Field Replaceable Unit (FRU) and Sensor Data Records (SDR) Load Utility Configuration File

Developer’s Guide

Order Number: 739110-001

Contents

1 Overview

Compatibility 5
System and Software Requirements 5
Reference Documents 5
About the FRUSDR Load Utility 6
- When to Run the FRUSDR Load Utility 7
- How to Run the FRUSDR Load Utility 7

2 Command Line Format

Parsing the Command Line 11
- Command Line Precedence 11

Displaying Usage Information 12
- `/?` or `/h` Command 12
- `/d fru` Command 12

Displaying a Given Area 12
Displaying FRU Area 13
Displaying SDR Area 14

3 Configuration File

About the Configuration File 15
Invoking the Configuration File 15
Configuration File Format 15
Configuration File Ordering 16
Probing the Product Configuration 17
- DEVICE_ID Option 17
- NVS_TYPE Option 17
- Example Probe Commands 18

Checking the FRU Data Integrity 20
Using COMPARE 20
Working with Strings 25
Processing the Configuration File 27
- Programming FRU Files 27
- FRU File Filtering 27
- Programming SDR Files 28
- SDR File Filtering 29
- SM BIOS (DMI) Programming and Mapping 31
- FRU Field Second ASCII String Formatting 33
- Input of Data 34
- Updating the FRU Non-Volatile Storage Area 35
- Updating the SDR Non-Volatile Storage Area 35
- Updating the SM BIOS (DMI) Non-Volatile Storage Area 35
- Sample Configuration File 36

Cleaning Up and Exiting 37

4 Solving Problems
Error Messages 39
Warning Messages 39

Appendix A
Glossary of Terms 47

Index 49

Figures

1. Example 1, Server System Field Replaceable Unit Locations 8
2. Example 2, Server System Field Replaceable Unit Locations 9

Tables

1. List of Commands 21
2. FRU Field Maximum Allowed Lengths 26
3. FRU/SM BIOS (DMI) Conversion Table 31
4. FRU Area String Specifications 32
5. FRU Field First String Specifications 32
6. FRU Field Second String Specification 33
7. TYPE Code Specification 33

1 Overview

This manual describes the Field Replacement Unit (FRU) and Sensor Data Record (SDR) Load Utility, a DOS-based program that updates or modifies the server management subsystem's Field Replacement Unit (FRU), Sensor Data Record (SDR) repository, and the Desktop Management Interface (DMI) non-volatile storage (NVS) components. The Desktop Management Interface is also known as the System Management BIOS (SM BIOS). The terms "DMI" and "SM BIOS" are used interchangeably throughout this manual.

Compatibility

The utility:

- Is Intelligent Platform Management (IPMI) 1.0 compatible
- Is compatible with ROM-DOS Version 6.22 and MS-DOS† Version 6.22
- Accepts `.cfg`, `.fru`, and `.sdr` files
- Supports Non-Volatile Storage Format Specification, Version 1.5

System and Software Requirements

- 512K of conventional memory
- Access to a read/write disk
- The following supporting files:
    - — a `.cfg` file describing the system configuration
    - — one or more `.fru` files describing the system's field replaceable units
    - — an `.sdr` file describing the sensors in the system

Reference Documents

OEM customers should contact their Intel field representative for access to the following reference documents. The *System Management BIOS Reference Specification* can be found on the Phoenix web site at: http://www.phoenix.com.

IPMI Platform Management FRU Information Storage Definition Version 0.9. Revision 0.17. November, 1997. © Intel Hewlett-Packard NEC Dell.

IPMI Platform Management FRU Information Storage Definition Version 1.0. Revision 1.0. September, 1998. © Intel Hewlett-Packard NEC Dell.

IPMI Platform Management Interface Specification, Version 0.9. Revision 0.17. March 1998. © Intel Hewlett-Packard NEC Dell.

IPMI Platform Management Interface Specification, Version 1.0. Revision 1.0. September, 1998. © Intel Hewlett-Packard NEC Dell.

Non-Volatile Storage Load File Format, Version 1.50. Revision 0.1. December, 1998. © Intel Corporation.

System Management BIOS Reference Specification, Version 2.1.

About the FRUSDR Load Utility

The executable for the utility is named `frusdr.exe`, and is designed to run in DOS non-protected mode. Because many of the changes made by the utility do not take effect until after the system has been cold/hard booted, turn the system off just prior to and after running the utility. Do not run the utility in a Window's DOS box, or in protected mode.

The utility is command line driven, and receives command line and prompted input from stdin, and directs its output to stdout.

The utility performs the following tasks:

- Discovers the product configuration based on instructions in a master configuration file.
- Verifies that FRU and SDR files meet basic format requirements.
- Displays the FRU, SDR, and DMI areas.
- Updates the non-volatile storage device associated with the Baseboard Management Controller (BMC) that holds the FRU and SDR area.
- Generically handles FRU devices that might not be associated with the BMC.
- Updates the SMBIOS area located in the BIOS non-volatile storage device.

⚠ **CAUTION**

The utility is designed so the FRU and SDR areas can be programmed repeatedly. However, since each FRU area is programmed immediately after the utility reads in that area, there is a possibility that a FRU file can be partially programmed due to invalid information discovered during the FRU file programming process. Therefore, you must be careful not to program invalid information into the FRU and SDR areas.

The utility does not allow a Ctrl-Break and Ctrl-C during critical sections of the application. You can stop execution by using Ctrl-Alt-Del, but doing so can cause the server to malfunction because incomplete FRU and SDR information may be programmed.

When to Run the FRUSDR Load Utility

You should run the FRUSDR load utility each time you upgrade or replace the hardware in your server, excluding ISA/PCI add-in boards, hard drives, and RAM. For example, if you replace an array of fans, you need to run the utility; it programs the sensors that need to be monitored for server management.

How to Run the FRUSDR Load Utility

You can run the utility directly from the configuration software CD or from diskettes you create from the CD.

If you choose to run the FRUSDR Load Utility from a diskette, you must create the utility disk from the CD and follow the instructions in the README.TXT file included on the CD.

NOTE

If your diskette drive is disabled, or improperly configured, you must use BIOS Setup to enable it. If necessary, you can disable the drive after you are finished with the FRUSDR utility.

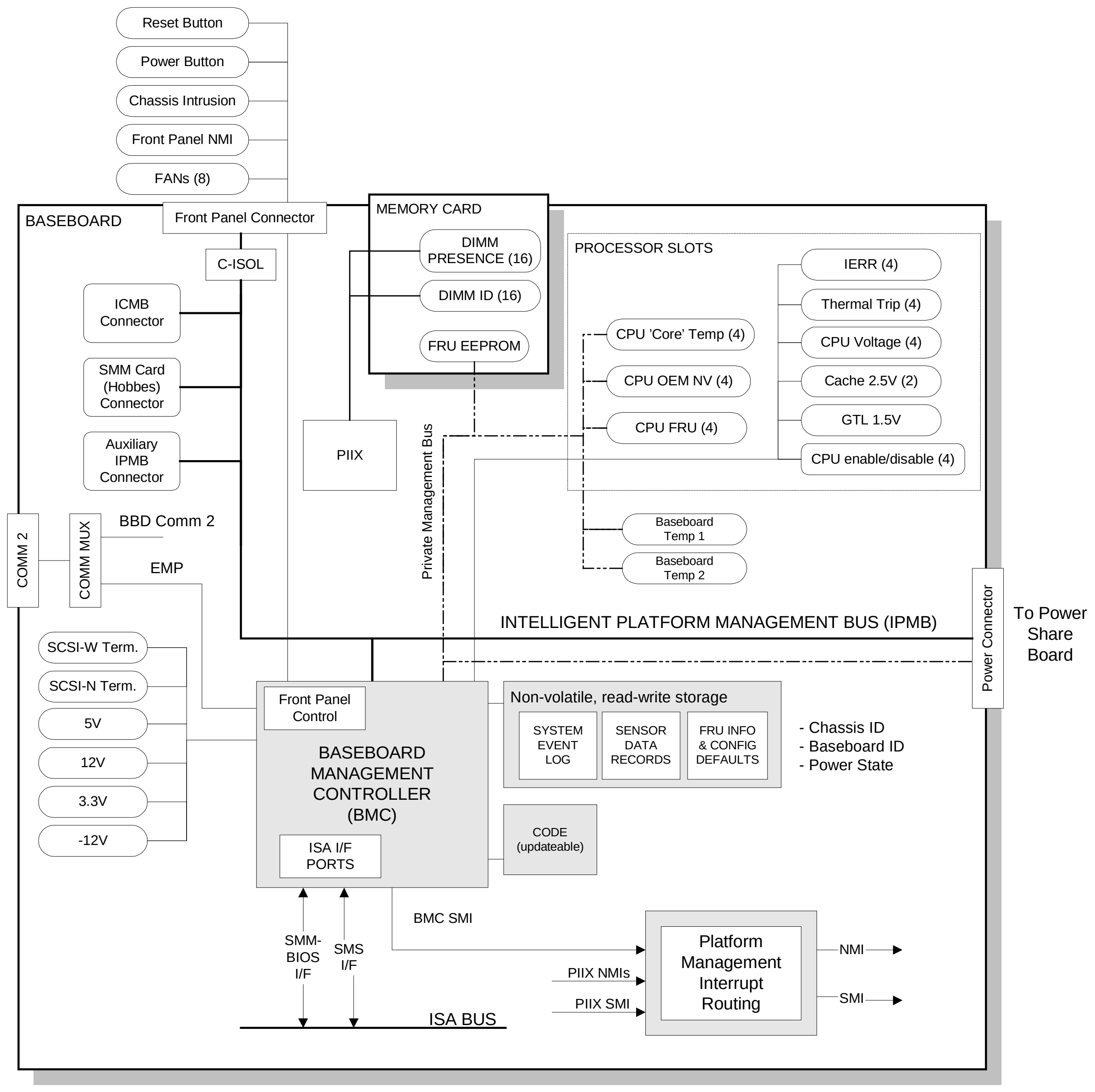

Figure 1. Example 1, Server System Field Replaceable Unit Locations

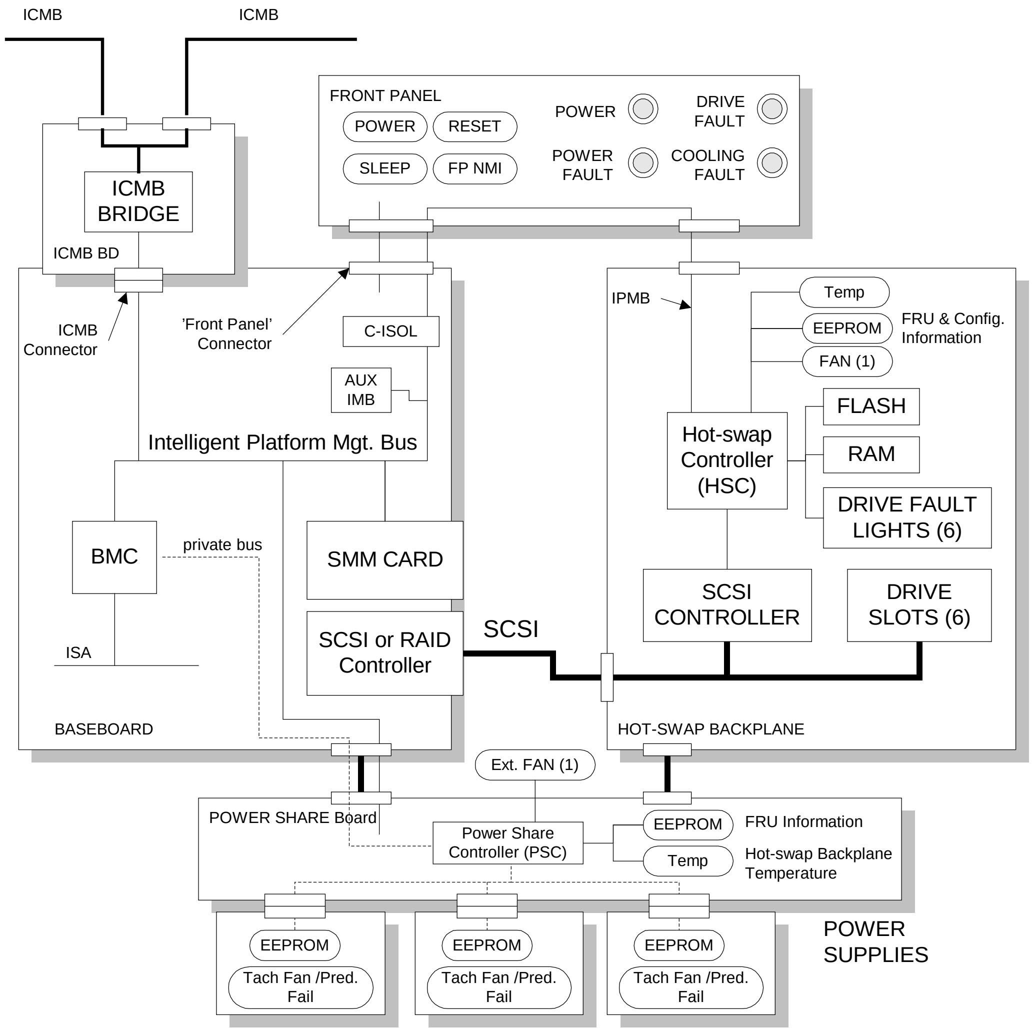

Figure 2. Example 2, Server System Field Replaceable Unit Locations

2 Command Line Format

Parsing the Command Line

The utility parses the command line, retrieving the specified source files and setting internal flags to control operation. Only one command line functions at a time. A command line function can consist of two parameters, for example: `/cfg` filename.cfg. Invalid parameters cause an error message and the utility exits. You can use either a slash (/) or a minus sign (-) to specify command line options. The `/p` flag can be used in conjunction with any of the other options.

The basic command line format is:

```
frusdr [/?] [/h] [/d {fru, sdr, dmi}] [/cfg filename.cfg]
```

where:

`Frusdr`	Is the name of the utility
`/? or /h`	Displays usage information
`/d {fru, sdr, dmi}`	Displays requested area only of the specified device
`/cfg filename.cfg`	Uses custom configuration file
`/p`	Pause between blocks of data

Command Line Precedence

Command line precedence means that the first command found is executed, followed by the next command, and so on. There are two categories of commands: Flag Commands and Action Commands. The only Flag Command is the Pause (`/p`). The Action Commands are: `/?`, `/d`, and `/cfg`. The Pause command only affects the execution path for the help (`/?` or `/h`) and display (`/d`) FRU, SDR, and DMI, commands. If an Action Command does not use the Pause flag (if there isn't enough data displayed to warrant a pause for example) no error results. If more than one Action Command is listed on the command line, an error message is displayed and the utility exits.

Displaying Usage Information

/? or /h Command

When the utility is run with the help command (`/?` or `/h`), the following message is displayed:

```
FRU & SDR Load Utility Version 4.0
 Usage:   FRUSDR
          /? or /h            Displays usage information
          /d {dmi,fru,sdr}    Only displays requested area
          /cfg filename.cfg   Uses custom CFG file
          /p                  Pause between blocks of data

Copyright (c) 1997-1999, Intel Corporation, All Rights Reserved
```

/d fru Command

The `/d fru` command can be followed with up to 16 device addresses so you can view up to 16 different FRU areas. If no device addresses are specified, the default displays the Baseboard Management Controller FRU. The arguments following the `/d fru` command are in the same order and value as the `NVS_TYPE`, `NVS_LUN`, `DEV_BUS`, and `DEV_ADDRESS` which are found in the header in each FRU file. The LUN address is optional. If the LUN address is used, it must start with an `L`.

Example:

```
FRUSDR /D FRU IMBDEVICE L00 00 C0 C2
```

where:

```
/d fru (device) [lun] (bus) (addr) (addr2) (etc)
```

Displaying a Given Area

When the utility is run with the, /d FRU, /d SDR, or /d DMI command line flag, the specified area is displayed. Each area represents one sensor for each instrumented device in the server. If the given display function fails because of an inability to parse the data present, or as a result of a hardware failure, the utility displays an error message and exits.

NOTE

Not all server products support the displaying or programming of the SM BIOS/DMI area. In those cases an error message is displayed.

Displaying FRU Area

The FRU area is displayed in ASCII format when the field is ASCII, or as a number when the field is a number. Each FRU area displayed is headed with the FRU area designated name. Each field has a field name header followed by the field in ASCII or as a number. The board, chassis, and product FRU areas end with an END OF FIELDS CODE that indicates there are no more data in the area. The internal use area is displayed in hex format, 16 bytes per line.

To display the FRU area, type **frusdr /d fru** and press <Enter>.

Example:

```
FRU IMBDEVICE on bus FFh, IMB address 20h, LUN 00
Display Header Area
Common Header Area (Version 1, Length 8)
 Internal Area Offset     = 01h
 Chassis Area Offset      = 0Ah
 Board Area Offset        = 0Eh
 Product Area Offset      = 16h
 MultiRecord Area Offsert = 00h
 PAD                      = 00h
 CHECKSUM                 = D0h

Displaying Internal Use Area
Internal Information Area (Version 0, Length 72)

 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 00 00 00 00 00 00 00 00

Displaying Chassis Area
Chassis Information Area (Version 1, Length 32)
 Chassis Type          = 11h
 Part Number       (ASCII) = SBALMADSTD02PP
 Serial Number     (ASCII) = A05884265
END OF FIELDS CODE

Displaying Board Area
Board Information Area (Version 1, Length 64)
 Unicode Country Base    = 00h
 Manufacturing Time (mins) = 733803
 Manufacturer Name (ASCII) = Intel
 Product Name      (ASCII) = B440FX DP
 Serial Number     (ASCII) = N03121530
 Part Number       (ASCII) = 664653-001
END OF FIELDS CODE
```

```
Displaying Product Area
Product Information Area (Version 1, Length 80)
 Unicode Country Base      = 00h
 Manufacturer Name (ASCII) = Intel
 Product Name      (ASCII) = B440FX DPServer
 Part Number       (ASCII) = SBALMADSTD02PP
 Product Version   (ASCII) =
 Serial Number     (ASCII) = A05884265
 Asset Tag         (ASCII) =
END OF FIELDS CODE
```

Displaying SDR Area

The SDR non-volatile storage area is displayed in the following hex format. The data are separated by a sensor record number X header, where X is the number of that sensor record in the SDR area. The next line after the header is the sensor record data in hex format delineated by spaces. Each line holds up to 16 bytes. The data on each line are followed by the same data in ASCII format; nonprintable characters (ch < 32 || ch > 126) are substituted by a period (.). To display the SDR area, type **frusdr /d sdr** and press <Enter>.

Example:

```
FRU & SDR Load Utility Version 4.0

Reading SDR Repository.........
Displaying SDR area

Reading SDR Record #1
 0E 00 10 01 37 20 00 0F 05 00 10 F1 F8 02 01 85      ....7 ..........
 02 00 00 00 04 00 00 C4 02 00 08 30 C2 07 91 95      ...........0....
 8E FF 00 1B 1B 00 99 95 00 8A 8E 02 02 00 01 CC      ................
 53 43 53 49 2D 42 2D 54 65 72 6D 33                  SCSI-B-Term3
Reading SDR Record #2
 0E 40 10 01 30 20 00 13 05 00 10 F1 F8 04 01 05      .@..0 ..........
 00 00 00 20 29 00 00 1E 02 00 00 00 00 00 00 00      ...  )...........
 00 FF 00 03 03 00 00 00 00 42 49 02 02 00 01 C5      .........BI.....
 46 41 4E 2D 32                                       FAN-2
```

3 Configuration File

About the Configuration File

The configuration file is an editable ASCII text file used by the load utility to execute the commands in the file. This file can be used to perform the following tasks:

- Probe the product to identify all the boards, subassemblies, and components in the product
- Load the proper SDRs into the non-volatile storage of the BMC and other generic FRU devices
- Load multiple FRU and SDR files to be programmed
- Define FRU and SDR areas to be programmed
- Request information or ask you to choose which areas to program

The configuration file has the following restrictions:

- The maximum line length is 255 characters.
- The maximum file size is 64K bytes.
- The configuration file does not support first time programming of FRU areas. This area should initially be programmed by the manufacturer of the device.
- The configuration file allows you to override values contained in an association FRU file, but it does not allow you to add areas to a FRU. The one exception is the Manufacturing Time. If it is zero, it is programmed with the current machine time.

Invoking the Configuration File

The utility is invoked with the command line parameter `/cfg filename.cfg`. The filename can be any DOS-accepted, eight-character filename string, with a three character extension (.cfg). If the argument `/cfg` is used without a filename, the default file MASTER.CFG is used, if it exists. If it does not exist, an error message is displayed.

Configuration File Format

The configuration file consists of lists of commands, data, and comment fields. There are two data types: strings and numbers. The numeric data values are represented in hexadecimal. The strings and numeric value are formed into fields that make up commands, requests, and prompts, which are used to obtain the information necessary to establish a system's configuration. See Working with Strings on page 25 and the sample configuration file on page 36 for detailed information.

Configuration File Ordering

1. The CONFIGURATION string name, if present, is recommended to be the first non-comment field in the file and usually describes the file's purpose.
2. The SET, PROBE, MENU, and PROMPT group of commands should be towards the beginning of the configuration file. The reason for this is because only a single pass is done on the file by the FRUSDR utility, therefore an IFSET statement will not be triggered on a tag set further on in the file.
3. Order the following fields: FRUNAME, FRUADDRESS, FRUAREA, FRUFIELD. Up to 16 FRU addresses can follow the FRU name. Multiple FRU areas can exist, and each FRU area can be followed by multiple FRU fields. If the FRUNAME is not followed by any FRU areas, no FRU areas are programmed.
4. Order the menu by starting with MENUTITLE, then MENU (one or more), and ending with MENUPROMPT.
5. The master SDR file is programmed before the FRU files are programmed, even if it's listed last in the file. The SDR file must be programmed before other FRU files are programmed, because the firmware uses the FRU locator Sensor Data Records for knowing the location of other FRU devices.

The Configuration File Commands are implemented as follows: If there is a primary command needed before a secondary command, and the secondary command comes first, an error message is generated. If the secondary command never comes after the primary command, then the primary command is ignored and no error message is generated.

Examples of Primary & Secondary commands:

Primary Commands	**Secondary Commands**
PROBE	FOUND
MENUTITLE	MENUPROMPT
PROMPT	YES & NO

Probing the Product Configuration

The PROBE command can be used to help determine which information is to be programmed and which is not. The command also works with Intelligent Platform Management Bus (IPMB) and non-IPMB FRU devices. There are several possible formats for the PROBE command when using the DEVICE_ID or the NVS_TYPE options. See the sample probe configuration file on page 18.

Minimal parameter checking is done on the PROBE command address arguments, because the utility has no knowledge of which addresses are valid or invalid. If the arguments are determined to be invalid, an error message is displayed. If no response is received, a timeout occurs and a FALSE is returned to the FOUND command.

There are several possible formats for the PROBE command. The first ASCII string denotes either the DEVICE_ID or NVS_TYPE. The second byte is dependent on what the first ASCII string is. The parameters used here are the same as in the header of the FRU file header definition for both the FRU and SDR. The address formats are also used with the FRUADDRESS command.

DEVICE_ID Option

When the DEVICE_ID option is used, the utility performs a firmware GET-DEVICE-ID command to the specified device. If the device is found, a TRUE is returned to be used by the FOUND command. The FOUND command follows the PROBE command, and determines which tag is set. These tags can be used later by the IFSET command to change the execution path or to set SDR tags.

Example:

```
PROBE    “DEVICE_ID”  01 D0
FOUND    “PSC”        “
```

NVS_TYPE Option

When the NVS_TYPE option is used, the utility probes for the specified non-volatile storage device. There are three possible formats for the PROBE command when an NVS_TYPE is used. The reason for the varied address formats is because of the different address requirements needed to read different FRU devices.

The possible NVS_TYPEs are:

```
IMBDEVICE    LUN    DEV_BUS    DEV_ADDRESS
DS1624S             DEV_BUS    DEV_ADDRESS
AT24C02             DEV_BUS    DEV_ADDRESS
PROCESSOR    SLOT#  DEV_BUS    DEV_ADDRESS
```

⟹ **NOTES**

The LUN address is only provided if the NVS_TYPE equals IMBDEVICE. Even then the LUN parameter is optional. If it is left out or not needed, it defaults to zero. Only LUN address 0, 1, 2, and 3 are allowed; any others cause an error message.

When using the NVS_TYPE PROCESSOR, the SLOT# starts at one. Therefore, in a four processor system, you would probe for slots 1, 2, 3, and 4.

There is a difference between an **invalid address** and an **unimplemented address**. An invalid address is an address that is not valid on any server platform. The utility should catch these addresses and flag with an error message. An unimplemented address is a valid address, but one that is not supported on the specific server platform the utility is running on. In this case the utility will not flag the address as invalid, instead it will try to program the area and probably display several communication errors.

In this case you should double-check the address for correctness. A good place to look is in the FRU file for that area. For example the complete address IMBDEVICE, bus=0xFF can only have a valid device address of 0x20 (the utility knows that the only device on bus=0xFF is the BMC at address 0x20). On the other hand IMBDEVICE, bus=0x00 can have any value as a device address. Therefore in this case, if you select the wrong device address, you will receive communication errors.

Example Probe Commands

```
PROBE     IMBDEVICE FF 20        //Probe the device encoded the same as the
                                // header, with the NVS_TYPE, then DEV_BUS,
                                // finally DEV_ADDRESS.

FOUND     "BMC"     ""           //Sets 1st string if probe found device.  Sets
                                 // 2nd if not found.  Either string could be
                                 // empty.
PROBE     “AT24C02”   01   A2
FOUND     "F16"     ""

PROBE     "DEVICE_ID" 01   D0
FOUND     "PSC"     ""

PROBE     “DS1624S”   01   9A
FOUND     "MEM"     ""

PROBE     “IMBDEVICE”  00  22
FOUND     "FP"      ""

PROBE     “IMBDEVICE”  02  00  22
FOUND     "PWRDST"   ""

PROBE     “PROCESSOR” 01
FOUND     “PROC2”       “”
```

```
IFSET    "BMC"
DISPLAY  "BMC found"
ELSE
DISPLAY  "BMC not found"
ENDIF

IFSET    "F16"
DISPLAY  "The Interconnect Backplane was found"
ELSE
DISPLAY  "The Interconnect Backplane was not found"
ENDIF

IFSET    "PSC"
DISPLAY  "The power share board was found"
ELSE
DISPLAY  "The power share board was not found”
ENDIF

IFSET    "MEM"
DISPLAY  "The memory module was found"
ELSE
DISPLAY  "The memory module was not found"
ENDIF

IFSET    "FP"
DISPLAY  "Front Panel was found"
ELSE
DISPLAY  "Front Panel was not found"
ENDIF

IFSET    "PWRDST"
DISPLAY  "Power distribution board was found"
ELSE
DISPLAY  "Power distribution board was not found"
ENDIF

IFSET    "PROC2"
DISPLAY  "Processor 2 was found"
ELSE
DISPLAY  "Processor 2 was not found"
ENDIF
```

Checking the FRU Data Integrity

The utility requires the FRU Common Header offsets to be correct. The utility checks the Common Header Area in each NVS device against the FRU file, and runs a checksum on it. If the Common Header Area in the FRU file is correct and matches what is in the NVS device, the information is programmed. An incorrect Common Header means the FRU area is corrupted or has never been initialized.

Using COMPARE

The COMPARE argument is placed in the configuration file on the same line after the FRUNAME or SDRNAME. This argument allows you to validate the information you're programming against that which exists in the non-volatile storage device, without actually programming the information. You can use either upper or lower case when using the COMPARE argument.

A byte by byte comparison is done with what is to be programmed. In the case of an SDR file, no checksum is used. The first two bytes of each Sensor Data Record are ignored because when the SDR Repository was programmed, the first two SDR bytes were modified by the BMC and a pointer inserted.

Example:

```
FRUNAME or SDRNAME    “abcsdr.sdr”   COMPARE
```

NOTE

The internal use area is never compared. The utility considers all bytes of the internal use area to be dynamic and subject to change at will by the firmware. In the board area, neither the manufacturing date and time or the board area checksum are compared. The rest of the board area bytes are compared.

Table 1. List of Commands

Command Name	Description
//	`// Comment` Ignores text following two forward slash marks (//) on a command line
BOOT	Performs a cold boot before exiting the program
CLEAR	`CLEAR “TAG STRING”` Removes the designated TAG STRING from the Master Tag link list accumulated by the SET command and other tag setting commands.
CONFIGURATION	`CONFIGURATION    "ASCII STRING"` Descriptive name for the configuration file. Should indicate which product the configuration file is for. The string is displayed to the user by the FRUSDR Load Utility.
DISPLAY	`DISPLAY    “ASCII STRING”` Displays an ASCII STRING to the screen. This command can be used to request the user to enter data such as a serial number, for example. The ASCII STRING poses the request. This command is meant to be used in conjunction with a command that is able to access stdin such as after a string of MENU commands. The length limit is 80 ASCII characters.
ELSE	Follows the IFSET command and is used when an IFSET condition is not TRUE. One ELSE statement is allowed for each IFSET statement, and the ELSE statement is always associated with the last IFSET statement unmatched by an ENDIF statement.
ENDIF	This command is associated with the IFSET command. It lets the FRUSDR Load Utility know where the end of the IFSET block is located. If there was no previous IFSET command, using the ENDIF flags an error.
FOUND	`FOUND    “TAG STRING”    “TAG STRING”` A FOUND command should immediately follow a preceding PROBE command. This command saves the first TAG STRING (just like the SET command), if the PROBE command returns a TRUE (indicating the device was found). If no device was found, the PROBE command returns a FALSE, and the FOUND command saves the second TAG STRING (just like the SET command). Either or both strings may be empty (“ ”). If a PROBE command was not executed, using the FOUND command will flag an error. Note: If there is no FOUND command following the PROBE command, the information returned from the PROBE command will never be used. No error message is generated under this circumstance.

continued

Table 1. List of Commands (continued)

Command Name	Description

FRUADDRESS	`FRUADDRESS    "ASCII STRING"    HEX BYTE    HEX BYTE    HEX BYTE.`
	The first ASCII string denotes the NVS_TYPE, the second byte is dependent on what the NVS_TYPE is. The parameters used here are the same as in the header of the FRU file header definition for both FRU and SDR files. There may only be one FRUADDRESS command containing NVS type, LUN address, and bus address per FRU file, and it must follow after the FRUNAME. Additional FRUADDRESS commands may follow, but they must only contain additional DEV_ADDRESS's. If multiple NVS type FRUADDRESS commands exist after a FRUNAME, only the information contained in the last one is used and no error message is displayed. Note 1: The LUN address is only provided if the NVS_TYPE equals IMBDEVICE. Note 2: If you want to program a FRU file into two or more FRU areas with different NVS_TYPEs, the series of FRU commands for that file need to be implemented for each different NVS area.
	`FRUADDRESS    HEX BYTE`
	The FRUADDRESS command designates an I2C address that overrides the _DEV_ADDRESS entries that may be present in the FRU file named by a preceding "FRUNAME" command. This allows you to force the FRUSDR utility to only access the FRUADDRESS. FRUADDRESS must follow the FRUNAME command. If no FRUADDRESS is specified, the FRUSDR utility attempts to program the FRU file to the DEV_ADDRESS found in the FRU file. Multiple DEV_ADDRESS's in FRU files are not supported. Instead only one address should exist in any FRU file. Use the Configuration file to program multiple FRUADDRESS's.

FRUAREA	`FRUAREA    "ASCII STRING"    <OPT-CMD>`
	The FRUAREA commands must come after the FRUNAME or FRUADDRESS commands they are associated with. The ASCII string contains the name of the FRU area that needs to be programmed. See Table 4 for a list of FRU areas.
	`<OPT-CMD> DMI`
	If the DMI command is added after the FRUAREA command, the FRU information for the specified Board, Chassis, or Product area is programmed into the associated SM BIOS area. Therefore, in the current implementation, the default state is that the DMI information is not programmed.
	When the DMI command is used after the Internal Use, Header, and MultiRecord area definitions, the DMI option is ignored and no error message is generated. This is intended to make the utility more user-friendly.

FRUFIELD	`FRUFIELD    "ASCII STRING"    "ASCII STRING"` This command must come directly after the FRUAREA command that it is associated with. FRUFIELD commands found for fields not belonging to the previous FRUAREA are not allowed. FRUFIELD commands missing from the list of fields for the FRUAREA defaults to what is in the non-volatile storage device. The first ASCII string delineates which FRU field, and the second ASCII string specifies the string to be inserted into that field. Specify an empty field in the second ASCII string by using empty quotes (" "). If no second ASCII string is found, the field information from the FRU file is used. A list of all possible FRU fields for the first ASCII string is in Table 5. The formatting for the second string is listed in Table 6.

continued

Table 1. List of Commands (continued)

Command Name	Description

FRUNAME	`FRUNAME "FILENAME.FRU" <OPT-CMD>`
	A file name of the FRU file to use as a source of FRU information. This file will be verified to be present and used during the configuration process. The string must conform to the file name designation for DOS. Quotes are optional around this file name.
	If FILENAME is replaced by the key word “SYSTEM”, no FRU file name is required. The existing FRU information that was previously programmed is used. This enables you to replace the ASSET Tag without having a FRU File. It is recommended that you follow the FRUNAME command with the FRUADDRESS information. By default the FRUADDRESS information used is the BMC FRU address.
	`<OPT-CMD> COMPARE`
	If the COMPARE argument is added after the FRU name, the information, which would be programmed, is compared to what is in the non-volatile storage device and an appropriate message returned. No information is programmed in this case. This feature is for use by Manufacturing.

IFSET	`IFSET “TAG STRING” “TAGSTRING” “TAG STRING” “TAG STRING”` Looks for the TAG STRING(s) in the existing Master Tag list. Support is provided for up to four tags with an AND implied between each tag. If the tag(s) have been set, all the commands until the first ELSE or ENDIF command are performed. If the tag(s) have not been set, all the commands until the first ELSE or ENDIF command are skipped. Nested IFSET/ELSE/ENDIF statements are supported. The level of nesting is determined by the amount of conventional memory available in your system.
MENU	`MENU “TAG STRING” <OPT-STRING>` Repeated MENU commands presents a list of tag strings. The user is able to select from this list. Numbers for selecting the items are automatically generated for each MENU command found. The selected “TAG STRING” is a descriptive tag used to denote a Sensor Data Record or control the path of execution in the configuration file. The tag is accumulated in the Master Tag link list the same as if it set with the SET command. `<OPT-STRING>` Optional String to be displayed in place of the tag string. Up to 128 byte string allowed.
MENUPROMPT	Displays a prompt after the previous MENU strings, then waits for one of the valid numbers to be entered. Once a valid number has been entered, execution continues and the selected MENU item is stored in the Master Tag link list. Note: If there is no MENUPROMPT command following the MENUTITLE command, the MENUTITLE and MENU’s commands are displayed, but there will be no menu prompt for a response. No error message is generated under this circumstance.
MENUTITLE	`MENUTITLE “ASCII STRING”` A descriptive string explaining the purpose of selecting from the subsequent MENU commands. This command should be placed just prior to the MENU commands. The length limit is 80 ASCII characters.
NO	`NO “TAG STRING”` Looks at a bit accessed by the PROMPT command. If the bit is reset, the tag string denoted in the NO field is accumulated in the Master Tag link list in the same manner as the SET command. The tag string may be a null string (“ “) if no action is desired. If there has been no previous PROMPT command, an error is flagged.

continued

Table 1. List of Commands (continued)

Command Name	Description
PROBE	`PROBE    “ASCII STRING” HEX BYTE    HEX BYTE    HEX BYTE` Used to help determine which information is to be programmed. This command also works with Intelligent Platform Management Bus (IPMB) and non-IPMB FRU devices. See the section on Probing the Product Configuration on page 17 for details.
PROMPT	`PROMPT    “ASCII STRING”` The ASCII STRING is intended to be a question to the user concerning the presence of a device needing a Sensor Data Record. The user answers yes or no (Y/N). A bit is set/reset depending on the user’s answer. Set on yes, reset on no. Another bit is used to designate if a PROMPT command has been given, and is set upon finding the PROMPT command. The YES and NO commands then know whether a PROMPT command has been found. If the YES or NO commands do not follow the PROMPT command, the PROMPT command is ignored. The length limit is 80 ASCII characters.

SDRNAME	`SDRNAME    “FILENAME.SDR”    <OPT-CMD1>    <OPT-CMD2>`
	A file name of the SDR file to use as a source of Sensor Data Records when the product configuration is determined. The string must conform to the file name designation for DOS. No quotes are needed around this file name.
	
	`<OPT-CMD1> LAST`
	If the LAST argument is added after the SDR name, the SDR file will be programmed after the FRU files are programmed. This feature exists for backward compatibility when using the current FRUSDR utility on older platforms which require the SDR file to be programmed last.
	
	`<OPT-CMD2> COMPARE`
	If the COMPARE argument is added after the SDR name, the information, which would be programmed, is compared to what is in the non-volatile storage device, and an appropriate message is returned. No information is programmed in this case. This feature is for use by Manufacturing. The LAST argument, if used, must precede the COMPARE argument.

SET	`SET    “TAG STRING”` “TAG STRING” is a descriptive name normally used to filter Sensor Data Records. These tag strings are accumulated internally by the FRUSDR Load Utility in the Master Tag link list as part of the system configuration.
YES	`YES    “TAG STRING”` Looks at a bit accessed by the PROMPT command. If the bit is set, the tag string denoted in the YES field is accumulated in the Master Tag link list in the same manner as described in for the SET command. The tag string may be a null string (“ ”) if no action is desired. If there has been no previous PROMPT command, an error is flagged.

Working with Strings

- All ASCII strings must be delineated with two double quotes (" ") if they are more than one word in length. Double quotes are also required if you want to use a NULL ASCII string. Quotes are optional for single word strings. The correct number of quotes is checked (and tagged as an error if there is an incorrect number) prior to checking the correct number of arguments.
- If quotes are not used, use white space to separate arguments. White space is defined as space, tab, or Carriage Return-Line Feed characters (CR-LF). Non-white space characters on lines that don't start with comment characters are considered data.
- Lines in the configuration file can be commented out by placing two forward slash marks (//) at the beginning of the line. The comment characters are effective until the end-of-line character is reached. The CR-LF is the end-of-line sequence.
- HEX and STRING data is delimited by white-space or end-of-line characters.
- HEX BYTE denotes a length of 1 byte.
- The string length limit for the DISPLAY, PROMPT, and MENUTITLE commands is 80 ASCII characters.
- All other ASCII strings are 32 bytes in length, 31 ASCII characters and 1 byte for the NULL character.
- Supported ASCII string characters are encoded between 26 and 126, except the double quote (“).
- You must supply an argument after a command that requires one, or an error message is displayed, although it is acceptable in some cases for the argument to be a NULL string shown by two double quotes (" ").
- TAG STRINGS are limited to 32 characters in length, 31 ASCII characters and 1 byte for the NULL character.
- All tags are placed into the Master Tag link list and used to filter the configuration file and all SDR files.

Examples:

```
CONFIGURATION        BEAR.CFG        // OK, quotes are optional.
CONFIGURATION        “BEAR.CFG”      // OK, quotes are optional.

IFSET    BEAR   BMC  FRONT PANEL     // Will be taken as having 4 arguments:
                                     //   BEAR, BMC, FRONT and PANEL.

IFSET    BEAR   BMC  “FRONT PANEL”   // Will be taken as having 3 arguments:
                                     //   BEAR, BMC and FRONT PANEL.

IFSET    “BEAR” “BMC” “FRONT PANEL”  // Will be taken as having 3 arguments:
                                     //   BEAR, BMC, and FRONT PANEL.

IFSET    “BEAR BMC  FRONT PANEL”     // Will be taken as having 1 arguments:
                                     //   ‘BEAR BMC FRONT PANEL’

IFSET    BEAR   BMC  “FRONT PANEL    // Error, will be to indicate the second
                                     //  quote is missing..

IFSET    BEAR   BMC“                 // Error, will still indicate the second
                                     //  quote is missing, Because it could
                                     //  be the first quote of a 3rd argument.
```

Table 2. FRU Field Maximum Allowed Lengths

FRU Fields	FRU Field Max Allowed Length	SM BIOS Field Names	SM BIOS Field Lengths (includes NULL terminator)
Product Manufacturer	1Fh bytes	SysInfo Manufacturer	20h Bytes
Product Name	1Fh bytes	SysInfo Product Name	20h Bytes
Product Part Number / Code	17h bytes	SysInfo Version Number	18h Bytes
Product Serial Number	1Fh bytes	SysInfo Serial Number	20h Bytes
Board Manufacturer	1Fh bytes	BaseBrd Manufacturer	20h Bytes
Board Product Name	1Fh bytes	BaseBrd Product Name	20h Bytes
Board Part Number / Code	17h bytes	BaseBrd Version Number	18h Bytes
Board Serial Number	1Fh bytes	BaseBrd Serial Number	20h Bytes
NULL	1Fh bytes	Chassis Manufacturer	20h Bytes
Chassis Type	1h byte	Chassis Type	1h Byte
Chassis Part Number / Code	17h bytes	Chassis Version Number	18h Bytes
Chassis Serial Number	1Fh bytes	Chassis Serial Number	20h Bytes
Product Asset Tag	1Fh bytes	Chassis Asset Tag	20h Bytes
Product Version	1Fh bytes		

Note: Actual allowed lengths are a factor of the FRU area size and the size of the other FRU fields. A FRU area has limited space, therefore maximum size strings may not be allowed for any string. Each FRU file designer estimates the field sizes that are needed, then creates a FRU file with dummy fields of the needed size. In some cases there may be a padded space at the end of a FRU area. In those instances the FRU fields in that area may be increased in size to use up the padded area.

Processing the Configuration File

The configuration file is first read in by the utility, then parsed. At that time if errors are discovered, the line number of the error in the configuration file is displayed. The configuration file is processed next. During processing, all PROMPT, PROBE, SET, CLEAR and MENU commands are executed. All FRUNAME and SDRNAME commands are put into a link list, along with their associated information.

Once the configuration file has been processed, the SDR file and the FRU file are programmed. The SDR file is programmed before the FRU files. The last command to be executed is the BOOT command, if specified.

Programming FRU Files

You can use the configuration file to program one or more FRU areas providing the FRU file is valid and contains only one FRU address. Multiple FRU addresses cannot be programmed. FRU files are programmed using the FRUADDRESS command for each FRU file. In programming the FRU fields in the board, chassis, and product areas, the appropriate fields must be specified after each FRU area is given. If no FRU fields are specified, the default disregards all FRU fields in the FRU area and retains what was previously in the non-volatile storage device.

The configuration file allows you to override values contained in a FRU file, but it does not allow you to add areas to a FRU. The one exception is the Manufacturing Time. If it is zero, it is programmed with the current time**.** In general, whatever information is contained in the non-volatile storage device should be considered the default values.

FRU File Filtering

You can filter a FRU file using the PROMPT, PROBE, MENU, IFSET and FRU programming statements in the configuration file. By default no FRU areas are programmed, even if a FRUNAME is given. If you want the entire FRU file programmed, you need to specify all the FRU areas and fields to be programmed.

Example:

```
IFSET       "FRU"
      // Program using the generic BMC FRU file
      FRUNAME     "H820BMC.FRU"          // H820 board product only
      FRUAREA     "HEADER"
      FRUAREA     "INTERNALUSE"
      FRUAREA     "CHASSIS"
      FRUFIELD    "CT"
      IFSET       "HUDSON"
        PROMPT      "Would you like to enter the chassis part number?"
        YES         "PARTNUM"
        NO          ""
        IFSET       "PARTNUM"
          FRUFIELD    "P#"  "@STDIN:ASCII"
        ENDIF
        PROMPT      "Would you like to enter the chassis serial number?"
        YES         "SERIALNUM"
        NO          ""
```

```
    IFSET       "SERIALNUM"
      FRUFIELD    "S#"  "@STDIN:ASCII"
    ENDIF
  ELSE
    // Don’t write over any existing part or serial numbers
  ENDIF
  FRUAREA     "BOARD"
  FRUFIELD    "MD"
  FRUFIELD    "MN"
  FRUFIELD    "PN"
  // Don’t write over any existing part or serial numbers
  FRUAREA     "PRODUCT"
  IFSET       "HUDSON"
    FRUFIELD    "MN" "Intel"          // Product manufacturer name
    FRUFIELD    "PN" "H820+ DP"       // Hudson product name
  ELSE
      FRUFIELD    "MN"
      FRUFIELD    "PN"
    ENDIF
    FRUFIELD    "PV"
    // Don’t write over any existing part or serial numbers
    PROMPT      "Would you like to enter an asset tag?"
    YES         "ASSET"
    NO          ""
    DISPLAY     " "
    IFSET       "ASSET"
      FRUFIELD    "AT" "@STDIN:ASCII"
    ENDIF
    FRUAREA     "MULTIREC"
  ENDIF
```

Programming SDR Files

The MASTER.SDR file contains all the possible SDRs for the system. These records might need to be filtered based on the current product configuration. You can use the configuration file to program SDR files using the SDRNAME command. Each record of the SDR file may selectively be programmed by using tags. The maximum allowable length of any SDR is 64 bytes. Any larger records are flagged as an error.

As the utility processes the configuration file, tags are accumulated and saved. These tags are used later to associate and compare with Sensor Data Records Tags in the SDR file for filtering purposes.

By default the utility programs all non-tagged records in an SDR file. In order to program tagged fields, these tags must be set in the configuration file.

Example:

```
Writing SDR Record #36
Reading SDR Repository...........
SDR file was successfully written!
Programming complete, reboot server for normal operation
```

SDR File Filtering

Use the configuration file to filter a SDR file using the SET, PROMPT, PROBE, MENU, and IFSET statements to control tags being set. If you supply tags in a SDR file and in the configuration file, all tagged fields are filtered out, and only non-tagged records are programmed. If you want the entire SDR file programmed, you need to either remove all the tags from the SDR file or set all the tags in the configuration file.

There are no differences between tags used to filter SDRs or tags used to control the configuration file execution path. All tags are stored in the same link list. Therefore, if a tag is set to be used while in the configuration file and the tag does not filter the SDR file, the tag should be removed by using the CLEAR command after the tag has completed its usefulness.

Example:

```
IFSET         "SDR"
      // The master sensor data record file
      SDRNAME       "H820SDR.SDR"
      // If they are using an Intel Hudson chassis
      IFSET       "HUDSON"
        SET         "BB_FAN_1"             // Set SDR tag for baseboard fan 1
        SET         "BB_FAN_2"             // Set SDR tag for baseboard fan 2
        SET         "BB_FAN_3"             // Set SDR tag for baseboard fan 3
        SET         "BB_FAN_4"             // Set SDR tag for baseboard fan 4
      ELSE          //   Else it is a custom system, so ask fan questions
        DISPLAY     "The H820+ DP server board provides instrumentation support
for tachometer"
        DISPLAY     "fans (speed sensing) and for digital fans (on/off sensing
only)."
        PROMPT      "Do you have fans connected to the baseboard?"
        YES         "FANS"
        NO          ""
        DISPLAY     " "
        IFSET       "FANS"
          MENUTITLE   "Are your baseboard fans tachometer fans or digital fans?"
          MENU        "TACH"  "Tachometer fans"
          MENU        "DIGT"  "Digital fans"
          MENU        "NONE"  "Neither"
          MENUPROMPT
          DISPLAY     " "
          IFSET       "NONE"
          // Neither was selected, so skip the next fan section.
          ELSE
            MENUTITLE   "How many baseboard fans do you have connected?"
            MENU        "ONE"   "One baseboard fan"
            MENU        "TWO"   "Two baseboard fans"
            MENU        "THREE" "Three baseboard fans"
            MENU        "FOUR"  "Four baseboard fans"
            MENUPROMPT
            DISPLAY     " "
            IFSET       "TACH"
              IFSET       "ONE"
                SET       "BB_FAN_1"
              ENDIF
              IFSET       "TWO"
                SET       "BB_FAN_1"
                SET       "BB_FAN_2"
              ENDIF
              IFSET       "THREE"
                SET       "BB_FAN_1"
                SET       "BB_FAN_2"
                SET       "BB_FAN_3"
              ENDIF
              IFSET       "FOUR"
                SET       "BB_FAN_1"
                SET       "BB_FAN_2"
                SET       "BB_FAN_3"
                SET       "BB_FAN_4"
              ENDIF
            ENDIF
            IFSET       "DIGT"
              IFSET       "ONE"
                SET       "DIG_BB_FAN_1"
              ENDIF
              IFSET       "TWO"
                SET       "DIG_BB_FAN_1"
                SET       "DIG_BB_FAN_2"
              ENDIF
              IFSET       "THREE"
                SET       "DIG_BB_FAN_1"
                SET       "DIG_BB_FAN_2"
                SET       "DIG_BB_FAN_3"
              ENDIF
              IFSET       "FOUR"
                SET       "DIG_BB_FAN_1"
                SET       "DIG_BB_FAN_2"
                SET       "DIG_BB_FAN_3"
                SET       "DIG_BB_FAN_4"
              ENDIF
            ENDIF
          ENDIF
        ENDIF
        DISPLAY     " "
      ENDIF
    ENDIF //End of IFSET SDR
```

SM BIOS (DMI) Programming and Mapping

Although the utility supports programming the SM BIOS or DMI area, this feature is not available on all platforms. On many platforms, the BIOS automatically pulls in the related FRU information into the DMI area when the system boots. On these system there is no need to program the DMI area and therefore this feature of the utility will fail if attempted.

In order to program the DMI area with DMI information, you must use the “DMI” switch parameter on the same line after the specified “FRUAREA”. The DMI and FRU areas do not have a direct mapping, therefore a "best fit" is done when programming the DMI area as shown in Table 3.

NOTE

The default for programming the DMI is "off".

The DMI switch is only meaningful after the Chassis, Board and Product FRU areas have been defined in the configuration file. The utility does not care which FRU file gets mapped to the DMI area. Therefore, in one case the Product area from one FRU file may be mapped to the DMI area, then later on in the same configuration file the Board area of another FRU file may be mapped. If the configuration file has multiple areas mapped to the same DMI area, the information from the last one programmed prevails in the DMI area.

Table 3. FRU/SM BIOS (DMI) Conversion Table

FRU Area	**SM BIOS (DMI) Area**
Product Information:	System Information:
Manufacturer Name	Manufacturer
Product Name	Product Name
Product Version	(No Mapping)
Part Number	Version Number
Serial Number	Serial Number
Asset Tag	(Mapped to Chassis Asset Tag)
Board Area:	**Board Information:**
Date and Time	(No Mapping)
Board Manufacturer	Manufacturer
Product Name	Product
Part Number	Version Number
Serial Number	Serial Number
Chassis Area:	**Chassis Information:**
(No Mapping)	Manufacturer
Type	Chassis Type
Part Number	Version Number
Serial Number	Serial Number
(Mapped to Product Asset Tag)	Asset Tag

Table 4. FRU Area String Specifications

FRU Area ASCII Strings
“HEADER”
“INTERNALUSE”
“CHASSIS”
“BOARD”
“PRODUCT”
“MULTIREC”

Table 5. FRU Field First String Specifications

FRU Field First ASCII String	String Description
“CT”	Chassis Type. In Chassis area only. ASCII String must represent a hex byte. Note: Regardless of the input type specified in the configuration file, the chassis type will always be BINARY/HEX and displayed in HEX format. Therefore, if input is “@STDIN:ASCII” and the ASCII characters “0d” are accepted as input, the number used will be 0x0D Hex. If the ASCII characters “13” are accepted as input, the number used will be 0x13 Hex.
“MN”	Manufacturer Name. In Board and Product areas only.
“PN”	Product Name. In Board and Product areas only.
“P#”	Part Number. In Chassis, Board and Product areas only.
“S#”	Serial Number. In Chassis, Board and Product areas only.
“PV”	Product Version. In Product area only. Note: The Product Version FRU Field does not map to a SM BIOS field and therefore is not displayed with the SM BIOS information.
“AT”	Asset Tag. In Product area only.
“MD”	Manufacturing Date & Time. In Board area only. Note: Date and time are taken from the Server, therefore no arguments are allowed following “MD”.
“AMx”	Additional Manufacturing Info. There can be multiple instances of this field, where ‘x’ represents a decimal number from 1-9.

Note: If the same field string is used multiple times in an area, the last string found is the one used. No error message is displayed. Except for the “CT” field, a field length must not equal one.

FRU Field Second ASCII String Formatting

Whether entering information by way of STDIN, an environment variable, or from a file, if an input length of zero is entered, the utility treats it the same as an empty string. An empty string is entered into the selected FRU field, thus wiping out any previously stored input. If the DMI switch is set, the empty string is also entered into the appropriate SM BIOS field.

Table 6. FRU Field Second String Specification

NULL (does not exist)	If a second string does not exist, the string in FRU file is used by default.
“ ” (empty string)	An empty string means that an empty field is inserted into the FRU field. Any other number of ASCII characters can be entered up to 31 bytes.
“@FILE:TYPE:NAME:#”	@FILE designator lets the utility know that it needs to obtain the string from a file designated by NAME. NAME may be a complete path up to 64 bytes, filename is standard DOS format. NAME may also be IFICS, this lets the utility know it needs to find the IFICS file to get the string from it. The ‘#’ specifies which line of the file to insert as the string. If no line number is supplied, an error will occur. TYPE specifies what coding designation the ASCII string from the file needs to be translated to. Possible TYPE designations specified in Table 7. Valid input terminates with any ASCII character entered outside the bounds of 20h to 7Fh.
“@ENV:TYPE:NAME”	@ENV designator lets the utility know it needs to obtain the ASCII string from the environment variable specified by NAME. TYPE specifies what coding designation the ASCII string from the file needs to be translated to. Possible TYPE designations are specified in Table 7. Valid input terminates with any ASCII character entered outside the bounds of 20h to 7Fh.
“@STDIN:TYPE”	@STDIN designator lets the utility know it needs to obtain the ASCII string from operator input. TYPE specifies what coding designation the ASCII string from the file needs to be translated to. Possible TYPE designations are specified in Table 7. Valid input terminates with any ASCII character entered outside the bounds of 20h to 7Fh.

Table 7. TYPE Code Specification

TYPE	Description	FRUSDR Version
ASCII	Straight 8 bit ASCII.	Ver 2.0 and later
ASCII6	6 bit ASCII packed bytes.	Support TBD
BCD	Binary Coded Decimal.	Ver 2.0 and later
BIN	Binary bytes. Note: Binary data will be displayed in HEX format.	Support TBD
UNICODE	Unicode.	Support TBD

Note: The allowable IPMI type codes are specified by the IPMI Platform Management FRU Information Storage Definition, but this table specifies which subset of type codes are supported by the FRUSDR utility.

Input of Data

The format for taking input data from the FRU and SDR files is defined in their respective specifications, which are listed in the reference documents section of this manual. When taking in data from a file, an environment variable, or from stdin, the input type is always ASCII. If in the Configuration File the input type is specified as something other than ASCII, it is converted after it has been taken as input. The maximum number of characters taken does not change based on the input type. For example, if 31 characters is the maximum ASCII length for a field, 31 characters is also the maximum length for Binary Coded Decimal (BCD), Binary, and ASCII6 input. Therefore, using different input types does not allow more characters to be stored, but rather the same number of characters to be stored in fewer bytes.

BCD input data should not have any separation characters such as a space as it does when taken from a FRU file. Two numbers are stored in each BCD byte, and only even length BCD numbers are valid. If one or two BCD characters are entered, they will not be accepted because when translated they take only one byte. If an odd number of BCD characters are input, they will be pre-appended with a zero in order to make the length even. The maximum number of BCD characters that can be entered is 30, or 15 bytes of characters (2 characters per byte).

Example 1:

BCD taken from FRU file:

```
99 12 34 56 78
```

Same BCD value taken from a file, an environment variable, or stdin:

```
9912345678
```

Example 2:

Odd number of BCD characters entered:

```
12345
```

Converts to:

```
012345
```

NOTE

UNICODE characters are not supported.

Updating the FRU Non-Volatile Storage Area

After the system configuration is determined, a typical configuration file updates the FRU non-volatile storage area. It will first verify the Common Header area and the checksum from the specified FRU file with what is programmed in the FRU non-volatile storage device. If specified, the Internal Use Area is read out of the specified FRU file and is programmed into the non-volatile storage device. Then chassis, board and product areas are read out of the specified FRU file and programmed into the non-volatile storage device. Finally the Multi-Record Area is read out of the specified FRU file, and the area is programmed into the FRU non-volatile storage device. If the temporary files option is enabled, all the areas are also written to the FRU.TMP file, which happens before each area get programmed.

Updating the SDR Non-Volatile Storage Area

After the utility validates the header area of the supplied SDR file, it updates the SDR repository area. Before programming, the utility clears the SDR repository area. When loading an SDR file from a Configuration File, the utility filters all tagged SDRs using a list of tags you set based on the product's configuration. Nontagged SDRs are automatically programmed.

If the temporary files option is enabled, the utility also copies all written SDRs to the SDR.TMP file. Generally, the SDR file is ordered from lowest to highest type, ending with a C0h type record containing the SDR Version information.

Updating the SM BIOS (DMI) Non-Volatile Storage Area

The utility adds the following chassis and product FRU information to the SM BIOS fields.

Example:

```
Loading SM BIOS System Area
  Manufacturer Name : Intel
  Name : AD450NX Server System
  Version Number : SMADN000BN00
  Serial Number : 0123456789
Loading SM BIOS Board Area
  Manufacturing Name : Intel
  Name : BMAD440LX
  Serial Number : 0123456789
  Version Number : 661880-303
Loading SM BIOS Chassis Area
  Chassis Part Number : 665001-002
  Chassis Serial Number :
  Asset Tag :
```

Sample Configuration File

The following is a sample configuration file for the FRUSDR Load Utility.

```
// SAMPLE CONFIGURATION FILE - This example contains bogus data.  It should be
// suitable to illustrate the use of the file format, and should be able to be
// saved as a separate ASCII text file that may be used as an aid to the
// development and editing of configuration files.

CONFIGURATION   “Rainy Day System”   // The name for this configuration file

// Do all prompts and probing here
// All sets are associated with all SDR files
SET    “Fan sensor 1”        // The name associated with SDR record
SET    “Fan sensor 2”        // The name associated with SDR record
SET    “Fan sensor 3”        // The name associated with SDR record

MENUTITLE   “Select one of the following items:”
MENU        “HSC1”  “Hot Swap Backplane one” //one of the possible selections
MENU        “HSC2”  “Hot Swap Backplane two” //one of the possible selections
MENU        “RPX”   “The RPX board”          //one of the possible selections
MENU        “PWS1”  “Power Supply 1”         //one of the possible selections
MENUPROMPT

// The FRUSDR Load Utility would interrogate the user with the string:
// “Is AT24C64 EEPROM present? [Y/N]”  The user types ‘Y’ or ‘N’ and return key.
PROMPT    “Is AT24C64 EEPROM present?”  // ATMEL 8K byte Serial EEPROM
YES       “AT24C64 EEPROM”  // If device present, use this data
NO         “”               // This field is blank indicating no action is
                            // desired if 'AT24C64 EEPROM’ isn’t there.

PROBE   “IMBDEVICE”  FF  20     // Probe the IMB device on the ISA bus at adx 20h
FOUND   “Columbus H/S”  “”      // set 1st string if probe found device, 2nd if
                                // not either string could be empty.

SDRNAME    “abcsdr.sdr”               // The SDR file name used by this
                                      // configuration file to program.
SDRNAME    “abcsdr.sdr”   COMPARE     // The SDR file name used by this
                                      // configuration file to compare.

IFSET   “Columbus H/S”
FRUNAME  HS001.FRU
FRUADDRESS   IMBDEVICE 01 02 22
FRUAREA      “HEADER”
FRUAREA      “INTERNALUSE”
FRUAREA      “BOARD”
FRUFIELD     “PN”  “@FILE:ASCII:PRODUCT.TXT:2”
FRUFIELD     “P#”  “@ENV:ASCII:PRODUCTCODE”
FRUFIELD     “S#”  “@FILE:ASCII:IFICS:2”
FRUAREA            “CHASSIS”
FRUFIELD     “P#”   @ENV:ASCII:PRODUCTCODE”
FRUFIELD     “S#”   @FILE:ASCII:IFICS:2”
FRUAREA            “PRODUCT”
FRUFIELD     “PN”   “M440LX Server System”
FRUFIELD     “P#”          // Use what is in the FRU file
FRUFIELD     “S#”   “@FILE:ASCII:IFICS:2”
FRUFIELD     “PV”   “”
FRUFIELD     “AT”   “N00000001”
ENDIF

FRUNAME     master.fru                  // The FRU file name used by this
FRUADDRESS  0x20                        // configuration file.
FRUADDRESS  0x22
FRUADDRESS  0x24
IFSET       “BOARD LEVEL”
FRUAREA     “HEADER”
FRUAREA     “INTERNALUSE”
FRUAREA     “BOARD”
FRUFIELD     “PN”  “@FILE:ASCII:PRODUCT.TXT:2”
FRUFIELD     “P#”  “@ENV:ASCII:PRODUCTCODE”
FRUFIELD     “S#”  “@FILE:ASCII:IFICS:2”
ENDIF

IFSET        “PRODUCT LEVEL”
FRUAREA      “INTERNALUSE”
FRUAREA      “CHASSIS”
FRUFIELD     “P#”  “@ENV:ASCII:PRODUCTCODE”
FRUFIELD     “S#”                    // Use what is in the FRU file
FRUAREA      “PRODUCT”
FRUFIELD     “PN”  “M440LX Server System”
FRUFIELD     “P#”  “@ENV:ASCII:PRODUCTCODE”
FRUFIELD     “S#”  “@FILE:ASCII:IFICS:2”
FRUFIELD     “PV”  “”
FRUFIELD     “AT”  “N00000001”
ENDIF

FRUNAME     “frutest.fru” COMPARE    // The name of a FRU file this configuration
                                     // file will use to compare to a FRU NVS
device
DISPLAY     “Enter serial number:  ”
FRUFIELD    “S#”      “@STDIN:ASCII”

// End of Configuration File
```

Cleaning Up and Exiting

If an update was successfully performed, the utility displays a single message and then exits. If the utility fails, it immediately exits with an error message and exit code.

4 Solving Problems

Error Messages

When errors occur during the processing of the configuration file one or more error messages are displayed. When the utility exits, an error code related to the last error message is returned to the OS shell which you can retrieve. If the utility has run successfully, the error code is zero, otherwise the exit code is some number above zero.

Warning Messages

Any of the following error messages may also be displayed as warning messages. The difference between the two is that warning messages do not cause the application utility to stop processing input and exit with an error code.

Miscellaneous Errors & Warnings	**Exit Code 1**
Error, unsuccessful write of FRU File to FRU area	
Error, EEPROM Board area contains an invalid checksum	
Error, EEPROM Chassis area contains an invalid checksum	
Error, exceeded maximum allowed BCD string length	
Error, expected data in record data portion	
Error, failed to retrieve header information from FRU file	
Error, failed to send message	
Error, FRU file appears to be invalid	
Error, insufficient memory	
Error, invalid Binary Coded Decimal (BCD) number was found	
Error, invalid chassis field	
Error, invalid checksum found in EEPROM	
Error, invalid field	
Error, an invalid field length of one was found	
Error, invalid header	
Error, invalid number of bytes used to indicate hex number	
Error, invalid string entered. You must enter a valid ASCII string	
Error, address values must be even	
Error, odd BCD length entered, so a zero was pre-appended	
Error, only 16 FRU addresses are allowed per FRU file	
Error, EEPROM Product area contains an invalid checksum	
Error, reading string length	
Error, reading record from file	

continued

Miscellaneous Errors & Warnings (continued)	**Exit Code 1**
Error, reading SDR record from file	
Error, processing FRU file	
Error, failed to read FRU string	
Error, security of program has been breached	
Error, type not found	
Error, this utility does not support saving the field in the requested type	
Error, all addresses must be unique	
Error, in verifying the written FRU	
Error, wrong number of bytes read	
Error, in verifying the written string	

Command Line Errors	**Exit Code 2**
Error, expected FRU file name	
Error, too many command line arguments	
Error, invalid command line parameter(s)	
Error, invalid file name extension entered	
Error, invalid LUN address found after NVS Type on command line	
Error, invalid DEV Ctrl after NVS Type on the command line	
Error, missing command line parameter(s)	
Error, no command line arguments were given	
Error, no filename extension entered	
Error, too many device addresses, only 16 device addresses are allowed	

FRU & SDR Programming Errors	**Exit Code 3**
Error, selected area not present	
Error, board FRU data is larger than the board FRU area	
Error, chassis FRU data is larger than the chassis FRU area	
Error, Automatic-iFICS is not available	
Error, Internal Use area size overflow	
Error, MultiRecord area size overflow	
Error, product FRU data is larger than the product FRU area	

Configuration File Parsing Errors	**Exit Code 4**
Error, argument follows a command in the configuration file when it should not	
Error, argument is too long that follows command in the configuration file	
Error, exceeded maximum argument length	
Error, incorrectly formatted input '@' string	
Error, invalid argument following FRU area in the configuration file	
Error, invalid argument following FRU name in the configuration file	

continued

Configuration File Parsing Errors (continued)	**Exit Code 4**
Error, invalid argument following SDR name in the configuration file	
Error, invalid argument found in configuration file	
Error, invalid BMCaddress found, the BMC address should be IMBDEVICE FF 20	
Error, invalid Board Area in file	
Error, invalid bus address	
// Special cases for printing out line number	
Error, invalid command found in the configuration file on line %d	
Error, invalid comment in configuration file on line %d	
Error, invalid configuration file string found	
Error, invalid Chassis Area in file	
Error, invalid chassis type found, must be a hex number from 01 to 17 hex	
Error, invalid device address found	
Error, invalid field input type specified in the configuration file	
Error, invalid FRU address found in configuration file	
Error, invalid FRU address format in configuration file	
Error, invalid FRU area found in configuration file	
Error, invalid Header Area in file	
Error, invalid Internal Use Area in file	
Error, invalid line number found	
Error, invalid MultiRecord Area in file	
Error, invalid device address, must be an even number	
Error, invalid number of arguments following the FRUADDRESS command	
Error, invalid number of arguments following the FOUND command	
Error, invalid number of arguments following the PROBE command	
Error, invalid Product Area in file	
Error, matching quote missing after command in the configuration file	
Error, the FRU file area must come before the FRU field	
Error, a FRU file name was found, but no FRU areas were specified	
Error, no valid argument found after command in the configuration file	
Error, no conditional statement found after IFSET	
Error, invalid FRU field was found	
Error, an IFSET statement must precede the ELSE statement	
Error, an IFSET statement must precede the ENDIF statement	
Error, the FRU file name must come before the FRU address	
Error, the FRU file name must come before the FRU area	
Error, a PROMPT command must precede the YES and NO commands	
Error, no previous corresponding PROBE for the FOUND command	
Error, non-matching IfSet and Endif statements	
Error, non-matching quotes found on configuration line	

continued

Configuration File Parsing Errors (continued)	**Exit Code 4**
Error, parsing configuration information	
Error, problem in parsing configuration file	
Error, parsing configuration line	
Error, problem in reading configuration file	
Error, too many arguments found after command in the configuration file	

Protocol / Communication Errors	**Exit Code 5**
Error, busy bit timeout	
Error, DATA READY timeout.	
Error, could not detect BMC through ISA driver	
Error, EEPROM not initialized	
Error, failed to get Read End from SMIC	
Error, failed to get Read Start from SMIC	
Error, failed to get Ready from SMIC	
Error, failed parsing message	
Error, failed receiving message	
Error, failed sending message	
Error, I2C bus problem	
Error, reading EEPROM string	
Error, reading internal thresholds	
Error, failed to receive BMC response	
Error, failed to receive BMC response, bad completion code	
Error, unknown failure while trying to receive BMC response	
Error, failed to send BMC command	
Error, in writing string to EEPROM	
Error, in verifying write to EEPROM	
Error, transmit buffer full	

DMI Problems	**Exit Code 6**
Error, INT15 (DA92h) Call not supported for CPU Config Info	
Error, “Get DMI Information” request failed	
Error, “Get DMI Structure” request failed	
Error, invalid DMI type requested: %d	
Error, “Set DMI Structure” request failed	
Error, writing to DMI space	

continued

Verify/Compare Errors	**Exit Code 7**
Error, FRU file does not match with what was programmed into memory	
Error, board areas defined in the file and in memory are different	
Error, board area checksums defined in the file and in memory are different	
Error, board area lengths defined in the file and in memory are different	
Error, board area offsets defined in the file and in memory are different	
Error, chassis areas defined in the file and in memory are different	
Error, chassis area checksums defined in the file and in memory are different	
Error, chassis area lengths defined in the file and in memory are different	
Error, chassis area offsets defined in the file and in memory are different	
Error, common header checksums defined in the file and in memory are different	
Error, file and EEPROM header areas must be identical, but are different	
Error, header length does not match with the sum lengths of each area	
Error, header format versions defined in the file and in memory are different	
Error, internal use area lengths defined in the file and in memory are different	
Error, internal use area offsets defined in the file and in memory are different	
Error, MultiRecord areas defined in the file and in memory are different	
Error, MultiRecord area lengths defined in the file and in memory are different	
Error, MultiRecord area offsets defined in the file and in memory are different	
Error, product areas defined in the file and in memory are different	
Error, product area checksums defined in the file and in memory are different	
Error, product area lengths defined in the file and in memory are different	
Error, product area offsets defined in the file and in memory are different	
Error, SDR file does not match with what was programmed into memory	
Error, SDR file unsuccessfully written	

FRU & SDR File Parsing Errors	**Exit Code 8**
Error, padded Board Area in FRU area on this server must be all zeros	
Error, padded Chassis Area in the FRU area on this server must be all zeros	
Error, expected valid hexadecimal data after _DEV_ADDRESS	
Error, expected valid hexadecimal data after _DEV_BUS	
Error, expected valid hexadecimal data after _DATA_LEN	
Error, expected valid hexadecimal data after _REC_LEN	
Error, expected valid hexadecimal data after _SDR_TYPE	
Error, expected valid hexadecimal data after _START_ADDR	
Error, _DATA_LEN should follow _START_ADDR	
Error, _DEV_ADDRESS should follow _DEV_BUS	
Error, _DEV_BUS should follow _NVS_TYPE	
Error, expected _FRU or _SDR tag	
Error, expected a left parenthesis '('	

continued

FRU & SDR File Parsing Errors (continued)	**Exit Code 8**
Error, _LF_FMT_VER should follow _LF_VERSION	
Error, _LF_NAME should be the first tag in the file	
Error, _LF_VERSION should follow _LF_NAME	
Error, _NVS_TYPE should follow _DATA_LEN	
Error, _REC_LEN should follow _SDR_TAG	
Error, _IPMI_VERSION should follow _LF_FMT_VER	
Error, _START_ADDR should be first entry in SEL header info	
Error, expected ASCII text after _LF_FMT_VER	
Error, expected ASCII text after _LF_NAME	
Error, invalid or unsupported _NVS_TYPE	
Error, expected a four digit hex number after _IPMI_VERSION	
Error, expected a four digit hex number after _IPMI_VERSION	
Error, expected ASCII text after _LF_VERSION	
Error, FRU Board Area may only be defined once per FRU file	
Error, FRU Chassis Area may only be defined once per FRU file	
Error, FRU Header Area may only be defined once per FRU file	
Error, FRU Internal Use Area may only be defined once per FRU file	
Error, FRU MultiRecord Area may only be defined once per FRU file	
Error, FRU Product Area may only be defined once per FRU file	
Error, invalid Board Area length in the FRU file	
Error, invalid Board Area length in the FRU area on this server	
Error, invalid common header checksum	
Error, invalid Chassis Area length in the FRU file	
Error, invalid Chassis Area length in the FRU area on this server	
Error, invalid environment variable length	
Error, invalid field length found in the FRU file	
Error, invalid FRU file header was found	
Error, invalid MultiRecord Area header block was found	
Error, invalid Product Area length in the FRU area on this server	
Error, invalid Sensor Data Record length found	
Error, no arguments permitted after the Manufacturing Date & Time field	
// Special case, leave on Error & cr lf	
Error, this program only supports file format version %s	
// Special case, leave on Error & cr lf	
Error, this program only supports SDR format version %s	
Error, padded Product Area in the FRU area on this server must be all zeros	
Error, expected ASCII text after _SDR_TAG	

continued

FRU & SDR File Parsing Errors (continued)	**Exit Code 8**
Error, parsing SDR or FRU file	
Error, parsing SDR or FRU file	
Error, unsupported field type found	
Error, unknown token keyword in file.	

File Processing Errors	**Exit Code 9**
Error, failed to open CFG.TMP file	
Error, failed to open DMI.TMP file	
Error, failed to open FRU.TMP file	
Error, failed to open SDR.TMP file	
Error, failed to open the default master input file: MASTER.CFG	
Error, failed to open the default master input file: MASTER.FRU	
Error, failed to open the default master input file: MASTER.SDR	
Error, failed to open the specified input file	
Error, failed to open the specified configuration file	
Error, failed to open the specified FRU file	
Error, failed to open the specified SDR file	
// Special case for printing out line number	
Error, an invalid character was found on line %d	
Error, invalid file size. FRU, SDR and CFG files must be less than 256 KBytes	
// Special case for printing out line number	
Error, line %d is too long. Lines lengths must be 255 characters or less	
Error, failed when writing to temporary file	

General Error	**Exit Code 10**

NOTE

In some cases, an error exit code 10 is returned where more than one error has occurred.

Appendix A

Glossary of Terms

Term	Definition
Baseboard Management Controller (BMC)	The primary microcontroller that controls the operation of Intel's server management subsystem.
Baseboard Management Controller (BMC) Configuration Defaults	Power-up configuration default values used by BMC upon power-up. Stored in the Internal Use Area of the BMC FRU Inventory area. Configuration parameters include default settings for timer intervals, timer enables, retry counts, messaging timeout intervals, buffer sizes, etc. The location and format of this information within the Internal Use Area varies by BMC code version.
Common Header	An area within a FRU Inventory area. Stores information for the FRU Inventory storage area format, overall size of the area, and offsets to the other areas within the FRU Inventory area (e.g., Internal Use Area, chassis Info Area, etc.). The BMC copies the Common Headers from FRU files to non-volatile storage.
Cold/Hard Boot	A Cold or Hard boot is defined in this document as turning off the system power via the on/off switch. Pulling the power cord is not necessary.
DMI area	See SM BIOS area.
Field Replaceable Unit (FRU)	A part which can be replaced in the field by service personnel or perhaps by a customer.
Field Replaceable Unit (FRU) Inventory area	An area in non-volatile storage that holds FRU information, including the serial number and part number of the board (e.g., Baseboard).
Internal Use Area	A portion of the BMC Inventory area. Holds device-specific configuration parameters. This area is NOT for OEM use.
Master Configuration File	A file containing the information that directs the FRU/SDR Load Utility to determine the configuration of the product. The configuration is then used to select the appropriate Sensor Data Records.
Sensor Data Record (SDR)	A record which completely describes a particular sensor. Each sensor has an associated SDR.
SDR Repository	Stores Sensor Data Records for all the sensors in a system. Resides in non-volatile storage under the control of the BMC.
SDR Table	The area in non-volatile storage dedicated to SDR Repository.
SM BIOS (DMI)	Desktop Management Interface software. An interface developed by industry consortium led by Intel. It provides dynamic system configuration and status, prepares and executes queries of system resources, and performs fault diagnostics.
SM BIOS area	Holds FRU information for SM BIOS/DMI software). Located in the non-volatile storage area of the BIOS.

Index

B

Baseboard Management Controller (BMC)
  configuration defaults, 47
  definition, 47
  non-volatile storage device, 6
BIOS, non-volatile storage device, 6

C

Cold/Hard Boot, definition, 47
Command Line Format, 11
  parsing the command line, 11
  precedence, 11
Commands
  // Comment, 21
  /? or /h MENU command, 12
  /d dmi command, 12
  /d fru command, 12
  /d sdr command, 12
  BOOT, 21
  CLEAR, 21
  CONFIGURATION, 21
  DISPLAY, 21
  ELSE, 21
  ENDIF, 21
  FOUND, 21
  FRUADDRESS, 22
  FRUAREA, 22
  FRUFIELD, 22
  FRUNAME, 23
  IFSET, 23
  List of, 21
  MENU, 23
  MENUPROMPT, 23
  MENUTITLE, 23
  NO, 23
  primary, 16
  PROBE, 24
  PROMPT, 24
  SDRNAME, 24
  secondary, 16
  SET, 24
  YES, 24
Common Header, definition, 47
COMPARE, using, 20
Compatibility, 5
Configuration File, 15
  about the configuration file, 15
  file format, 15
  file ordering, 16
  invoking the configuration file, 15
  processing the configuration file, 27
  sample configuration file, 36
  sample probe configuration file, 18
  strings, working with, 25

D

Desktop Management Interface (DMI), also known as System Management BIOS (SM BIOS), 5
Displaying
  FRU area, 13
  given area, 12
  SDR area, 14
  usage information, 12

E - F

Error Messages, 39
Field Replacement Unit (FRU), 5
FRU
  checking the FRU data integrity, 20
  definition, 47
  displaying FRU area, 13
  FRU file filtering, 27
  inventory area, 47
  programming files, 27
  updating the FRU non-volatile storage area, 35
FRUSDR Load Utility
  about, 6
  how to run, 7
  when to run, 7

G - I

Glossary of Terms, 47
Intelligent Chassis Management Bus (ICMB), 8
Intelligent Platform Management Bus (IPMB), 17
Intelligent Platform Management Interface (IPMI), 5
Internal Use Area, 47

M - N

manufacturing time, 15
master configuration file, 6
  definition, 47
non-volatile storage (NVS), 5

P

Probing the Product Configuration
  GET-DEVICE-ID, 17
  using the DEVICE_ID option, 17
  using the NVS_TYPE option, 17
  using the PROBE command, 17

S

SDR
  definition, 47
  displaying SDR area, 14
  programming files, 28
  SDR file filtering, 29
  SDR repository, definition, 47
  SDR table, definition, 47
  updating the SDR non-volatile storage area, 35
Sensor Data Record (SDR), 5
SM BIOS (DMI)
  definition, 47
  programming and mapping, 31
  updating the SM BIOS (DMI) non-volatile storage area, 35
SM BIOS (DMI) area, definition, 47
Solving Problems, 39
  Error Messages, 39
  Warning Messages, 39
strings, working with, 25
System and Software Requirements, 5
System Management BIOS (SM BIOS) also known as Desktop Management Interface (DMI), 5

W

Warning Messages, 39