System Management BIOS Reference Specification

previously known as

Desktop Management BIOS Specification

Version 2.2 — 16 March 1998

Co-authored by American Megatrends Inc., Award Software International Inc., Compaq Computer Corporation, Dell Computer Corporation, Hewlett-Packard Company, Intel Corporation, International Business Machines Corporation, Phoenix Technologies Limited, and SystemSoft Corporation.

This document is provided AS IS for informational purposes only. AMERICAN MEGATRENDS INC., AWARD SOFTWARE INTERNATIONAL INC., COMPAQ COMPUTER CORPORATION, DELL COMPUTER CORPORATION, HEWLETT-PACKARD COMPANY, INTEL CORPORATION, INTERNATIONAL BUSINESS MACHINES CORPORATION, PHOENIX TECHNOLOGIES LIMITED, AND SYSTEMSOFT CORPORATION MAKE NO WARRANTIES, EXPRESS OR IMPLIED, IN THIS DOCUMENT.

American Megatrends Inc., Award Software International Inc., Compaq Computer Corporation, Dell Computer Corporation, Hewlett-Packard Company, Intel Corporation, International Business Machines Corporation, Phoenix Technologies Limited, or SystemSoft Corporation may have patents or pending patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. The furnishing of this document does not give you any license to such patents, trademarks, copyrights, or other intellectual property rights.

AMERICAN MEGATRENDS INC., AWARD SOFTWARE INTERNATIONAL INC., COMPAQ COMPUTER CORPORATION, DELL COMPUTER CORPORATION, HEWLETT-PACKARD COMPANY, INTEL CORPORATION, INTERNATIONAL BUSINESS MACHINES CORPORATION, PHOENIX TECHNOLOGIES LIMITED, AND SYSTEMSOFT CORPORATION DO NOT MAKE ANY REPRESENTATION OR WARRANTY REGARDING SPECIFICATIONS IN THIS DOCUMENT OR ANY PRODUCT OR ITEM DEVELOPED BASED ON THESE SPECIFICATIONS. AMERICAN MEGATRENDS INC., AWARD SOFTWARE INTERNATIONAL INC., COMPAQ COMPUTER CORPORATION, DELL COMPUTER CORPORATION, HEWLETT-PACKARD COMPANY, INTEL CORPORATION, INTERNATIONAL BUSINESS MACHINES CORPORATION, PHOENIX TECHNOLOGIES LIMITED, AND SYSTEMSOFT CORPORATION DISCLAIM ALL EXPRESS AND IMPLIED WARRANTIES, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OR MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND FREEDOM FROM INFRINGEMENT. WITHOUT LIMITING THE GENERALITY OF THE FOREGOING, AMERICAN MEGATRENDS INC., AWARD SOFTWARE INTERNATIONAL INC., COMPAQ COMPUTER CORPORATION, DELL COMPUTER CORPORATION, HEWLETT-PACKARD COMPANY, INTEL CORPORATION, INTERNATIONAL BUSINESS MACHINES CORPORATION, PHOENIX TECHNOLOGIES LIMITED, AND SYSTEMSOFT CORPORATION DO NOT MAKE ANY WARRANTY OF ANY KIND THAT ANY ITEM DEVELOPED BASED ON THESE SPECIFICATIONS, OR ANY PORTION OF A SPECIFICATION, WILL NOT INFRINGE ANY COPYRIGHT, PATENT, TRADE SECRET OR OTHER INTELLECTUAL PROPERTY RIGHT OF ANY PERSON OR ENTITY IN ANY COUNTRY. IT IS YOUR RESPONSIBILITY TO SEEK LICENSES FOR SUCH INTELLECTUAL PROPERTY RIGHTS WHERE APPROPRIATE. AMERICAN MEGATRENDS INC., AWARD SOFTWARE INTERNATIONAL INC., COMPAQ COMPUTER CORPORATION, DELL COMPUTER CORPORATION, HEWLETT-PACKARD COMPANY, INTEL CORPORATION, INTERNATIONAL BUSINESS MACHINES CORPORATION, PHOENIX TECHNOLOGIES LIMITED, OR SYSTEMSOFT CORPORATION SHALL NOT BE LIABLE FOR ANY DAMAGES ARISING OUT OF OR IN CONNECTION WITH THE USE OF THESE SPECIFICATIONS, INCLUDING LIABILITY FOR LOST PROFIT, BUSINESS INTERRUPTION, OR ANY OTHER DAMAGES WHATSOEVER. SOME STATES DO NOT ALLOW THE EXCLUSION OR LIMITATION OF LIABILITY OR CONSEQUENTIAL OR INCIDENTAL DAMAGES; THE ABOVE LIMITATION MAY NOT APPLY TO YOU.

ActiveMovie, ActiveX, BackOffice, Direct3D, DirectDraw, DirectInput, DirectMusic, DirectPlay, DirectSound, DirectX, Microsoft, NetMeeting, Win32, Windows, and Windows NT are trademarks or registered trademarks of Microsoft Corporation in the United States and/or other countries. Intel, Pentium, and MMX are trademarks or registered trademarks of Intel Corporation. *Other product and company names mentioned herein might be the trademarks of their respective owners.

Document Information

The softcopy version of this specification is available from the following World Wide Web sites:

- www.phoenix.com/techs
- developer.intel.com/ial/wfm/
- www.ibm.com/products/surepath

Document Revision History

Version	Release Date	Description
2.0D	09/14/1995	Initial Release of DRAFT COPY
2.0M	12/12/1995	Final draft released, with the following changes: • Specified that dmiStorageBase (Function 50h) and NVStorageBase (Function 55h) must be paragraph-aligned. • Added Command value to change a string to function 52h; Command enumeration values modified. • Removed redundant enumerations from Processor Family list • Corrected Memory Subsystem Example • Corrected/clarified Indexed I/O access-methods for event-log; Access Method enumeration values and Access Method Address union modified • Added clarifications to some of the event log types
2.00	03/06/1996	Final release, with the following changes: • Specified that all structures end with a terminating NULL, even if the formatted portion of the structure contains string-reference fields and all the string fields are set to 0. • Corrected the Memory Subsystem Example, handles are now correctly created with a 'dw'. • Fixed formatting of some bit definition fields and function examples.
2.00.1	07/18/1996	Minor updates for new technology and clarifications. • Added definitions for Pentium® Pro, Burst EDO, and SDRAM. • Added clarifications to the Memory Controller Error Status.
2.1	06/16/1997	Added definition for static table interface, to allow the information to be accessed from new operating systems, see *2.1 Table Convention* on page 9. In addition: • Changed references to DMI BIOS to SMBIOS throughout; these changes are unmarked. • Added SubFunction DMI_CLEAR_EVENT_LOG2 to Function 54h - SMBIOS Control. • For those structure entries which are string numbers, changed the Value field definition of the field from Varies to STRING throughout; these changes are unmarked. • BIOS Information structure: Added support for 4-digit year and additional BIOS Characteristics via Characteristics Extension Byte 1. • System Information structure: Added Wakeup Type and UUID fields. • System Enclosure and Chassis structure: Added Bootup State, Power Supply State, Thermal State, and Security Status to allow the DMTF|Physical Container Global Table to be populated. • Processor Information structure: Voltage value can now be specified, rather than using bit-flags, and added enumeration values for Pentium® Pro, Pentium® II, and Slot 1. Also added notes to this section, indicating that the enumerated values for the structure are assigned by the DMTF. This structure was also updated to include the Cache Information handles identifying the L1, L2, and L3 caches associated with the processor. • Memory Controller Information structure: Added Enabled Error Correcting field. Also added note that this structure can never be updated to add string values, to preserve backwards compatibility. • Cache Information structure: Added Speed, Error Correction Type, Type, and Associativity fields. • Port Connector Information structure: Added enumerated values to Connector Types and Port Types. • System Slots structure: Added AGP enumeration values to Slot Type field. • BIOS Language Information structure: Added abbreviated-format for language strings and corrected example. • System Event Log structure: OEM-specific Access Methods can now be defined, added standard log header definitions, and a mechanism to allow the log entry's variable data formats to be described. Added note that this structure can never be updated to include string values, to preserve backwards compatibility. • Added Physical Memory Array, Memory Device, Memory Error Information, Memory Array Mapped Address, and Memory Device Mapped Address structures to support the population of the DMTF Enhanced Physical Memory groups. • Added Built-in Pointing Device structure to support the population of the DMTF Pointing Device group. • Added Portable Battery structure to support the population of the DMTF Portable Battery group. • Added appendices which contain a structure checklist and table-convention parsing pseudo-code.

Version	Release Date	Description

2.2	03/16/1998	The following changes were made to v2.1 of the document to produce this version:
		• Accepted all changes introduced at Version 2.1
		• Added ACPI statement-of-direction for dynamic state and event notification
		• *Table-convention is required for v2.2 and later compliance*
		• Corrected Structure Table entry point length value.
		• Added Command type 06h to the Plug-and-Play Set SMBIOS Structure function (52h).
		• Added new processor enumerations from the updated DMTF MASTER.MIF
		• System Enclosure: Added enumeration value for "Sealed-case PC", to support Net PC-type chassis'.
		• Memory Controller Information: Corrected description of how the BIOS computes the structure Length.
		• System Event Log:
		  • Added definition for end-of-log data, Event Log Type 0FFh.
		  • Added generic system-management event type; the handle of an associated probe or cooling device identifies the specific failing device.
		• Memory Error Information: Corrected structure size and offsets.
		• Portable Battery: Corrected the structure length and some of the offsets, added Smart Battery-formatted fields
		• Memory Device: Added RIMM form factor
		• Added the following new structures
		  • System Reset structure to support the population of the DMTF Automatic System Reset group.
		  • Hardware Security structure to support the population of the DMTF System Hardware Security group.
		  • System Power Control structure to support the population of the DMTF System Power Control group.
		  • Added Voltage Probe structure to support the population of the DMTF Voltage Probe group.
		  • Cooling Device structure to support the population of the DMTF Cooling Device group.
		  • Temperature Probe structure to support the population of the DMTF Temperature Probe group.
		  • Electrical Current Probe structure to support the population of the DMTF Electrical Current Probe group.
		  • Out-of-Band Remote Access structure to support the population of the DMTF Out-of-Band Remote Access group.
		  • Inactive structure type to support standard structure superset definitions.
		  • End-of-Table structure type to facilitate easier traversing of the structure data.

Table Of Contents

1. Overview

Desktop Management Interface (DMI) is a method of managing computers in an enterprise. The main component of DMI is the Management Information Format Database, or MIF. This database contains all the information about the computing system and its components. Using DMI, a system administrator can obtain the types, capabilities, operational status, installation date, and other information about the system components.

The *Desktop Management Interface Specification* and its companion `MASTER.MIF` define "manageable attributes that are expected to be supported by DMI-enabled computer systems". Many of these attributes have no standard interface to the management software, but are known by the system BIOS. The *System Management BIOS Reference Specification* provides that interface via data structures through which the system attributes are reported — see *Accessing SMBIOS Information* on page 9 for the definition of these interfaces.

1.1 Statement of Direction — Dynamic Information and Events

The current version of this specification provides an inter-operable method (see *2.1 Table Convention* on page 9) for the access of static system data. A future version of this specification will define methods compatible with the *Automatic Configuration and Power Interface Specification* (ACPI) that provide access to dynamic system data and events.

1.2 References

- *Advanced Configuration and Power Interface Specification*, Version 1.0, December 23 1996, http://www.teleport.com/~acpi
- *Desktop Management Interface Specification*, Version 2.0, March 29, 1996, www.dmtf.org/tech/specs.html
- DMTF MASTER.MIF, last edited 01 January 1998 by fdante, www.dmtf.org/tech/specs.html
- *DMTF DMI 2.0 Conformance Requirements* Version 1.0, 6 August 1997, www.dmtf.org/tech/specs.html
- *DMTF Mobile Supplement to Standard Groups* Version 1.02, 23 September 1997, www.dmtf.org/tech/apps.html
- *Smart Battery Data Specification*, Version 1.0, 15 February 1995, www.sbs-forum.org
- *Plug and Play BIOS Specification*, Version 1.0A, May 5, 1994, ftp://ftp.microsoft.com/developr/drg/Plug-and-Play/Pnpspecs
- *PCI BIOS Specification*, Version 2.1, August 26, 1994, http://www.pcisig.com/specs.html
- *"El Torito" Bootable CD-ROM Format Specification*, Version 1.0, January 25 1995, http://www.ptltd.com/techs/specs.html

1.3 Conventions Used in this Document

1. All numbers specified in this document are in decimal format unless otherwise indicated. A number followed by the letter 'h' indicates hexadecimal format; a number followed by the letter 'b' indicates binary format.

   For example, the values 10, 0Ah, and 1010b are equivalent.

2. Any value not listed in an enumerated list is reserved for future assignment by either this specification or the DMTF (depending on the list-value controlling body).

3. Most of the enumerated values defined in this specification simply track the like values specified by the DMTF — either within the DMTF's `MASTER.MIF` or the *DMTF Mobile Supplement to Standard*

Groups. Enumerated values which are controlled by the DMTF are identified within their respective subsection; additional values for these fields are assigned by the DMTF, not this specification.

2. Accessing SMBIOS Information

There are two access methods defined for the SMBIOS structures:

1. The first method, defined in v2.0 of this specification, provides the SMBIOS structures through a Plug-and-Play function interface, see *2.2 Plug-and-Play Calling Convention* on page 10.
2. A table-based method, defined in v2.1 of this specification, provides the SMBIOS structures as a packed list of data referenced by a table entry point; see *2.1 Table Convention* on page 9.

A BIOS compliant with v2.1 of this specification can provide one or both methods. A BIOS compliant with v2.2 and later of this specification must provide the table-based method and can optionally provide the Plug-and-Play function interface.

Note: An SMBIOS implementation that provides only the table-based method might cause some existing DMI browsers to no longer work.

2.1 Table Convention

The table convention, provided as an addition or alternative to the calling interface described in *2.2 Plug-and-Play Calling Convention* on page 10, allows the SMBIOS structures to also be accessed under 32-bit protected-mode operating systems such as Microsoft Windows NT*. This convention provides a searchable entry-point structure which contains a pointer to the packed SMBIOS structures residing somewhere in 32-bit physical address space.

Note: The table convention is required for SMBIOS v2.2 implementations.

2.1.1 SMBIOS Structure Table Entry Point

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

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

Offset	Name	Length	Description
00h	Anchor String	4 BYTEs	_SM_, specified as four ASCII characters (5F 53 4D 5F).
04h	Entry Point Structure Checksum	BYTE	Checksum of the Entry Point Structure (EPS). This value, when added to all other bytes in the EPS, will result in the value 00h (using 8-bit addition calculations). Values in the EPS are summed starting at offset 00h, for Entry Point Length bytes.
05h	Entry Point Length	BYTE	Length of the Entry Point Structure, starting with the Anchor String field, in bytes, currently 1Fh. *Note*: This value was incorrectly stated in v2.1 of this specification as 1Eh. Because of this, there might be v2.1 implementations that use either the 1Eh or 1Fh value, but v2.2 or later implementations must use the 1Fh value.

Offset	Name	Length	Description
06h	SMBIOS Major Version	BYTE	Identifies the major version of this specification implemented in the table structures, e.g. the value will be 0Ah for revision 10.22 and 02h for revision 2.1.
07h	SMBIOS Minor Version	BYTE	Identifies the minor version of this specification implemented in the table structures, e.g. the value will be 16h for revision 10.22 and 01h for revision 2.1.
08h	Maximum Structure Size	WORD	Size of the largest SMBIOS structure, in bytes, and encompasses the structure's formatted area and text strings. This is the value returned as StructureSize from the Plug-and-Play *Get SMBIOS Information* function.
0Ah	Entry Point Revision	BYTE	Identifies the EPS revision implemented in this structure and identifies the formatting of offsets 0Bh to 0Fh, one of: 00h Entry Point is based on SMBIOS 2.1 definition, formatted area is reserved and set to all 00h. 01h-FFh Reserved for assignment via this specification
0Bh - 0Fh	Formatted Area	5 BYTEs	The value present in the Entry Point Revision field defines the interpretation to be placed upon these 5 bytes.
10h	Intermediate anchor string	5 BYTEs	_DMI_, specified as five ASCII characters (5F 44 4D 49 5F). Note: This field is paragraph-aligned, to allow legacy DMI browsers to find this entry point within the SMBIOS Entry Point Structure.
15h	Intermediate Checksum	BYTE	Checksum of Intermediate Entry Point Structure (IEPS). This value, when added to all other bytes in the IEPS, will result in the value 00h (using 8-bit addition calculations). Values in the IEPS are summed starting at offset 10h, for 0Fh bytes.
16h	Structure Table Length	WORD	Total length of SMBIOS Structure Table, pointed to by the Structure Table Address, in bytes.
18h	Structure Table Address	DWORD	The 32-bit physical starting address of the read-only SMBIOS Structure Table, which can start at any 32-bit address. This area contains all of the SMBIOS structures fully packed together. These structures can then be parsed to produce exactly the same format as that returned from a Get SMBIOS Structure function call.
1Ch	Number of SMBIOS Structures	WORD	Total number of structures present in the SMBIOS Structure Table. This is the value returned as NumStructures from the Get SMBIOS Information function.
1Eh	SMBIOS BCD Revision	BYTE	Indicates compliance with a revision of this specification. It is a BCD value where the upper nibble indicates the major version and the lower nibble the minor version. For revision 2.1, the returned value is 21h. If the value is 00h, only the Major and Minor Versions in offsets 6 and 7 of the Entry Point Structure provide the version information.

2.2 Plug-and-Play Calling Convention

To prevent the proliferation of interfaces for accessing information embedded in the System BIOS, the *System Management BIOS Reference Specification* will follow the System Device Node model used by Plug and Play, and use Plug and Play BIOS functions to access DMI information. Plug and Play functions 50h-5Fh have been assigned to the SMBIOS BIOS Interface.

Each of the SMBIOS BIOS Plug-and-Play functions is available both in real-mode and 16-bit protected-mode. A function called in 16-bit protected-mode supports both 16-bit and 32-bit stack segments.

2.2.1 SMBIOS Functions

This table defines the current SMBIOS Functions.

SMBIOS Function	Function Number	Description	Required/Optional
GET_DMI_INFORMATION	50h	Returns the Number of Structures, the Size of the Largest Structure, and the SMBIOS Revision.	Required for calling interface
GET_DMI_STRUCTURE	51h	Copies the information for the specified Structure into the buffer specified by the caller.	Required for calling interface
SET_DMI_STRUCTURE	52h	Copies the information for the specified SMBIOS structure from the buffer specified by the caller.	Optional
GET_DMI_STRUCTURE_ CHANGE_INFO	53h	Returns the SMBIOS Structure Change Information into a 16-byte buffer specified by the caller.	Required for Dynamic Structure-change Notification Support
DMI_CONTROL	54h	Controls a system action	Optional
GET_GPNV_INFORMATION	55h	Returns information about the General Purpose Non-Volatile Storage Area	Required for GPNV Support
READ_GPNV_DATA	56h	Reads the entire specified GPNV contents into a buffer specified by the caller.	Required for GPNV Support
WRITE_GPNV_DATA	57h	Copies the contents of the user specified buffer into the GPNV. The function causes the entire specified GPNV to be updated.	Required for GPNV Support
Reserved for Future Use	58h-5Fh	Reserved, will return DMI_FUNCTION_NOT_ SUPPORTED.	Reserved

2.2.2 Error Return Codes

After the call has been made, the following return codes are available in the AX Register.

Return Code	Value	Description
DMI_SUCCESS	00h	Function Completed Successfully
DMI_UNKNOWN_FUNCTION	81h	Unknown, or invalid, function number passed
DMI_FUNCTION_NOT_SUPPORTED	82h	The function is not supported on this system
DMI_INVALID_HANDLE	83h	SMBIOS Structure number/handle passed is invalid or out of range.
DMI_BAD_PARAMETER	84h	The function detected invalid parameter or, in the case of a “Set SMBIOS Structure” request, detected an invalid value for a to-be-changed structure field.
DMI_INVALID_SUBFUNCTION	85h	The SubFunction parameter supplied on a SMBIOS Control function is not supported by the system BIOS.
DMI_NO_CHANGE	86h	There are no changed SMBIOS structures pending notification.
DMI_ADD_STRUCTURE_FAILED	87h	Returned when there was insufficient storage space to add the desired structure.
DMI_READ_ONLY	8Dh	A “Set SMBIOS Structure” request failed because one or more of the to-be-changed structure fields are read-only.
DMI_LOCK_NOT_SUPPORTED	90h	The GPNV functions do not support locking for the specified GPNV handle.
DMI_CURRENTLY_LOCKED	91h	The GPNV lock request failed - the GPNV is already locked.
DMI_ INVALID_LOCK	92h	The caller has failed to present the predefined GPNVLock value which is expected by the BIOS for access of the GPNV area.

2.2.3 SMBIOS Structure Access Interface

2.2.3.1 Function 50h – Get SMBIOS Information

Synopsis:

```
short FAR (*entryPoint)(
  short Function,                          /* PnP BIOS Function 50h */
  unsigned char FAR *dmiBIOSRevision,      /* Revision of the SMBIOS Extensions */
  unsigned short FAR *NumStructures,       /* Max. Number of Structures the BIOS will return */
  unsigned short FAR *StructureSize,       /* Size of largest SMBIOS Structure */
  unsigned long FAR *dmiStorageBase,       /* 32-bit physical base address for memory-mapped */
                                           /*  SMBIOS data */
  unsigned short FAR *dmiStorageSize,      /* Size of the memory-mapped SMBIOS data */
  unsigned short BiosSelector );           /* PnP BIOS readable/writable selector */
```

Description:

Required for SMBIOS Calling Interface Support. This function will return the revision of the SMBIOS Extensions and the maximum number of SMBIOS structures that the system BIOS will return information for in *NumStructures*. These structures represent the SMBIOS information that is embedded in the System BIOS. In addition to the number of structures, the system BIOS will return the size, in bytes, of the largest SMBIOS structure (and all of its supporting data) in *StructureSize*. This information can be utilized by the system software to determine the amount of memory required to get all of the SMBIOS structures. *Note*: The system BIOS may return a value that is larger than the actual largest SMBIOS

structure to facilitate hot docking or other dynamic SMBIOS information. The BIOS may also return fewer than *NumStructures* when the structures are retrieved using Function 51h. If the BIOS does not support SMBIOS calling interface capability, DMI_FUNCTION_NOT_SUPPORTED (82h) will be returned.

The *dmiBIOSRevision* parameter indicates compliance with a revision of this specification. It is a BCD value where the upper nibble indicates the major version and the lower nibble the minor version. For revision 2.0 the returned value will be 20h.

dmiStorageBase is updated by the BIOS call with the paragraph-aligned, 32-bit absolute physical base address of any memory-mapped SMBIOS structure information. If non-zero, this value allows the caller to construct a 16-bit data segment descriptor with a limit of *dmiStorageSize* and read/write access for subsequent input to functions 51h to 54h. If *dmiStorageBase* is 0, protected-mode mapping is not required.

In addition, *dmiStorageSize* identifies the *dmiWorkBuffer* size for input to function 52h and the *Data* buffer size for function 54h's DMI_CLEAR_EVENT_LOG2 sub-function. ***Note***: This feature is SMBIOS version-specific; for v2.0 implementations, the value of dmiStorageSize has no meaning if dmiStorageBase is 0. In this case, the buffer-sizing is provided by (*NumStructures* * *StructureSize*).

The *BiosSelector* parameter enables the system BIOS, if necessary, to update system variables that are contained in the system BIOS memory space. If this function is called from protected mode, the caller must create a data segment descriptor using the 16-bit Protected Mode data segment base address specified in the Plug and Play Installation Check data structure, a limit of 64KB, and the descriptor must be read/write capable. If this function is called from real mode, *BiosSelector* should be set to the Real mode 16-bit data segment address as specified in the Plug and Play Installation Check Structure. Refer to section 4.4 of the Plug and Play BIOS Specification revision 1.0a for more information on the Plug and Play Installation Check Structure and the elements that make up the structure.

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

Returns:

If successful - DMI_SUCCESS
If an Error (Bit 7 set) or a Warning occurred the Error Code will be returned in AX, the FLAGS and all other registers will be preserved.

Example:

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

```
push          BiosSelector
push          segment/selector of dmiStorageSize       ; Pointer to DMIStorageSize
push          offset of dmiStorageSize
push          segment/selector of dmiStorageBase       ; Pointer to DMIStorageBase
push          offset of dmiStorageBase
push          segment/selector of StructureSize        ; Pointer to StructureSize
push          offset of StructureSize
push          segment/selector of NumStructures        ; Pointer to NumStructures
push          offset NumStructures
push          segment/selector of dmiBIOSRevision      ; Pointer to DMIBIOSRevision
push          offset dmiBIOSRevision
push          GET_DMI_INFORMATION                      ; Function number, 50h
call          FAR PTR entryPoint
add           sp, 24                                   ; Clean up stack
cmp           ax, DMI_SUCCESS                          ; Function completed successfully?
jne           error
```

2.2.3.2 Function 51h – Get SMBIOS Structure

Synopsis:

```
short FAR (*entryPoint)(
  short Function,                          /* PnP BIOS Function 51h */
  unsigned short FAR *Structure,           /* Structure number/handle to retrieve*/
  unsigned char FAR *dmiStrucBuffer,       /* Pointer to buffer to copy structure data to */
  unsigned short dmiSelector,              /* SMBIOS data read/write selector */
  unsigned short BiosSelector );           /* PnP BIOS readable/writable selector */
```

Description:

Required for SMBIOS Calling Interface Support. This function will copy the information for the specified SMBIOS Structure into the buffer specified by the caller. The *Structure* argument is a pointer to the unique SMBIOS Structure number (handle). If *Structure* contains zero, the system BIOS will return the first SMBIOS Structure. The *dmiStrucBuffer* argument contains the pointer to the caller's memory buffer. If the function returns either DMI_SUCCESS or DMI_INVALID_HANDLE, *Structure* is updated with either the next sequential structure handle or the end-of-list indicator 0FFFFh.

The protected-mode read/write selector *dmiSelector* has base equal to *dmiStorageBase* and limit of at least *dmiStorageSize* — so long as the *dmiStorageBase* value returned from Function 50h was non-zero.

The *BiosSelector* parameter enables the system BIOS, if necessary, to update system variables that are contained in the system BIOS memory space. If this function is called from protected mode, the caller must create a data segment descriptor using the 16-bit Protected Mode data segment base address specified in the Plug and Play Installation Check data structure, a limit of 64KB, and the descriptor must be read/write capable. If this function is called from real mode, *BiosSelector* should be set to the Real mode 16-bit data segment address as specified in the Plug and Play Installation Check Structure. Refer to section 4.4 of the Plug and Play BIOS Specification revision 1.0a for more information on the Plug and Play Installation Check Structure and the elements that make up the structure.

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

Returns:

If successful - DMI_SUCCESS
If an Error (Bit 7 set) or a Warning occurred, the Error Code will be returned in AX, the FLAGS and all other registers will be preserved

Example:

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

```
push          BiosSelector
push          dmiSelector
push          segment/selector of dmiStrucBuffer   ; Pointer to dmiStrucBuffer
push          offset of dmiStrucBuffer
push          segment/selector of Structure        ; Pointer to Structure
push          offset of Structure
push          GET_ DMI_STRUCTURE                   ; Function number, 51h
call          FAR PTR entryPoint
add           sp, 14                               ; Clean up stack
cmp           ax, DMI_SUCCESS                      ; Function completed successfully?
jne           error
```

2.2.3.3 Function 52h – Set SMBIOS Structure

Synopsis:

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

Description:

Optional. This function will set the SMBIOS structure identified by the type (and possibly handle) found in the SMBIOS structure header in the buffer pointed to by *dmiDataBuffer*. Values that the BIOS allows to be set in the supplied structure will either be updated by the call, or will cause the BIOS to perform some defined action (such as enabling a hardware option, etc.).

Unless otherwise specified, all structures and structure values defined in Section 3, *SMBIOS Structures*, are read-only and cannot be set. Attempts to set these structures will return a DMI_READ_ONLY error. A structure field that is composed of read/write and read-only subfields can still be set -- so long as the read-only portion of the field is unmodified. Attempting to write to a read-only subfield will also cause a DMI_READ_ONLY to be returned.

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

Offset	Field	Length	Description
00h	*Command*	BYTE	Identifies the structure-setting operation to be performed, one of: 00h A single byte of information is to be changed in the structure identified by StructureHeader 01h A word (two bytes) of information is to be changed in the structure identified by StructureHeader 02h A double-word (four bytes) of information is to be changed in the structure identified by StructureHeader 03h The structure identified by StructureHeader is to be added to the SMBIOS structure pool 04h The structure identified by StructureHeader is to be deleted from the SMBIOS structure pool 05h A string's value is to be changed in the structure identified by StructureHeader. 06h A block of information (other than byte, word, or dword in size) is to be changed in the structure identified by StructureHeader. 07h-0FFh Reserved for future assignment by this specification.
01h	*FieldOffset*	BYTE	For a structure change Command, identifies the starting offset within the changed structure's fixed data of the to-be-changed item. For a string-value change Command, identifies the offset within the structure's fixed data associated with the string's "number". This field is ignored for all other Commands.
02h	*ChangeMask*	DWORD	For a fixed-length structure-change Command, identifies the ANDing mask to be applied to the existing structure data prior to applying the ChangeValue. The number of significant bytes within this area is defined by the Command. This field is ignored for all other Commands.

Offset	Field	Length	Description
06h	*ChangeValue*	DWORD	For a fixed-length structure-change Command, identifies the data value to be ORed with the existing structure data – after applying the ChangeMask. The number of significant bytes within this area is defined by the Command. This field is ignored for all other Commands.
0Ah	*DataLength*	WORD	For a structure-add Command, identifies the full length of the to-be-added structure. The length includes the structure header, the fixed-length portion of the structure, and any string data which accompanies the added structure – including all null-terminators. For a string-value change Command, identifies the length of the string data (including the null-terminator); if the length is 1 (indicating that only the null-terminator is provided), the current string's data is deleted so long as the string's data-access rights are met. For a variable-length block change Command, identifies the length of the contiguous data block to be changed. This field is ignored for all other Commands.
0Ch	*StructureHeader*	4 BYTEs	Contains the structure header (see *Structure Header Format* on page 26) of the structure to be added, changed, or deleted.
10h	*StructureData*	Var	For a structure-add Command, contains the data to be associated with the SMBIOS Structure identified by the StructureHeader. For a string-value change Command, contains the string's data (the number of characters is identified by DataLength). For a variable-length block change Command, contains the block's data (the number of bytes is identified by DataLength). This field is ignored for all other Commands.

The *dmiWorkBuffer* parameter references a work buffer for use by the BIOS in performing the request; the contents of the buffer are destroyed by the BIOS' processing. This work buffer must be read/write and sized to hold the entire SMBIOS structure pool, based on the information returned by *Function 50h – Get SMBIOS Information* (see page 12) plus the size of any structure to be added by the request. For SMBIOS v2.0 implementations, the pool size is specified by the maximum of (*StructureSize* * *NumStructures*) and (when *dmiStorageBase* is non-zero) *dmiStorageSize*; for v2.1 and later implementations, the pool size is specified by *dmiStorageSize*.

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

Control is defined as:

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

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

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

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

be read/write capable. If this function is called from real mode, BiosSelector should be set to the Real mode 16-bit data segment address as specified in the Plug and Play Installation Check Structure. Refer to section 4.4 of the Plug and Play BIOS Specification revision 1.0a for more information on the Plug and Play Installation Check Structure and the elements that make up the structure.

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

Note: If the system BIOS supports structure-change notification, a structure-change event will be issued by the BIOS upon its successful completion of a structure-setting (rather than validation) function call. See *Structure Change Notification Interface* on page 17 for more information.

Returns:

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

Errors:

DMI_BAD_PARAMETER	A parameter contains an invalid or unsupported value.
DMI_READ_ONLY	A parameter is read-only and differs from the present value – an attempt was made to modify a read-only value.
DMI_ADD_STRUCTURE_FAILED	The desired structure could not be added due to insufficient storage space.
DMI_INVALID_HANDLE	For an add (03h) *Command*, the structure handle present in the *StructureHeader* already exists or, for a change (00h to 02h and 05h) or delete (04h) *Command*, the structure handle does not exist.

Example:

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

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

2.2.4 Structure Change Notification Interface

Certain classes of systems may provide the capability for the addition or removal of system devices while the system unit is powered on, such as inserting a Notebook unit into a Docking Station. System BIOS support is necessary for providing SMBIOS Structure Change Notification accessible to system software so that when devices are added or removed the system software will comprehend any changes in the SMBIOS Structures. Structure Change Notification can be implemented as either a polled method or as asynchronous Plug-and-Play events. For information on how Plug-and-Play event notification is accessed, see section 4.6 of the Plug and Play BIOS Specification revision 1.0a.

When system software is notified on an event by either mechanism, it can then call the BIOS runtime function (Plug and Play BIOS Function 3 - Get Event) to get the type of event. In addition to the events defined in the Plug and Play BIOS Specification, the following event has been defined.

Note: Some DMI structure values might be inherently changing (e.g. an OEM-specific structure which returns system temperature and voltage values). Due to the frequency of the values' change, the BIOS might not return Structure Change status for this type of structure.

DMI_STRUCTURE_CHANGE_EVENT 7FFFh

This message indicates that there has been a change in the DMI Information being maintained by the System BIOS. Upon receiving a DMI_STRUCTURE_CHANGE_EVENT, system software can call the BIOS runtime function 53h (Get Structure Change Information) to determine the exact cause of the SMBIOS structure-change event.

2.2.4.1 Function 53h – Get Structure Change Information

Synopsis:

```
short FAR (*entryPoint)(
  short Function,                          /* PnP BIOS Function 53h */
  unsigned char FAR *dmiChangeStructure,   /* Pointer to SMBIOS Change structure */
  unsigned short dmiSelector,              /* SMBIOS data read/write selector */
  unsigned short BiosSelector );           /* PnP BIOS readable/writable selector */
```

Description:

Required for SMBIOS Dynamic Structure Change Notification Support. This function will allow system software to get information about what type of SMBIOS structure-change occurred. The SMBIOS structure-change information will be returned in the 16-byte memory buffer pointed to by *dmiChangeStructure* in the following format:

Field	Offset	Length	Value
SMBIOS Change Status	00h	BYTE	ENUM
SMBIOS Change Type	01h	BYTE	Bit Field
SMBIOS Structure Handle	02h	WORD	Varies
Reserved	04h-0Fh	12 BYTEs	00h

SMBIOS Change Status:

00h	No Change
01h	Other
02h	Unknown
03h	Single SMBIOS Structure Affected
04h	Multiple SMBIOS Structures Affected
05h - 0FFh	Reserved

SMBIOS Change Type:

Bit 0	One or more structures was changed, when 1.
Bit 1	One or more structures was added, when 1. See "*Function 52h – Set DMI Structure*" for information about adding SMBIOS structures.
Byte 2:7	Reserved, must be 0

If DMI Change Status 03h (Single Structure Affected) is returned, the number (or handle) of the affected structure is present in the "DMI Structure Handle" field; DMI Change Type identifies whether the structure was changed (01h) or added (02h).

If DMI Change Status 04h (Multiple DMI Structures Affected) is returned, the caller must enumerate all the structures to determine what was changed and/or added. DMI Change Type identifies whether multiple structures were changed (01h), multiple structures were added (02h), or structures were both changed and added (03h).

The DMI Change Status Byte remains valid until Function 53h is called. The calling of Function 53h will reset the DMI Change Status Byte to zero. If the call is issued in the absence of a DMI event, the function returns error code 86h (DMI_NO_CHANGE).

The protected-mode read/write selector *dmiSelector* has base equal to *dmiStorageBase* and limit of at least *dmiStorageSize* — so long as the *dmiStorageBase* value returned from Function 50h was non-zero.

The *BiosSelector* parameter enables the system BIOS, if necessary, to update system variables that are contained in the system BIOS memory space. If this function is called from protected mode, the caller must create a data segment descriptor using the 16-bit Protected Mode data segment base address specified in the Plug and Play Installation Check data structure, a limit of 64KB, and the descriptor must be read/write capable. If this function is called from real mode, *BiosSelector* should be set to the Real mode 16-bit data segment address as specified in the Plug and Play Installation Check Structure. Refer to section 4.4 of the Plug and Play BIOS Specification revision 1.0a for more information on the Plug and Play Installation Check Structure and the elements that make up the structure.

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

Returns:

If successful - DMI_SUCCESS
If an Error (Bit 7 set) or a Warning occurred the Error Code will be returned in AX, the FLAGS and all other registers will be preserved

Example:

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

```
push          BiosSelector
push          dmiSelector
push          segment/selector of dmiChangeStructure
push          offset of dmiChangeStructure
push          GET_ DMI_STRUCTURE_CHANGE_INFO       ; Function number, 53h
call          FAR PTR entryPoint
add           sp, 10                               ; Clean up stack
cmp           ax, DMI_SUCCESS                      ; Function completed successfully?
jne           error
```

2.2.5 Control Interface

2.2.5.1 Function 54h – SMBIOS Control

Synopsis:

```
short FAR (*entryPoint)(
  short Function,                  /* PnP BIOS Function 54h */
  short SubFunction,               /* Defines the specific control operation */
  void FAR *Data,                  /* Input/output data buffer, SubFunction specific */
  unsigned char Control,           /* Conditions for setting the structure */
  unsigned short dmiSelector,      /* SMBIOS data read/write selector */
  unsigned short BiosSelector );   /* PnP BIOS readable/writeable selector */
```

Description:

Optional. This function provides the interface to perform implementation-specific functions for the system, as defined by the *SubFunction* parameter and its (optional) *Data* values.

SubFunction	Name	Description
0000h	DMI_CLEAR_EVENT_LOG	Clears the event log as described in *System Event Log (Type 15)* on page 52. The *Data* parameter is reserved and must be set to

SubFunction	Name	Description
		0.
0001h	DMI_CONTROL_LOGGING	Data points to a 2-word (4-byte) buffer that describes how to control event logging – see 2.2.5.1.1 for bit-wise definitions. The first word (offset 0:1) identifies the ANDing mask to be applied to the existing log-control value prior to ORing the second word (offset 2:3). The second word is modified by the BIOS to contain the log-control value on entry to this function.
0002h	DMI_CLEAR_EVENT_LOG2	Clears the event log as described in *System Event Log (Type 15)* on page 52. The *Data* parameter is the 32-bit physical address of a work buffer needed to perform this operation. The buffer must be read/write and sized to hold *dmiStorageSize* bytes. The contents of the buffer are destroyed by the BIOS' processing. This sub-function is defined for v2.1 and later implementations of this specification and is preferred over the DMI_CLEAR_EVENT_LOG (0000h) sub-function.
0003h-3FFFh	Reserved	Reserved for future definition by this specification.
4000h-7FFFh	Reserved for BIOS vendor	Available for use by the BIOS vendor.
8000h-FFFFh	Reserved for system vendor	Available for use by the system vendor.

Note: A BIOS might support the Log Control function but not support all the *SubFunction* values.

The *Control* flag provides a mechanism for indicating to the BIOS whether the operation is to be performed immediately, or if this is a check to validate the operation's availability and/or data.

Control is defined as:

Bit 0	0 = Do not perform the operation, but validate its parameters. 1 = Perform the operation immediately.
Bits 1:7	Reserved, must be 0.

If bit 0 of *Control* is 0, then the *SubFunction* and contents of *Data* are checked for validity. If any are not valid, then the function returns DMI_BAD_PARAMETER. Validity checking is useful to determine if the BIOS supports a specific DMI Control *SubFunction*.

The protected-mode read/write selector *dmiSelector* has base equal to *dmiStorageBase* and limit of at least *dmiStorageSize* — so long as the *dmiStorageBase* value returned from Function 50h was non-zero.

The *BiosSelector* parameter enables the system BIOS, if necessary, to update system variables that are contained in the system BIOS memory space. If this function is called from protected mode, the caller must create a data segment descriptor using the 16-bit Protected Mode data segment base address specified in the Plug and Play Installation Check data structure, a limit of 64KB, and the descriptor must be read/write capable. If this function is called from real mode, BiosSelector should be set to the Real mode 16-bit data segment address as specified in the Plug and Play Installation Check Structure. Refer to section 4.4 of the Plug and Play BIOS Specification revision 1.0a for more information on the Plug and Play Installation Check Structure and the elements that make up the structure.

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

Returns:

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

Errors:

DMI_BAD_PARAMETER The *Data* contents were not valid for the requested SubFunction.

DMI_INVALID_SUBFUNCTION The *SubFunction* requested is not supported by the system BIOS.

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

```
push          BiosSelector
push          dmiSelector
push          Control
push          segment/selector of Data            ; pointer to SubFunction data
push          offset of Data
push          SubFunction
push          DMI_CONTROL                         ; Function number, 54h
call          FAR PTR entryPoint
add           sp, 14                              ; clean stack
cmp           ax, DMI_SUCCESS                     ; Successful?
jne           error                               ; No, go handle error
```

2.2.5.1.1 DMI_CONTROL_LOGGING Control Word

Word Bit Position	Meaning if Set
0	Enable Event Logging (overall)
1	Enable Correctable Memory Error Events' Logging
2	Disable the logging of POST errors
3 - 15	Reserved for future assignment by this specification, set to 0.

2.2.6 General Purpose Nonvolatile Storage Interface

A General-Purpose NonVolatile (GPNV) area is a persistent general-purpose storage area managed by the System Management BIOS. Multiple GPNV areas can be supported by a particular BIOS implementation. The size, format and location of a GPNV are not defined by this specification nor is the number of GPNV areas — these attributes are OEM-specific.

A GPNV storage area is not a requirement for a System Management BIOS. It is one method that might be used to store the System Event Log (see section 3.2.16, page 52). A GPNV storage area is not necessarily dedicated to the System Management functions of the BIOS, it can also be used by other services which require non-volatile storage.

A *Handle* parameter is passed into the GPNV function calls to specify which GPNV area is to be accessed. The *Handle* for the first GPNV area is 0, with remaining GPNV areas identified by *Handle* values 1, 2, 3... *n,* where (*n+1)* is the total number of GPNV areas supported by a particular BIOS implementation.

A *GPNVLock* parameter provides a mechanism for cooperative use of the GPNV. The *GPNVLock* value is set on a Read GPNV request (function 56h) and cleared on a Write GPNV request (function 57h). The BIOS compares the value of the *GPNVLock* which is set on a Read GPNV request with the value of the *GPNVLock* passed as a parameter into the GPNV Write request — if they match, the GPNV Write request succeeds and the GPNV data area will be updated on completion of the GPNV Write; if the lock values do not match, the BIOS does not update the GPNV area and DMI_CURRENTLY_LOCKED is returned. *Note*: GPNV locks are held until unlocked, even through system power and reboot cycles. The method used to preserve the GPNV Locks through boot cycles is left up to the system designer.

A BIOS might choose to "hide" a GPNV area by defining a special lock value which is required to access the area. In this case, the special *GPNVLock* value must be supplied with the GPNV read and write requests or the function is failed by the BIOS with DMI_INVALID_LOCK.

A lock set request *succeeds* when there is no outstanding lock set at the time that the Read GPNV request (Function 56h) is made. A lock set request *fails* when there is already a lock set as the result of

a previous Read GPNV request (which has not yet been cleared with a Function 57h Write GPNV request) or when a predefined lock value is required in order to access a particular GPNV area and the *GPNVLock* value provided by the caller does not match the required value.

The BIOS makes no attempt to enforce mutually-exclusive access to the GPNV — it is up to callers of GPNV Read to ensure unique *GPNVLock* values (e.g. process ID).

2.2.6.1 Function 55H – Get General-Purpose NonVolatile Information

Synopsis:

```
short FAR (*entryPoint)(
  short Function,                    /* PnP BIOS Function 55h */
  unsigned short FAR *Handle,        /* Identifies which GPNV to access */
  unsigned short FAR *MinGPNVRWSize, /* Minimum buffer size in bytes for accessing GPNV */
  unsigned short FAR *GPNVSize,      /* Size allocated for GPNV within the R/W Block */
  unsigned long FAR *NVStorageBase,  /* 32-bit  physical base address for... */
                                     /* ... mem. mapped nonvolatile storage media */
  unsigned short BiosSelector );     /* PnP BIOS readable/writable selector */
```

Description: *Required for GPNV support.* This function returns information about a General Purpose NonVolatile (GPNV) area. The *Handle* argument is a pointer to a number that identifies which GPNV's information is requested, a value of 0 accesses the first (or only) area.
On return:

**Handle* is updated either with the handle of the next GPNV area or, if there are no more areas, 0FFFFh. GPNV handles are assigned sequentially by the system, from 0 to the total number of areas (minus 1).

**MinGPNVRW Size* is updated with the minimum size, in bytes, of any buffer used to access this GPNV area. For a Flash based GPNV area, this would be the size of the Flash block containing the actual GPNV.

**GPNVSize* is updated with the size, in bytes, of this GPNV area (which is less than or equal to the *MinGPNVRWSize* value).

*NVStorageBase is updated with the paragraph-aligned, 32-bit absolute physical base address of this GPNV. If non-zero, this value allows the caller to construct a 16-bit data segment descriptor with a limit of MinGPNVRWSize and read/write access. If the value is 0, protected-mode mapping is not required for this GPNV.

Returns:

If successful - DMI_SUCCESS
If an Error (Bit 7 set) or a Warning occurred the Error Code will be returned in AX, the FLAGS and all other registers will be preserved

Example:

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

```
push          BiosSelector
push          segment/selector of NVStorageBase
push          offset of NVStorageBase
push          segment/selector of GPNVSize
push          offset of GPNVSize
push          segment/selector of MinGPNVRWSize
push          offset of MinGPNVRWSize
push          segment/selector of Handle
push          offset of Handle
push          GET_GPNV_INFORMATION          ; Function number, 55h
call          FAR PTR entryPoint
add           sp, 20                        ; Clean up stack
cmp           ax, DMI_SUCCESS               ; Function completed successfully?
jne           error
```

2.2.6.2 Function 56H – Read General-Purpose NonVolatile Data

Synopsis:

```
short FAR (*entryPoint)(
  short Function,                   /* PnP BIOS Function 56h */
  unsigned short Handle,            /* Identifies which GPNV is to be read */
  unsigned char FAR *GPNVBuffer,    /* Address of buffer in which to return GPNV */
  short FAR *GPNVLock,              /* Lock value */
  unsigned short GPNVSelector,      /* Selector for GPNV Storage */
  unsigned short BiosSelector );    /* PnP BIOS readable/writable selector */
```

Description: *Required for GPNV support.* This function is used to read an entire GPNV area into the buffer specified by *GPNVBuffer.* It is the responsibility of the caller to ensure that *GPNVBuffer* is large enough to store the entire GPNV storage block - this buffer must be at least the *MinGPNVRWSize* returned by Function 55h - Get GPNV Information*.* The *Handle* argument identifies the specific GPNV to be read*.* On a successful read of a GPNV area, that GPNV area will be placed in the *GPNVBuffer* beginning at offset 0. The protected-mode selector *GPNVSelector* has base equal to *NVStorageBase* and limit of at least *MinGPNVRWSize* — so long as the *NVStorageBase* value returned from Function 55h was non-zero*.*

Passing a *GPNVLock* value of -1 to the GPNV Read causes the *GPNVLock* value to be ignored — in this case the underlying logic makes no attempt to store a lock value for comparison with lock values passed into GPNV Write. Any value provided for *GPNVLock* besides -1 is accepted as a valid value for a lock request.

Returns:

If the GPNV lock is supported and the lock set request succeeds, the caller's *GPNVLock* is set to the value of the current lock and the function returns DMI_SUCCESS.

If the GPNV request fails, one of the following values is returned:

- DMI_ LOCK_NOT_SUPPORTED
- DMI_ INVALID_LOCK
- DMI_ CURRENTLY_LOCKED

For return status codes DMI_SUCCESS, DMI_LOCK_NOT_SUPPORTED and DMI_CURRENTLY_LOCKED, the GPNV Read function returns the current contents of the GPNV associated with *Handle* as the first *GPNVSize* bytes within *GPNVBuffer*, starting at offset 0. If a lock request fails with DMI_CURRENTLY_LOCKED status, the caller's GPNVLock will be set to the value of the current lock*.*

Example:

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

```
push          BiosSelector
push          GPNVSelector
push          segment/selector of  GPNVLock
push          offset of GPNVLock
push          segment/selector of GPNVBuffer
push          offset of GPNVBuffer
push          Handle
push          READ_GPNV_DATA              ; Function number, 56h
call          FAR PTR entryPoint
add           sp, 16                      ; Clean up stack
cmp           ax, DMI_SUCCESS             ; Function completed successfully?
jne           error
```

2.2.6.3 Function 57H – Write General-Purpose NonVolatile Data

Synopsis:

```
short FAR (*entryPoint)(
  short Function,                   /* PnP BIOS Function 57h */
  unsigned short Handle,            /* Identifies which GPNV is to be written */
  unsigned char FAR *GPNVBuffer,    /* Address of buffer containing complete GPNV to write*/
  short GPNVLock,                   /* Lock value */
  unsigned short GPNVSelector,      /* Selector for GPNV Storage */
  unsigned short BiosSelector );    /* PnP BIOS readable/writable selector */
```

Description: *Required for GPNV support.* This function is used to write an entire GPNV from the *GPNVBuffer* into the nonvolatile storage area. The *Handle* argument identifies the specific GPNV to be written. The protected-mode selector *GPNVSelector* has base equal to *NVStorageBase* and limit of at least *MinGPNVRWSize* — so long as the *NVStorageBase* value returned from *Get GPNV Information* was non-zero. The caller should first call *Read GPNV Data* (with a lock) to get the current area contents, modify the data, and pass it into this function — this ensures that the *GPNVBuffer* which is written contains a complete definition for the entire GPNV area. If the BIOS uses some form of block erase device, the caller must also allocate enough buffer space for the BIOS to store all data from the part during the reprogramming operation, not just the data of interest.

The data to be written to the GPNV selected by *Handle* must reside as the first *GPNVSize* bytes of the *GPNVBuffer*. *Note*: The remaining (*MinGPNVRWSize-GPNVSize)* bytes of the *GPNVBuffer* area are used as a scratch-area by the BIOS call in processing the write request; the contents of that area of the buffer are destroyed by this function call.

The *GPNVLock* provides a mechanism for cooperative use of the GPNV, and is set during a GPNV Read (Function 56h). If the input *GPNVLock* value is -1 the caller requests a forced write to the GPNV area, ignoring any outstanding *GPNVLock*. If the caller is not doing a forced write, the value passed in *GPNVLock* to the GPNV Write must be the same value as that (set and) returned by a previous GPNV Read (Function 56h).

Returns:

The GPNV Write function returns a value of DMI_ LOCK_NOT_SUPPORTED when a *GPNVLock* value other than -1 is specified and locking is not supported. A return status of DMI_ CURRENTLY_LOCKED indicates that the call has failed due to an outstanding lock on the GPNV area which does not match the caller's *GPNVLock* value. Any outstanding *GPNVLock* value (which was set by a previous *Function 56H – Read General-Purpose NonVolatile Data*) gets cleared on a successful write of the GPNV.

Example:

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

```
push          BiosSelector
push          GPNVSelector
push          GPNVLock
push          segment/selector of GPNVBuffer
push          offset of GPNVBuffer
push          Handle
push          WRITE_GPNV_DATA             ; Function number, 57h
call          FAR PTR entryPoint
add           sp, 14                      ; Clean up stack
cmp           ax, DMI_SUCCESS             ; Function completed successfully?
jne           error
```

3. SMBIOS Structures

The total number of structures can be obtained either through the *Get SMBIOS Information* function (see 2.2.3.1 on page 12) or from the SMBIOS Entry Point Structure (see 2.1 *Table Convention* on page 9). The System Information is presented to an application as a set of structures that are obtained by either calling the *Get SMBIOS Structure* function once per structure (see 2.2.3.2 on page 14) or by traversing the SMBIOS structure table referenced by the SMBIOS Entry Point Structure (see 2.1 *Table Convention* on page 9).

3.1 Structure Standards

Each SMBIOS structure has a formatted section and an optional unformed section. The formatted section of each structure begins with a 4-byte header. Remaining data in the formatted section is determined by the structure type, as is the overall length of the formatted section.

3.1.1 Structure Evolution and Usage Guidelines

As the industry evolves, the structures defined in this specification will evolve. To ensure that the evolution occurs in a nondestructive fashion, the following guidelines must be followed:

1. If a new field is added to an existing structure, that field is added at the end of the formatted area of that structure and the structure's *Length* field is increased by the new field's size.
2. Any software which interprets a structure shall use the structure's *Length* field to determine the formatted area size for the structure rather than hardcoding or deriving the Length from a structure field.
3. Each structure shall be terminated by a double-null (0000h), either directly following the formatted area (if no strings are present) or directly following the last string. This includes system- and OEM-specific structures and allows upper-level software to easily traverse the structure table. See below for structure-termination examples.
4. The unformed section of the structure is used for passing variable data such as text strings, see *3.1.3 Text Strings* for more information.
5. When an enumerated field's values are controlled by the DMTF, new values can be used as soon as they are defined by the DMTF without requiring an update to this specification.

Example: BIOS Information with strings

```
BIOS_Info      LABEL BYTE
db      0                              ; Indicates BIOS Structure Type
db      13h                            ; Length of information in bytes
dw      ?                              ; Reserved for handle
db      01h                            ; String 1 is the Vendor Name
db      02h                            ; String 2 is the BIOS version
dw      0E800h                         ; BIOS Starting Address
db      03h                            ; String 3 is the BIOS Build Date
db      1                              ; Size of BIOS ROM is 128K (64K * (1 + 1))
dq      BIOS_Char                      ; BIOS Characteristics
db      0                              ; BIOS Characteristics Extension Byte 1
db      'System BIOS Vendor Name',0    ;
db      '4.04',0                       ;
db      '00/00/0000',0                 ;
db      0                              ; End of strings
```

Example: BIOS Information without strings (example-only)

```
BIOS_Info      LABEL BYTE
db      0                              ; Indicates BIOS Structure Type
db      13h                            ; Length of information in bytes
dw      ?                              ; Reserved for handle
db      00h                            ; No Vendor Name provided
db      00h                            ; No BIOS version provided
dw      0E800h                         ; BIOS Starting Address
db      00h                            ; No BIOS Build Date provided
db      1                              ; Size of BIOS ROM is 128K (64K * (1 + 1))
dq      BIOS_Char                      ; BIOS Characteristics
db      0                              ; BIOS Characteristics Extension Byte 1
dw      0000h                          ; Structure terminator
```

3.1.2 Structure Header Format

Each SMBIOS structure begins with a 4-byte header, as follows:

Offset	Name	Length	Description
00h	Type	BYTE	Specifies the type of structure. Types 0 through 127 (7Fh) are reserved for and defined by this specification. Types 128 through 256 (80h to FFh) are available for system- and OEM-specific information.
01h	Length	BYTE	Specifies the length of the formatted area of the structure, starting at the Type field. The length of the structure's string-set is not included.
02h	Handle	WORD	Specifies the structure's handle, a unique 16-bit number in the range 0 to 0FFFEh (for version 2.0) or 0 to 0FEFFh (for version 2.1 and later). The handle can be used with the *Get SMBIOS Structure* function to retrieve a specific structure; the handle numbers are not required to be contiguous. For v2.1 and later, handle values in the range 0FF00h to 0FFFFh are reserved for use by this specification. If the system configuration changes, a previously assigned handle might no longer exist. However once a handle has been assigned by the BIOS, the BIOS cannot re-assign that handle number to another structure.

3.1.3 Text Strings

Text strings associated with a given SMBIOS structure are returned in the *dmiStrucBuffer*, appended directly after the formatted portion of the structure. This method of returning string information eliminates the need for application software to deal with pointers embedded in the SMBIOS structure. Each string is terminated with a null (00h) BYTE and the set of strings is terminated with an additional null (00h) BYTE. When the formatted portion of a SMBIOS structure references a string, it does so by specifying a non-zero string number within the structure's string-set. For example, if a string field contains 02h, it references the second string following the formatted portion of the SMBIOS structure. If a string field references no string, a null (0) is placed in that string field. If the formatted portion of the structure contains string-reference fields and all the string fields are set to 0 (no string references), the formatted section of the structure is followed by two null (00h) BYTES. See *3.1.1 Structure Evolution and Usage Guidelines* on page 25 for a string-containing example.

Note: Each text string is limited to 64 significant characters due to system MIF limitations.

3.2 Structure Definitions

3.2.1 BIOS Information (Type 0)

Offset	Name	Length	Value	Description
00h	Type	BYTE	0	BIOS Information Indicator
01h	Length	BYTE	Varies	12h + number of *BIOS Characteristics Extension* Bytes. If no Extension Bytes are used the Length will be 12h. For v2.1 and later implementations, the length is at least 13h since one extension byte is now defined.
02h	Handle	WORD	Varies	
04h	Vendor	BYTE	STRING	String number of the BIOS Vendor's Name
05h	BIOS Version	BYTE	STRING	String number of the BIOS Version. This is a free form string which may contain Core and OEM version information.
06h	BIOS Starting Address Segment	WORD	Varies	Segment location of BIOS starting address, e.g.0E800h. Note: The size of the runtime BIOS image can be computed by subtracting the Starting Address Segment from 10000h and multiplying the result by 16.
08h	BIOS Release Date	BYTE	STRING	String number of the BIOS release date. The date string, if supplied, is in either mm/dd/yy or mm/dd/yyyy format. If the year portion of the string is two digits, the year is assumed to be 19yy.
09h	BIOS ROM Size	BYTE	Varies (n)	Size (n) where 64K * (n+1) is the size of the physical device containing the BIOS, in bytes
0Ah	BIOS Characteristics	QWORD	Bit Field	Defines which functions the BIOS supports. PCI, PCMCIA, Flash, etc. See 3.2.1.1.
12h	BIOS Characteristics Extension Bytes	Zero or more BYTEs	Bit Field	Optional space reserved for future supported functions. The number of Extension Bytes that are present is indicated by the Length in offset 1 minus 12h. See 3.2.1.2 for extensions defined for v2.1 and later implementations.

3.2.1.1 BIOS Characteristics

QWORD Bit Position	Meaning if Set
Bit 0	Reserved
Bit 1	Reserved
Bit 2	Unknown
Bit 3	BIOS Characteristics Not Supported
Bit 4	ISA is supported
Bit 5	MCA is supported
Bit 6	EISA is supported
Bit 7	PCI is supported
Bit 8	PC Card (PCMCIA) is supported
Bit 9	Plug and Play is supported
Bit 10	APM is supported
Bit 11	BIOS is Upgradeable (Flash)
Bit 12	BIOS shadowing is allowed
Bit 13	VL-VESA is supported
Bit 14	ESCD support is available
Bit 15	Boot from CD is supported
Bit 16	Selectable Boot is supported
Bit 17	BIOS ROM is socketed
Bit 18	Boot From PC Card (PCMCIA) is supported
Bit 19	EDD (Enhanced Disk Drive) Specification is supported
Bit 20	Int 13h - Japanese Floppy for NEC 9800 1.2mb (3.5”, 1k Bytes/Sector, 360 RPM) is supported
Bit 21	Int 13h - Japanese Floppy for Toshiba 1.2mb (3.5”, 360 RPM) is supported
Bit 22	Int 13h - 5.25” / 360 KB Floppy Services are supported
Bit 23	Int 13h - 5.25” /1.2MB Floppy Services are supported
Bit 24	Int 13h - 3.5” / 720 KB Floppy Services are supported
Bit 25	Int 13h - 3.5” / 2.88 MB Floppy Services are supported
Bit 26	Int 5h, Print Screen Service is supported
Bit 27	Int 9h, 8042 Keyboard services are supported
Bit 28	Int 14h, Serial Services are supported
Bit 29	Int 17h, Printer Services are supported
Bit 30	Int 10h, CGA/Mono Video Services are supported
Bit 31	NEC PC-98
Bits32:47	Reserved for BIOS Vendor
Bits 48:63	Reserved for System Vendor

3.2.1.2 BIOS Characteristics Extension Byte 1

Note: All Characteristics Extension Bytes are reserved for assignment via this specification.

Byte Bit Position	Spec Version	Meaning if Set
Bit 0	2.1+	ACPI supported
Bit 1	2.1+	USB Legacy is supported
Bit 2	2.1+	AGP is supported
Bit 3	2.1+	I2O boot is supported
Bit 4	2.1+	LS-120 boot is supported
Bit 5	2.1+	ATAPI ZIP Drive boot is supported
Bit 6	2.1+	1394 boot is supported
Bit 7	2.1+	Smart Battery supported

3.2.2 System Information (Type 1)

The information in this structure defines attributes of the overall system and is intended to be associated with the *Component ID* group of the system's MIF.

Offset	Spec Version	Name	Length	Value	Description
00h	2.0+	Type	BYTE	1	Component ID Information Indicator
01h	2.0+	Length	BYTE	08h or 19h	Length dependent on version supported, 08h for 2.0 or 19h for 2.1 and later.
02h	2.0+	Handle	WORD	Varies	
04h	2.0+	Manufacturer	BYTE	STRING	Number of Null terminated string
05h	2.0+	Product Name	BYTE	STRING	Number of Null terminated string
06h	2.0+	Version	BYTE	STRING	Number of Null terminated string
07h	2.0+	Serial Number	BYTE	STRING	Number of Null terminated string
08h	2.1+	UUID	16 BYTEs	Varies	Universal Unique ID number. If the value is all FFh, the ID is not currently present in the system, but is settable. If the value is all 00h, the ID is not present in the system.
18h	2.1+	Wake-up Type	BYTE	ENUM	Identifies the event that caused the system to power up. See 3.2.2.1.

3.2.2.1 System — Wake-up Type

Byte Value	Meaning
00h	Reserved
01h	Other
02h	Unknown
03h	APM Timer
04h	Modem Ring
05h	LAN Remote
06h	Power Switch
07h	PCI PME#

3.2.3 Base Board Information (Type 2)

The information in this structure defines attributes of the system's baseboard (also known as the motherboard or planar).

Offset	Name	Length	Value	Description
00h	Type	BYTE	2	Base Board Information Indicator
01h	Length	BYTE	08h	
02h	Handle	WORD	Varies	
04h	Manufacturer	BYTE	STRING	Number of Null terminated string
05h	Product	BYTE	STRING	Number of Null terminated string
06h	Version	BYTE	STRING	Number of Null terminated string
07h	Serial Number	BYTE	STRING	Number of Null terminated string

3.2.4 System Enclosure or Chassis (Type 3)

The information in this structure defines attributes of the system's mechanical enclosure(s). For example, if a system included a separate enclosure for its peripheral devices, two structures would be returned: one for the main, system enclosure and the second for the peripheral device enclosure. The additions to this structure in v2.1 of this specification support the population of the DMTF|Physical Container Global Table group.

Offset	Spec Version	Name	Length	Value	Description
00h	2.0+	Type	BYTE	3	System Enclosure Indicator
01h	2.0+	Length	BYTE	Varies	09h for v2.0 implementations; 0Dh for v2.1 and later implementations.
02h	2.0+	Handle	WORD	Varies	
04h	2.0+	Manufacturer	BYTE	STRING	Number of Null terminated string
05h	2.0+	Type	BYTE	Varies	Bit 7 Chassis lock present if 1. Otherwise, either a lock is not present or it is unknown if the enclosure has a lock. Bits 6:0 Enumeration value, see below.
06h	2.0+	Version	BYTE	STRING	Number of Null terminated string
07h	2.0+	Serial Number	BYTE	STRING	Number of Null terminated string
08h	2.0+	Asset Tag Number	BYTE	STRING	Number of Null terminated string
09h	2.1+	Bootup State	BYTE	ENUM	Identifies the state of the enclosure when it was last booted. See 3.2.4.2 for definitions.
0Ah	2.1+	Power Supply State	BYTE	ENUM	Identifies the state of the enclosure's power supply (or supplies) when last booted. See 3.2.4.2 for definitions.
0Bh	2.1+	Thermal State	BYTE	ENUM	Identifies the enclosure's thermal state when last booted. See 3.2.4.2 for definitions.
0Ch	2.1+	Security Status	BYTE	ENUM	Identifies the enclosure's physical security status when last booted. See 3.2.4.3 for definitions.

3.2.4.1 System Enclosure or Chassis Types

Important Note: Enumerated values are controlled by the DMTF, not this specification.

Byte Value	Meaning
01h	Other
02h	Unknown
03h	Desktop
04h	Low Profile Desktop
05h	Pizza Box
06h	Mini Tower
07h	Tower
08h	Portable
09h	LapTop
0Ah	Notebook
0Bh	Hand Held
0Ch	Docking Station
0Dh	All in One
0Eh	Sub Notebook
0Fh	Space-saving
10h	Lunch Box
11h	Main Server Chassis
12h	Expansion Chassis
13h	SubChassis
14h	Bus Expansion Chassis
15h	Peripheral Chassis
16h	RAID Chassis
17h	Rack Mount Chassis
18h	Sealed-case PC

3.2.4.2 System Enclosure or Chassis States

Important Note: Enumerated values are controlled by the DMTF, not this specification.

Byte Value	Meaning
01h	Other
02h	Unknown
03h	Safe
04h	Warning
05h	Critical
06h	Non-recoverable

3.2.4.3 System Enclosure or Chassis Security Status

Important Note: Enumerated values are controlled by the DMTF, not this specification.

Byte Value	Meaning
01h	Other
02h	Unknown
03h	None
04h	External interface locked out
05h	External interface enabled

3.2.5 Processor Information (Type 4)

The information in this structure defines the attributes of a single processor; a separate structure instance is provided for each system processor socket/slot. For example, a system with an IntelDX2™ processor would have a single structure instance while a system with an IntelSX2™ processor would have a structure to describe the main CPU and a second structure to describe the 80487 co-processor.

Offset	Spec Version	Name	Length	Value	Description
00h	2.0+	Type	BYTE	4	Processor Information Indicator
01h	2.0+	Length	BYTE	Varies	The Length is 1Ah for v2.0 implementations or 20h for v2.1 and later implementations.
02h	2.0+	Handle	WORD	Varies	
04h	2.0+	Socket Designation	BYTE	STRING	String number for Reference Designation. Example string 'J202',0
05h	2.0+	Processor Type	BYTE	ENUM	See 3.2.5.1 on page 33
06h	2.0+	Processor Family	BYTE	ENUM	See 3.2.5.2 on page 34
07h	2.0+	Processor Manufacturer	BYTE	STRING	String number of Processor Manufacturer
08h	2.0+	Processor ID	QWORD	Varies	Raw processor identification data. See 3.2.5.3 for details.
10h	2.0+	Processor Version	BYTE	STRING	String number describing the Processor
11h	2.0+	Voltage	BYTE	Varies	See 3.2.5.4.
12h	2.0+	External Clock	WORD	Varies	External Clock Frequency, in MHz. If the value is unknown, the field is set to 0.
14h	2.0+	Max Speed	WORD	Varies	Maximum internal processor speed, as supported by the system. 0E9h for a 233MHz processor. If the value is unknown, the field is set to 0.
16h	2.0+	Current Speed	WORD	Varies	Same format as Max Speed
18h	2.0+	Status	BYTE	Varies	Bit 7 Reserved, must be 0 Bit 6 CPU Socket Populated 1 - CPU Socket Populated 0 - CPU Socket Unpopulated Bits 5:3 Reserved, must be zero Bits 2:0 CPU Status 0h - Unknown 1h - CPU Enabled 2h - CPU Disabled by User via BIOS Setup 3h - CPU Disabled By BIOS (POST Error) 4h - CPU is Idle, waiting to be enabled. 5-6h - Reserved 7h - Other
19h	2.0+	Processor Upgrade	BYTE	ENUM	See 3.2.5.5

Offset	Spec Version	Name	Length	Value	Description
1Ah	2.1+	L1 Cache Handle	WORD	Varies	The handle of a Cache Information structure which defines the attributes of the primary (Level 1) cache for this processor. The value is 0FFFFh if the processor has no L1 cache.
1Ch	2.1+	L2 Cache Handle	WORD	Varies	The handle of a Cache Information structure which defines the attributes of the secondary (Level 2) cache for this processor. The value is 0FFFFh if the processor has no L2 cache.
1Eh	2.1+	L3 Cache Handle	WORD	Varies	The handle of a Cache Information structure which defines the attributes of the tertiary (Level 3) cache for this processor. The value is 0FFFFh if the processor has no L3 cache.

3.2.5.1 Processor Information - Processor Type

Important Note: Enumerated values are controlled by the DMTF, not this specification.

Byte Value	Meaning
01h	Other
02h	Unknown
03h	Central Processor
04h	Math Processor
05h	DSP Processor
06h	Video Processor

3.2.5.2 Processor Information - Processor Family

Important Note: Enumerated values are controlled by the DMTF, not this specification.

Byte Value	Meaning
01h	Other
02h	Unknown
03h	8086
04h	80286
05h	Intel386™ processor
06h	Intel486™ processor
07h	8087
08h	80287
09h	80387
0Ah	80487
0Bh	Pentium® processor Family
0Ch	Pentium® Pro processor
0Dh	Pentium® II processor
0Eh	Pentium® processor with MMX™ technology
0Fh	Intel® Celeron™ processor
10h-11h	Reserved for specific Pentium® processor versions
12h	M1 Family
13h-18h	Reserved for specific M1 versions
19h	K5 Family
1Ah-1Fh	Reserved for specific K5 versions
20h	Power PC Family
21h	Power PC 601
22h	Power PC 603
23h	Power PC 603+
24h	Power PC 604
30h	Alpha Family[1]
40h	MIPS Family
50h	SPARC Family
60h	68040 Family
61h	68xxx
62h	68000
63h	68010
64h	68020
65h	68030
70h	Hobbit Family
80h	Weitek
90h	PA-RISC Family
A0h	V30 Family

[1] Some v2.0 specification implementations used *Processor Family* type value 30h to represent a Pentium ® Pro processor.

3.2.5.3 Processor ID Field Format

The Processor ID field contains processor-specific information which describes the processor's features.

3.2.5.3.1 X86-Class CPUs

For x86 class CPUs, the field's format depends on the processor's support of the CPUID instruction. If the instruction is supported, the *Processor ID* field contains two DWORD-formatted values. The first (offsets 08h-0Bh) is the EAX value returned by a CPUID instruction with input EAX set to 1; the second (offsets 0Ch-0Fh) is the EDX value returned by that instruction.

Otherwise, only the first two bytes of the *Processor ID* field are significant (all others are set to 0) and contain (in WORD-format) the contents of the DX register at CPU reset.

3.2.5.4 Processor Information – Voltage

Two forms of information can be specified by the SMBIOS in this field, dependent on the value present in bit 7 (the most-significant bit). If bit 7 is 0 (legacy mode), the remaining bits of the field represent the specific voltages that the processor socket can accept, as follows:

Bit 7 Set to 0, indicating 'legacy' mode for processor voltage
Bits 6:4 Reserved, must be zero
Bits 3:0 *Voltage Capability*. A Set bit indicates that the voltage is supported.
Bit 0 - 5V
Bit 1 - 3.3V
Bit 2 - 2.9V
Bit 3 - Reserved, must be zero.
Note: Setting of multiple bits indicates the socket is configurable

If bit 7 is set to 1, the remaining seven bits of the field are set to contain the processor's current *voltage times 10*. For example, the field value for a processor voltage of 1.8 volts would be 92h = 80h + (1.8 * 10) = 80h + 18 = 80h +12h.

3.2.5.5 Processor Information - Processor Upgrade

Important Note: Enumerated values are controlled by the DMTF, not this specification.

Byte Value	Meaning
01h	Other
02h	Unknown
03h	Daughter Board
04h	ZIF Socket
05h	Replaceable Piggy Back
06h	None
07h	LIF Socket
08h	Slot 1
09h	Slot 2

3.2.6 Memory Controller Information (Type 5)

The information in this structure defines the attributes of the system's memory controller(s) and the supported attributes of any memory-modules present in the sockets controlled by this controller.

Note: This structure, and its companion Memory Module Information (Type 6), are *obsolete* starting with version 2.1 of this specification; the Physical Memory Array (Type 16) and Memory Device (Type 17) structures should be used instead to allow proper population of the DMI 2.0 required groups. BIOS providers might choose to implement both memory description types to allow existing DMI browsers to properly display the system's memory attributes.

Offset	Spec Version	Name	Length	Value	Description
00h	2.0+	Type	BYTE	5	Memory Controller Indicator
01h	2.0+	Length	BYTE	Varies	Computed by the BIOS as either 15 + (2 * *x*) for v2.0 implementations or 16 + (2 * x) for v2.1 and later implementations, where *x* is the value present in offset 0Eh.
02h	2.0+	Handle	WORD	Varies	
04h	2.0+	Error Detecting Method	BYTE	ENUM	See 3.2.6.1
05h	2.0+	Error Correcting Capability	BYTE	Bit Field	See 3.2.6.2
06h	2.0+	Supported Interleave	BYTE	ENUM	See 3.2.6.3
07h	2.0+	Current Interleave	BYTE	ENUM	See 3.2.6.3
08h	2.0+	Maximum Memory Module Size	BYTE	Varies (n)	The size of the largest memory module supported (per slot), specified as n, where 2**n is the maximum size in MB. The maximum amount of memory supported by this controller is that value times the number of slots, as specified in offset 0Eh of this structure.
09h	2.0+	Supported Speeds	WORD	Bit Field	See 3.2.6.4 for bit-wise descriptions.
0Bh	2.0+	Supported Memory Types	WORD	Bit Field	See 3.2.7.1 on page 39 for bit-wise descriptions.
0Dh	2.0+	Memory Module Voltage	BYTE	Bit Field	This field describes the required voltages for each of the memory module sockets controlled by this controller: Bits 7:3 Reserved, must be zero Bit 2 2.9V Bit 1 3.3V Bit 0 5V Note: Setting of multiple bits indicates the sockets are configurable

Offset	Spec Version	Name	Length	Value	Description
0Eh	2.0+	Number of Associated Memory Slots (x)	BYTE	Varies	Defines how many of the *Memory Module Information* blocks are controlled by this controller.
0Fh to 0Fh + (2*x) - 1	2.0+	Memory Module Configuration Handles	x WORDs	Varies	A list of memory information structure handles controlled by this controller. Value in offset 0Eh (x) defines the count.
0Fh + (2*x)	2.1+	Enabled Error Correcting Capabilities	BYTE	Bit Field	Identifies the error-correcting capabilities which were enabled when the structure was built. See 3.2.6.2 for bit-wise definitions.

3.2.6.1 Memory Controller Error Detecting Method

Byte Value	Meaning
01h	Other
02h	Unknown
03h	None
04h	8-bit Parity
05h	32-bit ECC
06h	64-bit ECC
07h	128-bit ECC
08h	CRC

3.2.6.2 Memory Controller Error Correcting Capability

Byte Bit Position	Meaning
Bit 0	Other
Bit 1	Unknown
Bit 2	None
Bit 3	Single Bit Error Correcting
Bit 4	Double Bit Error Correcting
Bit 5	Error Scrubbing

3.2.6.3 Memory Controller Information - Interleave Support

Byte Value	Meaning
01h	Other
02h	Unknown
03h	One Way Interleave
04h	Two Way Interleave
05h	Four Way Interleave
06h	Eight Way Interleave
07h	Sixteen Way Interleave

3.2.6.4 Memory Controller Information - Memory Speeds

This bit-field describes the speed of the memory modules supported by the system.

Word Bit Position	Meaning
Bit 0	Other
Bit 1	Unknown
Bit 2	70ns
Bit 3	60ns
Bit 4	50ns
Bits 5:15	Reserved, must be zero

3.2.7 Memory Module Information (Type 6)

One *Memory Module Information* structure is included for each memory-module socket in the system. The structure describes the speed, type, size, and error status of each system memory module. The supported attributes of each module are described by the "owning" *Memory Controller Information* structure.

Note: This structure, and its companion Memory Controller Information (Type 5), are *obsolete* starting with version 2.1 of this specification; the Physical Memory Array (Type 16) and Memory Device (Type 17) structures should be used instead to allow proper population of the DMI 2.0 required groups. BIOS providers might choose to implement both memory description types to allow existing DMI browsers to properly display the system's memory attributes.

Offset	Name	Length	Value	Description
00h	Type	BYTE	6	Memory Module Configuration Indicator
01h	Length	BYTE	0Ch	
02h	Handle	WORD	Varies	
04h	Socket Designation	BYTE	STRING	String Number for Reference Designation. Example 'J202',0
05h	Bank Connections	BYTE	Varies	Each nibble indicates a bank (RAS#) connection, 0xF means no connection. Example: If banks 1 & 3 (RAS# 1 & 3) were connected to a SIMM socket the byte for that socket would be 13h. If only bank 2 (RAS 2) were connected the byte for that socket would be 2Fh.
06h	Current Speed	BYTE	Varies	The speed of the memory module, in ns (e.g. 70d for a 70ns module). If the speed is unknown, the field is set to 0.
07h	Current Memory Type	WORD	Bit Field	See 3.2.7.1
09h	Installed Size	BYTE	Varies	See 3.2.7.2
0Ah	Enabled Size	BYTE	Varies	See 3.2.7.2

Offset	Name	Length	Value	Description
0Bh	Error Status	BYTE	Varies	Bits 7:3 Reserved, set to 0's Bit 2 If set, the Error Status information should be obtained from the event log; bits 1and 0 are reserved. Bit 1 Correctable errors received for the module, if set. This bit will only be reset during a system reset. Bit 0 Uncorrectable errors received for the module, if set. All or a portion of the module has been disabled. This bit is only reset on power-on.

3.2.7.1 Memory Module Information - Memory Types

This bit-field describes the physical characteristics of the memory modules which are supported by (and currently installed in) the system.

Word Bit Position	Meaning
Bit 0	Other
Bit 1	Unknown
Bit 2	Standard
Bit 3	Fast Page Mode
Bit 4	EDO
Bit 5	Parity
Bit 6	ECC
Bit 7	SIMM
Bit 8	DIMM
Bit 9	Burst EDO
Bit 10	SDRAM
Bits 11:15	Reserved, must be zero

3.2.7.2 Memory Module Information - Memory Size

The Size fields of the Memory Module Configuration Information structure define the amount of memory currently installed (and enabled) in a memory-module connector.

The *Installed Size* fields identify the size of the memory module which is installed in the socket, as determined by reading and correlating the module's presence-detect information. If the system does not support presence-detect mechanisms, the *Installed Size* field is set to 7Dh to indicate that the installed size is not determinable. The *Enabled Size* field identifies the amount of memory currently enabled for the system's use from the module. If a module is known to be installed in a connector, but all memory in the module has been disabled due to error, the *Enabled Size* field is set to 7Eh.

Byte Bit Range	Meaning
Bits 0:6	Size (n), where 2**n is the size in MB with three special-case values: 7Dh Not determinable (Installed Size only) 7Eh Module is installed, but no memory has been enabled 7Fh Not installed
Bit 7	Defines whether the memory module has a single- (0) or double-bank (1) connection.

3.2.7.3 Memory Subsystem Example

A system utilizes a memory controller which supports up to 4-32MB 5V 70ns parity SIMMs. The memory module sockets are used in pairs A1/A2 and B1/B2 to provide a 64-bit data path to the CPU. No mechanism is provided by the system to read the SIMM IDs. RAS-0 and -1 are connected to the front- and back-size banks of the SIMMs in the A1/A2 sockets and RAS-2 and -3 are similarly connected to the B1/B2 sockets. The current installation is an 8MB SIMM in sockets A1 and A2, 16MB total.

```
db      5               ; Memory Controller Information
db      23              ; Length = 15 + 2*4
dw      14              ; Memory Controller Handle
db      4               ; 8-bit parity error detection
db      00000100b       ; No error correction provided
db      03h             ; 1-way interleave supported
db      03h             ; 1-way interleave currently used
db      5               ; Maximum memory-module size supported is 32MB (2**5)
dw      00000100b       ; Only 70ns SIMMs supported
dw      00A4h           ; Standard, parity SIMMs supported
db      00000001b       ; 5V provided to each socket
db      4               ; 4 memory-module sockets supported
dw      15              ; 1st Memory Module Handle
dw      16
dw      17
dw      18              ; 4th ...
db      6               ; Memory Module Information
db      0Ch
dw      15              ; Handle
db      1               ; Reference Designation string #1
db      01h             ; Socket connected to RAS-0 and RAS-1
db      00000010b       ; Current speed is Unknown, since can't read SIMM IDs
db      00000100b       ; Upgrade speed is 70ns, since that's all that's
                        ; supported
dw      00A4h           ; Current SIMM must be standard parity
db      7Dh             ; Installed size indeterminable (no SIMM IDs)
db      83h             ; Enabled size is double-bank 8MB (2**3)
db      0               ; No errors
db      "A1",0          ; String#1: Reference Designator
db      0               ; End-of-strings
db      6               ; Memory Module Information
db      0Ch
dw      16              ; Handle
db      1               ; Reference Designation string #1
db      01h             ; Socket connected to RAS-0 and RAS-1
db      0               ; Current speed is Unknown, since can't read SIMM IDs
dw      00A4h           ; Current SIMM must be standard parity
db      7Dh             ; Installed size indeterminable (no SIMM IDs)
db      83h             ; Enabled size is double-bank 8MB (2**3)
db      0               ; No errors
db      "A2",0          ; String#1: Reference Designator
db      0               ; End-of-strings
db      6               ; Memory Module Information
db      0Ch
dw      17              ; Handle
db      1               ; Reference Designation string #1
db      23h             ; Socket connected to RAS-2 and RAS-3
db      0               ; Current speed is Unknown, since can't read SIMM IDs
dw      0001h           ; Nothing appears to be installed (Other)
db      7Dh             ; Installed size indeterminable (no SIMM IDs)
db      7Fh             ; Enabled size is 0 (nothing installed)
db      0               ; No errors
db      "B1",0          ; String#1: Reference Designator
db      0               ; End-of-strings
```

```
db      6               ; Memory Module Information
db      0Ch
dw      18              ; Handle
db      1               ; Reference Designation string #1
db      23h             ; Socket connected to RAS-2 and RAS-3
db      0               ; Current speed is Unknown, since can't read SIMM IDs
dw      0001h           ; Nothing appears to be installed (Other)
db      7Dh             ; Installed size indeterminable (no SIMM IDs)
db      7Fh             ; Enabled size is 0 (nothing installed)
db      0               ; No errors
db      "B2",0          ; String#1: Reference Designator
db      0               ; End-of-strings
```

3.2.8 Cache Information (Type 7)

The information in this structure defines the attributes of CPU cache device in the system. One structure is specified for each such device, whether the device is internal to or external to the CPU module. Cache modules can be associated with a processor structure in one or two ways depending on the SMBIOS version, see *3.2.5 Processor Information (Type 4)* on page 32 and *3.2.15 Group Associations (Type 14)* on page 51 for more information.

Offset	Spec Version	Name	Length	Value	Description
00h	2.0+	Type	BYTE	7	Cache Information Indicator
01h	2.0+	Length	BYTE	Varies	The value is 0Fh for v2.0 implementations, or 13h for v2.1 implementations.
02h	2.0+	Handle	WORD	Varies	
04h	2.0+	Socket Designation	BYTE	STRING	String Number for Reference Designation Example: "CACHE1", 0
05h	2.0+	Cache Configuration	WORD	Varies	Bits 15:10 Reserved, must be zero Bits 9:8 Operational Mode 00b Write Through 01b Write Back 10b Varies with Memory Address 11b Unknown Bit 7 Enabled/Disabled (at boot time) 1b Enabled 0b Disabled Bits 6:5 Location, relative to the CPU module: 00b Internal 01b External 10b Reserved 11b Unknown Bit 4 Reserved, must be zero Bit 3 Cache Socketed 1b Socketed 0b Not Socketed Bits 2:0 Cache Level - 1 through 8, e.g. an L1 cache would use value 000b and an L3 cache would use 010b.

Offset	Spec Version	Name	Length	Value	Description
07h	2.0+	Maximum Cache Size	WORD	Varies	Maximum size that can be installed Bit 15 Granularity 0 - 1K granularity 1 - 64K granularity Bits 14:0 Max size in given granularity
09h	2.0+	Installed Size	WORD	Varies	Same format as Max Cache Size field, set to 0 if no cache is installed.
0Bh	2.0+	Supported SRAM Type	WORD	Bit Field	See 3.2.8.1
0Dh	2.0+	Current SRAM Type	WORD	Bit Field	See 3.2.8.1
0Fh	2.1+	Cache Speed	BYTE	Varies	The cache module speed, in nanoseconds. The value is 0 if the speed is unknown.
10h	2.1+	Error Correction Type	BYTE	ENUM	The error-correction scheme supported by this cache component, see 3.2.8.2.
11h	2.1+	System Cache Type	BYTE	ENUM	The logical type of cache, see 3.2.8.3.
12h	2.1+	Associativity	BYTE	ENUM	The associativity of the cache, see 3.2.8.4.

3.2.8.1 Cache Information - SRAM Type

Word Bit Position	Meaning
Bit 0	Other
Bit 1	Unknown
Bit 2	Non-Burst
Bit 3	Burst
Bit 4	Pipeline Burst
Bit 5	Synchronous
Bit 6	Asynchronous
Bits 7:15	Reserved, must be zero

3.2.8.2 Cache Information — Error Correction Type

Important Note: Enumerated values are controlled by the DMTF, not this specification.

Byte Value	Meaning
01h	Other
02h	Unknown
03h	None
04h	Parity
05h	Single-bit ECC
06h	Multi-bit ECC

3.2.8.3 Cache Information — System Cache Type

Important Note: Enumerated values are controlled by the DMTF, not this specification.

Byte Value	Meaning
01h	Other
02h	Unknown
03h	Instruction
04h	Data
05h	Unified

3.2.8.4 Cache Information — Associativity

Important Note: Enumerated values are controlled by the DMTF, not this specification.

Byte Value	Meaning
01h	Other
02h	Unknown
03h	Direct Mapped
04h	2-way Set-Associative
05h	4-way Set-Associative
06h	Fully Associative

3.2.9 Port Connector Information (Type 8)

The information in this structure defines the attributes of a system port connector, e.g. parallel, serial, keyboard, mouse ports. The port's type and connector information are provided. One structure is present for each port provided by the system.

Offset	Name	Length	Value	Description
00h	Type	BYTE	8	Connector Information Indicator
01h	Length	BYTE	9h	
02h	Handle	WORD	Varies	
04h	Internal Reference Designator	BYTE	STRING	String number for Internal Reference Designator, i.e. internal to the system enclosure, e.g. 'J101', 0
05h	Internal Connector Type	BYTE	ENUM	Internal Connector type. See 3.2.9.2
06h	External Reference Designator	BYTE	STRING	String number for the External Reference Designation external to the system enclosure, e.g. 'COM A', 0
07h	External Connector Type	BYTE	ENUM	External Connector type. See 3.2.9.2
08h	Port Type	BYTE	ENUM	Describes the function of the port. See 3.2.9.3

3.2.9.1 Port Information Example

The following structure shows an example where a DB-9 Pin Male connector on the System Backpanel (COM A) is connected to the System Board via a 9 Pin Dual Inline connector (J101).

```
db     8             ; Indicates Connector Type
db     9h            ; Length
dw     ?             ; Reserved for handle
db     01h           ; String 1 - Internal Reference Designation
db     18h           ; 9 Pin Dual Inline
db     02h           ; String 2 - External Reference Designation
db     08h           ; DB-9 Pin Male
db     09h           ; 16550A Compatible
db     'J101',0      ; Internal reference
db     'COM A',0     ; External reference
db     0
```

If an External Connector is not used (as in the case of a CD-ROM Sound connector) then the External Reference Designator and the External Connector type should be set to zero. If an Internal Connector is not used (as in the case of a soldered on Parallel Port connector which extends outside of the chassis) then the Internal Reference Designation and Connector Type should be set to zero.

3.2.9.2 Port Information - Connector Types

Byte Value	Meaning
00h	None
01h	Centronics
02h	Mini Centronics
03h	Proprietary
04h	DB-25 pin male
05h	DB-25 pin female
06h	DB-15 pin male
07h	DB-15 pin female
08h	DB-9 pin male
09h	DB-9 pin female
0Ah	RJ-11
0Bh	RJ-45
0Ch	50 Pin MiniSCSI
0Dh	Mini-DIN
0Eh	Micro-DIN
0Fh	PS/2
10h	Infrared
11h	HP-HIL
12h	Access Bus (USB)
13h	SSA SCSI
14h	Circular DIN-8 male
15h	Circular DIN-8 female
16h	On Board IDE
17h	On Board Floppy
18h	9 Pin Dual Inline (pin 10 cut)
19h	25 Pin Dual Inline (pin 26 cut)
1Ah	50 Pin Dual Inline
1Bh	68 Pin Dual Inline
1Ch	On Board Sound Input from CD-ROM
1Dh	Mini-Centronics Type-14
1Eh	Mini-Centronics Type-26

Byte Value	Meaning
1Fh	Mini-jack (headphones)
20h	BNC
21h	1394
A0h	PC-98
A1h	PC-98Hireso
A2h	PC-H98
A3h	PC-98Note
A4h	PC-98Full
FFh	Other - Use Reference Designator Strings to supply information.

3.2.9.3 Port Types

Byte Value	Meaning
00h	None
01h	Parallel Port XT/AT Compatible
02h	Parallel Port PS/2
03h	Parallel Port ECP
04h	Parallel Port EPP
05h	Parallel Port ECP/EPP
06h	Serial Port XT/AT Compatible
07h	Serial Port 16450 Compatible
08h	Serial Port 16550 Compatible
09h	Serial Port 16550A Compatible
0Ah	SCSI Port
0Bh	MIDI Port
0Ch	Joy Stick Port
0Dh	Keyboard Port
0Eh	Mouse Port
0Fh	SSA SCSI
10h	USB
11h	FireWire (IEEE P1394)
12h	PCMCIA Type II
13h	PCMCIA Type II
14h	PCMCIA Type III
15h	Cardbus
16h	Access Bus Port
17h	SCSI II
18h	SCSI Wide
19h	PC-98
1Ah	PC-98-Hireso
1Bh	PC-H98
1Ch	Video Port
1Dh	Audio Port
1Eh	Modem Port
1Fh	Network Port
A0h	8251 Compatible
A1h	8251 FIFO Compatible
0FFh	Other

3.2.10 System Slots (Type 9)

The information in this structure defines the attributes of a system slot. One structure is provided for each slot in the system.

Offset	Spec Version	Name	Length	Value	Description
00h	2.0+	Type	BYTE	9	System Slot Structure Indicator
01h	2.0+	Length	BYTE	Varies	0Ch for v2.0 implementations; 0Dh for v2.1 and later.
02h	2.0+	Handle	WORD	Varies	
04h	2.0+	Slot Designation	BYTE	STRING	String number for reference designation e.g. 'PCI-1',0
05h	2.0+	Slot Type	BYTE	ENUM	See 3.2.10.1
06h	2.0+	Slot Data Bus Width	BYTE	ENUM	See 3.2.10.2
07h	2.0+	Current Usage	BYTE	ENUM	See 3.2.10.3
08h	2.0+	Slot Length	BYTE	ENUM	See 3.2.10.4
09h	2.0+	Slot ID	WORD	Varies	See 3.2.10.5
0Bh	2.0+	Slot Characteristics 1	BYTE	Bit Field	See 3.2.10.6
0Ch	2.1+	Slot Characteristics 2	BYTE	Bit Field	See 3.2.10.7

3.2.10.1 System Slots - Slot Type

Byte Value	Meaning
01h	Other
02h	Unknown
03h	ISA
04h	MCA
05h	EISA
06h	PCI
07h	PC Card (PCMCIA)
08h	VL-VESA
09h	Proprietary
0Ah	Processor Card Slot
0Bh	Proprietary Memory Card Slot
0Ch	I/O Riser Card Slot
0Dh	NuBus
0Eh	PCI - 66MHz Capable
0Fh	AGP
10h	AGP 2X
11h	AGP 4X
A0h	PC-98/C20
A1h	PC-98/C24
A2h	PC-98/E
A3h	PC-98/Local Bus
A4h	PC-98/Card

3.2.10.2 System Slots - Slot Data Bus Width

Important Note: Enumerated values are controlled by the DMTF, not this specification.

Byte Value	Meaning
01h	Other
02h	Unknown
03h	8 bit
04h	16 bit
05h	32 bit
06h	64 bit
07h	128 bit

3.2.10.3 System Slots - Current Usage

Important Note: Enumerated values are controlled by the DMTF, not this specification.

Byte Value	Meaning
01h	Other
02h	Unknown
03h	Available
04h	In use

3.2.10.4 System Slots - Slot Length

Byte Value	Meaning
01h	Other
02h	Unknown
03h	Short Length
04h	Long Length

3.2.10.5 System Slots — Slot ID

The *Slot ID* field of the System Slot structure provides a mechanism to correlate the physical attributes of the slot to its logical access method (which varies based on the *Slot Type* field). The Slot ID field has meaning only for the slot types described below:

Slot Type	Slot ID Field Meaning
MCA	Identifies the logical Micro Channel slot number, in the range 1 to 15, in offset 09h. Offset 0Ah is set to 0.
EISA	Identifies the logical EISA slot number, in the range 1 to 15, in offset 09h. Offset 0Ah is set to 0.
PCI/AGP	Identifies the value present in the Slot Number field of the PCI Interrupt Routing table entry that is associated with this slot, in offset 09h — offset 0Ah is set to 0. The table is returned by the “Get PCI Interrupt Routing Options” BIOS function call. Note: This definition also applies to the 66MHz-capable PCI slots.
PCMCIA	Identifies the Adapter Number (offset 09h) and Socket Number (offset 0Ah) to be passed to PCMCIA Socket Services to identify this slot.

3.2.10.6 Slot Characteristics 1

BYTE Bit Position	Meaning if Set
Bit 0	Characteristics Unknown
Bit 1	Provides 5.0 Volts
Bit 2	Provides 3.3 Volts
Bit 3	Slot's opening is shared with another slot, e.g. PCI/EISA shared slot.
Bit 4	PC Card slot supports PC Card-16
Bit 5	PC Card slot supports CardBus
Bit 6	PC Card slot supports Zoom Video
Bit 7	PC Card slot supports Modem Ring Resume

3.2.10.7 Slot Characteristics 2

BYTE Bit Position	Meaning if Set
Bit 0	PCI slot supports Power Management Enable (PME#) signal
Bits 1:7	Reserved, set to 0

3.2.11 On Board Devices Information (Type 10)

The information in this structure defines the attributes of devices which are onboard (soldered onto) a system element, usually the baseboard. In general, an entry in this table implies that the BIOS has some level of control over the enabling of the associated device for use by the system.

Important Note: Since this structure was originally defined with the *Length* implicitly defining the number of devices present, no further fields can be added to this structure without adversely affecting existing software's ability to properly parse the data. Thus, if additional fields are required for this structure type a brand new structure must be defined to add a device count field, carry over the existing fields, and add the new information.

Offset	Name	Length	Value	Description
00h	Type	BYTE	10	On Board Devices Information Indicator
01h	Length	BYTE	Varies	Computed by the BIOS as 4 + 2 * (Number of Devices). The user of this structure determines the number of devices as (Length - 4) / 2.
02h	Handle	WORD	Varies	
4+2*(n-1)	$Device_n$ Type, n ranges from 1 to Number of Devices	BYTE	Varies	Bit 7 $Device_n$ Status 1 - Device Enabled 0 - Device Disabled Bits 6:0 Type of Device (See 3.2.11.1)
5+2*(n-1)	Description String	BYTE	STRING	String number of device description

Note: There may be a single structure instance containing the information for all onboard devices or there may be a unique structure instance for each onboard device.

3.2.11.1 Onboard Device Types

Byte Value	Meaning
01h	Other
02h	Unknown
03h	Video
04h	SCSI Controller
05h	Ethernet
06h	Token Ring
07h	Sound

3.2.12 OEM Strings (Type 11)

Offset	Name	Length	Value	Description
00h	Type	BYTE	11	OEM Strings Indicator
01h	Length	BYTE	5h	
02h	Handle	WORD	Varies	
04h	Count	BYTE	Varies	Number of strings

This structure contains free form strings defined by the OEM. Examples of this are: Part Numbers for Reference Documents for the system, contact information for the manufacturer, etc.

3.2.13 System Configuration Options (Type 12)

Offset	Name	Length	Value	Description
00h	Type	BYTE	12	Configuration Information Indicator
01h	Length	BYTE	5h	
02h	Handle	WORD	Varies	
04h	Count	BYTE	Varies	Number of strings

This structure contains information required to configure the base board's Jumpers and Switches.
Examples of this are: "JP2: 1-2 Cache Size is 256K, 2-3 Cache Size is 512K"
"SW1-1: Close to Disable On Board Video"

3.2.14 BIOS Language Information (Type 13)

The information in this structure defines the installable language attributes of the BIOS.

Offset	Spec Version	Name	Length	Value	Description
00h	2.0+	Type	BYTE	13	Language Information Indicator
01h	2.0+	Length	BYTE	16h	
02h	2.0+	Handle	WORD	Varies	
04h	2.0+	Installable Languages	BYTE	Varies	Number of languages available. Each available language will have a description string. This field contains the number of strings that follow the formatted area of the structure.
05h	2.1+	Flags	BYTE	Bit Field	Bits 7:1 Reserved Bit 0 If set to 1, the Current Language strings use the abbreviated format. Otherwise, the strings use the long format. See below for details.
06h	2.0+	Reserved	15 BYTEs	0	Reserved for future use
015h	2.0+	Current Language	BYTE	STRING	String number (one-based) of the currently installed language.

The strings describing the languages follow the *Current Language* byte. The format of the strings depends on the value present in bit 0 of the byte at offset 05h in the structure.

If the bit is 0, each language string is in the form “ISO 639 Language Name | ISO 3166 Territory Name | Encoding Method”. See the Example 1 below.

If the bit is 1, each language string consists of the 2-character ISO 639 Language Name directly followed by the 2-character ISO 3166 Territory Name. See Example 2 below.

Note: Refer to the Desktop Management Interface Specification, V1.0, Appendix A (ISO 639) and Appendix B (ISO 3166) for additional information.

Example 1: BIOS Language Information (Long Format)

```
db      13                          ; language information
db      16h                         ; length
dw      ??                          ; handle
db      3                           ; three languages available
db      0                           ; use long-format for language strings
db      15 dup (0)                  ; reserved
db      2                           ; current language is French Canadian
db      ‘en|US|iso8859-1’,0         ; language 1 is US English
db      ‘fr|CA|iso8859-1’,0         ; language 2 is French Canadian
db      ‘ja|JP|unicode’,0           ; language 3 is Japanese
db      0                           ; Structure termination
```

Example 2: BIOS Language Information (Abbreviated Format)

```
db      13                          ; language information
db      16h                         ; length
dw      ??                          ; handle
db      3                           ; three languages available
db      01h                         ; use abbreviated format for language strings
db      15 dup (0)                  ; reserved
db      2                           ; current language is French Canadian
db      ‘enUS’,0                    ; language 1 is US English
db      ‘frCA’,0                    ; language 2 is French Canadian
db      ‘jaJP’,0                    ; language 3 is Japanese
db      0                           ; Structure termination
```

3.2.15 Group Associations (Type 14)

Important Note: Since this structure was originally defined with the Length implicitly defining the number of items present, no further fields can be added to this structure without adversely affecting existing software's ability to properly parse the data. Thus, if additional fields are required for this structure type a brand new structure must be defined to add an item count field, carry over the existing fields, and add the new information.

Offset	Name	Length	Value	Description
00h	Type	BYTE	14	Group Associations Indicator
01h	Length	BYTE	Varies	Computed by the BIOS as 5 + (3 bytes for each item in the group). The user of this structure determines the number of items as (*Length* - 5) / 3.
02h	Handle	WORD	Varies	
04h	Group Name	BYTE	STRING	String number of string describing the group
05h	Item Type	BYTE	Varies	Item (Structure) Type of this member
06h	Item Handle	WORD	Varies	Handle corresponding to this structure

The Group Associations structure is provided for OEMs who want to specify the arrangement or hierarchy of certain components (including other Group Associations) within the system. For example, you can use the Group Associations structure to indicate that two CPU's share a common external cache system. These structures might look as follows:

First Group Association Structure:

```
db      14      ; Group Association structure
db      11      ; Length
dw      28h     ; Handle
db      01h     ; String Number (First String)
db      04      ; CPU Structure
dw      08h     ; CPU Structure's Handle
db      07      ; Cache Structure
dw      09h     ; Cache Structure's Handle
db      'Primary CPU Module', 0
db      0
```

Second Group Association Structure:

```
db      14      ; Group Association structure
db      11      ; Length
dw      29h     ; Handle
db      01h     ; String Number (First String)
db      04      ; CPU Structure
dw      0Ah     ; CPU Structure's Handle
db      07      ; Cache Structure
dw      09h     ; Cache Structure's Handle
db      'Secondary CPU Module', 0
db      0
```

In the examples above, CPU structures 08h and 0Ah are associated with the same cache, 09h. This relationship could also be specified as a single group:

```
db      14      ; Group Association structure
db      14      ; Length (5 + 3 * 3)
dw      28h     ; Structure handle for Group Association
db      1       ; String Number (First string)
db      4       ; 1st CPU
dw      08h     ; CPU structure handle
db      4       ; 2nd CPU
dw      0Ah     ; CPU structure handle
db      7       ; Shared cache
dw      09h     ; Cache structure handle
db      'Dual-Processor CPU Complex', 0
db      0
```

3.2.16 System Event Log (Type 15)

The presence of this structure within the SMBIOS data returned for a system indicates that the system supports an event log. An event log is a fixed-length area within a non-volatile storage element, starting with a fixed-length (and vendor-specific) header record, followed by one or more variable-length log records. See *3.2.16.4 Event Log Organization* on page 57 for more information. Refer also to *2.2.5 Function 54h – SMBIOS Control* on page 19 for interfaces that can be used to control the event-log.

An application can implement event-log change notification by periodically reading the System Event Log structure (via its assigned handle) looking for a change in the *Log Change Token*. This token uniquely identifies the last time the event log was updated. When it sees the token changed, the application can retrieve the entire event log and determine the changes since the last time it read the event log.

Offset	Spec Version	Name	Length	Value	Description
00h	2.0+	Type	BYTE	15	Event Log Type Indicator
01h	2.0+	Length	BYTE	Var	Length of the structure, including the Type and Length fields. The Length is 14h for v2.0 implementations or computed by the BIOS as 17h+(x*y) for v2.1 and higher implementations — where x is the value present at offset 15h and y is the value present at offset 16h.
02h	2.0+	Handle	WORD	Var	The handle, or instance number, associated with the structure.
04h	2.0+	Log Area Length	WORD	Var	The length, in bytes, of the overall event log area, from the first byte of header to the last byte of data.
06h	2.0+	Log Header Start Offset	WORD	Var	Defines the starting offset (or index) within the nonvolatile storage of the event-log's header, from the Access Method Address. For single-byte indexed I/O accesses, the most-significant byte of the start offset is set to 00h.

Offset	Spec Version	Name	Length	Value	Description
08h	2.0+	Log Data Start Offset	WORD	Var	Defines the starting offset (or index) within the nonvolatile storage of the event-log's first data byte, from the Access Method Address. For single-byte indexed I/O accesses, the most-significant byte of the start offset is set to 00h. ***Note***: The data directly follows any header information. Therefore, the header length can be determined by subtracting the *Header Start Offset* from the *Data Start Offset*.

Offset	Spec Version	Name	Length	Value	Description
0Ah	2.0+	Access Method	BYTE	Var	Defines the Location and Method used by higher-level software to access the log area, one of: 00h Indexed I/O: 1 8-bit index port, 1 8-bit data port. The Access Method Address field contains the 16-bit I/O addresses for the index and data ports. See 3.2.16.2.1 for usage details. 01h Indexed I/O: 2 8-bit index ports, 1 8-bit data port. The Access Method Address field contains the 16-bit I/O address for the index and data ports. See 3.2.16.2.2 for usage details. 02h Indexed I/O: 1 16-bit index port, 1 8-bit data port. The Access Method Address field contains the 16-bit I/O address for the index and data ports. See 3.2.16.2.3 for usage details. 03h Memory-mapped physical 32-bit address. The Access Method Address field contains the 4-byte (Intel DWORD format) starting physical address. 04h Available via General-Purpose NonVolatile Data functions, see 2.2.6 on page 21 for more information. The Access Method Address field contains the 2-byte (Intel WORD format) GPNV handle. 05h-7Fh Available for future assignment via this specification 80h-FFh BIOS Vendor/OEM-specific
0Bh	2.0+	Log Status[2]	BYTE	Var	This bit-field describes the current status of the system event-log: Bits 7:2 Reserved, set to 0's Bit 1 Log area full, if 1 Bit 0 Log area valid, if 1
0Ch	2.0+	Log Change[2] Token	DWORD	Var	Unique token that is reassigned every time the event log changes. Can be used to determine if additional events have occurred since the last time the log was read.

[2] The *Log Status* and *Log Change Token* fields might not be up-to-date (dynamic) when the structure is accessed using the table interface.

Offset	Spec Version	Name	Length	Value	Description

10h	2.0+	Access Method Address	DWORD	Var	The address associated with the access method; the data present depends on the Access Method field value. The area's format can be described by the following 1-byte-packed 'C' union:
					`union`
					`{`
					`struct`
					`{`
					`short IndexAddr;`
					`short DataAddr;`
					`} IO;`
					`long PhysicalAddr32;`
					`short GPNVHandle;`
					`} AccessMethodAddress;`

14h	2.1+	Log Header Format	BYTE	ENUM	Identifies the format of the log header area, see 3.2.16.5 for details.
15h	2.1+	Number of Supported Log Type Descriptors (x)	BYTE	Varies	Number of supported event log type descriptors that follow. If the value is 0, the list that starts at offset 17h is not present.
16h	2.1+	Length of each Log Type Descriptor (y)	BYTE	2	Identifies the number of bytes associated with each type entry in the list below. The value is currently "hard-coded" as 2, since each entry consists of two bytes. This field's presence allows future additions to the type list. Software that interprets the following list should not assume a list entry's length.
17h to 17h+(x*y))-1	2.1+	List of Supported Event Log Type Descriptors	Varies	Var	Contains a list of Event Log Type Descriptors (see 3.2.16.1), so long as the value specified in offset 15h is non-zero.

3.2.16.1 Supported Event Log Type Descriptors

Each entry consists of a 1-byte type field and a 1-byte data-format descriptor, as defined by the following table. The presence of an entry identifies that the Log Type is supported by the system and the format of any variable data which accompanies the first bytes of the log's variable data — a specific log record might have more variable data than specified by its Variable Data Format Type.

Offset	Name	Length	Value	Description
00h	Log Type	BYTE	ENUM	See 3.2.16.6.1 on page 59 for list.
01h	Variable Data Format Type	BYTE	ENUM	See 3.2.16.6.2 on page 60 for list

3.2.16.2 Indexed I/O Access Method

This section contains examples (in x86 assembly language) which detail the code required to access the "indexed I/O" event-log information.

3.2.16.2.1 1 8-bit Index, 1 8-bit Data (00h)

To access the event-log, the caller selects 1 of 256 unique data bytes by

1) Writing the byte data-selection value (index) to the *IndexAddr* I/O address
2) Reading or writing the byte data value to (or from) the *DataAddr* I/O address

```
mov    dx, IndexAddr       ;Value from event-log structure
mov    al, WhichLoc        ;Identify offset to be accessed
out    dx, al
mov    dx, DataAddr        ;Value from event-log structure
in     al, dx              ; Read current value
```

3.2.16.2.2 2 8-bit Index, 1 8-bit Data (01h)

To access the event-log, the caller selects 1 of 65536 unique data bytes by

1) Writing the least-significant byte data-selection value (index) to the *IndexAddr* I/O address
2) Writing the most-significant byte data-selection value (index) to the (*IndexAddr+1*) I/O address
3) Reading or writing the byte data value to (or from) the *DataAddr* I/O address

```
mov    dx, IndexAddr       ;Value from event-log structure
mov    ax, WhichLoc        ;Identify offset to be accessed
out    dx, al              ;Select LSB offset
inc    dx
xchg   ah, al
out    dx, al              ;Select MSB offset
mov    dx, DataAddr        ;Value from event-log structure
in     al, dx              ;Read current value
```

3.2.16.2.3 1 16-bit Index, 1 8-bit Data (02h)

To access the event-log, the caller selects 1 of 65536 unique data bytes by

1) Writing the word data-selection value (index) to the *IndexAddr* I/O address
2) Reading or writing the byte data value to (or from) the *DataAddr* I/O address

```
mov    dx, IndexAddr       ;Value from event-log structure
mov    ax, WhichLoc        ;Identify offset to be accessed
out    dx, ax
mov    dx, DataAddr        ;Value from event-log structure
in     al, dx              ;Read current value
```

3.2.16.3 Access Method Address — DWORD Layout

Access Type	BYTE 3	BYTE 2	BYTE 1	BYTE 0
00:02 — Indexed I/O	Data MSB	Data LSB	Index MSB	Index LSB
03- Absolute Address	Byte 3	Byte 2	Byte 1	Byte 0
04 - Use GPNV	0	0	Handle MSB	Handle LSB

3.2.16.4 Event Log Organization

The event log is organized as an optional (and implementation-specific) fixed-length header, followed by one or more variable-length event records, as illustrated below. From one implementation to the next, the format of the log header and the size of the overall log area might change; all other required fields of the event log area will be consistent across all systems.

Log Header (Optional)								
Type	Length	Year	Month	Day	Hour	Minute	Second	Log Variable Data
Reqd	**Reqd**	**Reqd**	**Reqd**	**Reqd**	**Reqd**	**Reqd**	**Reqd**	Optional

3.2.16.5 Log Header Format

The following table contains the byte enumeration values (available for SMBIOS v2.1 and later) which identify the standard formats of the event log headers.

Byte Value	Meaning	See …
00h	No header, e.g. the header is 0 bytes in length.	
01h	Type 1 log header	3.2.16.5.1
02h-7Fh	Available for future assignment via this specification	
80h-FFh	BIOS Vendor or OEM-specific format	

3.2.16.5.1 Log Header Type 1 Format

The type 1 event log header consists of the following fields:

Offset	Name	Length	Value	Description
00h	OEM Reserved	5 BYTES	Varies	Reserved area for OEM customization, not assignable by this specification.
05h	Multiple Event Time Window	BYTE	Varies	The number of minutes which must pass between duplicate log entries which utilize a multiple-event counter, specified in BCD. The value ranges from 00h to 99h to represent 0 to 99 minutes. See *3.2.16.6.3 Multiple-Event Counter* on page 60 for usage details.
06h	Multiple Event Count Increment	BYTE	Varies	The number of occurrences of a duplicate event which must pass before the multiple-event counter associated with the log entry is updated, specified as a numeric value in the range 1 to 255 (the value 0 is reserved). See *3.2.16.6.3 Multiple-Event Counter* on page 60 for usage details.
07h	Pre-boot Event Log Reset — CMOS Address	BYTE	Varies	Identifies the CMOS RAM address (in the range 10h - FFh) associated with the Pre-boot Event Log Reset; the value is 00h if the feature is not supported. See below for usage details.

Offset	Name	Length	Value	Description
08h	Pre-boot Event Log Reset — CMOS Bit Index	BYTE	Varies	Identifies the bit within the above CMOS RAM location which is set to indicate that the log should be cleared. The value is specified in the range 0 to 7, where 0 specifies the LSB and 7 specified the MSB. See below for usage details.
09h	CMOS Checksum — Starting Offset	BYTE	Varies	Identifies the CMOS RAM address associated with the start of the area which is to be checksummed, if the value is non-0. If the value is 0, the CMOS Address field lies outside of a checksummed region in CMOS. See below for usage details.
0Ah	CMOS Checksum — Byte Count	BYTE	Varies	Identifies the number of consecutive CMOS RAM addresses, starting at the Starting Offset, which participate in the CMOS Checksum region associated with the pre-boot event log reset. See below for usage details.
0Bh	CMOS Checksum — Checksum Offset	BYTE	Varies	Identifies the CMOS RAM address associated with the start of two consecutive bytes into which the calculated checksum value is stored. See below for usage details.
0Ch - 0Eh	Reserved	3 BYTEs	000000h	Available for future assignment via this specification.
0Fh	Header Revision	BYTE	01h	Identifies the version of Type 1 header implemented.

The Type 1 Log Header also provides pre-boot event log reset support. Application software can set a system-specific location of CMOS RAM memory (accessible via I/O ports 70h and 71h) to cause the event log to be cleared by the BIOS on the next reboot of the system.

To perform the field setting, application software follows these steps, so long as the *Pre-boot Event Log Reset — CMOS Address* field of the header is non-zero:

- Read the address specified from CMOS RAM set the bit specified by the *CMOS Bit Index* field to 1. Rewrite the CMOS RAM address with the updated data.
- If the *CMOS Checksum — Starting Offset* field is non-zero, recalculate the CMOS RAM checksum value for the range starting at the *Starting Offset* field for *Byte Count* bytes into a 2-byte value. Subtract that value from 0 to create the checksum value for the range and store that 2-byte value into the CMOS RAM; the least-significant byte of the value is stored at the CMOS RAM *Checksum Offset* and the most-significant byte of the value is stored at (*Checksum Offset*)+1.

3.2.16.6 Log Record Format

Each log record consists of a *required* fixed-length record header, followed by (optional) additional data which is defined by the event type. The fixed-length log record header is present as the first 8-bytes of each log record, regardless of event type, and consists of:

Offset	Name	Format	Description
00h	Event Type	BYTE	Specifies the "Type" of event noted in an event-log entry as defined in 3.2.16.6.1
01h	Length	BYTE	Specifies the byte length of the event record, including the record's Type and Length fields. The most-significant bit of the field specifies whether (0) or not (1) the record has been read. The implication of the record having been read is that the information in the log record has been processed by a higher software layer.
02h-07h	Date/Time Fields	BYTE	These fields contain the BCD representation of the date and time (as read from CMOS) of the occurrence of the event. The information is present in year, month, day, hour, minute, second order.
08h+	Log Variable Data	Var	This field contains the (optional) event-specific additional status information.

3.2.16.6.1 Event Log Types

Value	Description
00h	Reserved.
01h	Single-bit ECC memory error
02h	Multi-bit ECC memory error
03h	Parity memory error
04h	Bus time-out
05h	I/O Channel Check
06h	Software NMI
07h	POST Memory Resize
08h	POST Error
09h	PCI Parity Error
0Ah	PCI System Error
0Bh	CPU Failure
0Ch	EISA FailSafe Timer time-out
0Dh	Correctable memory log disabled
0Eh	Logging disabled for a specific Event Type – too many errors of the same type received in a short amount of time.
0Fh	Reserved
10h	System Limit Exceeded (e.g. voltage or temperature threshold exceeded).
11h	Asynchronous hardware timer expired and issued a system reset.
12h	System configuration information
13h	Hard-disk information
14h	System reconfigured
15h	Uncorrectable CPU-complex error
16h	Log Area Reset/Cleared
17h	System boot. If implemented, this log entry is guaranteed to be the first one written on any system boot.
18h-7Fh	Unused, available for assignment by this specification.
80h-FEh	Available for system- and OEM-specific assignments.

Value	Description
FFh	End-of-log. When an application searches through the event-log records, the end of the log is identified when a log record with this type is found.

3.2.16.6.2 Event Log Variable Data Format Types

The Variable Data Format Type, specified in the Event Log structure's Supported Event Type fields, identifies the standard-format that application software can apply to the first *n* bytes of the associated Log Type's variable data. Additional, OEM-specific, data might follow in the log's variable data field.

Value	Name	Description
00h	None	No standard format data is available; the first byte of the variable data (if present) contains OEM-specific unformatted information.
01h	Handle	The first WORD of the variable data contains the handle of the SMBIOS structure associated with the hardware element which failed.
02h	Multiple-Event	The first DWORD of the variable data contains a multiple-event counter (see 3.2.16.6.3 for details).
03h	Multiple-Event Handle	The first WORD of the variable data contains the handle of the SMBIOS structure associated with the hardware element which failed; it is followed by a DWORD containing a multiple-event counter (see 3.2.16.6.3 for details).
04h	POST Results Bitmap	The first 2 DWORDs of the variable data contain the POST Results Bitmap, as described in 3.2.16.6.3.1 on page 61.
05h	System Management Type	The first DWORD of the variable data contains a value which identifies a system-management condition. See 3.2.16.6.3.2on page 62 for the enumerated values.
06h	Multiple-Event System Management Type	The first DWORD of the variable data contains a value which identifies a system-management condition (see 3.2.16.6.3.2 on page 62 for the enumerated values). This DWORD is directly followed by a DWORD which contains a multiple-event counter (see 3.2.16.6.3 for details).
07h-7Fh	Unused	Unused, available for assignment by this specification.
80h-FFh	OEM assigned	Available for system- and OEM-specific assignments.

3.2.16.6.3 Multiple-Event Counter

Some system events can be persistent; once they occur, it is possible to quickly fill the log with redundant multiple logs. The Multiple Event Count Increment (*MECI*) and Multiple Event Time Window (*METW*) values can be used to reduce the occurrence of these multiple logs while providing multiple event counts.

Note: These values are normally specified within the event log header, see *3.2.16.5.1 Log Header Type 1 Format* on page 57 for an example; if the values aren't specified in the header, the application software can assume that the *MECI* value is 1 and the *METW* value is 60 (minutes).

The multiple-event counter is a DWORD (32-bit) value which tracks the number of logs of the same type which have occurred within *METW* minutes. The counter value is initialized (in the log entry) to FFFFFFFFh, implying that only a single event of that type has been detected, and the internal BIOS counter[3] specific to that log type is reset to 0. When the BIOS receives the next event of that type, it

[3] All BIOS counters which support the Multiple-Event Counters are reset to zero each time the system boots.

increments its internal counter and checks to see what recording of the error is to be performed:

1. *A new log entry is written* … and the internal BIOS counter reset to 0, if the date/time of the original log entry is outside of *METW* minutes.
2. *No recording* … (other than the internal counter increment) if the log's current multiple-event counter is 00000000h or if the internal BIOS counter is less than *MECI*.
3. *The next non-zero bit of the multiple-event counter is set to 0* … otherwise.

3.2.16.6.3.1 POST Results Bitmap

This variable data type, when present, is expected to be associated with the POST Error (08h) event log type and identifies that one or more error types have occurred. The bitmap consists of two DWORD values, described in the table below. Any bit within the DWORD pair that is specified as Reserved is set to 0 within the log data and is available for assignment via this specification. A set bit ('1'b) at a DWORD bit position implies that the error associated with that position has occurred.

Bit Position	First DWORD	Second DWORD
0	Channel 2 Timer error	Normally 0; available for OEM assignment
1	Master PIC (8259 #1) error	Normally 0; available for OEM assignment
2	Slave PIC (8259 #2) error	Normally 0; available for OEM assignment
3	CMOS Battery Failure	Normally 0; available for OEM assignment
4	CMOS System Options Not Set	Normally 0; available for OEM assignment
5	CMOS Checksum Error	Normally 0; available for OEM assignment
6	CMOS Configuration Error	Normally 0; available for OEM assignment
7	Mouse and Keyboard Swapped	PCI Memory Conflict
8	Keyboard Locked	PCI I/O Conflict
9	Keyboard Not Functional	PCI IRQ Conflict
10	Keyboard Controller Not Functional	PNP Memory Conflict
11	CMOS Memory Size Different	PNP 32 bit Memory Conflict
12	Memory Decreased in Size	PNP I/O Conflict
13	Cache Memory Error	PNP IRQ Conflict
14	Floppy Drive 0 Error	PNP DMA Conflict
15	Floppy Drive 1 Error	Bad PNP Serial ID Checksum
16	Floppy Controller Failure	Bad PNP Resource Data Checksum
17	Number of ATA Drives Reduced Error	Static Resource Conflict
18	CMOS Time Not Set	NVRAM Checksum Error, NVRAM Cleared
19	DDC Monitor Configuration Change	System Board Device Resource Conflict
20	Reserved, set to 0	Primary Output Device Not Found
21	Reserved, set to 0	Primary Input Device Not Found
22	Reserved, set to 0	Primary Boot Device Not Found
23	Reserved, set to 0	NVRAM Cleared By Jumper
24	Second DWORD has valid data	NVRAM Data Invalid, NVRAM Cleared
25	Reserved, set to 0	FDC Resource Conflict
26	Reserved, set to 0	Primary ATA Controller Resource Conflict
27	Reserved, set to 0	Secondary ATA Controller Resource Conflict
28	Normally 0; available for OEM assignment	Parallel Port Resource Conflict
29	Normally 0; available for OEM assignment	Serial Port 1 Resource Conflict
30	Normally 0; available for OEM assignment	Serial Port 2 Resource Conflict
31	Normally 0; available for OEM assignment	Audio Resource Conflict

3.2.16.6.3.2 System Management Types

The following table defines the system management types present in event log record's variable data. In general, each type is associated with a management event that occurred within the system.

Value	Name
00000000h	+2.5V Out of range, #1
00000001h	+2.5V Out of range, #2
00000002h	+3.3V Out of range
00000003h	+5V Out of range
00000004h	-5V Out of range
00000005h	+12V Out of range
00000006h	-12V Out of range
00000007h - 0000000Fh	Reserved for future out-of-range voltage levels, assigned via this specification
00000010h	System board temperature out of range
00000011h	Processor #1 temperature out of range
00000012h	Processor #2 temperature out of range
00000013h	Processor #3 temperature out of range
00000014h	Processor #4 temperature out of range
00000015h - 0000001Fh	Reserved for future out-of-range temperatures, assigned via this specification
00000020h - 00000027h	Fan n (n = 0 to 7) Out of range
00000028h - 0000002Fh	Reserved for future assignment via this specification
00000030h	Chassis secure switch activated
00000031h - 0000FFFFh	Reserved for future assignment via this specification
0001xxxxh	A system-management probe or cooling device is out-of-range. The *xxxx* portion of the value contains the handle of the SMBIOS structure associated with the errant device.
00020000h - 7FFFFFFFh	Reserved for future assignment via this specification
80000000h - FFFFFFFFh	OEM assigned

3.2.17 Physical Memory Array (Type 16)

This structure supports the population of the DMTF|Physical Memory Array group, as defined in theDMTF's `MASTER.MIF`.

Offset	Spec Version	Name	Length	Value	Description
00h	2.1+	Type	BYTE	16	Physical Memory Array type
01h	2.1+	Length	BYTE	0Fh	Length of the structure.
02h	2.1+	Handle	WORD	Varies	The handle, or instance number, associated with the structure.
04h	2.1+	Location	BYTE	ENUM	The physical location of the Memory Array, whether on the system board or an add-in board. See 3.2.17.1 for definitions.
05h	2.1+	Use	BYTE	ENUM	Identifies the function for which the array is used. See 3.2.17.2 for definitions.
06h	2.1+	Memory Error Correction	BYTE	ENUM	The primary hardware error correction or detection method supported by this memory array. See 3.2.17.3 for definitions.
07h	2.1+	Maximum Capacity	DWORD	Varies	The maximum memory capacity, in kilobytes, for this array. If the capacity is unknown, this field contains 8000 0000h.

Offset	Spec Version	Name	Length	Value	Description
0Bh	2.1+	Memory Error Information Handle	WORD	Varies	The handle, or instance number, associated with any error which was previously detected for the array. If the system does not provide the error information structure, the field contains FFFEh; otherwise, the field contains either FFFFh (if no error was detected) or the handle of the error-information structure. See also *3.2.19 Memory Error Information (Type 18)* on page 67
0Dh	2.1+	Number of Memory Devices	WORD	Varies	The number of slots or sockets available for Memory Devices in this array. This value represents the number of Memory Device structures which comprise this Memory Array. Each Memory Device has a reference to the 'owning' Memory Array.

3.2.17.1 Memory Array — Location

Important Note: Enumerated values are controlled by the DMTF, not this specification.

Byte Value	Meaning
01h	Other
02h	Unknown
03h	System board or motherboard
04h	ISA add-on card
05h	EISA add-on card
06h	PCI add-on card
07h	MCA add-on card
08h	PCMCIA add-on card
09h	Proprietary add-on card
0Ah	NuBus
A0h	PC-98/C20 add-on card
A1h	PC-98/C24 add-on card
A2h	PC-98/E add-on card
A3h	PC-98/Local bus add-on card

3.2.17.2 Memory Array — Use

Important Note: Enumerated values are controlled by the DMTF, not this specification.

Byte Value	Meaning
01h	Other
02h	Unknown
03h	System memory
04h	Video memory
05h	Flash memory
06h	Non-volatile RAM
07h	Cache memory

3.2.17.3 Memory Array — Error Correction Types

Important Note: Enumerated values are controlled by the DMTF, not this specification.

Byte Value	Meaning
01h	Other
02h	Unknown
03h	None
04h	Parity
05h	Single-bit ECC
06h	Multi-bit ECC
07h	CRC

3.2.18 Memory Device (Type 17)

This structure supports the population of the DMTF|Memory Device group, as defined in the DMTF's `MASTER.MIF`.

Note: If a system includes memory-device sockets, the SMBIOS implementation includes a *Memory Device* structure instance for each slot whether or not the socket is currently populated.

Offset	Spec Version	Name	Length	Value	Description
00h	2.1+	Type	BYTE	17	Memory Device type
01h	2.1+	Length	BYTE	15h	Length of the structure.
02h	2.1+	Handle	WORD	Varies	The handle, or instance number, associated with the structure.
04h	2.1+	Memory Array Handle	WORD	Varies	The handle, or instance number, associated with the Memory Array to which this device belongs.

Offset	Spec Version	Name	Length	Value	Description
06h	2.1+	Memory Error Information Handle	WORD	Varies	The handle, or instance number, associated with any error which was previously detected for the device. If the system does not provide the error information structure, the field contains FFFEh; otherwise, the field contains either FFFFh (if no error was detected) or the handle of the error-information structure. See also *3.2.19 Memory Error Information (Type 18)* on page 67
08h	2.1+	Total Width	WORD	Varies	The total width, in bits, of this memory device, including any check or error-correction bits. If there are no error-correction bits, this value should be equal to Data Width. If the width is unknown, the field is set to FFFFh.
0Ah	2.1+	Data Width	WORD	Varies	The data width, in bits, of this memory device. A Data Width of 0 and a Total Width of 8 indicates that the device is being used solely to provide 8 error-correction bits. If the width is unknown, the field is set to FFFFh.
0Ch	2.1+	Size	WORD	Varies	The size of the memory device. If the value is 0, no memory device is installed in the socket; if the size is unknown, the field value is FFFFh. The granularity in which the value is specified depends on the setting of the most-significant bit (bit 15). If the bit is 0, the value is specified in megabyte units; if the bit is 1, the value is specified in kilobyte units. For example, the value 8100h identifies a 256KB memory device and 0100h identifies a 256MB memory device.
0Eh	2.1+	Form Factor	BYTE	ENUM	The implementation form factor for this memory device. See 3.2.18.1 for definitions.
0Fh	2.1+	Device Set	BYTE	Varies	Identifies when the Memory Device is one of a set of Memory Devices that must be populated with all devices of the same type and size, and the set to which this device belongs. A value of 0 indicates that the device is not part of a set; a value of FFh indicates that the attribute is unknown. ***Note***: A *Device Set* number must be unique within the context of the Memory Array containing this Memory Device.
10h	2.1+	Device Locator	BYTE	STRING	The string number of the string that identifies the physically-labelled socket or board position where the memory device is located, e.g. “SIMM 3”.
11h	2.1+	Bank Locator	BYTE	STRING	The string number of the string that identifies the physically-labelled bank where the memory device is located, e.g. “Bank 0” or “A”.
12h	2.1+	Memory Type	BYTE	ENUM	The type of memory used in this device, see 3.2.18.2 for definitions.
13h	2.1+	Type Detail	WORD	Bit Field	Additional detail on the memory device type, see 3.2.18.3 for definitions.

3.2.18.1 Memory Device — Form Factor

Important Note: Enumerated values are controlled by the DMTF, not this specification.

Byte Value	Meaning
01h	Other
02h	Unknown
03h	SIMM
04h	SIP
05h	Chip
06h	DIP
07h	ZIP
08h	Proprietary Card
09h	DIMM
0Ah	TSOP
0Bh	Row of chips
0Ch	RIMM

3.2.18.2 Memory Device — Type

Important Note: Enumerated values are controlled by the DMTF, not this specification.

Byte Value	Meaning
01h	Other
02h	Unknown
03h	DRAM
04h	EDRAM
05h	VRAM
06h	SRAM
07h	RAM
08h	ROM
09h	FLASH
0Ah	EEPROM
0Bh	FEPROM
0Ch	EPROM

3.2.18.3 Memory Device — Type Detail

Important Note: Bit-field values are controlled by the DMTF, not this specification.

Note: Multiple bits are set if more than one attribute applies.

Word Bit Position	Meaning
Bit 0	Reserved, set to 0.
Bit 1	Other
Bit 2	Unknown
Bit 3	Fast-paged
Bit 4	Static column
Bit 5	Pseudo-static
Bit 6	RAMBUS
Bit 7	Synchronous
Bit 8	CMOS
Bit 9	EDO
Bit 10	Window DRAM
Bit 11	Cache DRAM
Bit 12	Non-volatile
Bits 13:15	Reserved, set to 0.

3.2.19 Memory Error Information (Type 18)

This structure supports the population of the DMTF|Physical Memory Array and DMTF|Memory Device groups, as defined in the DMTF's `MASTER.MIF`. The *Last Error Update* field, present in those groups, is not supplied in this structure since that field's attribute is known at the system-management application layer, not the BIOS.

Offset	Spec Version	Name	Length	Value	Description
00h	2.1+	Type	BYTE	18	Memory Error Information type
01h	2.1+	Length	BYTE	17h	Length of the structure.
02h	2.1+	Handle	WORD	Varies	The handle, or instance number, associated with the structure.
04h	2.1+	Error Type	BYTE	ENUM	The type of error that is associated with the current status reported for the memory array or device. See 3.2.19.1 for definitions..
05h	2.1+	Error Granularity	BYTE	ENUM	Identifies the granularity, e.g. device vs. Partition, to which the error can be resolved. See 3.2.19.2 for definitions.
06h	2.1+	Error Operation	BYTE	ENUM	The memory access operation that caused the error. See 3.2.19.3 for definitions.
07h	2.1+	Vendor Syndrome	DWORD	Varies	The vendor-specific ECC syndrome or CRC data associated with the erroneous access. If the value is unknown, this field contains 0000 0000h.
0Bh	2.1+	Memory Array Error Address	DWORD	Varies	The 32-bit physical address of the error based on the addressing of the bus to which the memory array is connected. If the address is unknown, this field contains 8000 0000h.

Offset	Spec Version	Name	Length	Value	Description
0Fh	2.1+	Device Error Address	DWORD	Varies	The 32-bit physical address of the error relative to the start of the failing memory device, in bytes. If the address is unknown, this field contains 8000 0000h.
13h	2.1+	Error Resolution	DWORD	Varies	The range, in bytes, within which the error can be determined, when an error address is given. If the range is unknown, this field contains 8000 0000h.

3.2.19.1 Memory Error — Error Type

Important Note: Enumerated values are controlled by the DMTF, not this specification.

Byte Value	Meaning
01h	Other
02h	Unknown
03h	OK
04h	Bad read
05h	Parity error
06h	Single-bit error
07h	Double-bit error
08h	Multi-bit error
09h	Nibble error
0Ah	Checksum error
0Bh	CRC error
0Ch	Corrected single-bit error
0Dh	Corrected error
0Eh	Uncorrectable error

3.2.19.2 Memory Error — Error Granularity

Important Note: Enumerated values are controlled by the DMTF, not this specification.

Byte Value	Meaning
01h	Other
02h	Unknown
03h	Device level
04h	Memory partition level

3.2.19.3 Memory Error — Error Operation

Important Note: Enumerated values are controlled by the DMTF, not this specification.

Byte Value	Meaning
01h	Other
02h	Unknown
03h	Read
04h	Write
05h	Partial write

3.2.20 Memory Array Mapped Address (Type 19)

This structure supports the population of the DMTF|Memory Array Mapped Addresses group, as defined in the DMTF's MASTER.MIF. One structure is present for each contiguous address range described.

See also *3.2.17 Physical Memory Array (Type 16)* on page 62, *3.2.18 Memory Device (Type 17)* on page 64, and *3.2.21 Memory Device Mapped Address (Type 20)* on page 69.

Offset	Spec Version	Name	Length	Value	Description
00h	2.1+	Type	BYTE	19	Memory Array Mapped Address indicator
01h	2.1+	Length	BYTE	0Fh	Length of the structure.
02h	2.1+	Handle	WORD	Varies	The handle, or instance number, associated with the structure.
04h	2.1+	Starting Address	DWORD	Varies	The physical address, in kilobytes, of a range of memory mapped to the specified *Physical Memory Array.*
08h	2.1+	Ending Address	DWORD	Varies	The physical ending address of the last kilobyte of a range of addresses mapped to the specified *Physical Memory Array.*
0Ch	2.1+	Memory Array Handle	WORD	Varies	The handle, or instance number, associated with the *Physical Memory Array* to which this address range is mapped. Multiple address ranges can be mapped to a single *Physical Memory Array*.
0Eh	2.1+	Partition Width	BYTE	Varies	Identifies the number of *Memory Devices* that form a single row of memory for the address partition defined by this structure.

3.2.21 Memory Device Mapped Address (Type 20)

This structure supports the population of the DMTF|Memory Device Mapped Addresses group, as defined in the DMTF's MASTER.MIF. One structure is present for each contiguous address range described.

See also *3.2.17 Physical Memory Array (Type 16)* on page 62, *3.2.18 Memory Device (Type 17)* on page 64, and *3.2.20 Memory Array Mapped Address (Type 19)* on page 69.

Offset	Spec Version	Name	Length	Value	Description
00h	2.1+	Type	BYTE	20	Memory Device Mapped Address indicator
01h	2.1+	Length	BYTE	13h	Length of the structure.
02h	2.1+	Handle	WORD	Varies	The handle, or instance number, associated with the structure.
04h	2.1+	Starting Address	DWORD	Varies	The physical address, in kilobytes, of a range of memory mapped to the referenced *Memory Device.*
08h	2.1+	Ending Address	DWORD	Varies	The physical ending address of the last kilobyte of a range of addresses mapped to the referenced *Memory Device*.
0Ch	2.1+	Memory Device Handle	WORD	Varies	The handle, or instance number, associated with the *Memory Device* structure to which this address range is mapped. Multiple address ranges can be mapped to a single *Memory Device*.

Offset	Spec Version	Name	Length	Value	Description
0Eh	2.1+	Memory Array Mapped Address Handle	WORD	Varies	The handle, or instance number, associated with the *Memory Array Mapped Address* structure to which this device address range is mapped. Multiple address ranges can be mapped to a single *Memory Array Mapped Address*.
10h	2.1+	Partition Row Position	BYTE	Varies	Identifies the position of the referenced *Memory Device* in a row of the address partition. For example, if two 8-bit devices form a 16-bit row, this field's value will be either 1 or 2. The value 0 is reserved; if the position is unknown, the field contains FFh.
11h	2.1+	Interleave Position	BYTE	Varies	The position of the referenced *Memory Device* in an interleave. The value 0 indicates non-interleaved, 1 indicates first interleave postion, 2 the second, and so on. If the position is unknown, the field contains FFh. For example: in a 2:1 interleave, the value 1 indicates the device in the 'even' position; in a 4:1 interleave, the value 1 indicates the first of four possible positions.
12h	2.1+	Interleaved Data Depth	BYTE	Varies	The maximum number of consecutive rows from the referenced *Memory Device* that are accessed in a single interleaved transfer. If the device is not part of an interleave, the field contains 0; if the interleave configuration is unknown, the value is FFh. For example, if a device transfers two rows each time it is read, its *Interleaved Data Depth* is set to 2. If that device is 2:1 interleaved and in Interleave Position 1, the rows mapped to that device are 1, 2, 5, 6, 9, 10, etc.

3.2.22 Built-in Pointing Device (Type 21)

This structure supports the population of the DMTF|Pointing Device group, as defined in the *DMTF Mobile Supplement to Standard Groups, v1.0* and describes the attributes of the built-in pointing device for the system — the presence of this structure does not imply that the built-in pointing device is active for the system's use!

Offset	Spec Version	Name	Length	Value	Description
00h	2.1+	Type	BYTE	21	Built-in Pointing Device indicator
01h	2.1+	Length	BYTE	07h	Length of the structure.
02h	2.1+	Handle	WORD	Varies	The handle, or instance number, associated with the structure.
04h	2.1+	Type	BYTE	ENUM	The type of pointing device, see 3.2.22.1.
05h	2.1+	Interface	BYTE	ENUM	The interface type for the pointing device, see 3.2.22.2.

Offset	Spec Version	Name	Length	Value	Description
06h	2.1+	Number of Buttons	BYTE	Varies	The number of buttons on the pointing device. If the device has three buttons, the field value is 03h.

3.2.22.1 Pointing Device — Type

Important Note: Enumerated values are controlled by the DMTF, not this specification.

Byte Value	Meaning
01h	Other
02h	Unknown
03h	Mouse
04h	Track Ball
05h	Track Point
06h	Glide Point
07h	Touch Pad

3.2.22.2 Pointing Device — Interface

Important Note: Enumerated values are controlled by the DMTF, not this specification.

Byte Value	Meaning
01h	Other
02h	Unknown
03h	Serial
04h	PS/2
05h	Infrared
06h	HP-HIL
07h	Bus mouse
08h	ADB (Apple Desktop Bus)
A0h	Bus mouse DB-9
A1h	Bus mouse micro-DIN
A2h	USB

3.2.23 Portable Battery (Type 22)

This structure supports the population of the DMTF|Portable Battery group, as defined in the *DMTF Mobile Supplement to Standard Groups, v1.0* and describes the attributes of the portable battery(s) for the system. The structure contains the static attributes for the group. Each structure describes a single battery pack's attributes.

Offset	Spec Version	Name	Length	Value	Description
00h	2.1+	Type	BYTE	22	Portable Battery indicator
01h	2.1+	Length	BYTE	1Ah	Length of the structure.
02h	2.1+	Handle	WORD	Varies	The handle, or instance number, associated with the structure.
04h	2.1+	Location	BYTE	STRING	The number of the string that identifies the location of the battery, e.g. "in the back, on the left-hand side."
05h	2.1+	Manufacturer	BYTE	STRING	The number of the string that names the company that manufactured the battery.

Offset	Spec Version	Name	Length	Value	Description
06h	2.1+	Manufacture Date	BYTE	STRING	The number of the string that identifies the date on which the battery was manufactured. V2.2+ implementations which use a Smart Battery will set this field to 0 (no string) to indicate that the *SBDS Manufacture Date* field contains the information.
07h	2.1+	Serial Number	BYTE	STRING	The number of the string that contains the serial number for the battery. V2.2+ implementations which use a Smart Battery will set this field to 0 (no string) to indicate that the *SBDS Serial Number* field contains the information.
08h	2.1+	Device Name	BYTE	STRING	The number of the string that names the battery device, e.g. “DR-36”.
09h	2.1+	Device Chemistry	BYTE	ENUM	Identifies the battery chemistry, see 3.2.23.1. V2.2+ implementations which use a Smart Battery will set this field to 02h (*Unknown*) to indicate that the *SBDS Device Chemistry* field contains the information.
0Ah	2.1+	Design Capacity	WORD	Varies	The design capacity of the battery in mWatt-hours. If the value is unknown, the field contains 0. For v2.2+ implementations, this value is multiplied by the *Design Capacity Multiplier* to produce the actual value.
0Ch	2.1+	Design Voltage	WORD	Varies	The design voltage of the battery, in mVolts. If the value is unknown, the field contains 0.
0Eh	2.1+	SBDS Version Number	BYTE	STRING	The number of the string that contains the *Smart Battery Data Specification* version number supported by this battery. If the battery does not support the function, no string is supplied.
0Fh	2.1+	Maximum Error in Battery Data	BYTE	Varies	The maximum error (as a percentage in the range 0 to 100) in the Watt-hour data reported by the battery, indicating an upper bound on how much additional energy the battery might have above the energy it reports having. If the value is unknown, the field contains FFh.
10h	2.2+	SBDS Serial Number	WORD	Varies	The 16-bit value that identifies the battery’s serial number. This value, when combined with the Manufacturer, Device Name, and Manufacture Date will uniquely identify the battery. The *Serial Number* field must be set to 0 (no string) for this field to be valid.

Offset	Spec Version	Name	Length	Value	Description
12h	2.2+	SBDS Manufacture Date	WORD	Varies	The date the cell pack was manufactured, in packed format: Bits 15:9 Year, biased by 1980, in the range 0 to 127. Bits 8:5 Month, in the range 1 to 12. Bits 4:0 Date, in the range 1 to 31. For example, 01 February 2000 would be identified as 0010 1000 0100 0001b (0x2841). The *Manufacture Date* field must be set to 0 (no string) to for this field to be valid.
14h	2.2+	SBDS Device Chemistry	BYTE	STRING	The number of the string that identifies the battery chemistry, e.g. “PbAc”. The *Device Chemistry* field must be set to 02h (*Unknown*) for this field to be valid.
15h	2.2+	Design Capacity Multiplier	BYTE	Varies	The multiplication factor of the *Design Capacity* value and assures that the mWatt hours value does not overflow for SBDS implementations. The multiplier default is 1, SBDS implementations use the value 10 to correspond to the data as returned from the SBDS Function 18h.
16h	2.2+	OEM-specific	DWORD	Varies	Contains OEM- or BIOS vendor-specific information.

3.2.23.1 Portable Battery — Device Chemistry

Important Note: Enumerated values are controlled by the DMTF, not this specification.

Byte Value	Meaning
01h	Other
02h	Unknown
03h	Lead Acid
04h	Nickel Cadmium
05h	Nickel metal hydride
06h	Lithium-ion
07h	Zinc air
08h	Lithium Polymer

3.2.24 System Reset (Type 23)

This structure supports the population of the DMTF|System Reset group, as defined in the DMTF's MASTER.MIF and describes whether Automatic System Reset functions enabled (*Status*). If the system has a watchdog Timer and the timer is not reset (*Timer Reset*) before the *Interval* elapses, an automatic system reset will occur. The system will re-boot according to the *Boot Option*. This function may repeat until the *Limit* is reached, at which time the system will re-boot according to the *Boot Option at Limit*.

Note: This structure type was added for specification v2.2.

Offset	Name	Length	Value	Description
00h	Type	BYTE	23	System Reset indicator
01h	Length	BYTE	0Dh	Length of the structure.
02h	Handle	WORD	Varies	The handle, or instance number, associated with the structure.
04h	Capabilities	BYTE	Bit-field	Identifies the system reset capabilities for the system. Bits 7:6 Reserved for future assignment via this specification, set to 00b. Bit 5 System contains a watchdog timer, either True (1) or False (0). Bits 4:3 *Boot Option on Limit*. Identifies the system action to be taken when the Reset Limit is reached, one of: 00b Reserved, do not use. 01b Operating system 10b System utilities 11b Do not reboot Bits 2:1 *Boot Option*. Indicates the action to be taken following a watchdog reset, one of: 00b Reserved, do not use. 01b Operating system 10b System utilities 11b Do not reboot Bit 0 *Status*. Identifies whether (1) or not (0) the system reset is enabled by the user.
05h	Reset Count	WORD	Varies	The number of automatic system resets since the last intentional reset. A value of 0FFFFh indicates unknown.
07h	Reset Limit	WORD	Varies	The number of consecutive times the system reset will be attempted. A value of 0FFFFh indicates unknown.
09h	Timer Interval	WORD	Varies	The number of minutes to use for the watchdog timer. If the timer is not reset within this interval, the system reset timeout will begin. A value of 0FFFFh indicates unknown.
0Bh	Timeout	WORD	Varies	Identifies the number of minutes before the reboot is initiated. It is used after a system power cycle, system reset (local or remote), and automatic system reset. A value of 0FFFFh indicates unknown.

3.2.25 Hardware Security (Type 24)

This structure supports the population of the DMTF|Hardware Security group, as defined in the DMTF's MASTER.MIF and describes the system-wide hardware security settings.

Note: This structure type was added for specification v2.2.

Offset	Name	Length	Value	Description
00h	Type	BYTE	24	Hardware Security indicator
01h	Length	BYTE	05h	Length of the structure.
02h	Handle	WORD	Varies	The handle, or instance number, associated with the structure.
04h	Hardware Security Settings	BYTE	Bit-field	Identifies the password and reset status for the system: Bits 7:6 *Power-on Password Status*, one of: 00b Disabled 01b Enabled 10b Not Implemented 11b Unknown Bits 5:4 *Keyboard Password Status*, one of: 00b Disabled 01b Enabled 10b Not Implemented 11b Unknown Bits 3:2 *Administrator Password Status*, one of: 00b Disabled 01b Enabled 10b Not Implemented 11b Unknown Bits 1:0 *Front Panel Reset Status*, one of: 00b Disabled 01b Enabled 10b Not Implemented 11b Unknown

3.2.26 System Power Controls (Type 25)

This structure supports the population of the DMTF|System Power Controls group, as defined in the DMTF's `MASTER.MIF` and describes the attributes for controlling the main power supply to the system. Software that interprets this structure uses the month, day, hour, minute, and second values to determine the number of seconds until the next power-on of the system. The presence of this structure implies that a timed power-on facility is available for the system.

Note: This structure type was added for specification v2.2.

Offset	Name	Length	Value	Description
00h	Type	BYTE	25	System Power Controls indicator
01h	Length	BYTE	09h	Length of the structure.
02h	Handle	WORD	Varies	The handle, or instance number, associated with the structure.
04h	Next Scheduled Power-on Month	BYTE	Varies	Contains the BCD value of the month on which the next scheduled power-on is to occur, in the range 01h to 12h. See 3.2.26.1.
05h	Next Scheduled Power-on Day-of-month	BYTE	Varies	Contains the BCD value of the day-of-month on which the next scheduled power-on is to occur, in the range 01h to 31h. See 3.2.26.1.
06h	Next Scheduled Power-on Hour	BYTE	Varies	Contains the BCD value of the hour on which the next scheduled power-on is to occur, in the range 00h to 23h. See 3.2.26.1.

Offset	Name	Length	Value	Description
07h	Next Scheduled Power-on Minute	BYTE	Varies	Contains the BCD value of the minute on which the next scheduled power-on is to occur, in the range 00h to 59h. See 3.2.26.1.
08h	Next Scheduled Power-on Second	BYTE	Varies	Contains the BCD value of the second on which the next scheduled power-on is to occur, in the range 00h to 59h. See 3.2.26.1.

3.2.26.1 System Power Controls — Calculating the Next Scheduled Power-on Time

The DMTF *System Power Controls* group contains a *Next Scheduled Power-on Time*, specified as the number of seconds until the next scheduled power-on of the system. Management software uses the date and time information specified in the associated SMBIOS structure to calculate the total number of seconds.

Any date or time field in the structure whose value is outside of the field's specified range does not contribute to the total-seconds count. For example, if the Month field contains the value 0xFF the next power-on is scheduled to fall within the next month, perhaps on a specific day-of-month and time.

3.2.27 Voltage Probe (Type 26)

This structure supports the population of the DMTF|Voltage Probe group, as defined in the DMTF's `MASTER.MIF` and describes the attributes for a voltage probe in the system. Each structure describes a single voltage probe.

Note: This structure type was added for specification v2.2.

Offset	Name	Length	Value	Description
00h	Type	BYTE	26	Voltage Probe indicator
01h	Length	BYTE	14h	Length of the structure.
02h	Handle	WORD	Varies	The handle, or instance number, associated with the structure.
04h	Description	BYTE	STRING	The number of the string that contains additional descriptive information about the probe or its location.
05h	Location and Status	BYTE	Bit-field	Defines the probe's physical location and status of the voltage monitored by this voltage probe. See 3.2.27.1.
06h	Maximum Value	WORD	Varies	The maximum voltage level readable by this probe, in millivolts. If the value is unknown, the field is set to 0x8000.
08h	Minimum Value	WORD	Varies	The minimum voltage level readable by this probe, in millivolts. If the value is unknown, the field is set to 0x8000.
0Ah	Resolution	WORD	Varies	The resolution for the probe's reading, in tenths of millivolts. If the value is unknown, the field is set to 0x8000.
0Ch	Tolerance	WORD	Varies	The tolerance for reading from this probe, in plus/minus millivolts. If the value is unknown, the field is set to 0x8000.
0Eh	Accuracy	WORD	Varies	The accuracy for reading from this probe, in plus/minus $1/100^{th}$ of a percent. If the value is unknown, the field is set to 0x8000.
10h	OEM-defined	DWORD	Varies	Contains OEM- or BIOS vendor-specific information.

3.2.27.1 Voltage Probe — Location and Status

Important Note: Each of the bit-field values map to enumerated values which are controlled by the DMTF, not this specification.

Bit Range	Field Name	Value	Meaning
7:5	Status	001.....	Other
		010.....	Unknown
		011.....	OK
		100.....	Non-critical
		101.....	Critical
		110.....	Non-recoverable
4:0	Location	...00001	Other
		...00010	Unknown
		...00011	Processor
		...00100	Disk
		...00101	Peripheral Bay
		...00110	System Management Module
		...00111	Motherboard
		...01000	Memory Module
		...01001	Processor Module
		...01010	Power Unit
		...01011	Add-in Card

3.2.28 Cooling Device (Type 27)

This structure supports the population of the DMTF|Cooling Device group, as defined in the DMTF's MASTER.MIF and describes the attributes for a cooling device in the system. Each structure describes a single cooling device.

Note: This structure type was added for specification v2.2.

Offset	Name	Length	Value	Description
00h	Type	BYTE	27	Cooling Device indicator
01h	Length	BYTE	0Ch	Length of the structure.
02h	Handle	WORD	Varies	The handle, or instance number, associated with the structure.
04h	Temperature Probe Handle	WORD	Varies	The handle, or instance number, of the temperature probe (see *3.2.29 Temperature Probe (Type 28)* on page 78) monitoring this cooling device. A value of 0xFFFF indicates that no probe is provided.
06h	Device Type and Status	BYTE	Bit-field	Identifies the cooling device type and the status of this cooling device, see 3.2.28.1.
07h	Cooling Unit Group	BYTE	Varies	Identifies the cooling unit group to which this cooling device is associated. Multiple cooling devices in the same cooling unit implies a redundant configuration. The value is 00h if the cooling device is not a member of a redundant cooling unit, non-zero values imply redundancy and that at least one other cooling device will be enumerated with the same value.
08h	OEM-defined	DWORD	Varies	Contains OEM- or BIOS vendor-specific information.

3.2.28.1 Cooling Device —Device Type and Status

Important Note: Each of the bit-field values map to enumerated values which are controlled by the DMTF, not this specification.

Bit Range	Field Name	Value	Meaning
7:5	Status	001.....	Other
		010.....	Unknown
		011.....	OK
		100.....	Non-critical
		101.....	Critical
		110.....	Non-recoverable
4:0	Device Type	...00001	Other
		...00010	Unknown
		...00011	Fan
		...00100	Centrifugal Blower
		...00101	Chip Fan
		...00110	Cabinet Fan
		...00111	Power Supply Fan
		...01000	Heat Pipe
		...01001	Integrated Refrigeration
		...10100	Active Cooling
		...10101	Passive Cooling

3.2.29 Temperature Probe (Type 28)

This structure supports the population of the DMTF|Temperature Probe group, as defined in the DMTF's MASTER.MIF and describes the attributes for a temperature probe in the system. Each structure describes a single temperature probe.

Note: This structure type was added for specification v2.2.

Offset	Name	Length	Value	Description
00h	Type	BYTE	28	Temperature Probe indicator
01h	Length	BYTE	14h	Length of the structure.
02h	Handle	WORD	Varies	The handle, or instance number, associated with the structure.
04h	Description	BYTE	STRING	The number of the string that contains additional descriptive information about the probe or its location.
05h	Location and Status	BYTE	Bit-field	Defines the probe's physical location and the status of the temperature monitored by this temperature probe. See 3.2.29.1.
06h	Maximum Value	WORD	Varies	The maximum temperature readable by this probe, in $1/10^{th}$ degrees C. If the value is unknown, the field is set to 0x8000.
08h	Minimum Value	WORD	Varies	The minimum temperature readable by this probe, in $1/10^{th}$ degrees C. If the value is unknown, the field is set to 0x8000.
0Ah	Resolution	WORD	Varies	The resolution for the probe's reading, in $1/1000^{th}$ degrees C. If the value is unknown, the field is set to 0x8000.

Offset	Name	Length	Value	Description
0Ch	Tolerance	WORD	Varies	The tolerance for reading from this probe, in plus/minus 1/10th degrees C. If the value is unknown, the field is set to 0x8000.
0Eh	Accuracy	WORD	Varies	The accuracy for reading from this probe, in plus/minus 1/100th of a percent. If the value is unknown, the field is set to 0x8000.
10h	OEM-defined	DWORD	Varies	Contains OEM- or BIOS vendor-specific information.

3.2.29.1 Temperature Probe — Location and Status

Important Note: Each of the bit-field values map to enumerated values which are controlled by the DMTF, not this specification.

Bit Range	Field Name	Value	Meaning
7:5	Status	001.....	Other
		010.....	Unknown
		011.....	OK
		100.....	Non-critical
		101.....	Critical
		110.....	Non-recoverable
4:0	Location	...00001	Other
		...00010	Unknown
		...00011	Processor
		...00100	Disk
		...00101	Peripheral Bay
		...00110	System Management Module
		...00111	Motherboard
		...01000	Memory Module
		...01001	Processor Module
		...01010	Power Unit
		...01011	Add-in Card

3.2.30 Electrical Current Probe (Type 29)

This structure supports the population of the DMTF|Electrical Current Probe group, as defined in the DMTF's MASTER.MIF and describes the attributes for an electrical current probe in the system. Each structure describes a single electrical current probe.

Note: This structure type was added for specification v2.2.

Offset	Name	Length	Value	Description
00h	Type	BYTE	29	Electrical Current Probe indicator
01h	Length	BYTE	14h	Length of the structure.
02h	Handle	WORD	Varies	The handle, or instance number, associated with the structure.
04h	Description	BYTE	STRING	The number of the string that contains additional descriptive information about the probe or its location.
05h	Location and Status	BYTE	ENUM	Defines the probe's physical location and the status of the current monitored by this current probe. See 3.2.30.1.

Offset	Name	Length	Value	Description
06h	Maximum Value	WORD	Varies	The maximum current readable by this probe, in milliamps. If the value is unknown, the field is set to 0x8000.
08h	Minimum Value	WORD	Varies	The minimum current readable by this probe, in milliamps. If the value is unknown, the field is set to 0x8000.
0Ah	Resolution	WORD	Varies	The resolution for the probe's reading, in tenths of milliamps. If the value is unknown, the field is set to 0x8000.
0Ch	Tolerance	WORD	Varies	The tolerance for reading from this probe, in plus/minus milliamps. If the value is unknown, the field is set to 0x8000.
0Eh	Accuracy	WORD	Varies	The accuracy for reading from this probe, in plus/minus 1/100th of a percent. If the value is unknown, the field is set to 0x8000.
10h	OEM-defined	DWORD	Varies	Contains OEM- or BIOS vendor-specific information.

3.2.30.1 Current Probe — Location and Status

Important Note: Each of the bit-field values map to enumerated values which are controlled by the DMTF, not this specification.

Bit Range	Field Name	Value	Meaning
7:5	Status	001.....	Other
		010.....	Unknown
		011.....	OK
		100.....	Non-critical
		101.....	Critical
		110.....	Non-recoverable
4:0	Location	...00001	Other
		...00010	Unknown
		...00011	Processor
		...00100	Disk
		...00101	Peripheral Bay
		...00110	System Management Module
		...00111	Motherboard
		...01000	Memory Module
		...01001	Processor Module
		...01010	Power Unit
		...01011	Add-in Card

3.2.31 Out-of-Band Remote Access (Type 30)

This structure supports the population of the DMTF|Out-of-Band Remote Access group, as defined in the DMTF's MASTER.MIF and describes the attributes and policy settings of a hardware facility which may be used to gain remote access to a hardware system when the operating system is not available due to power-down status, hardware failures, or boot failures.

Note: This structure type was added for specification v2.2.

Offset	Name	Length	Value	Description
00h	Type	BYTE	30	Out-of-Band Remote Access indicator
01h	Length	BYTE	06h	Length of the structure.
02h	Handle	WORD	Varies	The handle, or instance number, associated with the structure.
04h	Manufacturer Name	BYTE	STRING	The number of the string that contains the manufacturer of the out-of-band access facility.
05h	Connections	BYTE	Bit-field	Identifies the current remote-access connections: Bits 7:2 Reserved for future definition by this specification, set to all zeros. Bit 1 *Outbound Connection Enabled*. Identifies whether (1) or not (0) the facility is allowed to initiate outbound connections to contact an alert management facility when critical conditions occur. Bit 0 *Inbound Connection Enabled*. Identifies whether (1) or not (0) the facility is allowed to initiate outbound connections to receive incoming connections for the purpose of remote operations or problem management

3.2.32 Inactive (Type 126)

This structure definition supports a system implementation where the SMBIOS structure-table is a superset of all supported system attributes and provides a standard mechanism for the system BIOS to signal that a structure is currently inactive and should not be interpreted by the upper-level software.

For example, a portable system might include *System Slot* structures which are reported only when the portable has docked. An undocked system would report those structures as *Inactive*. When the system was docked, the structure Type would be changed from *Inactive* to the *System Slot* equivalent by the system-specific software.

Upper-level software that interprets the SMBIOS structure-table should bypass an *Inactive* structure just like a structure type that the software does not recognize.

Note: This structure type was added for specification v2.2.

Offset	Name	Length	Value	Description
00h	Type	BYTE	126	Inactive structure indicator
01h	Length	BYTE	Varies	Length of the structure.
02h	Handle	WORD	Varies	The handle, or instance number, associated with the structure.

3.2.33 End-of-Table (Type 127)

This structure type identifies the end of the structure table, which might be earlier than the last byte within the buffer specified by the structure. To ensure backward compatibility with management software written to previous versions of this specification, a system implementation should use the end-of-table indicator in a manner similar to the *Inactive (Type 126)* structure type — the structure table is still reported as a fixed-length and the entire length of the table is still indexable. If the end-of-table indicator is used in the last physical structure in a table, the field's length is encoded as 4.

Note: This structure type was added for specification v2.2.

Offset	Name	Length	Value	Description
00h	Type	BYTE	127	End-of-table indicator.
01h	Length	BYTE	Varies	Length of the structure.
02h	Handle	WORD	Varies	The handle, or instance number, associated with the structure.

Appendices

4. Structure Checklist

This checklist identifies which System Management BIOS structures are needed to populate DMTF groups required by DMI 2.0 (per *DMTF Desktop Management Interface (DMI) 2.0 Conformance Requirements*, Version 1.0).

Structure Type	Needed for DMTF DMI 2.0	DMTF Group Required	DMTF Group Supported
BIOS Information (Type 0)	Recommended	Yes	DMTF|System BIOS|001
System Information (Type 1)	Recommended	Yes	DMTF|ComponentID|001
Base Board Information (Type 2)	Optional		
System Enclosure or Chassis (Type 3)	Recommended	Yes	DMTF|Physical Container Global Table|001
Processor Information (Type 4)	Recommended, one entry per processor socket	Yes	DMTF|Processor|003
Memory Controller Information (Type 5)	Optional, types 16 and 17 preferred		
Memory Module Information (Type 6)	Optional, types 16 and 17 preferred		
Cache Information (Type 7)	Recommended, one entry per processor cache	Yes	DMTF|System Cache|002
Port Connector Information (Type 8)	Recommended		
System Slots (Type 9)	Recommended	Yes	DMTF|System Slots|003
On Board Devices Information (Type 10)	Optional		
OEM Strings (Type 11)	Optional		
System Configuration Options (Type 12)	Optional		
BIOS Language Information (Type 13)	Optional		
Group Associations (Type 14)	Optional		
System Event Log (Type 15)	Recommended		
Physical Memory Array (Type 16)	Recommended, if types 5 and 6 are not present	Yes	DMTF|Physical Memory Array|001
Memory Device (Type 17)	Recommended, if types 5 and 6 are not present	Yes	DMTF|Memory Device|001
Memory Error Information (Type 18)	Recommended, if a memory-related error occurred		DMTF|Physical Memory Array|001 and DMTF|Memory Device|001
Memory Array Mapped Address (Type 19)	Recommended	Yes	DMTF|Memory Array Mapped Addresses|001
Memory Device Mapped Address (Type 20)	Recommended	Yes	DMTF|Memory Device Mapped Addresses|001

Structure Type	Needed for DMTF DMI 2.0	DMTF Group Required	DMTF Group Supported
Built-in Pointing Device (Type 21)	Recommended for portable systems	Mobile	DMTF|Pointing Device|001
Portable Battery (Type 22)	Recommended for portable systems.	Mobile	DMTF|Portable Battery|001
System Reset (Type 23)	Recommended[4]		DMTF|System Reset|001
Hardware Security (Type 24)	Recommended[4]		DMTF|Physical Hardware Security|001
System Power Controls (Type 25)	Recommended[4]		DMTF|System Power Controls|001
Voltage Probe (Type 26)	Recommended[4]		DMTF|Voltage Probe|001
Cooling Device (Type 27)	Recommended[4]	Server	DMTF|Cooling Device|001
Temperature Probe (Type 28)	Recommended[4]		DMTF|Temperature Probe|001
Electrical Current Probe (Type 29)	Recommended[4]		DMTF|Electrical Current Probe|001
Out-of-Band Remote Access (Type 30)	Recommended[4]		DMTF|Out-of-Band Remote Access|001
Inactive (Type 126)	Optional, use as needed		
End-of-Table (Type 127)	Required for SMBIOS 2.2 implementations.		

[4] If the feature is present in the system.

5. Using the Table Convention

This section contains pseudo-code that describes the method that application software can use to parse the table-based SMBIOS structures. The example searches for the first structure of the type specified, returning the handle of the structure found or 0xFFFF if no structure of the type was found in the list. *TableAddress* and *StructureCount* values are those previously found by locating the Table Entry Point structure in low memory.

```
typedef unsigned short ushort;
typedef unsigned char  uchar;
typedef struct
  {
    uchar  Type;
    uchar  Length;
    ushort Handle;

  }  HEADER;

ushort FindStructure( char *TableAddress, ushort StructureCount, uchar Type )
  {
    ushort i, handle;
    uchar  lasttype;

    i      = 0;
    handle = 0xFFFF;

    while( i < StructureCount && handle == 0xFFFF )
      {
        i++;
        lasttype = ((HEADER *)TableAddress)->Type;
        if( lasttype == Type )
          {
            handle = ((HEADER *)TableAddress)->Handle;

          }  /* Found first structure of the requested type */
        else
          {
            TableAddress += ((HEADER *)TableAddress)->Length;
             while( *((int *)TableAddress) != 0 )
               {
                 TableAddress++;

               }  /* Get past trailing string-list */

             TableAddress += 2;

          }  /* Increment address to start of next structure */

      }  /* END while-loop looking for structure type */

    return handle;

  }  /* END FindStructure */
```