EFI Application Toolkit SMBIOS Library External Product Specification

Revision 1.0

June 6, 2000

ESG Server Software Technologies (SST)

Revision History

Date	Revision	Modifications
10/27/99	0.01	Initial Version
11/04/99	0.02	Refine interface, add more data structures
11/12/99	0.03	Refine interface, adjust parameters in API
12/15/99	0.1	Updated after detailed research and design
01/24/00	0.3	Released as part of the 0.7 Application Developers Toolkit
05/05/00	0.4	Updated to reflect being able to get a structure by just its handle
06/06/00	1.0	Added *SMBIOS_FreeStructure* interface and updates to types 0 and 38 structures.

Disclaimers

Table Of Contents

Table Of Figures

This page intentionally left blank

1 Introduction

This document provides the Specification for the SMBIOS Library of EFI Application Toolkit.

1.1 Scope

This Specification defines the content and features of the major components of the SMBIOS Library of EFI Application Toolkit. It provides detailed information about the implementation and general use of the product.

1.2 Target Audience

This Specification targets individuals who wish to understand the product functionality provided and the implementation details of SMBIOS Library. It is not a user manual, because some architecture and design information are included. The reader will also find information in this document to aid in understanding the functionality provided by the SMBIOS Interface on EFI.

1.3 Reference Documents

The following documents were useful in preparing this specification:

- *System Management BIOS Reference Specification Version 2.3.1 — 16 March 1999*
- *Extensible Firmware Interface Specification*. Version 0.91, July 30, 1999.
- *EFI Developer's Guide*. Version 0.2, July 14, 1999.
- *Extensible Firmware Interface Library Specification.* Version 0.2, July 14, 1999.
- *EFI Application Toolkit Product Requirements Document*. Revision 0.97, Sept. 27, 1999.

1.4 Product Overview

EFI SMBIOS Library provides SMBIOS accessing capability of the EFI Application Toolkit. It provides interfaces that can be used to access the SMBIOS structure table entry point and specified SMBIOS structure.

The following sections describe the execution requirements and operating environment of EFI SMBIOS Interface.

1.5 SMBIOS Library Functionality

The SMBIOS library provides interfaces to access the SMBIOS structure table entry point and specified SMBIOS structure, through the following library functions:

- SMBIOS_Initialize()
- SMBIOS_GetInformation()
- SMBIOS_GetStructure()
- SMBIOS_GetRawStructure()

- SMBIOS_FreeStructure()

The detailed description of these 5 functions can be found in later chapter.

Through SMBIOS library, SMBIOS information is retrieved in structure format that released user from dealing with data alignment and table search overhead.

2 EFI SMBIOS Library Design

2.1 SMBIOS 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.

2.2 Accessing SMBIOS Information

There are two access methods defined for the SMBIOS structures:

The first method, defined in v2.0 of this specification, provides the SMBIOS structures through a Plug-and-Play function interface.

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.

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.

Currently EFI SMBIOS library support is based on the table-based method.

2.3 Table Convention

The table convention allows the SMBIOS structures to be accessed under 32-bit protected-mode operating systems such as Microsoft Windows NT. This convention provides a searchable entry-point structure that contains a pointer to the packed SMBIOS structures residing somewhere in 32-bit physical address space.

2.3.1 SMBIOS Information

SMBIOS Entry Pointer Structure, which resides in a certain physical memory address range, contains the general information of SMBIOS. The Entry Point Structure definition is listed below.

```
//
// SMBIOS 2.2 Structure Table Entry Point
//
typedef struct SMBIOSTableEntryPoint
{
      UINT8  AnchorString[4];    // _SM_
      UINT8  EPSChecksum;        // Checksum of Entry Point Structure
      UINT8  EPLengh;            // Length of the Entry Point Structure
      UINT8  MajorVersion;              // Major version of the spec
      UINT8  MinorVersion;              // Minor version of the spec
      UINT16 MaxStructSize;      // Size of the largest SMBIOS Structure
      UINT8  EPRevision;         // Entry Point Revision
      UINT8  FormattedArea[5];   // The value present in the EP Revision field
                                 // Defines the interpretation to be placed
      UINT8  InterAnchorString[5];     // _DMI_
      UINT8  InterChecksum;      // Checksum of Intermediate EPS
      UINT16 StructTableLen;     // Total length of SMBIOS Structure Table
      UINT32 StructTableAddr;    // 32-bit physical starting address of the
                                 // Read-only Structure Table
      UINT16 NumStruct;          // Total number of structures
      UINT8  BCDRevision;        // Compliance with a revision of this spec
} SMBIOSEntryPointTable;
```

2.4 SMBIOS Structures

The total number of structures can be obtained from the SMBIOS Entry Point Structure. The System Information is presented to an application as a set of structures that are obtained by traversing the SMBIOS structure table referenced by the SMBIOS Entry Point Structure.

2.4.1 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.

All structure shall NOT be packed.

2.4.2 Structure Header Format

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

```
typedef struct SMBIOSHeader
{
      UINT8        StructType;
      UINT8        StructLength;
      UINT16       StructHandle;
} SMBIOSHeader;
```

2.4.3 Structure Definitions

There are more than 30 types of SMBIOS structures. For detailed information about the structure of each type, please refer to Appendix 2 and *SMBIOS Reference Specification*.

2.5 EFI SMBIOS Interface Initialization

In order to call any SMBIOS functions, the application must first call the SMBIOS_Initialize() function to initialize EFI interfaces.

3 SMBIOS Interfaces Definition

3.1 SMBIOS Error Code

```
#define EFI_SMBIOSERR(val)            EFIERR_OEM(0x30000 | val)

//
//  In addition to standard EFI status codes, this specification
//  defines additional return values which are compatible with
//  the EFI_ERROR() macro
//
```

Define	**Value**	**Meaning**
`EFI_SMBIOSERR_FAILURE`	`EFI_SMBIOSERR(1)`	Implementation specific error
`EFI_SMBIOSERR_STRUCT_NOT_FOUND`	`EFI_SMBIOSERR(2)`	The specified structure not found
`EFI_SMBIOSERR_TYPE_UNKNOWN`	`EFI_SMBIOSERR(3)`	The specified the type is un-known
`EFI_SMBIOSERR_UNSUPPORTED`	`EFI_SMBIOSERR(4)`	System does not support SMBIOS

3.2 SMBIOS_Initialize()

```
EFI_STATUS

SMBIOS_Initialize (
      IN    EFI_HANDLE       ImageHandle,
      IN    EFI_SYSTEM_TABLE*SystemTable
      );
```

Parameters

ImageHandle	Initialization parameter for Lib C
SystemTable	Standard EFI SystemTable for this application

Description

This function Initializes EFI SMBIOS interfaces.

Status Codes Returned

EFI_SUCCESS	Successfully initialize SMBIOS interface
EFI_SMBIOSERR_FAILURE	Failed to initialize SMBIOS interface
EFI_SMBIOSERR_UNSUPPORTED	System does not support SMBIOS feature

3.3 SMBIOS_GetTableEntryPoint ()

```
EFI_STATUS
SMBIOS_GetTableEntryPoint (
      IN OUT SMBIOSTableEntryPoint **pSMBIOSInfo
      );
```

Parameters

pSMBIOSInfo	Point to SMBIOS information table pointer from SMBIOS Structure Entry Point.

Description

This function gets the SMBIOS information from the Structure Entry Point.

User shall not allocate memory for the structure. User is responsible for freeing the memory allocated by ****pSMBIOSInfo*** through the *SMBIOS_FreeStructure()* routine.

Status Codes Returned

EFI_SUCCESS	Successfully get the SMBIOS information from the Structure Entry Point.
EFI_SMBIOSERR_STRUCT_NOT_FOUND	SMBIOS TableEntryPoint structure is not found

3.4 SMBIOS_GetStructure()

```
EFI_STATUS
SMBIOS_GetStructure (
     IN          UINT16     Type,
     IN          UINT16     Handle,
     IN OUT      VOID       **pStructureBuffer,
     IN OUT      UINT16     *Length,
     IN OUT      UINT16     *Key
     );
```

Parameters

Type	SMBIOS Structure type
Handle	Handle of specified SMBIOS structure
pStructureBuffer	Pointer to the pointer of retrieved buffer of specified SMBIOS structure
Length	Actual length of the specified SMBIOS structure buffer allocated
Key	Opaque data used by the library to optimize search

Description

This function gets the corresponding SMBIOS structure buffer according to specified type and handle.

The following table describes the behavior of this function for various combinations of *Type*, *Handle*, and *Key* values:

Type	Handle	*pKey	Behavior
!= 0xFFFF	!= 0xFFFF	Don’t care	Returns the structure with the Type and Handle specified.
!= 0xFFFF	== 0xFFFF	== 0xFFFF	Returns the first structure with the type specified.
!= 0xFFFF	== 0xFFFF	!= 0xFFFF	Returns the next structure with the type specified.
== 0xFFFF	!= 0xFFFF	Don’t care	Returns the structure with the Handle specified.
== 0xFFFF	== 0xFFFF	== 0xFFFF	Returns the first structure.
== 0xFFFF	== 0xFFFF	!= 0xFFFF	Returns the next structure.

As noted in the parameter descriptions, the *Key* is opaque data used by the library to optimize its searches. The only thing callers should do with this parameter is set it to *0xFFFF* to indicate the start of a search. Where the table specifies the Key value does not equal *0xFFFF* it means that it is using the value returned by a previous successful call.

At any time if this function call returns with a status code of *EFI_SMBIOSERR_STRUCT_NOT_FOUND*, it means there are no more structures of the specified type. The user shall not call the function again with the returned _*Key_.

User shall not allocate memory for the structure. User is responsible for freeing the memory allocated by ***pStructureBuffer* through the *SMBIO_FreeStructure()* routine.

If the type is unknown by the library, it returns *EFI_SMBIOSERR_TYPE_UNKNOWN* and the raw data will be put in the *StructureBuffer*.

NOTE: The *StructLength* field in the *SMBIOSHeader* that is part of the returned structure must not be used to indicate the size of the returned structure. That field indicates the length of the raw structure in its packed, unaligned form. The value returned in the function's Length argument reflects the true size of the returned structure.

Status Codes Returned

EFI_SUCCESS	Successfully get the specified SMBIOS structure buffer
EFI_SMBIOSERR_STRUCT_NOT_FOUND	The specified structure not found
EFI_SMBIOSERR_TYPE_UNKNOWN	The SMBIOS structure was not converted to a structure type known by the library

3.5 SMBIOS_GetRawStructure()

```
EFI_STATUS
SMBIOS_GetRawStructure (
      IN          UINT16      Type,
      IN          UINT16      Handle,
      IN OUT      VOID        **pRawBuffer,
      IN OUT      UINT16      *Length,
      IN OUT      UINT16      *Key
      );
```

Parameters

Type	SMBIOS Structure type
Handle	Handle of specified SMBIOS structure
pRawBuffer	Pointer to the pointer of retrieved raw buffer of specified SMBIOS structure
Length	Actual length of the specified SMBIOS structure buffer allocated
Key	Opaque data used by the library to optimize search

Description

This function gets the corresponding SMBIOS structure raw buffer according to specified type and handle.

The following table describes the behavior of this function for various combinations of *Type*, *Handle*, and *Key* values:

Type	Handle	*pKey	Behavior
!= 0xFFFF	!= 0xFFFF	Don’t care	Returns the structure with the Type and Handle specified.
!= 0xFFFF	== 0xFFFF	== 0xFFFF	Returns the first structure with the type specified.
!= 0xFFFF	== 0xFFFF	!= 0xFFFF	Returns the next structure with the type specified.
== 0xFFFF	!= 0xFFFF	Don’t care	Returns the structure with the Handle specified.
== 0xFFFF	== 0xFFFF	== 0xFFFF	Returns the first structure.
== 0xFFFF	== 0xFFFF	!= 0xFFFF	Returns the next structure.

As noted in the parameter descriptions, the *Key* is opaque data used by the library to optimize its searches. The only thing callers should do with this parameter is set it to *0xFFFF* to indicate the start of a search. Where the table specifies the Key value does not equal *0xFFFF* it means that it is using the value returned by a previous successful call.

At any time if this function call returns with a status code of *EFI_SMBIOSERR_STRUCT_NOT_FOUND*, it means there are no more structures of the specified type. The user shall not call the function again with the returned _*Key_.

User shall not allocate memory for the structure. User is responsible for freeing the memory allocated by ***pRawBuffer*** through the *SMBIO_FreeStructure()* routine.

Status Codes Returned

EFI_SUCCESS	Successfully get the specified SMBIOS raw structure buffer
EFI_SMBIOSERR_STRUCT_NOT_FOUND	The specified structure not found

3.6 SMBIOS_FreeStructure ()

```
EFI_STATUS

SMBIOS_FreeStructure (
      IN VOID *pStructBuffer
      );
```

Parameters

pStructBuffer	Point to SMBIOS structure data returned by *SMBIOS_GetTableEntryPoint(), SMBIOS_GetStructure(),* or *SMBIOS_GetRawStructure()* calls

Description

This function returns memory allocated through the *SMBIOS_GetTableEntryPoint(), SMBIOS_GetStructure(),* and *SMBIOS_GetRawStructure()* routines.

Status Codes Returned

EFI_SUCCESS	Successfully get the SMBIOS information from the Structure Entry Point.
EFI_SMBIOSERR_STRUCT_NOT_FOUND	SMBIOS TableEntryPoint structure is not found

Appendix 1: Data Structures

```
typedef struct DeviceStruct
{
      UINT8             DeviceType;
      UINT8             DescriptionString;
} DeviceStruct;


typedef struct MEMORYDEVICE
{
      UINT8             DeviceLoad;
      UINT16            DeviceHandle;
} MEMORYDEVICE;


typedef struct EVENTLOGTYPE
{
      UINT8             LogType;
      UINT8             DataFormatType;
} EVENTLOGTYPE;


enum enumStructureType
{

      eSMBIOSType0 = 0,
      eSMBIOSType1 = 1,
      eSMBIOSType2 = 2,
      eSMBIOSType3 = 3,
      eSMBIOSType4 = 4,
      eSMBIOSType5 = 5,
      eSMBIOSType6 = 6,
      eSMBIOSType7 = 7,
      eSMBIOSType8 = 8,
      eSMBIOSType9 = 9,
      eSMBIOSType10 = 10,
      eSMBIOSType11 = 11,
      eSMBIOSType12 = 12,
      eSMBIOSType13 = 13,
      eSMBIOSType14 = 14,
      eSMBIOSType15 = 15,
      eSMBIOSType16 = 16,
      eSMBIOSType17 = 17,
      eSMBIOSType18 = 18,
      eSMBIOSType19 = 19,
      eSMBIOSType20 = 20,
      eSMBIOSType21 = 21,
      eSMBIOSType22 = 22,
      eSMBIOSType23 = 23,
      eSMBIOSType24 = 24,
      eSMBIOSType25 = 25,
      eSMBIOSType26 = 26,
      eSMBIOSType27 = 27,
      eSMBIOSType28 = 28,
      eSMBIOSType29 = 29,
      eSMBIOSType30 = 30,
      eSMBIOSType32 = 32,
      eSMBIOSType33 = 33,
      eSMBIOSType34 = 34,
      eSMBIOSType35 = 35,
      eSMBIOSType36 = 36,
      eSMBIOSType37 = 37,
      eSMBIOSType38 = 38,
      eSMBIOSType39 = 39,
      eSMBIOSType126 = 126,
      eSMBIOSType127 = 127
};
```

Appendix 3: SMBIOS Structure Definitions

```
typedef struct SMBIOSType0
{
      SMBIOSHeader      Header;
      CHAR8             Vendor[64];
      CHAR8             BIOSVersion[64];
      UINT16            BIOSStartAddrSeg;
      CHAR8             BIOSReleaseDate[64];
      UINT8             BIOSROMSize;
      UINT64            BIOSCharacteristics;
      CHAR8             CharacteristicsExtSize;
      UINT8             BIOSCharacteristicsExt[1];
} SMBIOSType0;

typedef struct SMBIOSType1
{
      SMBIOSHeader      Header;
      CHAR8             Manufacturer[64];
      CHAR8             ProductName[64];
      CHAR8             Version[64];
      CHAR8             SerialNumber[64];
      UINT8             UUID[16];
      UINT8             WakeUpType;
} SMBIOSType1;

typedef struct SMBIOSType2
{
      SMBIOSHeader      Header;
      CHAR8             Manufacturer[64];
      CHAR8             ProductName[64];
      CHAR8             Version[64];
      CHAR8             SerialNumber[64];
} SMBIOSType2;

typedef struct SMBIOSType3
{
      SMBIOSHeader      Header;
      CHAR8             Manufacturer[64];
      UINT8             ChassisType;
      CHAR8             Version[64];
      CHAR8             SerialNumber[64];
      CHAR8             AssetTagNumber[64];
      UINT8             BootupState;
      UINT8             PowerSupplyState;
      UINT8             ThermalState;
      UINT8             SecurityStatus;
      UINT32            OEMDefined;
} SMBIOSType3;

typedef struct SMBIOSType4
{
      SMBIOSHeader      Header;
      UINT8             SocketDesignation;
      UINT8             ProcessorType;
      UINT8             ProcessorFamily;
      UINT8             ProcessorManufacturer;
      UINT64            ProcessorID;
      UINT8             ProcessorVersion;
      UINT8             Voltage;
      UINT16            ExternalClock;
      UINT16            MaxSpeed;
      UINT16            CurrentSpeed;
      UINT8             Status;
      UINT8             ProcessorUpgrade;
      UINT16            L1CacheHandle;
      UINT16            L2CacheHandle;
      UINT16            L3CacheHandle;
      CHAR8             SerialNumber[64];
      CHAR8             AssetTag[64];
      CHAR8             PartNumber[64];
} SMBIOSType4;

typedef struct SMBIOSType5
{
      SMBIOSHeader      Header;
      UINT8             ErrorDetectingMethod;
      UINT8             ErrorCorrectingCapability;
      UINT8             SupportedInterleave;
      UINT8             CurrentInterleave;
      UINT8             MaximumMemoryModuleSize;
      UINT16            SupportedSpeeds;
      UINT16            SupportedMemoryTypes;
      UINT8             MemoryModuleVoltage;
      UINT8             AssociatedMemorySlots;
      UINT16            MemoryModuleConfigHandle;
      UINT8             EnabledErrorCorrectingCapabilities;
} SMBIOSType5;

typedef struct SMBIOSType6
{
      SMBIOSHeader      Header;
      CHAR8             SocketDesignation[64];
      UINT8             BankConnections;
      UINT8             CurrentSpeed;
      UINT16            CurrentMemoryType;
      UINT8             InstalledSize;
      UINT8             EnabledSize;
      UINT8             ErrorStatus;
} SMBIOSType6;

typedef struct SMBIOSType7
{
      SMBIOSHeader      Header;
      CHAR8             SocketDesignation[64];
      UINT16            CacheConfiguration;
      UINT16            MaximumCacheSize;
      UINT16            InstalledSize;
      UINT16            SupportedSRAMType;
      UINT16            CurrentSRAMType;
      UINT8             CacheSpeed;
      UINT8             ErrorCorrectionType;
      UINT8             SystemCacheType;
      UINT8             Associativity;
```

```
} SMBIOSType7;

typedef struct SMBIOSType8
{
      SMBIOSHeader      Header;
      CHAR8             InternalReferenceDesignator[64];
      UINT8             InternalConnectorType;
      CHAR8             ExternalReferenceDesignator[64];
      UINT8             ExternalConnectorType;
      UINT8             PortType;
} SMBIOSType8;

typedef struct SMBIOSType9
{
      SMBIOSHeader      Header;
      CHAR8             SlotDesignation[64];
      UINT8             SlotType;
      UINT8             SlotDataBusWidth;
      UINT8             CurrentUsage;
      UINT8             SlotLength;
      UINT16            SlotID;
      UINT8             SlotCharacteristics;
      UINT8             SlotCharacteristics2;
} SMBIOSType9;

typedef struct SMBIOSType10
{
      SMBIOSHeader      Header;
      DeviceStruct      Device;
} SMBIOSType10;

typedef struct SMBIOSType11
{
      SMBIOSHeader      Header;
      UINT8             Count;
      CHAR8             bufOEMString[1][64];
} SMBIOSType11;

typedef struct SMBIOSType12
{
      SMBIOSHeader      Header;
      UINT8             Count;
      CHAR8             bufSysConfigurations[1][64];
} SMBIOSType12;

typedef struct SMBIOSType13
{
      SMBIOSHeader      Header;
      UINT8             InstallableLanguages;
      UINT8             Flags;
      UINT8             reserved[15];
      UINT8             CurrentLanguageIndex;
      CHAR8             IntalledLanguages[1][64];
} SMBIOSType13;
```

```
typedef struct SMBIOSType14
{
      SMBIOSHeader      Header;
      CHAR8             GroupName[64];
      UINT8             ItemType;
      UINT16            ItemHandle;
} SMBIOSType14;

typedef struct SMBIOSType15
{
      SMBIOSHeader      Header;
      UINT16            LogAreaLength;
      UINT16            LogHeaderStartOffset;
      UINT16            LogDataStartOffset;
      UINT8             AccessMethod;
      UINT8             LogStatus;
      UINT32            LogChangeToken;
      UINT32            AccessMethodAddress;
      UINT8             LogHeaderFormat;
      UINT8             NumberOfSupportedLogTypeDescriptors;
      UINT8             LengthOfLogTypeDescriptor;
      EVENTLOGTYPE      EventLogTypeDescriptors[1];
} SMBIOSType15;

typedef struct SMBIOSType16
{
      SMBIOSHeader      Header;
      UINT8             Location;
      UINT8             Use;
      UINT8             MemoryErrorCorrection;
      UINT32            MaximumCapacity;
      UINT16            MemoryErrorInformationHandle;
      UINT16            NumberOfMemoryDevices;
} SMBIOSType16;

typedef struct SMBIOSType17
{
      SMBIOSHeader      Header;
      UINT16            MemoryArrayHandle;
      UINT16            MemoryErrorInformationHandle;
      UINT16            TotalWidth;
      UINT16            DataWidth;
      UINT16            Size;
      UINT8             FormFactor;
      UINT8             DeviceSet;
      CHAR8             DeviceLocator[64];
      CHAR8             BankLocator[64];
      UINT8             MemoryType;
      UINT16            TypeDetail;
      UINT16            Speed;
      CHAR8             Manufacturer[64];
      CHAR8             SerialNumber[64];
      CHAR8             AssetTag[64];
      CHAR8             PartNumber[64];
} SMBIOSType17;
```

```
typedef struct SMBIOSType18
{
      SMBIOSHeader      Header;
      UINT8             ErrorType;
      UINT8             ErrorGranularity;
      UINT8             ErrorOperation;
      UINT32            VendorSyndrome;
      UINT32            MemoryArrayErrorAddress;
      UINT32            DeviceErrorAddress;
      UINT32            ErrorResolution;
} SMBIOSType18;

typedef struct SMBIOSType19
{
      SMBIOSHeader      Header;
      UINT32            StartingAddress;
      UINT32            EndingAddress;
      UINT16            MemoryArrayHandle;
      UINT8             PartitionWidth;
} SMBIOSType19;

typedef struct SMBIOSType20
{
      SMBIOSHeader      Header;
      UINT32            StartingAddress;
      UINT32            EndingAddress;
      UINT16            MemoryDeviceHandle;
      UINT16            MemoryArrayMappedAddressHandle;
      UINT8             PartitionRowPosition;
      UINT8             InterleavePosition;
      UINT8             InterleavedDataDepth;
} SMBIOSType20;

typedef struct SMBIOSType21
{
      SMBIOSHeader      Header;
      UINT8             Type;
      UINT8             Interface;
      UINT8             NumberOfButtons;
} SMBIOSType21;

typedef struct SMBIOSType22
{
      SMBIOSHeader      Header;
      CHAR8             Location[64];
      CHAR8             Manufacturer[64];
      CHAR8             ManufactureDate[64];
      CHAR8             SerialNumber[64];
      CHAR8             DeviceName[64];
      CHAR8             BankLocator[64];
      UINT8             DeviceChemistry;
      UINT16            DeviceCapacity;
      UINT16            DesignVoltage;
      CHAR8             SBDSVersionNumber[64];
      UINT8             MaximumErrorInBatteryData;
      UINT16            SBDSSerialNumber;
      UINT16            SBDSManufactureDate;
      CHAR8             SBDSDeviceChemistry[64];
      UINT8             DesignCapacityMultiplier;
      UINT32            OEMSpecific;
} SMBIOSType22;

typedef struct SMBIOSType23
{
      SMBIOSHeader      Header;
      UINT8             Capabilities;
      UINT16            ResetCount;
      UINT16            ResetLimit;
      UINT16            TimerInterval;
      UINT16            Timeout;
} SMBIOSType23;

typedef struct SMBIOSType24
{
      SMBIOSHeader      Header;
      UINT8             HardwareSecuritySettings;
} SMBIOSType24;

typedef struct SMBIOSType25
{
      SMBIOSHeader      Header;
      UINT8             NextScheduledPowerOnMonth;
      UINT8             NextScheduledPowerOnDayOfMonth;
      UINT8             NextScheduledPowerOnHour;
      UINT8             NextScheduledPowerOnMinute;
      UINT8             NextScheduledPowerOnSecond;
} SMBIOSType25;

typedef struct SMBIOSType26
{
      SMBIOSHeader      Header;
      CHAR8             Description[64];
      UINT8             LocationAndStatus;
      UINT16            MaximumValue;
      UINT16            MinimumValue;
      UINT16            Resolution;
      UINT16            Tolerance;
      UINT16            Accuracy;
      UINT32            OEMDefined;
      UINT16            NominalValue;
} SMBIOSType26;

typedef struct SMBIOSType27
{
      SMBIOSHeader      Header;
      UINT16            TemperatureProbeHandle;
      UINT8             DeviceTypeAndStatus;
      UINT8             CoolingUnitGroup;
      UINT32            OEMDefined;
      UINT16            NominalSpeed;
} SMBIOSType27;
```

```
typedef struct SMBIOSType28
{
      SMBIOSHeader      Header;
      CHAR8             Description[64];
      UINT8             LocationAndStatus;
      UINT16            MaximumValue;
      UINT16            MinimumValue;
      UINT16            Resolution;
      UINT16            Tolerance;
      UINT16            Accuracy;
      UINT32            OEMDefined;
      UINT32            NominalValue;
} SMBIOSType28;

typedef struct SMBIOSType29
{
      SMBIOSHeader      Header;
      CHAR8             Description[64];
      UINT8             LocationAndStatus;
      UINT16            MaximumValue;
      UINT16            MinimumValue;
      UINT16            Resolution;
      UINT16            Tolerance;
      UINT16            Accuracy;
      UINT32            OEMDefined;
      UINT16            NominalValue;
} SMBIOSType29;

typedef struct     SMBIOSType30
{
      SMBIOSHeader      Header;
      CHAR8             ManufacturerName[64];
      UINT8             Connections;
} SMBIOSType30;

typedef struct SMBIOSType32
{
      SMBIOSHeader      Header;
      UINT8             Reserved[6];
      UINT8             BootStatus;
} SMBIOSType32;

typedef struct SMBIOSType33
{
      SMBIOSHeader      Header;
      UINT8             ErrorType;
      UINT8             ErrorGranularity;
      UINT8             ErrorOperation;
      UINT32            VendorSyndrome;
      UINT64            MemoryArrayErrorAddress;
      UINT64            DeviceErrorAddress;
      UINT32            ErrorResolution;
} SMBIOSType33;

typedef struct SMBIOSType34
{
      SMBIOSHeader      Header;
      CHAR8             Description[64];
      UINT8             Type;
      UINT32            Address;
      UINT8             AddressType;
} SMBIOSType34;

typedef struct SMBIOSType35
{
      SMBIOSHeader      Header;
      CHAR8             Description[64];
      UINT16            ManagementDeviceHandle;
      UINT16            ComponentHandle;
      UINT16            ThresholdHandle;
} SMBIOSType35;

typedef struct SMBIOSType36
{
      SMBIOSHeader      Header;
      UINT16            LowerThresholdNonCritical;
      UINT16            UpperThresholdNonCritical;
      UINT16            LowerThresholdCritical;
      UINT16            UpperThreaholdCritical;
      UINT16            LowerThresholdNonRecoverable;
      UINT16            UpperThresholdNonRecoverable;
} SMBIOSType36;

typedef struct SMBIOSType37
{
      SMBIOSHeader      Header;
      UINT8             ChannelType;
      UINT8             MaximumChannelLoad;
      UINT8             MemoryDeviceCount;
      MEMORYDEVICE      MemoryDevice[1];
} SMBIOSType37;

typedef struct SMBIOSType38
{
      SMBIOSHeader      Header;
      UINT8             InterfaceType;
      UINT8             IPMISpecificationRevision;
      UINT8             I2CSlaveAddress;
      UINT8             NVStorageDeviceAddress;
      UINT64            BaseAddress;
      UINT8             BaseAddressModifier_InterruptInfo;
      UINT8             InterruptNumber;
} SMBIOSType38;
```

Note: The last two fields of the SMBIOSType38 structure correspond to the fields added by the Revision 3 addendum of the IPMI V1.0, revision 1.1 specification. To determine if the SMBIO supports this addendum, the call must check the *StructLength* field of the *Header*. If the value is > 0x10, the two new fields will contain valid data. Otherwise, the values of these fields are undetermined.

```
typedef struct SMBIOSType39
{
      SMBIOSHeader      Header;
      UINT8             PowerUnitGroup;
      CHAR8             Location[64];
      CHAR8             DeviceName[64];
      CHAR8             Manufacturer[64];
      CHAR8             SerialNumber[64];
      CHAR8             AssetTagNumber[64];
      CHAR8             ModelPartNumber[64];
      CHAR8             RevisionLevel[64];
      CHAR8             Description[64];
      UINT16            MaxPowerCapacity;
      UINT16            PowerSupplyCharacteristics;
      UINT16            InputVoltageProbeHandle;
      UINT16            CoolingDeviceHandle;
      UINT16            InputCurrentProbeHandle;
} SMBIOSType39;

typedef struct SMBIOSType126
{
      SMBIOSHeader      Header;
} SMBIOSType126;

typedef struct SMBIOSType127
{
      SMBIOSHeader      Header;
} SMBIOSType127;
```