device that hooks Int 19h, the last device that does so becomes the boot device.

## 9.5 Memory Management

Historically, in legacy BIOSes, Int 18h was used to invoke the built-in ROM BASIC. This was to be done by any option ROM, or bootstrap program, that was invoked by Int 19h. It was also done by the system BIOS, if there were no option ROMs or bootstrap programs to run.

Before the Plug and Play/BIOS Boot Specification was defined, some BIOS vendors added to their BIOSes the ability to select which devices could be boot devices, and in what order to try to boot them.

At first, you could select from Floppy and Hard disk. Then, CD-ROMs were added to the list. Finally, Network was added. The limitation here, was that you could only have one of each type of bootable device. If you had two bootable network cards, there was no way to define the order that they should be tried; or even which one should be tried.

### 9.5.1 POST Memory Manager

If POST Memory Manager (PMM) is available, it must be used by PXE option ROMs. How PMM is detected and used is covered in the "POST Memory Manager Specification, Version 1.0".

#### 9.5.1.1 Detecting PMM Services

A data structure exists within the BIOS for PMM presence detection. The PMM Structure is located in the system BIOS address space on a paragraph boundary between segment addresses E000h and FFFFh. There will be only one PMM Structure in the system BIOS. The presence of the PMM Structure indicates that the PMM Services are present and available for calling. For now, the only PMM service used by PXE option ROMs is pmmAllocate.

#### PMM Structure

Offset	Name	Size	Value	Description
00h	Signature	4 Bytes	'\$PMM'	PMM Structure signature. This signature starts on a paragraph boundary. '\$' is at byte 0.
04h	Revision	1 Byte	01h	Structure revision.
05h	Length	1 Byte	varies	Length of this structure in bytes.
06h	Checksum	1 Byte	varies	Checksum update field. For the structure to be valid, the sum of all bytes, including this one, must be zero.
07h	EntryPoint	4 Bytes	varies	Far pointer to PMM API entry point. This call is only available during option ROM initialization.
0Bh	Reserved	5 Bytes	0	Reserved

#### 9.5.1.2 pmmAllocate

The pmmAllocate function attempts to allocate a memory block of the specified type and size, and return the address of the memory block to the caller. The memory block is a contiguous array of paragraphs whose size is specified by the length parameter. The contents of the allocated memory block are undefined.

```
UINT32 (far *pmm.EntryPoint)(
    UINT16 function,           // 0 for pmmAllocate
    UINT32 length,           // Number of 16-byte paragraphs to allocate
    UINT32 handle,           // 32-bit handle to assign to memory block
    UINT16 flags              // Bit flags specifying options
);
```

#### function

0 for pmmAllocate. Invalid values for the function parameter (3..65535) cause an error value of FFFFFFFFh to be returned, signaling that the function is not supported.

**length**

The size of the requested memory block in paragraphs. If the length is zero, no memory is allocated and the value returned is the size of the largest memory block available for the memory type specified in the flags parameter. The alignment bit in the flags register is ignored when calculating the largest memory block available.

**handle**

A client-specified identifier to be associated with the allocated memory block. A handle of FFFFFFFFh indicates that no identifier should be associated with the block. Such a memory block is known as an 'anonymous' memory block and cannot be found using the pmmFind function. If a specified handle for a requested memory block is already used in a currently allocated memory block, an error value of zero is returned.

**flags**

A bitmap used by the client to designate options regarding memory allocation.

Bits	Field	Value	Description
1:0	flags.MemoryType	1..3	0 = Invalid 1 = Conventional memory (0 to 1MB) 2 = Extended memory (1MB to 4GB) 3 = Conventional or extended memory
2	flags.Alignment	0..1	0 = No alignment 1 = Use alignment from the length parameter
15:3	flags.Reserved	0	Reserved. Must be zero.

**9.5.2 Int 15h**

If PMM is not available, PXE option ROMs should assume that they are in a legacy BIOS and allocate extended memory by chaining Int 15h, service AH=88h (Get Extended Memory Size), and reduce the amount of memory reported by this service.

This method of extended memory management is not recommended, because not all option ROMs and applications will respect the reduced memory size.

---

## 10. Third Party Design Support

---

The following independent SW vendors (ISV) are both available to provide design support for PXE compliant boot roms.

Lanworks Technologies Inc  
Contact: Mark Kuess  
2425 Skymark Ave  
Mississauga, ON,  
CANADA L4W 4Y6  
Tel: (905) 238-5528 ext 166 Fax: (905) 238-9407  
Email: [mark@lanworks.com](mailto:mark@lanworks.com)  
Internet: <http://www.lanworks.com/>

Dirk Koeppen Incom-Beratungs-GmbH  
Contact: Dirk Koeppen  
Holzwiesenweg 22  
D-63073 Offenbach,  
Germany  
Tel: 011 49 69 89 3000 Fax: 011 49 69 89 3004  
Email: [dirk@incom.de](mailto:dirk@incom.de)  
Internet: <http://www.incom.de/>

In addition, Incom-Beratungs-GmbH provides design support for PXE services and OS loaders.

---

## 11. LSA2 Operation and Troubleshooting FAQs

---

Q1. How does the LSA2 boot PROM begin execution?

A1. LSA2 is implemented as a standard PC/AT x86 boot PROM. It is called by the system BIOS during option ROM scan. If it is running in a system that supports the Plug and Play (PnP) BIOS Boot Specification (BBS), it returns control to the system BIOS identifying itself as a valid Initial Program Load (IPL) device. If it is not in a PnP/BBS system, it hooks interrupt vector 18h and returns control to the BIOS.

Q2. Why are there two different versions of the LSA2 for the Intel 82557/82558-based network controllers?

A2. The first version of the LSA2 (e100b.nic) is designed to run from the boot PROM of a NIC. This code will only take 2 KB of upper memory. When the device boots, code is copied from the boot PROM into the top of free base memory.

The second version of the LSA2 (e100b.ld) is designed to run from the BIOS boot PROM on the motherboard, like a system with built-in video. This code will take 23 KB of upper memory during option ROM scan, all but 2 KB will be copied into the top of extended memory (see Q3) before control is returned to the BIOS. When the device boots, code is copied from extended memory into the top of free base memory.

Q3. Why does the BIOS lock up during option ROM scan after running e100b.ld?

A3. Some BIOS's are not restoring the correct Global Descriptor Table (GDT) when the Protected Mode Copy (Int 15h, AH=87h) service is run during option ROM scan. These BIOS's are usually in a flat memory model during option ROM scan, and the Protected Mode Copy service returns with the processor in real-mode.

You can use the NIC version (e100b.nic) to test the rest of the LSA2 code with a BIOS that has the Protected Mode Copy bug.

To fix this bug, the BIOS needs to restore the correct GDT while in option ROM scan.

Q4. How is LSA2 PROM selected to be the boot device?

A4. The LSA2 is a standard PCI/PnP option ROM.

If the LSA2 is placed into a BIOS that supports PnP/BBS (BIOS Boot Specification), the BIOS should insert a network boot device into the boot order list.

If the LSA2 is placed into a BIOS that does not support PnP/BBS, that BIOS must support network devices that hook interrupt vector 18h. After the LSA2 returns control to the BIOS (at the end of the option ROM initialization call), the BIOS should check to see if Int 18h has changed. If it has, the BIOS should assume that a network boot device has hooked Int 18h and give the user the ability to select network boot in the CMOS setup. Normally, a BIOS gives the user the ability to boot drive A:, C:, and the CD-ROM drive. It now needs to add network to the list of boot devices.

Q5. Our BIOS supports PnP/BBS. Why does the LSA2 still hook bootstrap interrupt 18h (or 19h)?

A5. If you are putting the LSA2 into a BIOS that supports PnP/BBS v1.01 and the LSA2 is still hooking bootstrap interrupt 18h, then the LSA2 cannot detect PnP/BBS support in your BIOS.

Please verify that:

1. PnP installation check structure is paragraph (16 byte) aligned between addresses E0000h and FFFF0h.
2. The PnP installation check structure is valid during option ROM initialization.

3. The BIOS supports function 60h, Get Version and Installation Check. This is the only way that LSA2 can detect PnP/BBS support.

If you are implementing your own PXE boot ROM image that will be going into a LOM (LAN-on-motherboard), and you know the BIOS supports PnP/BBS, you do not have to use function 60h to check for PnP/BBS. This is only a requirement for NIC boot ROM images. They can be placed into systems that have a PnP BIOS, but do not support PnP.

Q6. Does LSA2 use POST Memory Manager if it is supported in the system BIOS?

A7. Yes, the LOM version (e100b.ld) will use POST Memory Manager if it is available. LSA2 will allocate a 64KB block using POST Memory Manager, if this allocation fails, it will fall back to hooking the extended memory size service (Interrupt 15h, AH=88h) and reducing the reported size of extended memory by 64KB.