EFI How To Guide

Draft for Review

Version 1.0
March 15, 2004

Intel order number xxxxxx-001

Revision History

Revision	Revision History	Date
1.0	First release.	3/15/04

intel

Contents

Tables

1 Introduction

Overview

This help system provides information on how to perform numerous activities within EFI. The topic areas that are discussed include the following:

- Disk Utilities
- Boot Manager
- Configuration Driver Boot
- Console Redirection
- EFI Shell and File System
- EFI Drivers
- Identification
- USB
- Network Interface

For more information on the EFI Shell commands that are referenced in this help system, see the *EFI 1.1 Shell Commands Specification,* which is available in the EFI Sample Implementation from the EFI web site.

Available Tools and References

The following tools, utilities, and sources of information are referenced throughout this help system. See Available EFI Documentation and Related Information from Intel Corporation in the master EFI Documentation help system for the URLs. All of the following are available from the EFI web site at:
http://developer.intel.com/technology/efi/

Documentation

- *Extensible Firmware Interface Specification*, version 1.10
- *EFI 1.10 Specification Update*, version -001 (11/26/03)
- *EFI 1.1 Shell Commands Specification*, version 0.3: Available in the EFI Sample Implementation

Tools and Utilities

- EFI Application Toolkit, version 1.10.14.62
- EFI Disk Utilities, which contains the **`Diskpart`**, **`Efichk`**, and **`Efifmt`** utilities
- EFI Sample Implementation, version 1.10.14.62
- EFI Shell: Available in the EFI Sample Implementation

Typographic Conventions

In addition to the typographic conventions described in the EFI Documentation master help system, this help system also uses the typographic and illustrative conventions described below:

`Bold Monospace` — In examples showing how to use a command, words in a **`Bold Monospace`** typeface that is green indicate text that you need to type at the command prompt or select out of a list of options. Occasionally this convention may instead highlight information within the example for you to note and is not text you need to type or select. This typeface typically appears within separate paragraphs that have a `Plain Monospace` typeface.

2
Disk Utilities

How to Partition an EFI GPT Disk

Partitioning an EFI GUID Partition Table (GPT) disk requires the **diskpart.efi** program from the EFI web site utilities.

This example is a step-by-step guide for creating a disk partition. Type **diskpart** or **diskpart.efi** from the EFI Shell to run the program. Then, type the following commands in green below, in the order listed, to partition the disk. Some screen output has been omitted for brevity.

```
fs0:>diskpart
DiskPart Version 1.0
Based on EFI core release Version 1.2.1.0
```

`DiskPart>help` ←	Displays a list of commands for DiskPart.
`DiskPart>Select 0` ←	Disk number depending on which drive you want.
`DiskPart>clean [all]` ←	May need to wipe all partitions for OS install. **clean 0** may also work if there is only one partition.
`DiskPart>new gpt` ←	Selects the GPT partition type.
`DiskPart>create` ←	Without parameters a help list for the create command will display.
`DiskPart>create name=partition type=efisys size=100 attr=001`	**name** is the partition name. **type** must be **efisys**. **size** is in megabytes. **attr=001** enables the partition to be present. (Bug will be removed in a later utility.)
`DiskPart>Inspect` ←	Should show you a GUID with a partition present. Repeat for all GPTs desired.
`DiskPart>exit`	
`fs0:>map -r` ←	Registers new partitions with the EFI Shell **map** command.

The disk partition will now have a unique GUID that will show up in the device path.

Help List for DiskPart

The following table lists the commands that you can use in **`Diskpart.efi`**.

Available Commands for DiskPart

Command	Description
LIST	Shows a list of partitionable disks.
SELECT <number>	Selects a disk (spindle) on which to work.
INSPECT	Dumps the partition data on the selected spindle.
CLEAN [ALL}	Cleans all data off the disk. Note that this command **destroys the data.**
ALL	Writes zeroes to the whole disk, without just first and last MB.
NEW [MBR | GPT]	Creates either a Master Boot Record (MBR) or GPT partition.
FIX	Fixes a fragmented partition.
CREATE	Creates a new partition.
DELETE	Deletes a partition.
HELP	Prints this screen. Type **`<cmd> Help`** for details on a command.
H	Prints this screen. Type **`<cmd> H`** for details on a command.
?	Prints this screen. Type **`<cmd> ?`** for details on a command.
EXIT	Exits the program.
SYMBOLS [VER]	Shows a list of predefined type GUIDs. VER is for verbose.
REM	Prints a remark.
MAKE [LIST] [make script]	**`make list`** lists the make script. **`make scriptname args`** runs the script.
DEBUG (null)	Changes the debug level. The syntax is **`DEBUG #`**, where the # will be the debug level used for the **`DiskPart`** utility. The null default will be level 0.
ABOUT	Gives information about this version.

Click here to return to How to Partition an EFI GPT Disk.

Help List for Create Command in DiskPart

To view the help for **DiskPart**, type the following in green at the prompt:

```
DiskPart>create help
```

The following options are available for **DiskPart**, which are explained in the table below:

```
create [help]
```

or

```
create name=namestr (type=typename | typeguid=guid) [offset=ooo]
[size=sss] [attr=aaaa] [ver]
```

Options for the DiskPart Create Command

Option	Description
name=namestr	Required. **namestr** may be in quotation marks (e.g., **name="p01"**).
type=typename	One of **type=typename** or **typeguid=guid** is required. **Typename** is one of the types listed by the **symbols** command.
typeguid=guid	One of **type=typename** or **typeguid=guid** is required. GUID is of form XXXYYYZZZ.
offset=ooo	Optional. **ooo** is hexadecimal block offset. If **ooo** is absent, the partition will start at the end of the last partition.
size=sss	Optional. **sss** is the size in megabytes (decimal number). If **sss** is 0, **sss** is greater than the free space, or the option is absent, the partition will fill end of disk.
attr=aaa	Optional. **aaa** is a 64-bit hexadecimal string of attribute flags. Attributes bits [0..31] are reserved for EFI. Bits [32..63] are partition-type specific. Bit 0 is required for the platform to function properly
ver	Optional command to turn on the verbose status.

Examples

```
create name="a partition" type=MSDATA size=0 attr=1
create name=part2 type=efisys size=400
```

Click here to return to How to Partition an EFI GPT Disk.

How to Format an EFI GPT Disk

Formatting an EFI GUID Partition Table (GPT) disk requires the **efifmt.efi** program from the EFI web site utilities.

This example is a step-by-step guide for formatting a disk partition. Type **efifmt** or **efifmt.efi** at the Shell prompt to run the **efifmt.efi** program. Then, type the text indicated in green below to format the disk.

```
fs0:>efifmt
```

Displays help for the **efifmt** utility.

```
fs0:>efifmt blk# /FS:FAT32 /Q
```

Where:

blk# is the block device shown in the **map** command with *hd/sigxxx*.

/FS:FAT32 (must be in all caps) formats the FAT32 partition.

```
fs0:>efifmt blk0 /FS:FAT32 /Q
EFI Disk Format Version 1.0
Based on EFI Core Version 1.2.12.38
WARNING, ALL DATA ON EFI DEVICE
blk1 WILL BE LOST!
Proceed with Format (Y/N)? Y
QuickFormatting 4000M
Initializing the File Allocation Table (FAT)...
Volume label (11 characters, ENTER for none)? EXIT
Format complete.
          4186660864 bytes total disk space.
          4186656768 bytes available on disk.
                4096 bytes in each allocation unit.
             1022133 allocation units available on disk.
                  32 bits in each FAT entry.
Volume Serial Number is 00008CBE-000069C9
fs0:>map -r
```

Should now show **fsX** with the new file system.

Help List for Efifmt

The table below lists the options that you can use with the **efifmt.efi** program.

```
EFI Disk Format Version 1.0
Based on EFI Core Version 1.2.12.38
Formats a disk for use with EFI.

EFIFMT device [/FS:file-system] [/V:label] [/Q] [/A:size]
```

Options for Efifmt

Option	Description
device	Specifies the EFI device to format without a colon. Example: **blk0**.
/FS:filesystem	Specifies the type of the file system (FAT, FAT32).
/V:label	Specifies the volume label.
/Q	Performs a quick format.
/A:size	Overrides the default allocation unit size. Default settings are **strongly** recommended for general use, instead of using the **/A** override. FAT supports 512, 1024, 2048, 4096, 8192, 16K, and 32K bytes per sector. FAT32 supports 512, 1024, 2048, 4096, 8192, 16K, and 32K bytes per sector. Note that the FAT and FAT32 file systems impose the following restrictions on the number of clusters on a volume: • FAT: Number of clusters must be less than or equal to 65526. • FAT32: The number of clusters must be greater than 65526 and less than 268435446. The format will stop processing if the above requirements cannot be met using the specified cluster size.

Click here to return to How to Format an EFI GPT Disk.

How to Make an EFI Bootable CD-ROM/DVD-ROM

CDs and DVDs must be burned using the El Torito format. Ahead Software, Inc. has several Nero* CD-ROM burning programs that allow you to burn CDs and DVDs using this format. You can use any of the following programs:

- Nero Enterprise Edition
- Nero 5.5 Edition
- Nero 6 Ultra Edition (the latest version that is available)

Nero Enterprise Edition contains built-in wizards for burning EFI bootable and visible CD-ROMs.

Nero 5.5 Edition does not have built-in wizards to help you burn CD-ROMs, but the instructions below will help you burn a CD-ROM for EFI. It is recommended that you use Nero Enterprise Edition or Nero 6 Ultra Edition, which are much more user friendly than the Nero 5.5 Edition.

Go to the Nero web site (http://www.nero.com*) to purchase a copy of the software. Note that only Nero 6 Ultra Edition will be available and instructions are not covered in this guide

El Torito, Operating Systems, and EFI

For Linux*-based systems, other programs are also available to burn CDs in the El Torito format; this topic does not cover these programs, however.

CDs that are burned with the El Torito format are visible to both operating systems (Windows* and Linux) and EFI.

Instructions for Burning EFI Bootable CD-ROMs

Click the links below for instructions on burning EFI visible/bootable CD-ROMs using the following Nero products:

- Nero Enterprise Edition
- Nero 5.5 Edition
- Nero 6 Ultra Edition

NOTE

Intel has not yet tested Nero 6 Ultra Edition and cannot provide instructions, but it should behave similar to Nero Enterprise Edition.

Click the links below for additional instructions on burning CD-ROMs for the following:

- Itanium®-based systems
- IA-32 processor-based systems

Using Nero* Enterprise Edition to Burn EFI Visible/Bootable CD-ROMs

Do the following to burn an EFI visible/bootable CD-ROM or DVD-ROM using Nero* Enterprise Edition:

1. Start Nero Enterprise Edition.
2. Close any wizard that pops up when the program started.
3. On the top menu, click **File > New** for a new compilation.
4. In the upper left pane of the **New Compilation** window, select **CD-ROM** or **DVD-ROM.**
5. In the lower left pane of the **New Compilation** window, select **CD-ROM (EFI boot).**
6. In the **New Compilation** window, click the **Label** tab. Fill in the CD-ROM volume label information and any other relevant text you want on the CD-ROM. The top pull-down should say **ISO9660.**
7. Click the **New** button in the **New Compilation** window, which should open a left-hand window pane with the contents of what will be burned on the CD-ROM/DVD-ROM. The right-hand pane should be the **File Browser.**
8. From the **File Browser** pane, right-click on any files/directories you want and drag them to the left-hand pane.
9. If you want the CD-ROM to be bootable, you must copy the boot file from the **File Browser** pane into the following directory and rename it as indicated:
10. For Itanium®-based systems: **`\EFI\BOOT\BOOTIA64.efi`**
11. For IA-32 processor-based systems: **`\EFI\BOOT\BOOTIA32.efi`**
12. After all the files have been copied to the left-hand pane, click the **Burn** button to burn the CD-ROM. The bottom bar will tell you how much of the CD-ROM will be used by the files that you have put into the left-hand pane or NEW FATISO1.

Using Nero* 5.5 Edition to Burn EFI Visible/Bootable CD-ROMs

Do the following to burn an EFI visible/bootable CD-ROM or DVD-ROM using Nero* 5.5 Edition:

1. Before burning the CD-ROM, you must partition a hard drive as FAT32. The size of the drive partition must be less than the capacity of a CD-ROM (approximately 640 MB) using the CD-ROM burning PC.

   It is best to create multiple partition sizes on the disk because the burn time of the CD-ROM will be proportional to the entire image of the source hard drive partition. (i.e., 100 MB, 200 MB, 300 MB, ... 600 MB). You will burn the full size of the disk partition on the CD-ROM regardless of how many files you put on the FAT32 partition.

   You can also use a USB flash device as the source partition. You will need to use a USB flash device that is big enough to hold all the files you want to burn on the CD-ROM. You will also need to format the USB flash device as a FAT partition and make sure that any security encryption software is not enabled on the USB flash device (i.e., disable the security driver). To format a flash device as a FAT partition, you can use Windows* Explorer. The USB flash device entry should be *D: Removable device* (or something similar). Right-click on the USB flash device in the Windows Explorer window. Use the pull-down option **Format** and make sure the **Quick format** box is not checked. When done formatting, use Windows Explorer to copy the files you want on the CD-ROM onto the USB flash device. Choose a partition that is large enough to contain all the files you want on the CD-ROM (must be less than 640 MB).

2. Recall the letter of the partition (e.g., d: or e:) to which you copied all of your files. According to the *EFI Specification*, the system will attempt to boot from a removable media `FilePath` by adding a default file name in the form of **`\EFI\BOOT\BOOT{machine-type short name}.EFI`**, where **`{machine-type short name}`** defines a PE32+ image format architecture. Each file contains only one EFI image type, and a system may support booting from one or more image types.

   If you want the CD-ROM to be bootable, you must copy the boot file into the following directory and rename it as indicated:

   - For Itanium®-based systems: **`\EFI\BOOT\BOOTIA64.efi`**
   - For IA-32 processor-based systems: **`\EFI\BOOT\BOOTIA32.efi`**

   For example, if you wanted the CD-ROM to boot to the EFI Shell, copy **`nshell.efi`** from the EFI Application Toolkit and rename it **`\EFI\BOOT\BOOTIA64.EFI`** on the FAT32 partition from which you are going to burn the CD-ROM. In this case, if the CD-ROM was selected from the EFI boot manager, you would boot to the truncated version of the Shell that you copied from the Toolkit. If **`startup.nsh`** was also included at the root level on the partition, then **`startup.nsh`** would be processed as well.
3. Start Nero Burning ROM on the PC. Make sure you have set up Nero to select the CD-ROM burner and set up its preferences.

   When you start Nero, it should automatically open a **New Compilation** window with the tabs **Info**, **Multisession**, **ISO**, **Label**, and others showing. If a new compilation does not start automatically, go to **File > New** on the menu bar.
4. In the left-hand column of the **New Compilation** window, use the vertical slider and select **CD-ROM (Boot).** This option should automatically select the **Boot** tab and pick the first partition that is FAT32 and small enough to fit a 640 MB CD-ROM on it. You may have multiple FAT32 partitions that are less than 640 MB, so there may be more than one.

   Select the **Bootable logical drive** to which you copied all of your EFI files (the partition that has the **`\EFI\BOOT\BOOTIA64.EFI`** file on it; see step 2 above). The program may give you a few selections. Make sure you select the drive with the partition that you created on the drive that you added above for the EFI boot files. Typically, you would want this partition to be 640 MB for a full-sized CD-ROM.

   On Nero versions newer than 5.5.19.x, do the following:

   - Enable the **Expert** setting.
   - Set **Kind of emulation** to **No emulation.**
   - Fill in the boot message with what you want the user to see when the CD-ROM is booting.
   - Set the platform identifier (if this box is there) to **EFI boot partition.**
5. Click the **New** button. A new window named **New** should appear, which is the destination window.

   Open a **File Browser** window for the source files if one is not already open (this button is the yellow folder icon next to the **?**).

   Drag and drop the desired files to the destination CD-ROM image.

   With the right button on the mouse, select all the desired files on the FAT32 partition that were created in step 2 above from the **File Browser** window and drag them to the **New** window

(right-hand column side). Make sure that **\EFI\BOOT\BOOTIA64.EFI** is at the root of the CD-ROM; it should not be in a subdirectory.

When you release the mouse, a **Filter Files** box will pop up and allow you to filter certain undesirable directories or files from getting onto the CD-ROM or destination image (for example, you generally would not want the **Recycled** folder on your CD-ROM). Click **OK** if you want all the files you have selected with the mouse to show up in the **New** CD-ROM folder.

Make sure a blank CD-ROM is in the burner.

When you have finished dragging and dropping all the files/folders you want on the CD-ROM, select **File >Write CD-ROM** (or select the **Write** icon).

A **Write CD** window should open.

Select the **Boot** tab and make sure it is going to burn from the drive partition to which you copied all the EFI files (in step 2 above).

Select the **Burn** tab and select the maximum write speed that will work for your burner.

It is recommended that you have the following options selected or not selected, as indicated:

- **Determine max speed:** Not selected.
- **Simulation:** Not selected.
- **Write:** Selected.
- **Disc-at-once:** Selected.
- **Finalize CD:** Selected. You will not be able to add new files to the CD-ROM, but using **Finalize CD** will make the CD-ROM more compatible with many of the older CD-ROM players that might be used in a system.

Select the **Label CD** tab and type in a name for the CD-ROM. **ISO9660** should be selected in the top tab box (default).

6. Click the **Write** button to burn the CD-ROM.

Making an EFI Bootable CD-ROM on Itanium®-Based Systems

On an Itanium®-based system, if a CD-ROM is burned with a file on it named **\EFI\BOOT\BOOTIA64.EFI**, the EFI boot manager will automatically execute the **Bootia64.efi** file if the CD-ROM device path is selected from the boot manager.

Regardless of what file you want to execute (for example, **nshell.efi**), simply rename it **Bootia64.efi** and place it in the **\EFI\BOOT** directory on the CD-ROM or DVD-ROM.

Making an EFI Bootable CD-ROM on IA-32 Processor-Based Systems

For IA-32 processor-based systems with EFI, put the file on the CD-ROM in **`\EFI\BOOT\BOOTIA32.EFI`**.

A CD-ROM can have both of the following files:

- **`Bootia32.efi`**
- **`Bootia64.efi`**

This CD-ROM would be bootable by both Itanium®-based systems and IA-32 processor-based systems with EFI.

3
Boot Manager

How to Add an Executable to the Boot Menu

NOTE

Other OEM boot managers may have different boot menus. This example is only for the reference EFI boot manager that is included in the EFI Sample Implementation code from the EFI web site.

This example will add a boot option to execute the **`ia64ldr.efi`** OS loader on the SCSI hard drive. This procedure can be done for any **`.efi`** executable file, not just the OS loader.

To add an executable to the boot menu, do the following:

1. Start the EFI boot manager and select the **Boot option maintenance menu.**
2. At the boot option maintenance menu, select **Add a boot option.**
3. Select a volume from a possible boot device list.
4. Enter a text description of the new boot option.

The resulting boot menu should then show the new boot option.

The following examples are from screen snapshots from the reference EFI boot manager and illustrate the procedure listed above.

Boot Manager

Select the option in green below and hit the enter key to go to the boot option maintenance menu.

```
EFI Boot Manager ver 1.10 [14.62]
Please select a boot option
Floppy/Pci(1F|1)/Ata(Primary,Slave)
CD/DVD ROM/Pci(1F|1)/Ata(Primary,Master)
Hard Drive/Pci(6|2|0)/Scsi(Pun0,Lun0)/HD
(Part1,Sig8983DFE0-F474-...)
Network Boot/Pci(1|0|0)/Mac(0002B300076A)
EFI Shell [Built-in]
Boot option maintenance menu

Use ▲ and ▼ to change option(s). Use Enter to select an option.
```

Boot Option Maintenance Menu

At the boot option maintenance menu, select the option in green below:

```
BIOS Version S870BN4A.86B.0882.P06.0303311722
Main Menu. Select an Operation

Boot from a File
Add a Boot Option
Delete Boot Option(s)
Change Boot Order

Manage BootNext setting
Set Auto Boot TimeOut

Cold Reset
Exit

Timeout-->[7] sec SystemGuid-->[01034B34-47E9-D711-BEA6-0002B300076A]
SerialNumber-->[1234567890123 ]
```

Possible Boot Device List

Select the volume that you want to add to the boot menu, as shown in green below:

```
BIOS Version S870BN4A.86B.0882.P06.0303311722

Boot From a File. Select a Volume

NO VOLUME LABEL [Acpi(PNP0A03,1)/Pci(1F|0)/Pci(2|0)/Scsi
Removable Media Boot [Acpi(PNP0A03,0)/Pci(1F|1)/Ata(Primary,Slav
Removable Media Boot [Acpi(PNP0A03,0)/Pci(1F|1)/Ata(Primary,Mast
Load File [VenHw(6D9FEEB1-E585-11D3-BC22-0080C73C8881)]
Load File [VenHw(6D9FEEB1-E585-22D3-BC22-0080C73C8881)]
Load File [EFI Shell [Built-in]]
Load File [Acpi(PNP0A03,0)/Pci(1E|0)/Pci(0|0)/Mac(0002B300076A)]
Legacy Boot
Exit
```

Select the directory that contains the file you want to add, as shown in green below:

```
BIOS Version S870BN4A.86B.0882.P06.0303311722
selected No volume label
03/27/03 03:24p <DIR> 4,096 EFI
03/27/03 03:24p <DIR> 4,096 MSUtil
Exit
```

[Treat like a removable media boot]

Select the file you want to add, as shown in green below:

```
BIOS Version S870BN4A.86B.0882.P06.0303311722
selected /efi/Microsoft/winnt50/
03/27/03 03:24p <DIR> 4,096 .
03/27/03 03:24p <DIR> 4,096 ..
05/28/03 08:25p 742,400 ia64ldr.efi
05/28/03 08:25p 330,240 fpswa.efi
Exit
```

Enter Text Description

The file that you just selected will then appear below the **Add a Boot Option** menu, as shown in green below. Then, make the additional selections also shown in green.

```
BIOS Version
S870BN4A.86B.0882.P06.0303311722

Add a Boot Option. Select a Volume

NO VOLUME LABEL [Acpi(PNP0A03,1)/Pci(1F|0)/Pci(2|0)/Scsi(Pun0,Lu
Removable Media Boot [Acpi(PNP0A03,0)/Pci(1F|1)/Ata(Primary,Slav
Removable Media Boot [Acpi(PNP0A03,0)/Pci(1F|1)/Ata(Primary,Mast
Load File [VenHw(6D9FEEB1-E585-11D3-BC22-0080C73C8881)]
Load File [VenHw(6D9FEEB1-E585-22D3-BC22-0080C73C8881)]
Load File [EFI Shell [Built-in]]
Load File [Acpi(PNP0A03,0)/Pci(1E|0)/Pci(0|0)/Mac(0002B300076A)]
Legacy Boot
Exit

Filename: \EFI\Microsoft\WINNT50\ia64ldr.efi
DevicePath:
[Acpi(PNP0A03,1)/Pci(1F|0)/Pci(2|0)/Scsi(Pun0,Lun0)/HD(Part1,Sig89
83DFE0-F474-01C2-507B-9E5F8078F531)/\EFI\Microsoft\WINNT50\ia64ldr.efi]
IA-64 EFI Application 05/28/03 08:25p 742,400 bytes

Enter New Description: Boot to 64-bit
Windows Server 2003
```

Type the new option to appear on the boot manager at the prompt.

ASCII/Unicode strings only, with maximum of 80 characters.

```
Enter BootOption Data Type [A-Ascii U-
Unicode N-No BootOption]: N
```

Type N for none.

```
Save changes to NVRAM [Y-Yes N-No]: Y
```

Press Y to save this selection.

Resulting Boot Menu

The following is the resulting boot menu. Note that the option in green is the one added by this procedure.

```
EFI Boot Manager ver 1.10 [14.62]
Please select a boot option
Floppy/Pci(1F|1)/Ata(Primary,Slave)
CD/DVD ROM/Pci(1F|1)/Ata(Primary,Master)
Hard Drive/Pci(6|2|0)/Scsi(Pun0,Lun0)/HD(Part1,Sig8983DFE0-F474-...)
Network Boot/Pci(1|0|0)/Mac(0002B300076A)
EFI Shell [Built-in]
Boot to 64-bit Windows Server 2003
Boot option maintenance menu

Use ▲ and ▼ to change option(s). Use Enter to select an option.
```

How to Delete an Executable from the Boot Menu

This example shows how to delete the **Boot to 64-bit Windows Server* 2003** entry that was added in How to Add an Executable to the Boot Menu.

To delete an executable from the boot menu, do the following:

1. Start the EFI boot manager and select the **Boot option maintenance menu.**
2. At the boot option maintenance menu, select **Delete boot option(s).**
3. Select the boot option to delete from the boot device list.

The resulting boot menu should then show the new boot option.

The following examples are from screen snapshots from the reference EFI boot manager and illustrate the procedure listed above.

Boot Manager

Select the option in green below and hit the enter key to go to the boot option maintenance menu.

```
EFI Boot Manager ver 1.10 [14.62]
Please select a boot option
Floppy/Pci(1F|1)/Ata(Primary,Slave)
CD/DVD ROM/Pci(1F|1)/Ata(Primary,Master)
Hard Drive/Pci(6|2|0)/Scsi(Pun0,Lun0)/HD
(Part1,Sig8983DFE0-F474-...)
Network Boot/Pci(1|0|0)/Mac(0002B300076A)
EFI Shell [Built-in]
Boot to 64-bit Windows Server 2003
Boot option maintenance menu

Use ▲ and ▼ to change option(s). Use Enter to select an option.
```

Boot Option Maintenance Menu

At the boot option maintenance menu, select the option in green below:

```
BIOS Version S870BN4A.86B.0882.P06.0303311722
Main Menu. Select an Operation

Boot from a File
Add a Boot Option
Delete Boot Option(s)
Change Boot Order

Manage BootNext setting
Set Auto Boot TimeOut

Cold Reset
Exit

Timeout-->[7] sec SystemGuid-->[01034B34-47E9-D711-BEA6-0002B300076A]
SerialNumber-->[1234567890123 ]
```

Possible Boot Device List

Select the volume that you want to delete from the boot menu, as shown in green below:

```
BIOS Version S870BN4A.86B.0882.P06.0303311722

Delete Boot Option(s). Select an Option

Floppy/Pci(1F|1)/Ata(Primary,Slave)
CD/DVD ROM/Pci(1F|1)/Ata(Primary,Master)
Hard Drive/Pci(6|2|0)/Scsi(Pun0,Lun0)/HD(Part1,Sig8983DFE0-F474-...)
Network Boot/Pci(1|0|0)/Mac(0002B300076A)
EFI Shell [Built-in]
Boot to 64-bit Windows Server 2003
Delete All Boot Options
Save Settings to NVRAM
Help
Exit

Acpi(PNP0A03,1)/Pci(1F|0)/Pci(2|0)/Scsi(Pun0,Lun0)/HD(Part1,Sig8983DFE0
-F474-01CBoot0005
```

Resulting Boot Menu

The following is the resulting boot menu. Note that the **Boot to 64-bit Windows Server 2003** option that was deleted by this procedure is now gone.

```
EFI Boot Manager ver 1.10 [14.62]
Please select a boot option
Floppy/Pci(1F|1)/Ata(Primary,Slave)
CD/DVD ROM/Pci(1F|1)/Ata(Primary,Master)
Hard Drive/Pci(6|2|0)/Scsi(Pun0,Lun0)/HD(Part1,Sig8983DFE0-F474-...)
Network Boot/Pci(1|0|0)/Mac(0002B300076A)
EFI Shell [Built-in]

Boot option maintenance menu

Use ▲ and ▼ to change option(s). Use Enter to select an option.
```

The boot option will vanish from the main boot screen after saving to NVRAM.

How to Change the Execution Order of the Boot Menu

This example shows how to make the built-in EFI Shell the first boot option.

To change the order in which options on the boot menu are executed, do the following:

1. Start the EFI boot manager and select the **Boot option maintenance menu.**
2. At the boot option maintenance menu, select **Change Boot Order.**
3. Select the boot option to move from the Change Boot Order menu.

The resulting boot menu should then show the new order of the boot options.

The following examples are from screen snapshots from the reference EFI boot manager and illustrate the procedure listed above.

Boot Manager

Select the option in green below and hit the enter key to go to the boot option maintenance menu. Note that the **EFI Shell** is the last boot option.

```
EFI Boot Manager ver 1.10 [14.62]

Please select a boot option

Floppy/Pci(1F|1)/Ata(Primary,Slave)
CD/DVD ROM/Pci(1F|1)/Ata(Primary,Master)
Hard Drive/Pci(6|2|0)/Scsi(Pun0,Lun0)/HD(Part1,Sig8983DFE0-F474-...)
Network Boot/Pci(1|0|0)/Mac(0002B300076A)
EFI Shell [Built-in]

Boot option maintenance menu

Use ▲ and ▼ to change option(s). Use Enter to select an option.
```

Boot Option Maintenance Menu

At the boot option maintenance menu, select the option in green below:

```
BIOS Version S870BN4A.86B.0882.P06.0303311722
Main Menu. Select an Operation

Boot from a File
Add a Boot Option
Delete Boot Option(s)
Change Boot Order

Manage BootNext setting
Set Auto Boot TimeOut

Cold Reset
Exit

Timeout-->[7] sec SystemGuid-->[01034B34-47E9-D711-BEA6-0002B300076A]
SerialNumber-->[1234567890123 ]
```

Change Boot Option Help

This menu helps to set the boot order of options. To change the boot option order, select a boot option using the arrow keys. Press 'U' or 'u' to move the option up in the order chain and press 'D' or 'd' to move it down the order chain. Save the settings to NVRAM (if needed) before exiting.

In this example, highlight the option in green and type "UUUU" (the *U* key four times) to move it to the top of the list.

```
BIOS Version S870BN4A.86B.0882.P06.0303311722

Change boot order. Select an Option

Floppy/Pci(1F|1)/Ata(Primary,Slave)
CD/DVD ROM/Pci(1F|1)/Ata(Primary,Master)
Hard Drive/Pci(6|2|0)/Scsi(Pun0,Lun0)/HD(Part1,Sig8983DFE0-F474-...)
Network Boot/Pci(1|0|0)/Mac(0002B300076A)
EFI Shell [Built-in]

Save Settings to NVRAM
Help
Exit

Acpi(PNP0A03,0)/Pci(1F|1)/Ata(Primary,Slave)
Boot0000
```

The following is what the menu looks like after moving the EFI Shell option to the top of the boot order list.

```
BIOS Version S870BN4A.86B.0882.P06.0303311722

Change boot order. Select an Option

EFI Shell [Built-in]
Floppy/Pci(1F|1)/Ata(Primary,Slave)
CD/DVD ROM/Pci(1F|1)/Ata(Primary,Master)
Hard Drive/Pci(6|2|0)/Scsi(Pun0,Lun0)/HD(Part1,Sig8983DFE0-F474-...)
Network Boot/Pci(1|0|0)/Mac(0002B300076A)

Save Settings to NVRAM
Help
Exit

VenHw(D65A6B8C-71E5-4DF0-A909-F0D2992B5AA9)
Boot0004
```

Resulting Boot Menu

The following is the resulting boot menu. Note that the **EFI Shell [Built-in]** option in green that was moved by this procedure is now at the top of the list of boot options.

```
EFI Boot Manager ver 1.10 [14.62]

Please select a boot option

EFI Shell [Built-in]
Floppy/Pci(1F|1)/Ata(Primary,Slave)
CD/DVD ROM/Pci(1F|1)/Ata(Primary,Master)
Hard Drive/Pci(6|2|0)/Scsi(Pun0,Lun0)/HD(Part1,Sig8983DFE0-F474-...)
Network Boot/Pci(1|0|0)/Mac(0002B300076A)
Boot option maintenance menu

Use ▲ and ▼ to change option(s). Use Enter to select an option.
```

How to View an OS Boot Option in the Boot Manager

To view a OS boot option, enter the EFI boot manager, which should show the list of bootable devices.

The example below shows what an installed OS boot option looks like (device path with a GPT disk GUID) and also where that OS loader is in the system. In this example, the EFI boot manager displays a **Boot to 64-bit Windows Server* 2003** boot option. This boot option is user defined and would have been added by a user or administrator, as shown in How to Add an Executable to the Boot Menu.

Boot Manager

In this example, the OS boot option is the one in green below.

```
EFI Boot Manager ver 1.10 [14.62]

Please select a boot option

Floppy/Pci(1F|1)/Ata(Primary,Slave)
CD/DVD ROM/Pci(1F|1)/Ata(Primary,Master)
Hard Drive/Pci(6|2|0)/Scsi(Pun0,Lun0)/HD(Part1,Sig8983DFE0-F474-
Network Boot/Pci(1|0|0)/Mac(0002B300076A)
EFI Shell [Built-in]
Boot to 64-bit Windows Server 2003

Boot option maintenance menu

Use ▲ and ▼ to change option(s). Use Enter to select an option.
```

The NVRAM has the location of the loader as stored on the SCSI partition 1 on the hard drive. The disk has a unique ID of Sig8983DFE0-F474-01C2-507B-9E5F8078F531, which was created when the disk was partitioned with either **`diskpart.efi`** or with an EFI-aware OS installation program. In this case, it is stored in NVRAM flash as boot variable **`Boot0005`**. You can use **`dmpstore`** to see the raw NVRAM or the **Add boot option** menu to see the real device path of the boot option, along with any arguments passed into the command line of the boot option.

Microsoft provides a utility called **`nvrboot.efi`**, which allows you to export or import the boot option to a file that can be saved in case the NVRAM is erased. **`Nvrboot.efi`** may not be visible unless you use the EFI Shell command **`attrib +h +r`** from the **`MSutil`** directory.

Dmpstore

Dmpstore is an EFI Shell command to dump the raw NVRAM. Boot variables can be checked using this command.

This command displays all the environment variables that are being managed by EFI. The following example shows the typical output for help on this command.

Examples

```
Shell> dmpstore
 Dump NVRAM

 Variable RT+BS 'Efi:BootCurrent' DataSize = 2
   00000000: FF FF                                          *..*
 Variable NV+RT+BS 'Efi:LangCodes' DataSize = 2A
   00000000: 65 6E 67 65 6E 6D 61 6E-67 63 68 69 7A 68 6F 64
*engenmangchizhod*
   00000010: 65 75 67 65 6D 67 65 72-67 6D 68 67 6F 68 66 72
*eugemgergmhgohfr*
   00000020: 61 66 72 65 66 72 6D 66-72 6F                  *afrefrmfro*
 Variable NV+RT+BS 'Efi:Lang' DataSize = 3
   00000000: 65 6E 67                                       *eng*
  ...
 Variable NV+BS 'ShellAlias:del' DataSize = 6
   00000000: 72 00 6D 00 00 00                               *r.m...*
 Variable NV+BS 'ShellAlias:copy' DataSize = 6
   00000000: 63 00 70 00 00 00                               *c.p...*
 Variable NV+BS 'SEnv:path' DataSize = 4
   00000000: 2E 00 00 00                                     *....*
```

Click here to return to How to View an OS Boot Option in the Boot Manager.

4
Configuration Driver Boot

How to Install an EFI 1.10 Driver

To add an EFI 1.10 driver to the system so it always loads before executing the boot options in the EFI boot manager, use the Shell **bcfg** command. The driver NVRAM list will always be loaded first before processing the boot options in the reference code.

This action is done in case you need a driver to be loaded to be able to see the OS loader.

At the EFI Shell prompt, type the following in green to run the **bcfg** command, which displays and modifies the driver and boot configuration.

```
fs0:>bcfg
```

To view the help text for the **bcfg** command, type the following in green at the Shell prompt instead; see Bcfg Help List for the list of options that are available for the **bcfg** command.

```
fs0:>bcfg -?
```

Use the **bcfg** driver dump to view all drivers that are already loaded in NVRAM.

For example, if you want to load the **fpswa.efi** driver on **fs0:** at **\efi\microsoft\winnt50**, type the following in green at the Shell prompt:

```
fs0:>bcfg driver add 1 fs0:\efi\microsoft\winnt50\fpswa.efi
"fpswa.efi driver"
Target = 1
bcfg: Add boot driver as 1
```

To see if the “add” was successful, type the following in green:

```
fs0:\bcfg dump driver
The boot driver list is:
01.  Acpi(PNP0A03,1)/Pci(1F|0)/Pci(2|0)/Scsi(Pun0,Lu
n0)/HD(Part1,Sig8983DFE0-F474-01C2-507B-9E5F8078F531
)/\EFI\Microsoft\WINNT50\fpswa.efi "fpswa.efi driver"
```

To remove this driver from the driver option list in NVRAM, type the following in green:

```
fs0:\bcfg driver rm 1
```

How to Alter Boot Options in the EFI Shell

To alter an EFI 1.10 boot option to the system in the EFI boot manager, use the Shell **bcfg** command.

At the EFI Shell prompt, type the following in green to run the **bcfg** command, which displays and modifies the driver and boot configuration.

```
fs0:>bcfg
```

To view the help text for the **bcfg** command, type the following in green at the Shell prompt instead; see Bcfg Help List for the list of options that are available for the **bcfg** command.

```
fs0:>bcfg -?
```

To see the boot option in NVRAM, type the following in green:

```
fs0:>bcfg dump boot

The boot driver list is:
01. Acp(PNP0A03,0)/Pci(1E|0)/Pci(1|0)/Mac(0002B3645FE9) "load file"
02. Acpi(PNP0A03,0)/Pci(1F|0)/Acpi(PNP0604,0) "floppy"
```

The string **" .... "** will be the string displayed by the EFI boot manager. For this example, **"load file"** and **"floppy"** are the boot options that will show in the boot menu.

To remove a boot entry based on its number as displayed by the **dump** command, type the following in green:

```
fs0:>bcfg boot rm #
```

To add a new boot option, use the same example as above except use the **boot add** option instead of **driver**.

For example, to make Windows* the first boot option, type the following in green:

```
fs0:\bcfg boot add 1 fs0:\efi\microsoft\winnt50\ia64ldr.efi
"Boot to 64-bit Windows Server 2003"
```

Bcfg Help List

The following options are available from the **bcfg** command, which are then explained in the table below:

```
BCFG driver|boot [dump [-v]] [add # file "desc"] [rm #] [mv # #]
```

Options for the Bcfg Command

Option	Description
driver	Displays/modifies the driver option list.
boot	Displays/modifies the boot option list.
dump	Displays the option list.
-v	Displays the option list with extra information.
add	Adds an option.
#	The number of the option to add in hexadecimal.
file	The file name of the EFI application/driver for the option.
"desc"	The description of the option being added.
rm	Removes an option.
#	The number of the option to remove in hexadecimal.
mv	Moves an option.
#	The number of the option to move in hexadecimal.
#	The new number of the option being moved.

Click the links below to return to:

- How to Install an EFI 1.10 Driver
- How to Alter Boot Options in the EFI Shell
- How to Load, Start, and Stop an EFI 1.10 Driver
- How to Save and Restore a Windows* Boot Option

How to View an EFI Boot Device Path

Use the EFI Shell **map** command to display the mappings in NVRAM between block devices and the shorthand variables such as **blk*xx*** and **fs*yy***.

To run the **map** command and view the device mappings, type the following in green at the Shell prompt:

```
fs1:\>map

Device mapping table

fs0 : Acpi(PNP0A03,0)/Pci(1F|1)/Ata(Primary,Slave)

fs1 : Acpi(PNP0A03,1)/Pci(1F|0)/Pci(2|0)/Scsi(Pun0,Lun0)/HD(Part1,Sig89
83DFE0-F474-01C2-507B-9E5F8078F531)
fs2 : Acpi(PNP0A03,1)/Pci(1F|0)/Pci(2|0)/Scsi(Pun1,Lun0)/HD(Part1,Sig1A
AA84E00-1DD2-1000-848F-0002B300076A)
blk0 : Acpi(PNP0A03,0)/Pci(1F|1)/Ata(Primary,Master)

blk1 : Acpi(PNP0A03,0)/Pci(1F|1)/Ata(Primary,Slave)
blk2 : Acpi(PNP0A03,1)/Pci(1F|0)/Pci(2|0)/Scsi(Pun0,Lun0)
blk3 :
Acpi(PNP0A03,1)/Pci(1F|0)/Pci(2|0)/Scsi(Pun0,Lun0)/HD(Part1,Sig89
83DFE0-F474-01C2-507B-9E5F8078F531)
blk4 : Acpi(PNP0A03,1)/Pci(1F|0)/Pci(2|0)/Scsi(Pun0,Lun0)/HD(Part2,Sig8
98D07A0-F474-01C2-F1B3-12714F758821)
blk5 : Acpi(PNP0A03,1)/Pci(1F|0)/Pci(2|0)/Scsi(Pun0,Lun0)/HD(Part3,Sig8
9919B80-F474-01C2-D931-F8428177D974)
blk6 : Acpi(PNP0A03,1)/Pci(1F|0)/Pci(2|0)/Scsi(Pun1,Lun0)
blk7 : Acpi(PNP0A03,1)/Pci(1F|0)/Pci(2|0)/Scsi(Pun1,Lun0)/HD(Part1,Sig1
AA84E00-1DD2-1000-848F-0002B300076A)
blk8 : Acpi(PNP0A03,1)/Pci(1F|0)/Pci(2|0)/Scsi(Pun1,Lun0)/HD(Part2,Sig6
04B59A6-8B71-4F7F-AE13-58990CE63502)
```

The EFI Shell is replacing what is to the left of the colon (**:**) with the device path to the right of the colon (**:**).

For instance, in the example above, **FS0:** is being replaced with **Acpi(PNP0A03,0)/Pci(1F|1)/Ata(Primary,Slave)**.

In the example above, **fs0:** and **blk0** have the same Plug and Play and PCI function device numbering.

In this particular snapshot of the Shell NVRAM, **fs0:** shows that there is a floppy disk in the LS-120 drive with a FAT16 system on **blk0** (based on the device path).

In this example, there is an ATAPI disk controller on root bus bridge 0 that is on the primary IDE channel and it is a slave device.

PNP0A03 is the Plug and Play code for a root bus bridge, and the "**,0**" indicates that the IDE controller is on bus 0. The IDE controller is device 1F, function 0.

There is also an IDE controller on the primary master channel (a CD-ROM), but no CD-ROM is in the drive, so there is no **fsX:** corresponding to it. If it were, you would see a CD-ROM entry in the device path with an FS on it.

blk2 is the raw SCSI controller.

blk3, **blk4**, and **blk5** are the three partitions on the GPT hard disk on the SCSI drive, which is on SCSI bus 0 (**Scsi(Pun0,Lun0)**), target ID 0. Each partition has a different GUID or unique ID when it was created. Only partition 1 has an EFI system partition on it because only **fs1:** corresponds with the same disk GUID (the **blk3** device path matches **fs1:**).

blk7 matches with **fs2:** because they have the same disk GUID. In this system, there are two SCSI drives—one on SCSI bus 0, target ID 0 and one on SCSI bus 1, target ID 0. The second drive has only two partitions on it. It is unknown what file systems may be on **blk8**, **blk4**, and **blk5**. It may require that the OS has a native driver for the file system and EFI can read only FAT12, FAT16, and FAT32.

NOTES

1. *Default mappings are the mappings chosen by the EFI integrator of the system. They are typically stored in flash NVRAM in the system but may also reside on the EFI system partition in the* **\EFI\Boot** *directory. The core EFI implementation will determine which NVRAM source to use.*
2. *The mapping order of* **fsX:** *to block I/O devices is arbitrary. EFI applications should not rely on the system mapping block I/O devices to a particular FSx mapping. Adding or removing media may arbitrarily rename the FSx mapping when a* **map -r** *command occurs. Applications should create their own mappings.*

5
Console Redirection

How to Set up Console Redirection

On Intel® systems with a legacy BIOS, the F2 setup menus still control the serial port.

In the EFI Sample Implementation code on the EFI web site, there are separate boot maintenance menu options to control the console redirection, console output, and console input locations (through device paths).

The example below shows how to set up the legacy BIOS for console redirection on an Itanium®-based system. To get to this menu, press F2 after booting the system and then select the **Servers** menu. From this menu, select the **Console Redirection** menu, which will show you the view below. The items on the right side of this example are the settings that are needed to set up console redirection.

```
Console Redirection

Serial Console Redirection
Serial Console Redirection                  [Enabled]
Serial Port                                 [COM2 2F8 IRQ3]
Baud Rate                                   [115.2K]
Flow Control                                [CTS/RTS]
Terminal Type                               [PC-ANSI]
Remote Console Reset                        [Disabled]
ACPI OS Headless Operation                  [Disabled]
ACPI OS Baud Rate                           [19.2K]
ACPI OS Flow Control                        [CTS/RTS]
ACPI OS Terminal Type                       [PC-ANSI]
```

If enabled, BIOS uses the specified serial port to redirect the console to a remote ANSI terminal. Enabling this option disables quiet boot.

Only the first six lines control the EFI console redirection. It must be enabled and set to the correct terminal type that matches the terminal to which you are attaching the EFI system. Using a 9-pin serial Null modem cable will typically allow you to connect the serial port from the EFI system to another computer. For Windows*-based systems, PC_ANSI typically is used. For Linux*-based systems and Unix*-based systems, VT-100 is normally used. Future versions of Windows XP may support UTF8, which supports full Unicode, non-English character sets.

The following example menus show how to set up console redirection in the boot option maintenance menu on an EFI-based Intel® Server Platform SR870BN4.

Boot Option Maintenance Menu

NOTE

The boot option maintenance menu may be different from platform to platform based on platform policies during the build of the EFI implementation. Therefore, some of the other menus seen in this guide include menu entries that may differ depending on the platform policies that were used during the build time.

```
BIOS Version S870BN4A.86B.0882.P06.0303311722
Boot option maintenance menu
Main Menu. Select an Operation

Boot from a File
Add a Boot Option
Delete Boot Option(s)
Change Boot Order

Manage BootNext setting
Set Auto Boot TimeOut

Select Active Console Output Device      <-
Select Active Console Input Device       <-
Select Active Standard Error Device      <-

Cold Reset
Exit
```

Console Input Menu

This section shows the Console Input Device menu. Select the desired input from this menu.

```
EFI Boot Maintenance Manager ver 1.10 [14.62]

Select the Console Input Device(s)

Acpi(PNP0A03,0)/Pci(1F|0)/Acpi(PNP0303,0)
Acpi(PNP0A03,0)/Pci(1F|0)/Acpi(PNP0303,0)/Uart(115200 81)/VenM
Acpi(PNP0A03,0)/Pci(1F|0)/Acpi(PNP0303,0)/Uart(115200 81)/VenM
Acpi(PNP0A03,0)/Pci(1F|0)/Acpi(PNP0303,0)/Uart(115200 81)/VenM
Acpi(PNP0A03,0)/Pci(1F|0)/Acpi(PNP0303,0)/Uart(115200 81)/VenM
Acpi(PNP0A03,0)/Pci(1F|0)/Acpi(PNP0303,1)/Uart(115200 81)/VenM
Acpi(PNP0A03,0)/Pci(1F|0)/Acpi(PNP0303,1)/Uart(115200 81)/VenM
Acpi(PNP0A03,0)/Pci(1F|0)/Acpi(PNP0303,1)/Uart(115200 81)/VenM
Acpi(PNP0A03,0)/Pci(1F|0)/Acpi(PNP0303,1)/Uart(115200 81)/VenM
Save Sttings to NVRAM
Exit
```

Console Output Menu

This section shows the Console Output Device menu. Select the desired output device from this menu.

```
EFI Boot Maintenance Manager ver 1.10 [14.62]

Select the Console Output Device(s)

Acpi(PNP0A03,0)/Pci(1F|0)/Acpi(PNP0303,0)
Acpi(PNP0A03,0)/Pci(1F|0)/Acpi(PNP0303,0)/Uart(115200 81)/VenM
Acpi(PNP0A03,0)/Pci(1F|0)/Acpi(PNP0303,0)/Uart(115200 81)/VenM
Acpi(PNP0A03,0)/Pci(1F|0)/Acpi(PNP0303,0)/Uart(115200 81)/VenM
Acpi(PNP0A03,0)/Pci(1F|0)/Acpi(PNP0303,0)/Uart(115200 81)/VenM
Acpi(PNP0A03,0)/Pci(1F|0)/Acpi(PNP0303,1)/Uart(115200 81)/VenM
Acpi(PNP0A03,0)/Pci(1F|0)/Acpi(PNP0303,1)/Uart(115200 81)/VenM
Acpi(PNP0A03,0)/Pci(1F|0)/Acpi(PNP0303,1)/Uart(115200 81)/VenM
Acpi(PNP0A03,0)/Pci(1F|0)/Acpi(PNP0303,1)/Uart(115200 81)/VenM
Save Sttings to NVRAM
Exit
```

Console Standard Error Menu

This section shows the Console Standard Error Output Device menu. Select the desired standard output device from this menu.

```
EFI Boot Maintenance Manager ver 1.10 [14.62]

Select the Console Standard Error output Device(s)

Acpi(PNP0A03,0)/Pci(1F|0)/Acpi(PNP0303,0)
Acpi(PNP0A03,0)/Pci(1F|0)/Acpi(PNP0303,0)/Uart(115200 81)/VenM
Acpi(PNP0A03,0)/Pci(1F|0)/Acpi(PNP0303,0)/Uart(115200 81)/VenM
Acpi(PNP0A03,0)/Pci(1F|0)/Acpi(PNP0303,0)/Uart(115200 81)/VenM
Acpi(PNP0A03,0)/Pci(1F|0)/Acpi(PNP0303,0)/Uart(115200 81)/VenM
Acpi(PNP0A03,0)/Pci(1F|0)/Acpi(PNP0303,1)/Uart(115200 81)/VenM
Acpi(PNP0A03,0)/Pci(1F|0)/Acpi(PNP0303,1)/Uart(115200 81)/VenM
Acpi(PNP0A03,0)/Pci(1F|0)/Acpi(PNP0303,1)/Uart(115200 81)/VenM
Acpi(PNP0A03,0)/Pci(1F|0)/Acpi(PNP0303,1)/Uart(115200 81)/VenM
Save Sttings to NVRAM
Exit
```

How to Redirect Console Output to a File

In the EFI Shell environment, output from an EFI application can be redirected to a file.

The following is an example of a sample EFI application to show how any application can redirect its output to another file. To redirect output from a console application to a file, type the following in green at the EFI Shell prompt:

```
fs0:>xxxx.efi 1>a output.txt
```

The following table describes the options that are available for this command.

Options for Redirecting Output

Option	Description
xxxx.efi	Example EFI application.
1	There are two possible values for this number: • 1 indicates the standard console output the user would have seen. • 2 means the error output from the program.
>a output.txt	Sends the console output in ASCII to a file called **output.txt** at which the current file system is pointed. The Shell needs to have a valid FATxx file system for this command to work.
>u output.txt	Sends the console output in Unicode to a file called **output.txt**. Several systems are aware of Unicode in Windows*- and Linux*-based systems. All programs will understand ASCII but some will not be able to parse the Unicode strings. Default is Unicode.

In the EFI Shell in the EFI Sample Implementation, there is no means to redirect input from a file.

How to Change the Size of the Console

To show the screen resolutions that the output console supports, use the EFI Shell **mode** command.

The example below is the default setting for most systems.

To display the modes that are available, type the following in green at the EFI Shell prompt:

```
fs0:>mode
Available modes on standard output:
col 80 row 25 *
col 80 row 50
col 100 row 31
```

The asterisk (*) indicates the current mode that the output console is in.

To change the screen so it is bigger and will show more of the displayed files, type the following in green at the EFI Shell prompt:

```
fs0:>mode 100 31
```

This example will switch the screen to 100 characters wide by 31 lines long.

If console redirection is on, the only mode that is available is 80 characters by 25 lines.

6
EFI Shell and File System

How to Use NVRAM with the EFI Shell and File System

EFI uses part of the flash in the system to store nonvolatile information from boot to boot for the EFI Shell and the EFI boot manager. All of the boot options that are seen on the screen during boot are saved in this flash area. All the Shell variables and nonvolatile set variables are also stored in this flash area.

The purpose of the NVRAM is to store the following:

- The boot information to load any operating systems installed on the system
- The location of any EFI drivers in the system that are needed to boot or provide system management

At the Shell, the **map** command uses the flash to store all the block I/O and **fsx:** device path information. Every time the **map -r** command is executed, the NVRAM is updated with a new set of **blk** and **fsx:** information.

To see the raw NVRAM, you can use the EFI Shell **dmpstore** command, which will list every byte of NVRAM. The list can be redirected to a file (**dmpstore 1>A nv.txt**) and examined with any text editor. This command is not recommended for novice users. It is a way to determine what is stored in NVRAM for the EFI boot manager and EFI Shell.

EFI Shell NVRAM Usage

Shell users typically use the NVRAM to store information that will be repeatedly used in **.nsh** batch files or between various Shell scripts and tests.

To see all the current shell NVRAM variable assignments (for the Shell that is currently running, type **set** at the Shell prompt, as shown in green in the following:

```
fs0:>set
* path : fs0:\efi\tools
```

The path variable is always set so that every time you enter a command at the prompt, it will look in these directories for any EFI executables. This mechanism works similar to MS-DOS*. Some EFI commands will be built into the Shell itself. To see which commands are built in, type either of the following in green at the Shell prompt:

```
fs0:>set ?
```

or

```
fs0:>set help
```

If a Shell command is not found to be built into the Shell, it will then start looking in all the locations defined by the path to find it.

To add new Shell variables, use the Shell **set** command. At the EFI Shell prompt, type the following in green:

```
fs0:>set varname value
```

where **varname** is the variable name and **value** is the value of the variable name. For example:

```
fs0:\>set racecar 1234
```

Typing **set** at the Shell prompt again now shows the value of **path** and **racecar**. For example:

```
fs0:>set
path    : .;fs0:\efi\tools;fs0:\efi\boot;fs0:\;fs1:\efi\tools;
          fs1:\efi\boot;fs1:\
racecar : 1234
```

To delete **racecar**, type the following in green at the Shell prompt:

```
fs0:>set -d racecar
```

You can use Shell NVRAM variables at the Shell prompt and in Shell **.nsh** files. The following example sets the variable **harddrive** for fs1;, if FS1 is pointing at the SCSI hard drive:

```
fs0:>set harddrive fs1:

fs0:\>dir %harddrive%

Directory of: fs1:\
03/27/03 03:24p <DIR> 4,096 EFI
03/27/03 03:24p <DIR> 4,096 MSUtil
05/28/03 08:26p 322 entrsrv
05/28/03 08:26p 322 entrsrv2
05/28/03 09:00p 26 test.txt
06/09/03 04:06p <DIR> 4,096 lan
02/05/03 11:05p 144,384 LoadPciRom.efi
06/09/03 04:14p <DIR> 4,096 adaptec
06/12/03 12:02p <DIR> 4,096 test
4 File(s) 145,054 bytes
5 Dir(s)
```

It is important to note that the Shell variables take up flash space that is meant for the EFI boot manager to store information, such as where the operating system and boot devices are located (SCSI, floppy, CD-ROM, USB, etc.). Filling up the NVRAM with numerous variables is not recommended and may cause the system to fail if it is not cleaned.

It is recommended to delete any Shell variables at the end of any Shell scripts or Shell sessions. In Itanium®-based systems, the total NVRAM space is typically approximately 64 KB for both the EFI boot manager and the Shell. This amount will vary depending on flash availability and firmware installation components and is not guaranteed. Contact your system BIOS support person or platform TME to determine the actual size for your particular platform and BIOS.

How to Save and Restore a Windows* Boot Option

To save and restore a Windows* boot option, use the Shell **bcfg** command using the **add** and **boot** options. See How to Alter Boot Options in the EFI Shell and Bcfg Help List for more information.

How to Map an EFI File System

Use the EFI Shell **map** command to map a file system.

To run the **map** command and view the device mappings, type the following in green at the Shell prompt:

```
fs1:\>map

Device mapping table

fs0 : Acpi(PNP0A03,0)/Pci(1F|1)/Ata(Primary,Slave)

fs1 : Acpi(PNP0A03,1)/Pci(1F|0)/Pci(2|0)/Scsi(Pun0,Lun0)/HD(Part1,Sig89
83DFE0-F474-01C2-507B-9E5F8078F531)
fs2 : Acpi(PNP0A03,1)/Pci(1F|0)/Pci(2|0)/Scsi(Pun1,Lun0)/HD(Part1,Sig1A
AA84E00-1DD2-1000-848F-0002B300076A)
blk0 : Acpi(PNP0A03,0)/Pci(1F|1)/Ata(Primary,Master)
blk1 : Acpi(PNP0A03,0)/Pci(1F|1)/Ata(Primary,Slave)
blk2 : Acpi(PNP0A03,1)/Pci(1F|0)/Pci(2|0)/Scsi(Pun0,Lun0)
blk3 :
Acpi(PNP0A03,1)/Pci(1F|0)/Pci(2|0)/Scsi(Pun0,Lun0)/HD(Part1,Sig89
83DFE0-F474-01C2-507B-9E5F8078F531)
blk4 : Acpi(PNP0A03,1)/Pci(1F|0)/Pci(2|0)/Scsi(Pun0,Lun0)/HD(Part2,Sig8
98D07A0-F474-01C2-F1B3-12714F758821)
blk5 : Acpi(PNP0A03,1)/Pci(1F|0)/Pci(2|0)/Scsi(Pun0,Lun0)/HD(Part3,Sig8
9919B80-F474-01C2-D931-F8428177D974)
blk6 : Acpi(PNP0A03,1)/Pci(1F|0)/Pci(2|0)/Scsi(Pun1,Lun0)
blk7 : Acpi(PNP0A03,1)/Pci(1F|0)/Pci(2|0)/Scsi(Pun1,Lun0)/HD(Part1,Sig1
AA84E00-1DD2-1000-848F-0002B300076A)
blk8 : Acpi(PNP0A03,1)/Pci(1F|0)/Pci(2|0)/Scsi(Pun1,Lun0)/HD(Part2,Sig6
04B59A6-8B71-4F7F-AE13-58990CE63502)
```

To map the floppy to FS0:, type the first line in green below at the Shell prompt. Then, to view the resulting mappings, just type **map** at the Shell prompt, as shown in green in the second line.

```
Shell>map floppy FS0:
Shell>map

Device mapping table

fs0 : Acpi(PNP0A03,0)/Pci(1F|1)/Ata(Primary,Slave)

fs1 : Acpi(PNP0A03,1)/Pci(1F|0)/Pci(2|0)/Scsi(Pun0,Lun0)/HD(Part1,Sig89
83DFE0-F474-01C2-507B-9E5F8078F531)
fs2 : Acpi(PNP0A03,1)/Pci(1F|0)/Pci(2|0)/Scsi(Pun1,Lun0)/HD(Part1,Sig1A
AA84E00-1DD2-1000-848F-0002B300076A)
blk0 : Acpi(PNP0A03,0)/Pci(1F|1)/Ata(Primary,Master)
...
floppy : Acpi(PNP0A03,0)/Pci(1F|1)/Ata(Primary,Slave)
```

How to Mount an EFI File System

The Shell **mount** command will define a mapping between a user-defined name and a block device handle. The most common use of this command is to assign drive names to device handles that support a known file system protocol. Once these assignments are made, the drive names can be used with all the file manipulation commands. The following example shows typical output for help on this command.

Use the EFI Shell **map** command with the Shell **mount** command to mount a file system. To view the help for the **mount** command and display the options that are available, type the following in green at the Shell prompt:

```
Shell>mount -?

mount BlkDevice [sname]

    BlkDevice      - The name of the block device to mount
    sname          - The name of the newly mounted file system
```

To run the **map** command and view the device mappings, type the following in green at the Shell prompt:

```
Shell>map

Device mapping table

fs1 :
Acpi(PNP0A03,1)/Pci(1F|0)/Pci(2|0)/Scsi(Pun0,Lun0)/HD(Part1,Sig89
83DFE0-F474-01C2-507B-9E5F8078F531)
fs2 :
Acpi(PNP0A03,1)/Pci(1F|0)/Pci(2|0)/Scsi(Pun1,Lun0)/HD(Part1,Sig1A
AA84E00-1DD2-1000-848F-0002B300076A)
blk0 : Acpi(PNP0A03,0)/Pci(1F|1)/Ata(Primary,Master)
blk1 : Acpi(PNP0A03,0)/Pci(1F|1)/Ata(Primary,Slave)
blk2 : Acpi(PNP0A03,1)/Pci(1F|0)/Pci(2|0)/Scsi(Pun0,Lun0)
blk3 :
Acpi(PNP0A03,1)/Pci(1F|0)/Pci(2|0)/Scsi(Pun0,Lun0)/HD(Part1,Sig89
83DFE0-F474-01C2-507B-9E5F8078F531)
blk4 :
Acpi(PNP0A03,1)/Pci(1F|0)/Pci(2|0)/Scsi(Pun0,Lun0)/HD(Part2,Sig8
98D07A0-F474-01C2-F1B3-12714F758821)
blk5 :
Acpi(PNP0A03,1)/Pci(1F|0)/Pci(2|0)/Scsi(Pun0,Lun0)/HD(Part3,Sig8
9919B80-F474-01C2-D931-F8428177D974)
blk6 : Acpi(PNP0A03,1)/Pci(1F|0)/Pci(2|0)/Scsi(Pun1,Lun0)
blk7 :
Acpi(PNP0A03,1)/Pci(1F|0)/Pci(2|0)/Scsi(Pun1,Lun0)/HD(Part1,Sig1
AA84E00-1DD2-1000-848F-0002B300076A)
blk8 :
Acpi(PNP0A03,1)/Pci(1F|0)/Pci(2|0)/Scsi(Pun1,Lun0)/HD(Part2,Sig6
04B59A6-8B71-4F7F-AE13-58990CE63502)
```

To mount the floppy as **blk0:**, type the first line in green at the Shell prompt, and then type the second line in green to view the resulting mappings:

```
Shell>mount blk0: floppy
Shell>map

Device mapping table

fs0 : Acpi(PNP0A03,0)/Pci(1F|1)/Ata(Primary,Slave)
fs1 :
Acpi(PNP0A03,1)/Pci(1F|0)/Pci(2|0)/Scsi(Pun0,Lun0)/HD(Part1,Sig89
83DFE0-F474-01C2-507B-9E5F8078F531)
fs2 :
Acpi(PNP0A03,1)/Pci(1F|0)/Pci(2|0)/Scsi(Pun1,Lun0)/HD(Part1,Sig1A
AA84E00-1DD2-1000-848F-0002B300076A)
blk0 : Acpi(PNP0A03,0)/Pci(1F|1)/Ata(Primary,Master)
. . .
floppy : Acpi(PNP0A03,0)/Pci(1F|1)/Ata(Primary,Slave)
```

How to Identify a Boot Media Device with a File Mapping

The following example shows how to view a device or mapping to a file system. To identify the device or mapping, type the following in green at the Shell prompt:

```
fs1:\>map

Device mapping table

fs0 : Acpi(PNP0A03,0)/Pci(1F|1)/Ata(Primary,Slave)
fs1 :
Acpi(PNP0A03,1)/Pci(1F|0)/Pci(2|0)/Scsi(Pun0,Lun0)/HD
(Part1,Sig89 83DFE0-474-01C2-507B-9E5F8078F531)
fs2 :
Acpi(PNP0A03,1)/Pci(1F|0)/Pci(2|0)/Scsi(Pun1,Lun0)/HD(Part1,Sig1A
A84E00-1DD2-1000-848F-0002B300076A)
blk0 : Acpi(PNP0A03,0)/Pci(1F|1)/Ata(Primary,Master)
blk1 : Acpi(PNP0A03,0)/Pci(1F|1)/Ata(Primary,Slave)
blk2 : Acpi(PNP0A03,1)/Pci(1F|0)/Pci(2|0)/Scsi(Pun0,Lun0)
blk3 :
Acpi(PNP0A03,1)/Pci(1F|0)/Pci(2|0)/Scsi(Pun0,Lun0)/HD(Part1,Sig89
83DFE0-F474-01C2-507B-9E5F8078F531)
blk4 :
Acpi(PNP0A03,1)/Pci(1F|0)/Pci(2|0)/Scsi(Pun0,Lun0)/HD(Part2,Sig8
98D07A0-F474-01C2-F1B3-12714F758821)
blk5 :
Acpi(PNP0A03,1)/Pci(1F|0)/Pci(2|0)/Scsi(Pun0,Lun0)/HD(Part3,Sig8
9919B80-F474-01C2-D931-F8428177D974)
blk6 : Acpi(PNP0A03,1)/Pci(1F|0)/Pci(2|0)/Scsi(Pun1,Lun0)
blk7 :
Acpi(PNP0A03,1)/Pci(1F|0)/Pci(2|0)/Scsi(Pun1,Lun0)/HD(Part1,Sig1
AA84E00-1DD2-1000-848F-0002B300076A)
blk8 :
Acpi(PNP0A03,1)/Pci(1F|0)/Pci(2|0)/Scsi(Pun1,Lun0)/HD(Part2,Sig6
04B59A6-8B71-4F7F-AE13-58990CE63502)
```

SCSI

In this example, **fs1** is a SCSI disk drive with the attributes shown in green. For more information on how to interpret these attributes, see How to View an EFI Boot Device Path.

How to Add an EFI File System to the Shell Mappings

Use the EFI Shell **mount** command in combination with the **map -d** command to force the assignment of a blkx device to a specific known mapping.

That is, if you wanted partition 1 on the SCSI drive to be called **HD1:**, you would type the following in green at the Shell prompt to remove the file system:

```
fs0:>map -d fs1:
```

Then, to assign the **blk3** device as **HD1**, type the following in green at the Shell prompt:

```
fs0:>mount blk3 HD1
```

After doing the above, typing **map** at the Shell prompt would yield **HD1:**, pointing to partition 1 on the SCSI drive, as shown in green below:

```
fs1:\>map

Device mapping table

fs0 : Acpi(PNP0A03,0)/Pci(1F|1)/Ata(Primary,Slave)
HD1 :
Acpi(PNP0A03,1)/Pci(1F|0)/Pci(2|0)/Scsi(Pun0,Lun0)/HD(Part1,
Sig89 83DFE0-F474-01C2-507B-9E5F8078F531)
fs2 :
Acpi(PNP0A03,1)/Pci(1F|0)/Pci(2|0)/Scsi(Pun1,Lun0)/HD(Part1,
Sig1A A84E00-1DD2-1000-848F-0002B300076A)
blk0 : Acpi(PNP0A03,0)/Pci(1F|1)/Ata(Primary,Master)
blk1 : Acpi(PNP0A03,0)/Pci(1F|1)/Ata(Primary,Slave)
blk2 : Acpi(PNP0A03,1)/Pci(1F|0)/Pci(2|0)/Scsi(Pun0,Lun0)
blk3 :
Acpi(PNP0A03,1)/Pci(1F|0)/Pci(2|0)/Scsi(Pun0,Lun0)/HD(Part1,
Sig89 83DFE0-F474-01C2-507B-9E5F8078F531)
blk4 :
Acpi(PNP0A03,1)/Pci(1F|0)/Pci(2|0)/Scsi(Pun0,Lun0)/HD(Part2,
Sig8 98D07A0-F474-01C2-F1B3-12714F758821)
blk5 :
Acpi(PNP0A03,1)/Pci(1F|0)/Pci(2|0)/Scsi(Pun0,Lun0)/HD(Part3,
Sig8 9919B80-F474-01C2-D931-F8428177D974)
blk6 : Acpi(PNP0A03,1)/Pci(1F|0)/Pci(2|0)/Scsi(Pun1,Lun0)
blk7 :
Acpi(PNP0A03,1)/Pci(1F|0)/Pci(2|0)/Scsi(Pun1,Lun0)/HD(Part1,
Sig1 AA84E00-1DD2-1000-848F-0002B300076A)
blk8 :
Acpi(PNP0A03,1)/Pci(1F|0)/Pci(2|0)/Scsi(Pun1,Lun0)/HD(Part2,
Sig6 04B59A6-8B71-4F7F-AE13-58990CE63502)
```

To mount a block device without a name, type the following in green at the Shell prompt:

```
fs0:>mount blk1
```

NOTES

1. *The Shell* **`mount`** *command uses the Disk I/O Protocol to read the FATxx format on a device. The name of the mounted file system is stored in NVRAM for a given Shell environment.*
2. *The mounted names will be lost when the* **`map -r`** *command is called the next time.*
3. *If the* **`mount`** *command is used without the second argument, it mounts the block device. An* **`EFI_FILE_SYSTEM_PROTOCOL`** *is then on the handle, but a drive name from the Shell is not generated.*

How to Delete an EFI File System Mapping at the Shell

Use the Shell **`map -d`** command to delete an EFI file system mapping in the EFI Shell. To delete a mapped name, type the following in green at the Shell prompt:

```
fs0:>map -d floppy

fs0:>map

Device mapping table

  fs0 : VenHw(58C518B1-76F3-11D4-BCEA-0080C73C8881)/VenHw(0C95A92F-
A006-11D4-BCFA-0080C73C8881)
  blk0: VenHw(58C518B1-76F3-11D4-BCEA-0080C73C8881)/VenHw(0C95A92F-
A006-11D4-BCFA-0080C73C8881)
```

To rename a default mapping to a user-defined map name, complete the following steps.

1. To map a display of a system with EDD 3.0 implemented, type the following in green:

```
fs1:>map

Device mapping table

  fs0 : Acpi(PNP0A03,0)/PCI(3|1)/Ata(Secondary,Master)
  fs1 :
Acpi(PNP0A03,1)/PCI(0|0)/Scsi(Pun0,Lun0)/HD(Part1,Sig1B16CC00-ABD0-
01C)
  fs2 : Acpi(PNP0A03,1)/PCI(0|0)/Scsi(Pun0,Lun0)/HD(Part4,SigG0)
  blk0: Acpi(PNP0A03,0)/PCI(3|1)/Ata(Secondary,Master)
  blk1: Acpi(PNP0A03,1)/PCI(0|0)/Scsi(Pun0,Lun0)
  blk2:
Acpi(PNP0A03,1)/PCI(0|0)/Scsi(Pun0,Lun0)/HD(Part1,Sig1B16CC00-ABD0-
01C)
  blk3: Acpi(PNP0A03,1)/PCI(0|0)/Scsi(Pun0,Lun0)/HD(Part4,SigG0)
  blk4: Acpi(PNP0A03,0)/PCI(3|1)/Ata(Primary,Master)
fs1:\>
```

2. To remap **fs0:** from an LS-120 so that it is always called **floppy**, type the following in green at the Shell prompt:

```
fs1:\>map -d fs0
fs1:\>dh -p diskio
Handle dump by protocol 'diskio'
 15: diskio blkio fs DevPath(..i(3|1)/Ata(Secondary,Master))
 16: diskio blkio DevPath(..,1)/PCI(0|0)/Scsi(Pun0,Lun0))
 48: diskio blkio fs DevPath(..ABD0-01C0-507B-9E5F8078F531)) ESP
 49: diskio blkio fs DevPath(..i(Pun0,Lun0)/HD(Part4,SigG0)) ESP
 17: diskio blkio DevPath(..PCI(3|1)/Ata(Primary,Master))
fs1:\>map floppy 15
fs1:\>floppy:
floppy:\>map

Device mapping table

  fs1   :
Acpi(PNP0A03,1)/PCI(0|0)/Scsi(Pun0,Lun0)/HD(Part1,Sig1B16CC00-ABD0-0)
  fs2   : Acpi(PNP0A03,1)/PCI(0|0)/Scsi(Pun0,Lun0)/HD(Part4,SigG0)
  blk0  : Acpi(PNP0A03,0)/PCI(3|1)/Ata(Secondary,Master)
  blk1  : Acpi(PNP0A03,1)/PCI(0|0)/Scsi(Pun0,Lun0)
  blk2  :
Acpi(PNP0A03,1)/PCI(0|0)/Scsi(Pun0,Lun0)/HD(Part1,Sig1B16CC00-ABD0-0)
  blk3  : Acpi(PNP0A03,1)/PCI(0|0)/Scsi(Pun0,Lun0)/HD(Part4,SigG0)
  blk4  : Acpi(PNP0A03,0)/PCI(3|1)/Ata(Primary,Master)
  floppy:                                                    <-- Floppy
Acpi(PNP0A03,0)/PCI(3|1)/Ata(Secondary,Master)
```

Now the current directory is the root on **floppy**, which is the LS-120 drive on the ATAPI secondary channel master device.

How to Alias New Commands to the Shell

The Shell **alias** command displays, creates, or deletes aliases in the EFI Shell environment. An *alias* provides a new name for an existing EFI Shell command or an EFI application. Once the alias is created, it can be used to run the command or launch the EFI application. There are some aliases that are predefined in the EFI Shell environment. These aliases provide the equivalent MS-DOS* and Unix* names for the file manipulation commands. The examples below show typical output from help for this command.

Examples

To display all aliases in the current EFI environment, type the following in green at the Shell prompt:

```
Shell>alias
      dir  : ls
      md   : mkdir
      rd   : rm
      del  : rm
      copy : cp
```

To create an alias to the EFI environment, type the following in green at the Shell prompt:

```
Shell>alias myguid guid
Shell>alias
        dir    : ls
        md     : mkdir
        rd     : rm
        del    : rm
        copy   : cp
        myguid : guid
```

To delete an alias in the EFI environment, type the following in green at the Shell prompt:

```
Shell>alias -d myguid
Shell>alias
        dir    : ls
        md     : mkdir
        rd     : rm
        del    : rm
        copy   : cp
```

To add a volatile alias, which has an asterisk at the line head, in the current EFI environment, type the following in green at the Shell prompt:

```
Shell>alias -v fs0 floppy
Shell>alias
        dir  : ls
        md   : mkdir
        rd   : rm
        del  : rm
        copy : cp
      * fs0  : floppy
```

This volatile alias will disappear at the next boot.

To add an alias with parameters, type the following in green at the Shell prompt:

```
Shell>alias "dir /p" "ls -b"
Shell>alias
        dir    : ls
        md     : mkdir
        rd     : rm
        del    : rm
        copy   : cp
        dir /p : ls -b
Shell>"dir /p"
```

How to Set an EFI Shell Variable

To add new Shell variables, use the **set** command. At the Shell prompt, type the following in green:

```
Shell>set varname value
```

The **set** command is used to maintain the environment variables that are available from the EFI environment. This command can do the following:

- Display the environment variables.
- Create new environment variables.
- Change the value of existing environment variables.
- Delete environment variables.

The **set** command will set the environment variable specified by **sname** to **value**. This form of the command can be used to create a new environment variable or to modify an existing environment variable. If the **set** command is used without any parameters, then all the environment variables are displayed. If the set command is used with the **-d** option, then the environment variable specified by **sname** will be deleted. The following examples show the typical output from the help for this command.

Examples

To create an environment variable, type the following in green at the Shell prompt:

```
Shell>set diagnosticPath fs0:\efi\diag;fs1:\efi\diag
```

To display all the environment variables, type the following in green at the Shell prompt:

```
Shell>set
    path           : .
    diagnosticPath : fs0:\efi2.0\diag;fs1:\efi2.0\diag
```

To delete an environment variable, type the following in green at the Shell prompt:

```
Shell>set -d diagnosticPath
Shell>set
    path           : .
```

To change an environment variable, type the following in green at the Shell prompt:

```
fs0:\>set src efi
fs0:\>set
    path : .;fs0:\efi\tools;fs0:\efi\boot;fs0:\
    src  : efi
fs0:\>set src efi2.0
fs0:\>set
    path : .;fs0:\efi\tools;fs0:\efi\boot;fs0:\
    src  : efi2.0
```

How to Edit a File

The Edit program is provided as part of the EFI Shell to edit text files on EFI systems. Type **`edit`** at the Shell prompt to invoke the editor. Edit is commonly used to examine files or create **`.nsh`** Shell files.

For example, to edit a file **`shell.log`** using the Edit program, type the following in green at the Shell prompt:

```
fs0:\>edit shell.log
```

By default, the Edit program creates text files that are in Unicode format. Hitting F9 will change the format to standard ASCII text files that most other systems can read and write. To save a file, hit F2 and type in the file name to save the file as. To exit Edit, hit F3.

If you are using a remote console with Edit, the function keys must be assigned in the remote terminal program so that the input console can interpret them correctly. See Appendix B of the *EFI 1.10 Specification* on the EFI web site to correctly assign the esc characters for your terminal program. This step is necessary because most of the basic save and open functions require function keys in Edit.

NOTES

1. *If the file is not specified,* **`NewFile.txt`** *is edited.*
2. *The size of file must not be larger than 16 MB.*

How to Make a Batch Script File (.nsh)

In EFI, **.nsh** files batch files are very similar to MS-DOS* **.bat** files. The syntax is very similar to MS-DOS*, and most batch files can easily be converted with some minor changes. See the *EFI 1.1 Shell Commands Specification* that is included in the EFI 1.10 Sample Implementation for more information.

Any command at the command prompt can also be executed from a **.nsh** file. Using this file allows tasks to be scripted and allows them to be replayed by typing the **.nsh** file without the **.nsh** extension.

If you name the following file **helloworld.nsh**:

```
echo "Hello world"
pause
```

Then when **helloworld** is typed at the prompt, you would see the following:

```
Hello>echo Hello world
Hello world
hello> pause
Enter 'q' to quit, any other key to continue:
```

Hitting the enter key would then allow the **.nsh** shell script to finish and return to the Shell prompt.

Examples

The following example is a script that can be used to initialize the network stack:

```
echo Loading TCP/IP stack...
load \efi\tools\tcpipv4.efi
if not %lasterror% == 0 then
echo Error starting network stack
goto Done
echo loaded tcpipv4
cd \
:Done
```

The following is an example of creating a Shell variable called **PYTHON PATH** and adding the RAM drive to the current path:

```
set PYTHONPATH ram:\;%path%
```

The following is an example of printing all **.nsh** files using "for" statements:

```
echo -off
echo "For loop to print all scripts in this directory..."
for %x in *.nsh
echo For loop iteration for %x
endfor
echo "Nested for loops..."
for %i in 1 2 3
for %j in a b c
echo i == %i , j == %j
endfor
endfor
echo "Done."
echo -on
```

The following is an example of redirection in a script:

```
echo -on
echo "Output redirection of a command"
echo "The quick brown fox jumped over the purple cow" 1>a cow.txt
```

7
EFI Drivers

How to Identify Hardware in a System Managed by EFI 1.10 Drivers

You can use the following two Shell commands to identify the hardware in a system that is managed by EFI 1.10 drivers:

- **`Drivers`** command: Displays the list of drivers that follow the EFI Driver Model.
- **`DH`** command: Displays the device handles in the EFI environment.

Drivers Command Example

The **`drivers`** command lists all the drivers in the system.

An EFI 1.10 driver is a software program that manages a specific piece of hardware in the system. An EFI 1.10 driver includes additional information that allows the system or user to identify certain features of the driver.

To list all the EFI 1.10 drivers in the system at the current time, type the following in green at the Shell prompt:

```
fs0:\>drivers
            T   D
D           Y C I
R           P F A
V  VERSION  E G G #D #C DRIVER NAME IMAGE NAME
== ======== = = = == == ================================================
18 00000010 B - -  5 38 PCI Bus Driver PciBus
19 00000010 D - -  1  - PC-AT ISA Device Enumeration Driver PcatIsaAcpi
1A 00000010 B - -  1  3 ISA Bus Driver IsaBus
1B 00000010 B - -  2  2 ISA Serial Driver IsaSerial
1C 00000000 ? - -  -  - BIOS[INT10] VGA Mini Port Driver BiosVgaMiniPort
1D 00000010 ? - -  -  - VGA Class Driver VgaClass
1E 00000010 D - -  1  - UGA Console Driver GraphicsConsole
1F 00000010 B - -  2  1 Serial Terminal Driver Terminal
20 00000010 D - -  2  - Platform Console Management Driver ConPlatform
21 00000010 D - -  2  - Platform Console Management Driver ConPlatform
25 00000010 B - -  2  2 Console Splitter Driver ConSplitter
26 00000010 ? - -  -  - Console Splitter Driver ConSplitter
27 00000010 B - -  2  2 Console Splitter Driver ConSplitter
28 00000010 B - -  2  2 Console Splitter Driver ConSplitter
2C 00000010 D - -  2  - Usb Uhci Driver UsbUhci
2D 00000010 B - -  2  1 USB Bus Driver UsbBus
2E 00000010 ? - -  -  - <UNKNOWN> UsbCbi1
2F 00000010 ? - -  -  - Usb Bot Mass Storage Driver UsbBot
30 00000010 ? - -  -  - Usb Cbi0 Mass Storage Driver UsbCbi0
31 00000010 ? - -  -  - Generic USB Mass Storage Driver UsbMassStorage
32 00000010 D - -  1  - Usb Keyboard Driver UsbKeyboard
33 00000114 D - -  1  - ATI Rage XL UGA Driver PciRom Seg=00000000
```

```
61 00000010 D - -  6  - Generic Disk I/O Driver DiskIo
62 00000010 B - -  1  3 Partition Driver(MBR/GPT/El Torito) Partition
63 00000010 D - -  2  - FAT File System Driver Fat
64 00000010 ? - -  -  - Intel(R) PRO 100 UNDI Driver Undi
65 00000030 B - -  1  1 Intel(R) PRO 1000 UNDI Driver v1.20 GigUndi
66 00000010 D - -  1  - Simple Network Protocol Driver Snp3264
67 00000010 D - -  1  - PXE Base Code Driver PxeBc
68 00000010 D - -  1  - PXE DHCPv4 Driver PxeDhcp4
69 00000010 ? - -  -  - Usb Mouse Driver UsbMouse
75 01020B00 D X -  2  - LSI Logic Ultra320 SCSI Driver VenHw(6D9FEEB1...
```

The following table describes the columns that are shown in the example above.

Column Name	Description
DRV	Describes the handle number in the handle database at which the driver is loaded. The handle number is useful for commands such as **connect**, **disconnect**, **dh**, **drvcfg**, and **drvdiag**.
Version	Indicates which version number of the driver is loaded. If the driver number is less than 10 hex (or 0f), the driver is a legacy driver that does not adhere to the *EFI 1.10 Specification.* It may make a callback into the legacy BIOS such as an INT 16 or INT 10.
TYPE	Describes if the driver is a bus driver or device driver. If a question mark (**?**) is displayed, the driver may be loaded in the handle database but is not currently being used by the system.
CFG	Indicates if the driver includes a setup configuration program. If the driver has one, it can be invoked by executing the **drvcfg** Shell command and passing in the driver handle number and controller that the driver is managing.
DIAG	Indicates if the driver includes a diagnostic self test program. If the driver has one, it can be invoked by executing the **drvdiag** Shell command and passing the driver handle number and controller that the driver is managing.
#D	Indicates the number of devices the driver is managing currently.
#C	Indicates the number of child devices the driver has produced.
driver name	Lists the string pulled from the driver for a human to be able to identify what the driver is for and may include a version number.
image name	Name that is used when the driver is compiled.

If a driver is active and managing something, it will need to have some number of **#D** or **#C** active.

Use the Shell command **drivers -b** to halt the screen after one screen's worth of drivers being displayed.

DH Command Example

Use the Shell command **dh -d** to see the specific information on any given EFI 1.10 driver. Only fully compliant EFI 1.10 drivers will display any detailed information about the drivers. From this command you can determine the version number, name, and specific controllers that the driver is managing.

For example, using the **drivers** command example, type the following in green at the Shell prompt (75 is the handle number or drive number):

```
fs0:\>dh -d 75
75: Image(VenHw(6D9FEEB1-E585-22D3-BC22-0080C73C8881)) DriverBinding
ComponentName Configuration
      Driver Name       : LSI Logic Ultra320 SCSI Driver
      Image Name        : VenHw(6D9FEEB1-E585-22D3-BC22-0080C73C8881)
      Driver Version    : 01020B00
      Driver Type       : DEVICE
      Configuration     : YES
      Diagnostics       : NO
      Managing          :
      Ctrl[46]          : LSI Logic Ultra320 SCSI Controller
      Ctrl[47]          : LSI Logic Ultra320 SCSI Controller
```

Executing the **DH -d** command on the controller that the driver is managing will display further information about which driver(s) are managing the controller and which handle is the parent to the controller. The device path will also tell you where the controller is in the system (typically on a specific host bus bridge and at a specific PCI device and function number). This is also the same information that the **map** command will use if the device produces a block I/O that is managing a FAT32 EFI system partition. The **DH -d <controller #>** command will also tell you which device is a child device if it is produced by the driver.

```
fs0:\>dh -d 46
46: PciIo ScsiPassThru DevPath (..NP0A03,1)/Pci(1F|0)/Pci(2|0))
      Controller Name       : LSI Logic Ultra320 SCSI Controller
      Device Path           : Acpi(PNP0A03,1)/Pci(1F|0)/Pci(2|0)
      Controller Type       : DEVICE
      Configuration         : YES
      Diagnostics           : NO
      Managed by :
          Drv[75] : LSI Logic Ultra320 SCSI Driver
      Parent Controllers :
          Parent[14] : Acpi(PNP0A03,1)
      Child Controllers : <NONE>
```

In this particular case, driver 75 is the LSI Logic* U320 SCSI driver. It is active and managing two SCSI controllers (controller 46 and 47). It is version 01020B00 and it has a setup configuration program.

How to Load, Start, and Stop an EFI 1.10 Driver

This topic discusses the following:

- How to load an EFI 1.10 driver
- How to start an EFI 1.10 driver
- How to stop an EFI 1.10 driver

How to Load an EFI 1.10 Driver

The **load** command, by default, will run the **connect** command on the driver as soon as it is loaded. This action will effectively start the driver and it will start managing any devices in the system that are not already started. If the driver is one of many drivers that need to be loaded to function correctly, then the **-nc** option may be used to load the driver but not run **connect** or start the driver. As is typical of some drivers, you may need to load three of them for the drivers to function. For example:

```
fs0:/>load -nc usbuchi.efi
fs0:/>load -nc usbbus.efi
fs0:/>load -nc usbkeyboard.efi
fs0:/>connect -r
```

This example would load all three USB drivers but not start them until the **connect** command was executed. The load order is critical to **usbbus.efi** and **usbkeyboard.efi** because **usbbus.efi** must run after **usbuhci.efi** is loaded and **usbkeyboard.efi** must loaded after **usbbus.efi** is started.

See the **bcfg** command for loading a driver automatically on every boot.

How to Start an EFI 1.10 Driver

To start an EFI driver, type **connect** at the Shell prompt once you know at which handle number the driver is (see the **drivers** command).

For example, to start the LSI Logic* U320 SCSI driver from the **drivers** command example, you would type the following in green at the Shell prompt:

```
fs0:/>connect 75
```

This example would start the driver if it is not already managing some devices and controllers as it is in the **drivers** command example.

How to Stop an EFI 1.10 Driver

To stop an EFI driver, type **disconnect** at the Shell prompt once you know what controllers a driver is currently managing (see the **drivers** and **dh -d C#** command examples). In the **drivers** example, to stop the driver from managing all the SCSI drivers on the LSI U320 SCSI controller, you would type the following in green at the Shell prompt:

```
fs0:/>disconnect <device handle>
```

in this case:

```
fs0:/>disconnect 46
```

This example would stop the driver from managing the first SCSI bus on the LSI U320 SCSI controller and destroy any block I/O handles and file systems.

Type **disconnect -?** at the Shell prompt to see all variations of this command:

```
fs0:/>disconnect -?
disconnect DeviceHandle# [DriverHandle# [ChildHandle#]] | [-r]

    DeviceHandle#   - Device handle (hex)
    DriverHandle#   - Driver handle (hex)
    ChildHandle#    - Child handle of device (hex)
    -r              - Disconnect drivers from all devices
```

How to List All EFI Devices in a System

Use the Shell **devices** command to display the list of devices that are being managed by EFI drivers that follow the EFI Driver Model.

To run the **devices** command, type the following in green at the Shell prompt:

```
fs0:\>devices
C  T D
T  Y C I
R  P F A
L  E G G #P #D #C Device Name
== = = = == == == ============
=================================================
0A R - -  -  2  - FAT File System [FAT16] 120 MB
0B R - -  -  1  - Acpi(PNP0A03,0)/Pci(1F|1)/Ata(Primary,Master)
13 R - -  -  1  7 Acpi(PNP0A03,0)
14 R - -  -  1  8 Acpi(PNP0A03,1)
15 R - -  -  1  6 Acpi(PNP0A03,2)
16 R - -  -  1  6 Acpi(PNP0A03,3)
17 R - -  -  1 11 Acpi(PNP0A03,4)
29 D - -  2  -  - Primary Standard Error Device
2A D - -  2  -  - Primary Console Input Device
2B D - -  2  -  - Primary Console Output Device
34 B - -  1  2  1 Usb Universal Host Controller
35 D - -  1  2  - Usb Universal Host Controller
```

```
36 D - -  1  -  - Acpi(PNP0A03,0)/Pci(1D|7)
37 D - -  1  -  - Acpi(PNP0A03,0)/Pci(1E|0)
38 B - -  1  1  1 Acpi(PNP0A03,0)/Pci(1E|0)/Pci(0|0)
39 B - -  1  5  2 ATI Rage XL Video Adapter
3A B - -  1  2  3 Acpi(PNP0A03,0)/Pci(1F|0)
3B B - -  1  3  1 Generic Usb Keyboard
3C B - -  1  1  1 Acpi(PNP0A03,0)/Pci(1F|0)/Acpi(PNP0501,0)
3D B - -  1  1  1 Acpi(PNP0A03,0)/Pci(1F|0)/Acpi(PNP0501,1)
3E D - -  1  -  - Acpi(PNP0A03,0)/Pci(1F|0)/Acpi(PNP0303,0)
3F D - -  1  1  - Acpi(PNP0A03,0)/Pci(1F|0)/Acpi(PNP0501,0)/Uart(115200 N81)
40 B - -  1  1  1 Acpi(PNP0A03,0)/Pci(1F|0)/Acpi(PNP0501,1)/Uart(115200 N81)
41 D - -  1  -  - Acpi(PNP0A03,1)/Pci(1C|0)
42 D - -  1  -  - Acpi(PNP0A03,1)/Pci(1D|0)
43 D - -  1  -  - Acpi(PNP0A03,1)/Pci(1D|0)/Pci(1F|0)
44 D - -  1  -  - Acpi(PNP0A03,1)/Pci(1E|0)
45 D - -  1  -  - Acpi(PNP0A03,1)/Pci(1F|0)
46 D X -  1  1  - LSI Logic Ultra320 SCSI Controller
47 D X -  1  1  - LSI Logic Ultra320 SCSI Controller
48 D - -  1  -  - Acpi(PNP0A03,1)/Pci(1F|0)/Pci(1F|0)
49 D - -  1  -  - Acpi(PNP0A03,2)/Pci(1C|0)
4A D - -  1  -  - Acpi(PNP0A03,2)/Pci(1D|0)
4B D - -  1  -  - Acpi(PNP0A03,2)/Pci(1D|0)/Pci(1F|0)
4C D - -  1  -  - Acpi(PNP0A03,2)/Pci(1E|0)
4D D - -  1  -  - Acpi(PNP0A03,2)/Pci(1F|0)
4E D - -  1  -  - Acpi(PNP0A03,2)/Pci(1F|0)/Pci(1F|0)
4F D - -  1  -  - Acpi(PNP0A03,3)/Pci(1C|0)
50 D - -  1  -  - Acpi(PNP0A03,3)/Pci(1D|0)
51 D - -  1  -  - Acpi(PNP0A03,3)/Pci(1D|0)/Pci(1F|0)
52 D - -  1  -  - Acpi(PNP0A03,3)/Pci(1E|0)
53 D - -  1  -  - Acpi(PNP0A03,3)/Pci(1F|0)
54 D - -  1  -  - Acpi(PNP0A03,3)/Pci(1F|0)/Pci(1F|0)
55 D - -  1  -  - Acpi(PNP0A03,4)/Pci(18|0)
56 D - -  1  -  - Acpi(PNP0A03,4)/Pci(18|1)
57 D - -  1  -  - Acpi(PNP0A03,4)/Pci(18|2)
58 D - -  1  -  - Acpi(PNP0A03,4)/Pci(18|3)
59 D - -  1  -  - Acpi(PNP0A03,4)/Pci(1C|0)
5A D - -  1  -  - Acpi(PNP0A03,4)/Pci(1C|1)
5B D - -  1  -  - Acpi(PNP0A03,4)/Pci(1C|2)
5C D - -  1  -  - Acpi(PNP0A03,4)/Pci(1C|3)
5D D - -  1  -  - Acpi(PNP0A03,4)/Pci(1C|4)
5E D - -  1  -  - Acpi(PNP0A03,4)/Pci(1C|5)
5F D - -  1  -  - Acpi(PNP0A03,4)/Pci(1C|6)
60 B - -  1  5  3 PC-ANSI Serial Console
7C D - -  1  3  - Acpi(PNP0A03,0)/Pci(1E|0)/Pci(0|0)/Mac(0002B300076A)
7D R - -  2  3  - Acpi(PNP0A03,1)/Pci(1F|0)/Pci(2|0)/Scsi(Pun0,Lun0)
A5 D - -  1  2  - FAT File System [FAT16] 172 MB
A6 D - -  1  1  - Acpi(PNP0A03,1)/Pci(1F|0)/Pci(2|0)/Scsi
(Pun0,Lun0)/HD(Part2,Sig898D07A0-F474-01C2-F1B3-12714F758821)
A7 D - -  1  1  - Acpi(PNP0A03,1)/Pci(1F|0)/Pci(2|0)/Scsi(Pun0,Lun0)/HD
(Part3,Sig89919B80-F474-01C2-D931-F8428177D974)
```

This command lists all the devices in the system that have been started and that are being actively managed. It is sorted by controller number in the CTRL column. Every PCI bus slot being managed by the PCI bus driver, as well as every hard drive controller that is actively being managed by EFI drivers, is displayed.

How to List Devices in a System

To see the relationship between controllers, use the Shell **devtree** command. This command will display the hierarchical relationship between the controllers that are being managed.

In this example, each PCI host bus is seen managing PCI devices in the system. The PCI host bus, **cntrl[14]**, is managing the LSI Logic* U320 SCSI controllers. The first PCI host bus bridge, **cntrl[13]**, is managing the USB controller and all the console input and output devices on the south bridge. Each indentation in the listing reveals what devices are subordinate or are being managed by a given controller.

Examples

To display the tree of all devices below device 28 that follow the EFI Driver Model, type the following in green at the Shell prompt:

```
Shell>devtree 28
```

To display the tree of all devices that follow the EFI Driver Model and then break when the screen is full, type the following in green at the Shell prompt:

```
Shell>devtree -b
```

To display the tree of all devices that follow the EFI Driver Model, type the following in green at the Shell prompt:

```
Shell>devtree
 Device Tree
   Ctrl[03]
   Ctrl[09] Acpi(PNP0A03,0)/Pci(1F|1)
   Ctrl[0A] FAT File System [FAT16] 120 MB
   Ctrl[0B] Acpi(PNP0A03,0)/Pci(1F|1)/Ata(Primary,Master)
   Ctrl[13] Acpi(PNP0A03,0)
     Ctrl[34] Usb Universal Host Controller
       Ctrl[3B] Generic Usb Keyboard
         Ctrl[2A] Primary Console Input Device
     Ctrl[35] Usb Universal Host Controller
     Ctrl[36] Acpi(PNP0A03,0)/Pci(1D|7)
     Ctrl[37] Acpi(PNP0A03,0)/Pci(1E|0)
     Ctrl[38] Acpi(PNP0A03,0)/Pci(1E|0)/Pci(0|0)
       Ctrl[7C] Acpi(PNP0A03,0)/Pci(1E|0)/Pci(0|0)/Mac(0002B300076A)
     Ctrl[39] ATI Rage XL Video Adapter
       Ctrl[2B] Primary Console Output Device
       Ctrl[29] Primary Standard Error Device
     Ctrl[3A] Acpi(PNP0A03,0)/Pci(1F|0)
       Ctrl[3C] Acpi(PNP0A03,0)/Pci(1F|0)/Acpi(PNP0501,0)
```

```
          Ctrl[3F]
Acpi(PNP0A03,0)/Pci(1F|0)/Acpi(PNP0501,0)/Uart(115200 N
       Ctrl[3D] Acpi(PNP0A03,0)/Pci(1F|0)/Acpi(PNP0501,1)
          Ctrl[40]
Acpi(PNP0A03,0)/Pci(1F|0)/Acpi(PNP0501,1)/Uart(115200 N
           Ctrl[60] PC-ANSI Serial Console
             Ctrl[2A] Primary Console Input Device
             Ctrl[2B] Primary Console Output Device
             Ctrl[29] Primary Standard Error Device
       Ctrl[3E] Acpi(PNP0A03,0)/Pci(1F|0)/Acpi(PNP0303,0)
   Ctrl[14] Acpi(PNP0A03,1)
     Ctrl[41] Acpi(PNP0A03,1)/Pci(1C|0)
     Ctrl[42] Acpi(PNP0A03,1)/Pci(1D|0)
     Ctrl[43] Acpi(PNP0A03,1)/Pci(1D|0)/Pci(1F|0)
     Ctrl[44] Acpi(PNP0A03,1)/Pci(1E|0)
     Ctrl[45] Acpi(PNP0A03,1)/Pci(1F|0)
     Ctrl[46] LSI Logic Ultra320 SCSI Controller
     Ctrl[47] LSI Logic Ultra320 SCSI Controller
     Ctrl[48] Acpi(PNP0A03,1)/Pci(1F|0)/Pci(1F|0)
   Ctrl[15] Acpi(PNP0A03,2)
     Ctrl[49] Acpi(PNP0A03,2)/Pci(1C|0)
     Ctrl[4A] Acpi(PNP0A03,2)/Pci(1D|0)
     Ctrl[4B] Acpi(PNP0A03,2)/Pci(1D|0)/Pci(1F|0)
     Ctrl[4C] Acpi(PNP0A03,2)/Pci(1E|0)
     Ctrl[4D] Acpi(PNP0A03,2)/Pci(1F|0)
     Ctrl[4E] Acpi(PNP0A03,2)/Pci(1F|0)/Pci(1F|0)
   Ctrl[16] Acpi(PNP0A03,3)
     Ctrl[4F] Acpi(PNP0A03,3)/Pci(1C|0)
     Ctrl[50] Acpi(PNP0A03,3)/Pci(1D|0)
     Ctrl[51] Acpi(PNP0A03,3)/Pci(1D|0)/Pci(1F|0)
     Ctrl[52] Acpi(PNP0A03,3)/Pci(1E|0)
     Ctrl[53] Acpi(PNP0A03,3)/Pci(1F|0)
     Ctrl[54] Acpi(PNP0A03,3)/Pci(1F|0)/Pci(1F|0)
   Ctrl[17] Acpi(PNP0A03,4)
     Ctrl[55] Acpi(PNP0A03,4)/Pci(18|0)
     Ctrl[56] Acpi(PNP0A03,4)/Pci(18|1)
     Ctrl[57] Acpi(PNP0A03,4)/Pci(18|2)
     Ctrl[58] Acpi(PNP0A03,4)/Pci(18|3)
     Ctrl[59] Acpi(PNP0A03,4)/Pci(1C|0)
     Ctrl[5A] Acpi(PNP0A03,4)/Pci(1C|1)
     Ctrl[5B] Acpi(PNP0A03,4)/Pci(1C|2)
     Ctrl[5C] Acpi(PNP0A03,4)/Pci(1C|3)
     Ctrl[5D] Acpi(PNP0A03,4)/Pci(1C|4)
     Ctrl[5E] Acpi(PNP0A03,4)/Pci(1C|5)
     Ctrl[5F] Acpi(PNP0A03,4)/Pci(1C|6)
   Ctrl[71] VenHw(6D9FEEB1-E585-11D3-BC22-0080C73C8881)
   Ctrl[74] VenHw(6D9FEEB1-E585-22D3-BC22-0080C73C8881)
   Ctrl[79] VenHw(D65A6B8C-71E5-4DF0-A909-F0D2992B5AA9)
   Ctrl[7B] VenHw(43086B9B-6F1A-11D4-87AB-00062945C3B9)
   Ctrl[7D] Acpi(PNP0A03,1)/Pci(1F|0)/Pci(2|0)/Scsi(Pun0,Lun0)
     Ctrl[A5] FAT File System [FAT16] 172 MB
     Ctrl[A6] Acpi(PNP0A03,1)/Pci(1F|0)/Pci(2|0)/Scsi(Pun0,Lun0)
              /HD(Part2,Sig898D 07A0-F474-01C2-F1B3-12714F758821)
```

```
      Ctrl[A7] Acpi(PNP0A03,1)/Pci(1F|0)/Pci(2|0)/Scsi(Pun0,Lun0)
               /HD(Part3,Sig8991 9B80-F474-01C2-D931-F8428177D974)
    Ctrl[7E] Acpi(PNP0A03,1)/Pci(1F|0)/Pci(2|0)/Scsi(Pun6,Lun0)
```

How to Run EFI 1.10 Driver Configuration

To run the setup configuration program on a card that is managed by an EFI 1.10 driver, use the Shell **drvcfg** command. This command replaces the Ctrl-Alt-<key> sequence that most PCI card option ROMs used in legacy BIOS.

Using the **drivers** command, obtain the list of all the drivers.

If the LSI Logic* U320 SCSI driver (driver 75) had a diagnostic test program embedded in the driver, the CFG column in the **drivers** command would contain an *X* (as shown in green below):

```
...
75 01020B00 D X X  2  - LSI Logic Ultra320 SCSI Driver
VenHw(6D9FEEB1...
...
```

Use the **dh** command to obtain the list of controllers for the SCSI driver (driver 75), by typing the following in green at the Shell prompt:

```
fs0:\>dh -d 75
75: Image(VenHw(6D9FEEB1-E585-22D3-BC22-0080C73C8881)) DriverBinding
ComponentName Configuration
      Driver Name       : LSI Logic Ultra320 SCSI Driver
      Image Name        : VenHw(6D9FEEB1-E585-22D3-BC22-0080C73C8881)
      Driver Version    : 01020B00
      Driver Type       : DEVICE
      Configuration     : YES
      Diagnostics       : NO
      Managing          :
      Ctrl[46]          : LSI Logic Ultra320 SCSI Controller
      Ctrl[47]          : LSI Logic Ultra320 SCSI Controller

fs0:>drvcfg -s 75 46
```

In this last example, **-s** means to run setup on the driver; **75** is the driver handle for the LSI U320 SCSI driver, and **46** is the controller that the LSI U320 SCSI EFI driver is managing.

How to Run Diagnostics for EFI 1.10 Drivers

Using the **drivers** command, obtain the list all the drivers.

If the LSI Logic* U320 SCSI driver had a diagnostic test program embedded in it, the DIAG column in the **drivers** command would contain an X (in green below).

```
             T   D
D            Y C I
R            P F A
V  VERSION   E G G #D #C DRIVER NAME IMAGE NAME
== ======== = = = == == ================================================
...
75 01020B00 D X X  2  - LSI Logic Ultra320 SCSI Driver VenHw(6D9FEEB1...
...
```

To see all the devices that have diagnostics for them, type the following in green at the Shell prompt:

```
fs0:\>drvdiag -c
```

To run standard diagnostics on all drivers in the system, type the following in green at the Shell prompt:

```
fs0:\>drvdiag -s
```

Use the **dh** command for the SCSI driver (driver 75) to obtain the list of controllers, as shown in the first line in green below, and then type the command in green at the bottom of this example to run the standard diagnostic test on the driver at handle 75.

```
fs0:\>dh -d 75
75: Image(VenHw(6D9FEEB1-E585-22D3-BC22-0080C73C8881)) DriverBinding
ComponentName Configuration
      Driver Name       : LSI Logic Ultra320 SCSI Driver
      Image Name        : VenHw(6D9FEEB1-E585-22D3-BC22-0080C73C8881)
      Driver Version    : 01020B00
      Driver Type       : DEVICE
      Configuration     : YES
      Diagnostics       : NO
      Managing          :
      Ctrl[46]          : LSI Logic Ultra320 SCSI Controller
      Ctrl[47]          : LSI Logic Ultra320 SCSI Controller

fs0:\>drvdiag -s 75
```

The **drivers** command example would run the diagnostics on the LSI U320 SCSI card.

To run the standard diagnostic test on the LSI U320 SCSI controller 46 (the first SCSI bus on the LSI card), type the following in green at the Shell prompt:

```
fs0:\>drvdiag -s 75 46
```

How to Test an EFI Block Device

Use the Shell **dblk** command to test an EFI block device. This command dumps the first blocks from a Block I/O Protocol device (such as the EFI system partition on a SCSI disk).

First, use the **map** command to identify a block I/O device with a FAT32 file system on it. To do so, type the following in green at the Shell prompt:

```
Shell>map -r

Device mapping table

fs0 : Acpi(PNP0A03,0)/Pci(1F|1)/Ata(Primary,Slave)
fs1 : Acpi(PNP0A03,1)/Pci(1F|0)/Pci(2|0)/Scsi(Pun0,Lun0)/HD(Part1,Sig
       8983DFE0 -F474-01C2-507B-9E5F8078F531)
blk0 : Acpi(PNP0A03,0)/Pci(1F|1)/Ata(Primary,Slave)
blk1 : Acpi(PNP0A03,0)/Pci(1F|1)/Ata(Primary,Master)
blk2 : Acpi(PNP0A03,1)/Pci(1F|0)/Pci(2|0)/Scsi(Pun0,Lun0)
blk3 :
Acpi(PNP0A03,1)/Pci(1F|0)/Pci(2|0)/Scsi(Pun0,Lun0)/HD(Part1,Sig
       8983DFE0-F474-01C2-507B-9E5F8078F531)
blk4 :
Acpi(PNP0A03,1)/Pci(1F|0)/Pci(2|0)/Scsi(Pun0,Lun0)/HD(Part2,Sig
       898D07A0-F474-01C2-F1B3-12714F758821)
blk5 :
Acpi(PNP0A03,1)/Pci(1F|0)/Pci(2|0)/Scsi(Pun0,Lun0)/HD(Part3,Sig
       89919B80-F474-01C2-D931-F8428177D974)
```

In this case, **blk0**, which is the floppy, has a FAT file system on it because **FS0:** has exactly the same device path: **Acpi(PNP0A03,0)/Pci(1F|1)/Ata(Primary,Slave)**.

Next, use the **dblk** command with the block number, the starting block, and the number of blocks in the command to dump the blocks. To do so, type the following in green at the Shell prompt:

```
Shell>dblk blk0 0 1

LBA 0x0000000000000000 Size 0x00000200 bytes BlkIo 0x3D379818
00000000: EB 3C 90 4D 53 44 4F 53-35 2E 30 00 02 04 01 00 *.<.MSDOS5.0.....*
00000010: 02 00 02 00 00 F0 F1 00-20 00 08 00 00 00 00 00 *........ .......*
00000020: 00 C3 03 00 00 01 29 4A-D9 FE 92 53 75 70 65 72 *......)J...Super*
00000030: 44 69 73 6B 20 20 46 41-54 31 36 20 20 20 FA 33 *Disk FAT16 .3*
00000040: C0 8E D0 BC 00 7C 16 07-BB 78 00 36 C5 37 1E 56 *.........x.6.7.V*
00000050: 16 53 BF 3E 7C B9 0B 00-FC F3 A4 06 1F C6 45 FE *.S.>..........E.*
00000060: 0F 8B 0E 18 7C 88 4D F9-89 47 02 C7 07 3E 7C FB *......M..G...>..*
00000070: CD 13 72 79 33 C0 39 06-13 7C 74 08 8B 0E 13 7C *..ry3.9...t.....*
00000080: 89 0E 20 7C A0 10 7C F7-26 16 7C 03 06 1C 7C 13 *.. .....&.......*
00000090: 16 1E 7C 03 06 0E 7C 83-D2 00 A3 50 7C 89 16 52 *..........P...R*
000000A0: 7C A3 49 7C 89 16 4B 7C-B8 20 00 F7 26 11 7C 8B *..I...K.. ..&...*
000000B0: 1E 0B 7C 03 C3 48 F7 F3-01 06 49 7C 83 16 4B 7C *.....H....I...K.*
000000C0: 00 BB 00 05 8B 16 52 7C-A1 50 7C E8 92 00 72 1D *......R..P....r.*
000000D0: B0 01 E8 AC 00 72 16 8B-FB B9 0B 00 BE E6 7D F3 *.....r..........*
000000E0: A6 75 0A 8D 7F 20 B9 0B-00 F3 A6 74 18 BE 9E 7D *.u... .....t....*
000000F0: E8 5F 00 33 C0 CD 16 5E-1F 8F 04 8F 44 02 CD 19 *._.3...^....D...*
00000100: 58 58 58 EB E8 8B 47 1A-48 48 8A 1E 0D 7C 32 FF *XXX...G.HH....2.*
```

```
00000110: F7 E3 03 06 49 7C 13 16-4B 7C BB 00 07 B9 03 00 *....I...K.......*
00000120: 50 52 51 E8 3A 00 72 D8-B0 01 E8 54 00 59 5A 58 *PRQ.:.r....T.YZX*
00000130: 72 BB 05 01 00 83 D2 00-03 1E 0B 7C E2 E2 8A 2E *r..............*
00000140: 15 7C 8A 16 24 7C 8B 1E-49 7C A1 4B 7C EA 00 00 *....$...I..K....*
00000150: 70 00 AC 0A C0 74 29 B4-0E BB 07 00 CD 10 EB F2 *p....t).........*
00000160: 3B 16 18 7C 73 19 F7 36-18 7C FE C2 88 16 4F 7C *;...s..6......O.*
00000170: 33 D2 F7 36 1A 7C 88 16-25 7C A3 4D 7C F8 C3 F9 *3..6....%..M....*
00000180: C3 B4 02 8B 16 4D 7C B1-06 D2 E6 0A 36 4F 7C 8B *.....M......6O..*
00000190: CA 86 E9 8A 16 24 7C 8A-36 25 7C CD 13 C3 0D 0A *.....$..6%......*
000001A0: 4E 6F 6E 2D 53 79 73 74-65 6D 20 64 69 73 6B 20 *Non-System disk *
000001B0: 6F 72 20 64 69 73 6B 20-65 72 72 6F 72 0D 0A 52 *or disk error..R*
000001C0: 65 70 6C 61 63 65 20 61-6E 64 20 70 72 65 73 73 *eplace and press*
000001D0: 20 61 6E 79 20 6B 65 79-20 77 68 65 6E 20 72 65 * any key when re*
000001E0: 61 64 79 0D 0A 00 49 4F-20 20 20 20 20 20 53 59 *ady...IO SY*
000001F0: 53 4D 53 44 4F 53 20 20-20 53 59 53 00 00 55 AA *SMSDOS SYS..U.*
Fat 16 BPB FatLabel: 'SuperDisk ' SystemId: 'FAT16 ' OemId: 'MSDOS5.0'
SectorSize 0x200 SectorsPerCluster 4 ReservedSectors 1 # Fats 2
Root Entries 0x200 Media 0xF0 Sectors 0x3C300 SectorsPerFat 0xF1
SectorsPerTrack 0x20 Heads 8
```

This example dumps the first block on the floppy and lists only one block. **Dblk** can also crack the FAT16 information on the floppy and display the relevant FAT data at the bottom of the screen.

This method is also an effective way to test a block I/O device by dumping the first and last block on the system to the screen (the last block is 0x3C2FF).

8
Identification

How to Determine the Amount of Memory in a System

Use the Shell **memmap** command to determine the amount of memory in a system. This command displays the memory map that is maintained by the EFI environment, and it is the EFI environment that keeps track of all the physical memory in the system and how it is currently being used. The *EFI Specification* defines a set of memory type descriptors.

To run the **memmap** command, type the following in green at the Shell prompt:

```
fs1:\>memmap

Type       Start             End               # Pages          Attributes
BS_data    0000000000000000-0000000000000FFF 0000000000000001 0000000000000009
available  0000000000001000-0000000000006FFF 0000000000000006 0000000000000009
BS_data    0000000000007000-0000000000008FFF 0000000000000002 0000000000000009
available  0000000000009000-0000000000081FFF 0000000000000079 0000000000000009
RT_data    0000000000082000-0000000000083FFF 0000000000000002 8000000000000009
available  0000000000084000-0000000000084FFF 0000000000000001 0000000000000009
BS_data    0000000000085000-000000000009FFFF 000000000000001B 0000000000000009
RT_code    00000000000C0000-00000000000FFFFF 0000000000000040 8000000000000009
available  0000000000100000-000000000FF7FFFF 000000000000FE80 000000000000000B
BS_data    000000000FF80000-000000000FFFFFFF 0000000000000080 000000000000000B
available  0000000010000000-000000003C8E7FFF 000000000002C8E8 000000000000000B
LoaderCode 000000003C8E8000-000000003C92DFFF 0000000000000046 000000000000000B
LoaderData 000000003C92E000-000000003C937FFF 000000000000000A 000000000000000B
available  000000003C938000-000000003C96BFFF 0000000000000034 000000000000000B
LoaderData 000000003C96C000-000000003C97DFFF 0000000000000012 000000000000000B
available  000000003C97E000-000000003CE29FFF 00000000000004AC 000000000000000B
BS_data    000000003CE2A000-000000003D000FFF 00000000000001D7 000000000000000B
available  000000003D001000-000000003D03DFFF 000000000000003D 000000000000000B
BS_data    000000003D03E000-000000003D075FFF 0000000000000038 000000000000000B
available  000000003D076000-000000003D079FFF 0000000000000004 000000000000000B
BS_data    000000003D07A000-000000003D091FFF 0000000000000018 000000000000000B
available  000000003D092000-000000003D0A7FFF 0000000000000016 000000000000000B
BS_data    000000003D0A8000-000000003D0B1FFF 000000000000000A 000000000000000B
available  000000003D0B2000-000000003D0B3FFF 0000000000000002 000000000000000B
BS_data    000000003D0B4000-000000003D0B5FFF 0000000000000002 000000000000000B
available  000000003D0B6000-000000003D0B7FFF 0000000000000002 000000000000000B
BS_data    000000003D0B8000-000000003D37DFFF 00000000000002C6 000000000000000B
available  000000003D37E000-000000003D703FFF 0000000000000386 000000000000000B
BS_code    000000003D704000-000000003D77DFFF 000000000000007A 000000000000000B
available  000000003D77E000-000000003D8B9FFF 000000000000013C 000000000000000B
RT_data    000000003D8BA000-000000003D8FFFFF 0000000000000046 8000000000000009
BS_code    000000003D900000-000000003F9BFFFF 00000000000020C0 000000000000000B
available  000000003F9C0000-000000003FA3FFFF 0000000000000080 000000000000000B
RT_code    000000003FA40000-000000003FDFFFFF 00000000000003C0 8000000000000009
PAL_code   000000003FE00000-000000003FE3FFFF 0000000000000040 8000000000000009
RT_code    000000003FE40000-000000003FE95FFF 0000000000000056 8000000000000009
```

```
available  000000003FE96000-000000003FECDFFF  0000000000000038 000000000000000B
RT_code    000000003FECE000-000000003FF39FFF  000000000000006C 8000000000000009
RT_data    000000003FF3A000-000000003FFFFFFF  00000000000000C6 8000000000000009
MemMapIO   00000000FE000000-00000000FEFFFFFF  0000000000001000 8000000000000001
RT_data    00000000FF000000-00000000FFFFFFFF  0000000000001000 8000000000000001
MemMapIO   00000FFFF8000000-00000FFFFBFFFFFF  0000000000004000 8000000000000001
MemPortIO  00000FFFFC000000-00000FFFFFFFFFFF  0000000000004000 8000000000000001

   LoaderCode:      70 Pages (286,720)
   LoaderData:      28 Pages (114,688)
   BS_code    :   8,506 Pages (34,840,576)
   BS_data    :   1,431 Pages (5,861,376)
   RT_code    :   1,218 Pages (4,988,928)
   RT_data    :   4,366 Pages (17,883,136)
   available : 250,525 Pages (1,026,150,400)
   MemMapIO  :  20,480 Pages (83,886,080)
   MemPortIO :  16,384 Pages (67,108,864)
   PAL_code  :      64 Pages (262,144)
 Total Memory: 1,039 MB (1,090,387,968) Bytes
```

In this example, the total memory in the system is 1 GB. EFI will total the flash space as well as the system RAM reported to it. This command will take a snapshot of the EFI memory map and indicate exactly which regions of memory are being assigned for EFI use and which are available.

How to Identify the Version Number of EFI and Other System Firmware

The Shell **ver** command displays the version information for EFI firmware. This information is retrieved through the EFI System Table.

To run the **ver** command, type the following in green at the Shell prompt:

```
fs1:\>ver
BIOS Version S870BN4A.86B.0882.P06.0303311722

 EFI Specification Revision : 1.10
 EFI Vendor                 : INTEL
 EFI Revision               : 14.62

 SAL Specification Revision      3.01
 SAL_A Revision  3.00, SAL_B Revision  3.00
 PAL_A Revision  7.31, PAL_B Revision  7.40
 FPSWA Version 1.12

 Other modules mentioned in FIT (Firmware Interface Table)
 FIT_Entry Type   0, Revision    3.00
 FIT_Entry Type  14, Revision    7.31
 FIT_Entry Type  14, Revision    5.15
 FIT_Entry Type  20, Revision    3.00
 FIT_Entry Type  48, Revision    0.01
 FIT_Entry Type  56, Revision    3.00
 FIT_Entry Type  64, Revision    3.00
 FIT_Entry Type 126, Revision    3.00

  SalProc Entry 000000003FE48020 and GP 000000004008BD20
  PalProc Entry 000000003FE08010 IO Port Base 00000FFFFC000000
  Cache Enabled
```

The Intel® BIOS string in this case is **BIOS Version S870BN4A.86B.0882.P06.0303311722**. The elements in this string indicate the following:

- **S870BN4A**: The Intel® system product identifier to identify the platform.
- **86B**: It is a server BIOS.
- **0882**: The build number. (Note: all releases are usually done in increasing order.)
- **P06**: It is the sixth production release.
- **0303311722**: The release date was March 3, 2003, at 17:22, or 5:22 pm.

This string is the one that should be used to identify the platform and BIOS for the system. The version of EFI on the system is 1.10.14.62 and is from Intel. The SAL revision (for Itanium®-based systems only) is 3.01. The **PAL_A** revision is 7.31 and **PAL_B** is 7.40; these numbers determine the stepping of the Intel® Itanium® processor that the BIOS firmware supports. For support, store this information to a file and email it back to Intel® support. They will inform the support group exactly what firmware is in your particular system.

To store a copy of this information, use the Shell command line redirection. For example, type the following in green at the Shell prompt:

```
fs1:\>ver 1>A Version.txt
```

This information is stored in an ASCII text file that can be emailed to an Intel support organization.

How to Identify the System GUID

The system GUID is the system identifier that is unique to each system that is programmed at the factory.

At the EFI boot manager, when the system boots up, select the boot maintenance menu. It will be displayed at the bottom of the display. As shown in green in the boot option maintenance menu below, the text highlighted in green is the system GUID.

Boot Manager

Select the option in green below and hit the enter key to go to the boot option maintenance menu.

```
EFI Boot Manager ver 1.10 [14.62]
Please select a boot option
Floppy/Pci(1F|1)/Ata(Primary,Slave)
CD/DVD ROM/Pci(1F|1)/Ata(Primary,Master)
Hard Drive/Pci(6|2|0)/Scsi(Pun0,Lun0)/HD(Part1,Sig8983DFE0-F474-...)
Network Boot/Pci(1|0|0)/Mac(0002B300076A)
EFI Shell [Built-in]

Boot option maintenance menu

Use ▲ and ▼ to change option(s). Use Enter to select an option.
```

Boot Option Maintenance Menu

The text highlighted in green below is the system GUID.

```
BIOS Version S870BN4A.86B.0882.P06.0303311722
Main Menu. Select an Operation

Boot from a File
Add a Boot Option
Delete Boot Option(s)
Change Boot Order

Manage BootNext setting
Set Auto Boot TimeOut

Cold Reset
Exit

Timeout-->[7] sec SystemGuid-->[01034B34-47E9-D711-BEA6-
0002B300076A]
SerialNumber-->[1234567890123 ]
```

How to Identify an EFI 1.10 Native Driver

A *native EFI driver* is a driver that is compiled for a specific processor (Intel® Itanium® processor or IA-32 processor).

When compiling a driver, if you set the compiler flags for a specific processor architecture, then the resulting driver will be a native EFI driver. This distinction means that the driver will work on only one architecture and that the **`.efi`** executable will run only on the intended processor—either an Itanium®-based system or an IA-32 processor-based system.

If you compiled the driver with the Intel® C Compiler for EFI Byte Code, then the driver is not a native EFI driver and will execute on both Itanium-based systems and IA-32 processor-based systems.

However, if you do not know how a driver was compiled, you can determine if a driver is a native EFI driver in the following ways:

- Test the driver on different processor platforms. If a driver will work on an IA-32 processor-based system but not an Itanium-based system, then the driver is a native EFI driver for IA-32 only. If a driver will work on an Itanium-based system but not an IA-32 processor-based system, then the driver is a native EFI driver for the Itanium processor only. If the driver works on both architectures, then it is not a native EFI driver.
- If you used the EFI Sample Implementation, check where the driver is saved in the directory structure. In the EFI Sample Implementation, native EFI drivers are kept in separate directories; non-native drivers are stored in the **`\%EFI_SOURCE%\EDK\Drivers\Ebc`** directory.

How to Identify an EFI 1.10 EFI Byte Code (EBC) Driver

An *EFI Byte Code (EBC) driver* is a driver that was compiled with the Intel® C Compiler for EFI Byte Code, and the same **.efi** executable will run on multiple platforms (both Itanium®-based systems and IA-32 processor-based systems).

To identify a driver as being an EBC driver, use the Shell **dmem** command and enter the starting address for the driver handle image, as shown in green below. An EBC driver will show **0E BC** at offset 70h, as highlighted in green in the **3FF7D878** row:

```
fs0:\>dmem 37ff7d808

  Memory Address 000000003FF7D808 200 Bytes
  3FF7D808: 49 42 49 20 53 59 53 54-02 00 01 00 78 00 00 00  *IBI SYST....x...*
  3FF7D818: 5C 3E 6A FE 00 00 00 00-88 2E 1B 3F 00 00 00 00  *\>j........?....*
  3FF7D828: 26 00 0C 00 00 00 00 00-88 D3 1A 3F 00 00 00 00  *&..........?....*
  3FF7D838: A8 CE 1A 3F 00 00 00 00-88 F2 1A 3F 00 00 00 00  *...?.......?....*
  3FF7D848: 28 EE 1A 3F 00 00 00 00-08 DD 1A 3F 00 00 00 00  *(..?.......?....*
  3FF7D858: A8 EB 1A 3F 00 00 00 00-18 C3 3F 3F 00 00 00 00  *...?..........*
  3FF7D868: 00 4B 3F 3F 00 00 00 00-06 00 00 00 00 00 00 00  *.K............*
  3FF7D878: 0E BC F7 3F 00 00 00 00-70 74 61 6C 88 00 00 00
*...?....ptal....*
  3FF7D888: 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  *................*
  3FF7D898: 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  *................*
  3FF7D8A8: 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  *................*
  3FF7D8B8: 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  *................*
  3FF7D8C8: 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  *................*
  3FF7D8D8: 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  *................*
  3FF7D8E8: 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  *................*
```

How to Identify the EFI 1.10 Driver Version Number and Component Name Protocol

The Shell **drivers** command is useful for determining if a driver is an EFI 1.10 driver or an EFI 1.02 driver. If it is an EFI 1.10 driver, then the version number that is listed in the VERSION field in the Shell **drivers** command will be 00000010. For an earlier version of EFI, the number will be something different.

In the following example, type the first line in green at the Shell prompt to view the driver information for the system. The version number for the PCI bus driver is 00000010, which indicates it is an EFI 1.10 driver.

```
Shell>drivers

            T   D
D           Y C I
R           P F A
V  VERSION  E G G #D #C DRIVER NAME                          IMAME NAME
== ======== = = = == == ================================
==================
0D 00000010 B - -  1 11 PCI Bus Driver                       PciBus
0E 00000010 D - -  1  - PC-AT ISA Device Enumeration Driver PcatIsaAcpi
0F 00000010 B - -  1  6 ISA Bus Driver                       IsaBus
10 00000010 B - -  2  2 ISA Serial Driver                    IsaSerial
11 00000010 D - -  1  - PCI VGA Mini Port Driver             PciVgaMiniPort
```

How to Identify Device Handles

Use the Shell **dh** command to list all the handles that are currently in the system, as shown in green below:

```
fs0:\>dh
Handle dump
1: Varstore
2: Image(EFI Core)
3: DevIo DevPath ()
4: Varstore
5: UnicodeCollation
6: Image(NVRAM API Driver 1.00)
7:
8: Image(BiosDisk)
9: DevPath (Acpi(PNP0A03,0)/Pci(1F|1))
A: Fs DiskIo BlkIo DevPath (..Pci(1F|1)/Ata(Primary,Slave))
B: DiskIo BlkIo DevPath (..ci(1F|1)/Ata(Primary,Master))
C: Image(Ebc)
D: Ebc
E: DebugSupport
F:
10: Image(Decompress)
11: Decompress
12: Image(PcatPciRootBridge)
13: PciRootBridgeIo DevPath (Acpi(PNP0A03,0))
14: PciRootBridgeIo DevPath (Acpi(PNP0A03,1))
15: PciRootBridgeIo DevPath (Acpi(PNP0A03,2))
16: PciRootBridgeIo DevPath (Acpi(PNP0A03,3))
17: PciRootBridgeIo DevPath (Acpi(PNP0A03,4))
18: Image(PciBus) DriverBinding ComponentName
19: Image(PcatIsaAcpi) DriverBinding ComponentName
```

```
1A: Image(IsaBus) DriverBinding ComponentName
1B: Image(IsaSerial) DriverBinding ComponentName
1C: Image(BiosVgaMiniPort) DriverBinding ComponentName
1D: Image(VgaClass) DriverBinding ComponentName
1E: Image(GraphicsConsole) DriverBinding ComponentName
1F: Image(Terminal) DriverBinding ComponentName
20: Image(ConPlatform) DriverBinding ComponentName
21: DriverBinding ComponentName
22:
23:
24:
25: Image(ConSplitter) DriverBinding ComponentName
26: DriverBinding ComponentName
27: DriverBinding ComponentName
28: DriverBinding ComponentName
29: Txtout PrimaryStdErr
2A: Txtin SimplePointer PrimaryConIn
2B: Txtout PrimaryConOut UgaDraw
2C: Image(UsbUhci) DriverBinding ComponentName
2D: Image(UsbBus) DriverBinding ComponentName
2E: Image(UsbCbi1) DriverBinding
2F: Image(UsbBot) DriverBinding ComponentName
30: Image(UsbCbi0) DriverBinding ComponentName
31: Image(UsbMassStorage) DriverBinding ComponentName
32: Image(UsbKeyboard) DriverBinding ComponentName
33: Image(PciRom Seg=00000000 Bus=01 Dev=01 Func=00 Image=0001)
DriverBinding C
omponentName
34: PciIo UsbHostController DevPath (Acpi(PNP0A03,0)/Pci(1D|0))
35: PciIo UsbHostController DevPath (Acpi(PNP0A03,0)/Pci(1D|1))
36: PciIo DevPath (Acpi(PNP0A03,0)/Pci(1D|7))
37: PciIo DevPath (Acpi(PNP0A03,0)/Pci(1E|0))
38: PciIo DevPath (..NP0A03,0)/Pci(1E|0)/Pci(0|0))
39: Txtout PciIo ConOut StdErr BusSpecificDriverOverride UgaDraw
DevPath (..NP0
A03,0)/Pci(1E|0)/Pci(1|0))
3A: PciIo IsaAcpi DevPath (Acpi(PNP0A03,0)/Pci(1F|0))
3B: Txtin ConIn UsbIo DevPath (..P0A03,0)/Pci(1D|0)/Usb(1, 0))
3C: IsaIo DevPath (..0)/Pci(1F|0)/Acpi(PNP0501,0))
3D: IsaIo DevPath (..0)/Pci(1F|0)/Acpi(PNP0501,1))
3E: IsaIo DevPath (..0)/Pci(1F|0)/Acpi(PNP0303,0))
3F: SerialIo DevPath (..(PNP0501,0)/Uart(115200 N81))
40: SerialIo DevPath (..(PNP0501,1)/Uart(115200 N81))
41: PciIo DevPath (Acpi(PNP0A03,1)/Pci(1C|0))
42: PciIo DevPath (Acpi(PNP0A03,1)/Pci(1D|0))
43: PciIo DevPath (..P0A03,1)/Pci(1D|0)/Pci(1F|0))
44: PciIo DevPath (Acpi(PNP0A03,1)/Pci(1E|0))
45: PciIo DevPath (Acpi(PNP0A03,1)/Pci(1F|0))
46: PciIo ScsiPassThru DevPath (..NP0A03,1)/Pci(1F|0)/Pci(2|0))
47: PciIo ScsiPassThru DevPath (..NP0A03,1)/Pci(1F|0)/Pci(2|1))
48: PciIo DevPath (..P0A03,1)/Pci(1F|0)/Pci(1F|0))
49: PciIo DevPath (Acpi(PNP0A03,2)/Pci(1C|0))
4A: PciIo DevPath (Acpi(PNP0A03,2)/Pci(1D|0))
```

```
4B: PciIo DevPath (..P0A03,2)/Pci(1D|0)/Pci(1F|0))
4C: PciIo DevPath (Acpi(PNP0A03,2)/Pci(1E|0))
4D: PciIo DevPath (Acpi(PNP0A03,2)/Pci(1F|0))
4E: PciIo DevPath (..P0A03,2)/Pci(1F|0)/Pci(1F|0))
4F: PciIo DevPath (Acpi(PNP0A03,3)/Pci(1C|0))
50: PciIo DevPath (Acpi(PNP0A03,3)/Pci(1D|0))
51: PciIo DevPath (..P0A03,3)/Pci(1D|0)/Pci(1F|0))
52: PciIo DevPath (Acpi(PNP0A03,3)/Pci(1E|0))
53: PciIo DevPath (Acpi(PNP0A03,3)/Pci(1F|0))
54: PciIo DevPath (..P0A03,3)/Pci(1F|0)/Pci(1F|0))
55: PciIo DevPath (Acpi(PNP0A03,4)/Pci(18|0))
56: PciIo DevPath (Acpi(PNP0A03,4)/Pci(18|1))
57: PciIo DevPath (Acpi(PNP0A03,4)/Pci(18|2))
58: PciIo DevPath (Acpi(PNP0A03,4)/Pci(18|3))
59: PciIo DevPath (Acpi(PNP0A03,4)/Pci(1C|0))
5A: PciIo DevPath (Acpi(PNP0A03,4)/Pci(1C|1))
5B: PciIo DevPath (Acpi(PNP0A03,4)/Pci(1C|2))
5C: PciIo DevPath (Acpi(PNP0A03,4)/Pci(1C|3))
5D: PciIo DevPath (Acpi(PNP0A03,4)/Pci(1C|4))
5E: PciIo DevPath (Acpi(PNP0A03,4)/Pci(1C|5))
5F: PciIo DevPath (Acpi(PNP0A03,4)/Pci(1C|6))
60: Txtin Txtout ConIn ConOut StdErr DevPath (..t(115200
N81)/VenMsg(PcAnsi))
61: Image(DiskIo) DriverBinding ComponentName
62: Image(Partition) DriverBinding ComponentName
63: Image(Fat) DriverBinding ComponentName
64: Image(Undi) DriverBinding ComponentName
65: Image(GigUndi) DriverBinding ComponentName
66: Image(Snp3264) DriverBinding ComponentName
67: Image(PxeBc) DriverBinding ComponentName
68: Image(PxeDhcp4) DriverBinding ComponentName
69: Image(UsbMouse) DriverBinding ComponentName
6A: Image(EFI-IFlash Driver .20)
6B:
6C: Image(EMP64 Driver)
6D:
6E: Image(Platform VPD Driver 1.00)
6F:
70: Image(FitFpswa)
71: Load DevPath (..E585-11D3-BC22-0080C73C8881))
72: Image(VenHw(6D9FEEB1-E585-11D3-BC22-0080C73C8881))
73: Image(FitSCSI)
74: Load DevPath (..E585-22D3-BC22-0080C73C8881))
75: Image(VenHw(6D9FEEB1-E585-22D3-BC22-0080C73C8881))
     DriverBinding ComponentName Configuration
76:
77: Image(BmcWDTDrv)
78:
79: Load DevPath (..71E5-4DF0-A909-F0D2992B5AA9))
7A: LegacyBoot
7B: DevPath (..6F1A-11D4-87AB-00062945C3B9))
7C: Load Pxebc Net Nii DevPath (..)/Pci(0|0)/Mac(0002B300076A))
7D: DiskIo BlkIo DevPath (..|0)/Pci(2|0)/Scsi(Pun0,Lun0))
```

```
7E: DevPath (..|0)/Pci(2|0)/Scsi(Pun6,Lun0))
7F: Image(FitBtmgr)
80: Image(shellenv)
81: Image(attrib)
82: Image(ls)
83: Image(mkdir)
84: Image(mode)
85: Image(cp)
86: Image(mv)
87: Image(comp)
88: Image(rm)
89: Image(memmap)
8A: Image(type)
8B: Image(dmpstore)
8C: Image(load)
8D: Image(loadbmp)
8E: Image(ver)
8F: Image(err)
90: Image(time)
91: Image(date)
92: Image(stall)
93: Image(reset)
94: Image(vol)
95: Image(cls)
96: Image(edit)
97: Image(hexedit)
98: Image(dblk)
99: Image(mm)
9A: Image(dmem)
9B: Image(pci)
9C: Image(bcfg)
9D: Image(FitBtmnt)
9E: Image(FitBtmgr)
9F: Image(FitBtmnt)
A0: Image(FitBtmgr)
A1: Image(FitBtmnt)
A2: Image(FitBtmgr)
A3: Image(VenHw(3C510960-893A-4B46-A2DA-D1BC18AF7629))
A4: Image(nshell) ShellInt
A5: Fs DiskIo BlkIo ESP DevPath (..F474-01C2-507B-9E5F8078F531))
A6: DiskIo BlkIo DevPath (..F474-01C2-F1B3-12714F758821))
A7: DiskIo BlkIo DevPath (..F474-01C2-D931-F8428177D974))
```

Any handle that has an **Image** after it is an EFI boot service or driver that is loaded into memory. The rest of the handles have a list of EFI protocols showing on them (e.g., **Diskio**, **Blkio**, and **Devpath**). Use the Shell **guid** command to list all the known EFI standard protocols and the shorthand notation used by the **dh** command; see the *EFI 1.1 Shell Commands Specification* for the command syntax and an example. If a handle does not have any text by it, it means that the GUID is not an EFI 1.10 standard, published GUID and is unique to the system.

Not all the information on a given handle will fit on one line.

Type **dh <handle#>** to see all the detailed information for a given handle. For example, type the following in green at the Shell prompt to see the detailed information for handle A6:

```
fs0:\>dh a6
 Handle A6 (3D091188)
    DiskIo (3D091390)
    BlkIo (3D0E2320) Fixed MId:0 bsize 200, lblock 3EC0F (131,604,480),
              partition rw !cached
    Dpath (3D091208)
       ACPI Device Path for ACPI
        HID A0341D0, UID 1
       Hardware Device Path for PCI
        Function (0) Device (1F)
       Hardware Device Path for PCI
        Function (0) Device (2)
       Messaging Device Path for SCSI
        SCSI (PUN 0, LUN 0)
       Media Device Path for Hard Drive
        Partition (2) Start (0000000000056496) Size (000000000003EC10)
       AsStr: 'Acpi(PNP0A03,1)/Pci(1F|0)/Pci(2|0)/Scsi(Pun0,Lun0)/HD
              (Part2,Sig898 D07A0-F474-01C2-F1B3-12714F758821)'
```

This command will list each protocol that is loaded on the handle and any arguments that are relevant for the particular instance of the protocol on that handle. This instance may mean the memory location where the image is at or the specific memory location of the protocol.

Use the **dh -d** command to display all the details if the handle is an EFI 1.10 driver; it will say "driver binding" on that handle if it is an EFI 1.10 driver).

How to Identify EFI Protocols and EFI File Systems on a Handle

This topic discusses how to identify the following on a handle:

- EFI protocols
- EFI file systems

Protocols

Use the Shell **dh <handle#>** command to see all the detailed information for a given handle. For example, type the following in green at the Shell prompt:

```
fs0:\>dh a6
 Handle A6 (3D091188)
    DiskIo(3D091390)
    BlkIo (3D0E2320) Fixed MId:0 bsize 200, lblock 3EC0F (131,604,480),
              partition rw !cached
    Dpath (3D091208)
       ACPI Device Path for ACPI
        HID A0341D0, UID 1
       Hardware Device Path for PCI
        Function (0) Device (1F)
       Hardware Device Path for PCI
        Function (0) Device (2)
       Messaging Device Path for SCSI
        SCSI (PUN 0, LUN 0)
       Media Device Path for Hard Drive
        Partition (2) Start (0000000000056496) Size (000000000003EC10)
       AsStr: 'Acpi(PNP0A03,1)/Pci(1F|0)/Pci(2|0)/Scsi(Pun0,Lun0)/HD
              (Part2,Sig898 D07A0-F474-01C2-F1B3-12714F758821)'
```

In this case, driver A6 has three protocols loaded onto it:

```
A6: DiskIo BlkIo DevPath (..F474-01C2-F1B3-12714F758821))
DiskIo
BlkIo
DevPath
```

Using the **dh a6** command, as shown above, will show the specific parameters that are used for each of the three protocols. The Device Path Protocol details will tell you specifically which device the other drivers on the handle are managing. In this case, it is the second partition of the LSI Logic* U320 SCSI card SCSI drive.

File Systems

Use the Shell **map** command to identify all the block I/O devices on your system, by typing the following in green at the Shell prompt:

```
Shell>map -r

Device mapping table

fs0 : Acpi(PNP0A03,0)/Pci(1F|1)/Ata(Primary,Slave)
fs1 : Acpi(PNP0A03,1)/Pci(1F|0)/Pci(2|0)/Scsi(Pun0,Lun0)/HD(Part1,Sig
      8983DFE0-F474-01C2-507B-9E5F8078F531)
blk0 : Acpi(PNP0A03,0)/Pci(1F|1)/Ata(Primary,Slave)
blk1 : Acpi(PNP0A03,0)/Pci(1F|1)/Ata(Primary,Master)
blk2 : Acpi(PNP0A03,1)/Pci(1F|0)/Pci(2|0)/Scsi(Pun0,Lun0)
blk3 : Acpi(PNP0A03,1)/Pci(1F|0)/Pci(2|0)/Scsi(Pun0,Lun0)/HD(Part1,Sig
      8983DFE0-F474-01C2-507B-9E5F8078F531)
blk4 : Acpi(PNP0A03,1)/Pci(1F|0)/Pci(2|0)/Scsi(Pun0,Lun0)/HD(Part2,Sig
      898D07A0-F474-01C2-F1B3-12714F758821)
blk5 : Acpi(PNP0A03,1)/Pci(1F|0)/Pci(2|0)/Scsi(Pun0,Lun0)/HD(Part3,Sig
      89919B80-F474-01C2-D931-F8428177D974)
```

In this case, there are six block I/O devices:

- Floppy on **blk0** or the ATAPI primary slave.
- CD-ROM on **blk1** or the ATAPI primary master (but no CD-ROM in the drive)
- SCSI controller on **blk2**.
- Three hard drive partitions on the SCSI controller on **blk3**, **blk4**, and **blk5**. Each one has a different GUID.
- FS0: is the floppy, which is **blk0**.
- FS1: is the FAT32 partition on **blk3**, because it has the same device path and GUID (**part1** and **Sig 8983DFE0-F474-01C2-507B-9E5F8078F531**) and **blk3**.

Use the **map -r** command to rescan all the **blkx** devices in the system to check for new FAT file systems. This command will change the FSx assignments depending on what removable devices are inserted in the system. There is no guarantee that FS0: will be the floppy after the next time **map -r** is called.

Use the **mount** command in combination with the **map -d** command to force the assignment of a **blkx** device to a specific known mapping. That is, if you want partition 1 of the SCSI drive to be named HD1:, type the following in green at the Shell prompt:

```
Shell>map -d fs1

Shell>mount blk3 HD1
```

Now typing **map** would yield HD1: pointing to partition 1 on the SCSI drive.

9
USB

How to Use the USB Stack

Eight USB drivers are provided in the EFI 1.10 Sample Implementation. Not all the USB drivers may be loaded depending on your BIOS flash. Check to see which ones are loaded by using the **`drivers`** command.

The following table lists these eight USB drivers and what they are used for.

USB Driver Stack

USB Driver	Description
UsbUhci.efi	Must be started first. Controls the UHCI controller.
UsbBus.efi	Must be started after **`UsbUhci.efi`** is started.
UsbCbi0.efi	Uses **`UsbBus.efi`** and **`UsbUhci.efi`** drivers. 1.44 MB USB floppies use this driver.
UsbCbi1.efi	Uses **`UsbBus.efi`** and **`UsbUhci.efi`** drivers. Flash key devices use this driver.
UsbKeyboard.efi	For keyboards. Uses **`UsbBus.efi`** and **`UsbUhci.efi`** drivers.
UsbMouse.efi	For mice.
UsbMassStorage.efi	Uses **`UsbBus.efi`** and **`UsbUhci.efi`** drivers. Flash key devices use this driver.
UsbBot.efi	Flash key devices use this driver.

10
Network Interface

How to Use the LAN Stack and Boot to LAN PXE

Three drivers are required for the PXE LAN boot to function, which are listed in the table below.

LAN Driver Stack

Driver	Description
Undi.efi	For a specific LAN card. Intel supplies **`Undi.efi`** for all Intel® PRO/100 network adapters.
Snp3264.efi	Needed for PXE LAN boot (NII interface and PXE boot support).
PxeBc.efi	Needed for PXE LAN boot (NII interface and PXE boot support).
PxeDhcp4.efi	Optional. May be needed for networks with a DHCP server for the system to get an IP address.

If all three drivers (**`Undi`**, **`Snp3264`**, and **`PxeBc`**) are loaded for a given LAN card, you should be able to boot from the card. The boot option maintenance menu will show the MAC address of the card in the Add a Boot Option menu if the drivers have been started.

How to Use the Toolkit LAN Driver Stack

The EFI Application Toolkit provides a complete TCP/IPv4 network stack and configuration tools that take advantage of this protocol.

There are two ways to configure the network stack:

- Manual configuration
- Through the use of a DHCP server and client

The following sections outline the network configuration files that are used by the socket library and the two configuration procedures. Network stack configuration commands need to be executed after booting to the Shell. It is suggested that these commands be grouped together in an EFI batch script to make setting up the network a simple one-line command.

Manual Network Configuration

The first operation is to load the TCP/IP Protocol. This step is done with the Shell **load** command. The **load** command does not use the search path to locate protocols, so the path must be explicitly given if it is not in the current working directory.

Next, the network interfaces need to be configured. The TCP/IP stack contains a loopback interface **lo0**, which can be optionally configured along with the Ethernet interface **sni0** if a compatible UNDI Ethernet card is installed. Configuration is performed with the **ifconfig** command. The following is the simple form of the command:

```
fs0:/>ifconfig lo0 inet <ip address> up
```

where `<ip address>` is the address assigned to the system.

If the system is connected to a network that employs subnetting, a subnet mask would also need to be specified as follows:

```
fs0:/>ifconfig sni0 inet <ip address> netmask <netmask> up
```

where `<netmask>` is the network mask that is appropriate for your network.

Finally, if you wish to do networking across multiple networks or subnetworks, you must set a gateway address for the appropriate gateway(s) attached to your network. This is done with the route command as follows:

```
fs0:/>route add <destination> <gateway ip address>
```

where `<destination>` specifies the target network or host, and `<gateway ip address>` specifies the address for the gateway attached to your network responsible for routing data to the destination. Using default for `<destination>` will set a default route.

The following is a series of commands that might be found in a network configuration batch file. All IP addresses (except the loopback address) are fictional and would need to be changed to reflect your network configuration:

```
load fs0:\efi\tools\tcpipv4.efi
ifconfig lo0 inet 127.0.0.1 up
ifconfig sni0 inet 10.8.198.178 netmask 255.255.255.0 up
route add default 10.8.198.251
```

DHCP Client Network Configuration

If a DHCP server is accessible for the EFI system, the DHCP client protocol can be loaded to configure the network interface. The DHCP client protocol can be used only if it is run with the EFI Shell that is provided in the EFI Application Toolkit, release 0.80 or higher. Early BIOS releases for Itanium®–based systems containing 0.99 EFI implementations do not contain the required EFI Shell. The DHCP client constructs configuration scripts and runs them when it is first loaded. Temporary scripts are placed in the root directory of the file system and removed when configuration has completed. The temporary scripts will set some volatile Shell variables and invoke **`\etc\dhcp-script.nsh`** to perform network configuration based on the information returned by a DHCP server. There is little reason to modify this script.

The DHCP client uses the configuration file **`\etc\dhclient.conf`** to determine how it will interact with the DHCP server. This file may be edited to specify the host name of the EFI system. Some DHCP servers will use this information to register the host with the network DNS server.

The **`dhcp-script.nsh`** and **`dhclient.conf`** files can be found in the **`\protocols\dhclient\client\scripts`** source directory in the Toolkit.

A simple Shell batch script with the following lines in green is all that is required to configure the network stack when DHCP services are available:

```
# The following assumes the network stack and DHCP client
# protocols are in the current working directory.
load tcpipv4.efi
load dhclient.efi
```