Intel® Platform Innovation Framework for EFI Miscellaneous Subclass Specification

Draft for Review

Version 0.9

April 1, 2004

Intel order number xxxxxx-001

Revision History

Revision	Revision History	Date
0.9	First public release.	4/1/04

Contents

1 Introduction

Overview

This specification defines the core code that is required for an implementation of the miscellaneous data hub subclass of the Intel® Platform Innovation Framework for EFI (hereafter referred to as the "Framework"). This specification does the following:

- Describes the basic components of the miscellaneous data hub subclass and miscellaneous subclass data records
- Provides code definitions for type and record definitions for the miscellaneous subclass that are architecturally required by the *Intel® Platform Innovation Framework for EFI Architecture Specification*
- Provides examples of the data records for the miscellaneous subclass

This specification complies with the System Management BIOS (SMBIOS) Reference Specification, version 2.3.4.

Conventions Used in This Document

This document uses the typographic and illustrative conventions described below.

Data Structure Descriptions

Intel® processors based on 32-bit Intel® architecture (IA-32) are “little endian” machines. This distinction means that the low-order byte of a multibyte data item in memory is at the lowest address, while the high-order byte is at the highest address. Processors of the Intel® Itanium® processor family may be configured for both “little endian” and “big endian” operation. All implementations designed to conform to this specification will use “little endian” operation.

In some memory layout descriptions, certain fields are marked *reserved*. Software must initialize such fields to zero and ignore them when read. On an update operation, software must preserve any reserved field.

The data structures described in this document generally have the following format:

STRUCTURE NAME: The formal name of the data structure.

Summary: A brief description of the data structure.

Prototype: A “C-style” type declaration for the data structure.

Parameters: A brief description of each field in the data structure prototype.

Description: A description of the functionality provided by the data structure, including any limitations and caveats of which the caller should be aware.

Related Definitions: The type declarations and constants that are used only by this data structure.

Pseudo-Code Conventions

Pseudo code is presented to describe algorithms in a more concise form. None of the algorithms in this document are intended to be compiled directly. The code is presented at a level corresponding to the surrounding text.

In describing variables, a *list* is an unordered collection of homogeneous objects. A *queue* is an ordered list of homogeneous objects. Unless otherwise noted, the ordering is assumed to be First In First Out (FIFO).

Pseudo code is presented in a C-like format, using C conventions where appropriate. The coding style, particularly the indentation style, is used for readability and does not necessarily comply with an implementation of the *Extensible Firmware Interface Specification*.

Typographic Conventions

This document uses the typographic and illustrative conventions described below:

Plain text	The normal text typeface is used for the vast majority of the descriptive text in a specification.
Plain text (blue)	In the online help version of this specification, any plain text that is underlined and in blue indicates an active link to the cross-reference. Click on the word to follow the hyperlink. Note that these links are *not* active in the PDF of the specification.
Bold	In text, a **Bold** typeface identifies a processor register name. In other instances, a **Bold** typeface can be used as a running head within a paragraph.
Italic	In text, an *Italic* typeface can be used as emphasis to introduce a new term or to indicate a manual or specification name.
`BOLD Monospace`	Computer code, example code segments, and all prototype code segments use a **`BOLD Monospace`** typeface with a dark red color. These code listings normally appear in one or more separate paragraphs, though words or segments can also be embedded in a normal text paragraph.
`Bold Monospace`	In the online help version of this specification, words in a **`Bold Monospace`** typeface that is underlined and in blue indicate an active hyperlink to the code definition for that function or type definition. Click on the word to follow the hyperlink. Note that these links are *not* active in the PDF of the specification. Also, these inactive links in the PDF may instead have a **`Bold Monospace`** appearance that is underlined but in dark red. Again, these links are not active in the PDF of the specification.
`Italic Monospace`	In code or in text, words in *`Italic Monospace`* indicate placeholder names for variable information that must be supplied (i.e., arguments).
`Plain Monospace`	In code, words in a `Plain Monospace` typeface that is a dark red color but is not bold or italicized indicate pseudo code or example code. These code segments typically occur in one or more separate paragraphs.

text text text — In the PDF of this specification, text that is highlighted in yellow indicates that a change was made to that text since the previous revision of the PDF. The highlighting indicates only that a change was made since the previous version; it does not specify what changed. If text was deleted and thus cannot be highlighted, a note in red and highlighted in yellow (that looks like *(Note: text text text.)*) appears where the deletion occurred.

See the master Framework glossary in the Framework Interoperability and Component Specifications help system for definitions of terms and abbreviations that are used in this document or that might be useful in understanding the descriptions presented in this document.

See the master Framework references in the Interoperability and Component Specifications help system for a complete list of the additional documents and specifications that are required or suggested for interpreting the information presented in this document.

The Framework Interoperability and Component Specifications help system is available at the following URL:

http://www.intel.com/technology/framework/spec.htm

2 Design Discussion

Overview

This specification describes a group of data records that have similar characteristics. The data records are records that will be input to the data hub to be consumed by one or more drivers.

This specification complies with the *Intel® Platform Innovation Framework for EFI Data Hub Specification* and it is assumed that the consumer of this specification is well versed in the concept of the data hub. This specification describes in more detail how to implement a driver that will log all the miscellaneous data records and how to create drivers that will consume this data.

This specification also specifies the format of each data record. Each data record must be capable of being used by current and potential consumers of the data—in other words, the format should also be consumable by all potential agents. The unit of measurement, if applicable, should be the most common unit of measurement for the specific data record, and the range of values for a data record should be usable for the foreseeable future.

Additional References

In addition to the other Framework, industry, and Intel® specifications listed in References in the Interoperability and Component Specifications help system, the following sources are also referenced in this specification or may be useful to you:

- Boot Integrity Services: http://www.intel.com/labs/manage/wfm/tools/bis/index.htm
- Distributed Management Task Force, Inc.: http://www.dmtf.org/*
- *Smart Battery Data Specification:* http://www.sbs-forum.org/specs/*

Scope

This specification is the contract for miscellaneous data between the appropriate data drivers and the associated data consumers. Miscellaneous data includes information from the following:

- BIOS
- Physical system and baseboard
- Port
- Onboard device
- Security
- Power
- Cooling
- Management
- Power supply

This specification does not include cache, memory, or processor data, which can be found in their own specifications.

The data driver does not have to declare all the record types listed in this specification but should declare only those data types that are applicable. If the data is not applicable (for example, if no battery power exists in the system), the data consumer should make the necessary adjustment and not declare the associated data records.

A specific record defines the semantic context of the data. All records related to miscellaneous data are documented in this specification. The record type is numbered sequentially and a new record type can be appended to the end of the list. The record type in this specification defines the semantic context of the data. The data records in this specification comply to the **EFI_SUBCLASS_TYPE1_HEADER**.*Version* field of 1. Type **EFI_SUBCLASS_TYPE1_HEADER** is defined in the *Intel® Platform Innovation Framework for EFI Data Hub Subclass Design Guide*.

Data records can be added or deleted as long as backward compatibility is maintained. If a data record needs to be updated, a new entry to this specification can be added. At some point when the objectives of this subclass are not accomplished or become inconsistent, an entirely new subclass with a new Globally Unique Identifier (GUID) will be introduced.

NOTE

The data records and structures sometimes are System Management BIOS (SMBIOS)–centric as the initial document is based on the SMBIOS requirements.

Consumer versus Producer

It is possible that a producer of miscellaneous data is also a consumer and vice versa. An example is a producer of a management device component that consumes a thermal device or cooling device.

Compliancy Requirements

None of the record numbers described in this document are required by the *Extensible Firmware Interface Specification.* Some record number entries may be required by other industry-wide specifications such as the *System Management BIOS (SMBIOS) Reference Specification.*

Some **ENUM** definitions are controlled by the Distributed Management Task Force, Inc. (DMTF) organization. Refer to their Web site at www.dmtf.org/standards/dmi* and select the link to *Master MIF*.

Header Information

Miscellaneous Subclass Definition

The miscellaneous subclass belongs to the data class and is identified as the miscellaneous subclass by the GUID. See Miscellaneous Subclass Definition in Code Definitions for the definition.

Data Record Header

Each data record that is logged or read starts with a standard header of type **EFI_DATA_RECORD_HEADER**. The format of the header is defined in the *Intel® Platform Innovation Framework for EFI Data Hub Specification*.

Data Class Header

Each data record that is a member of the data class starts with a standard header of type **EFI_SUBCLASS_TYPE1_HEADER**. The format of the header is defined in the *Intel® Platform Innovation Framework for EFI Data Hub Subclass Design Guide*. This header follows the data record header **EFI_DATA_RECORD_HEADER**, which is defined in the *Intel® Platform Innovation Framework for EFI Data Hub Specification*.

The *Intel® Platform Innovation Framework for EFI Data Hub Subclass Design Guide* provides generic descriptions of the fields in **EFI_SUBCLASS_TYPE1_HEADER**. The following explanation further clarifies the descriptions for the miscellaneous subclass:

- The *Instance* is the instance number of the miscellaneous class in the system. A new producer of a subsystem should use the next incremental *Instance* number. The exact definition of what constitutes a new *Instance* number is a system architecture decision. An example is to partition the system into subsystems and assign an *Instance* to each subsystem. Examples of a subsystem could be a node in a hyper-cube system or a peripheral bay in a server.
- The *SubInstance* is a "device" in the miscellaneous subsystem, as multiple identical "devices" can exist in a miscellaneous subsystem. An example would be multiple onboard devices in a device bay. The *SubInstance* would be used to identify each SCSI controller. The *SubInstance* number should increment for each new identical "device" in the subsystem.

Raw Data

The raw data follows the **EFI_SUBCLASS_TYPE1_HEADER** header and its definition is specific to the *RecordNumber*. The syntax of the raw data is defined in Data Record Number in Code Definitions. Type **EFI_SUBCLASS_TYPE1_HEADER** is defined in the *Intel® Platform Innovation Framework for EFI Data Hub Subclass Design Guide*.

Data Record Number

The **EFI_SUBCLASS_TYPE1_HEADER** is followed by a data record. The data record format is specific to a *RecordNumber*. The sections in Data Record Number in Code Definitions define the data *RecordNumber*s for the miscellaneous subclass. The *RecordNumber*s are subdivided into different subsections on a per-SMBIOS type for ease of reference. See the Examples section for examples of these record numbers.

The **EFI_INTER_LINK_DATA** structure is used to link together multiple data hub items of the same subclass. An example is the use of the cooling device (see Cooling Device (Type 27)) to link a temperature probe (see Temperature Probe (Type 28)) with a fan. The *CoolingDeviceTemperatureLink* points to the temperature probe record.

All generic macros are defined in the *Intel® Platform Innovation Framework for EFI Data Hub Subclass Design Guide*. **STRING_REF** is defined in the *Intel® Platform Innovation Framework for EFI Human Interface Infrastructure Specification*.

3 Code Definitions

Introduction

This section contains the basic definitions of the data record header fields that are specific to the miscellaneous subclass, as well as definitions of miscellaneous data records. The following data types and data records are defined in this section:

- EFI_MISC_SUBCLASS
- EFI_MISC_SUBCLASS_VERSION
- EFI_MISC_LAST_PCI_BUS_DATA
- EFI_MISC_BIOS_VENDOR_DATA
- EFI_MISC_SYSTEM_MANUFACTURER_DATA
- EFI_MISC_BASE_BOARD_MANUFACTURER_DATA
- EFI_MISC_CHASSIS_MANUFACTURER_DATA
- EFI_MISC_PORT_INTERNAL_CONNECTOR_DESIGNATOR_DATA
- EFI_MISC_SYSTEM_SLOT_DESIGNATION_DATA
- EFI_MISC_ONBOARD_DEVICE_DATA
- EFI_MISC_OEM_STRING_DATA
- EFI_MISC_SYSTEM_OPTION_STRING_DATA
- EFI_MISC_NUMBER_OF_INSTALLABLE_LANGUAGES_DATA
- EFI_MISC_SYSTEM_LANGUAGE_STRING_DATA
- EFI_MISC_GROUP_NAME_DATA
- EFI_MISC_GROUP_ITEM_SET_DATA
- EFI_MISC_POINTING_DEVICE_TYPE_DATA
- EFI_MISC_BATTERY_LOCATION_DATA
- EFI_MISC_RESET_CAPABILITIES_DATA
- EFI_MISC_HARDWARE_SECURITY_SETTINGS_DATA
- EFI_MISC_SCHEDULED_POWER_ON_MONTH_DATA
- EFI_MISC_VOLTAGE_PROBE_DESCRIPTION_DATA
- EFI_MISC_COOLING_DEVICE_TEMP_LINK_DATA
- EFI_MISC_TEMPERATURE_PROBE_DESCRIPTION_DATA
- EFI_MISC_ELECTRICAL_CURRENT_PROBE_DESCRIPTION_DATA
- EFI_MISC_REMOTE_ACCESS_MANUFACTURER_DESCRIPTION_DATA
- EFI_MISC_BIS_ENTRY_POINT_DATA
- EFI_MISC_BOOT_INFORMATION_STATUS_DATA
- EFI_MISC_MANAGEMENT_DEVICE_DESCRIPTION_DATA
- EFI_MISC_MANAGEMENT_DEVICE_COMPONENT_DESCRIPTION_DATA

- **EFI_MISC_IPMI_INTERFACE_TYPE_DATA**
- **EFI_MISC_POWER_SUPPLY_UNIT_GROUP_DATA**
- **EFI_MISC_SMBIOS_STRUCT_ENCAPSULATION_DATA**

See the Examples section for examples using these data records.

Header Information

Miscellaneous Subclass Definition

EFI_MISC_SUBCLASS

Summary

The miscellaneous subclass belongs to the data class and is identified as the miscellaneous subclass by the GUID.

GUID

```
#define EFI_MISC_SUBCLASS_GUID \
{ 0x772484B2, 0x7482, 0x4b91, 0x9F, 0x9A, 0xAD, 0x43, 0xF8, 0x1C,
0x58, 0x81}
```

Class

```
#define        EFI_MISC_SUBCLASS        EFI_DATA_CLASS_DATA
```

Description

Summary

The miscellaneous subclass belongs to the data class and is identified as the miscellaneous subclass by the GUID.

For this subclass, the values defined above are used as follows:

- **EFI_DATA_RECORD_HEADER**.*DataRecordGuid* = **EFI_MISC_SUBCLASS_GUID**, which is the GUID that is specific to the miscellaneous subclass.
- **EFI_DATA_RECORD_HEADER**.*DataRecordClass* = **EFI_DATA_CLASS_DATA**. The "class" may be equal to the GUID "class" or a superset of the GUID "class."

Type **EFI_DATA_CLASS_DATA** is defined in **EFI_DATA_RECORD_HEADER**, which is defined in the *Intel® Platform Innovation Framework for EFI Data Hub Specification*.

Subclass Version

EFI_MISC_SUBCLASS_VERSION

Summary

Indicates the version of the miscellaneous subclass.

Prototype

```
#define EFI_MISC_SUBCLASS_VERSION   0x0100
```

Description

This value indicates the version of the miscellaneous subclass. It is used in **EFI_SUBCLASS_TYPE1_HEADER**.*Version*. Type **EFI_SUBCLASS_TYPE1_HEADER** is defined in the *Intel® Platform Innovation Framework for EFI Data Hub Subclass Design Guide*.

Data Record Number

Legacy Miscellaneous Record Numbers

Legacy Miscellaneous Record Numbers

This section describes information that is not part of the SMBIOS specification but of either other legacy specifications or information that is pertinent to an operating system.

Last PCI Bus Number

EFI_MISC_LAST_PCI_BUS_DATA

Summary

This data record refers to the last PCI bus that was found.

Prototype

```
typedef struct {
  UINT8                     LastPciBus;
} EFI_MISC_LAST_PCI_BUS_DATA;
```

Parameters

LastPciBus

A zero-based value that indicates the last PCI bus enumeration during power-on.

Description

This data record refers to the last PCI bus that was found.

For this data record, **EFI_SUBCLASS_TYPE1_HEADER**.*RecordType* = **EFI_MISC_LAST_PCI_BUS_RECORD_NUMBER**. Type **EFI_SUBCLASS_TYPE1_HEADER** is defined in the *Intel® Platform Innovation Framework for EFI Data Hub Subclass Design Guide*.

Related Definitions

```
//*******************************************************
// Record number
//*******************************************************
#define EFI_MISC_LAST_PCI_BUS_RECORD_NUMBER  0x00000001
```

BIOS Information (Type 0)

EFI_MISC_BIOS_VENDOR_DATA

Summary

This data record refers to the BIOS.

Prototype

```
typedef struct {
  STRING_REF                        BiosVendor;
  STRING_REF                        BiosVersion;
  STRING_REF                        BiosReleaseDate;
  EFI_PHYSICAL_ADDRESS              BiosStartingAddress;
  EFI_EXP_BASE2_DATA                BiosPhysicalDeviceSize;
  EFI_MISC_BIOS_CHARACTERISTICS     BiosCharacteristics1;
  EFI_MISC_BIOS_CHARACTERISTICS_EXTENSION
                                    BiosCharacteristics2;
} EFI_MISC_BIOS_VENDOR_DATA;
```

Parameters

BiosVendor

The BIOS vendor. Type **STRING_REF** is defined in the *Intel® Platform Innovation Framework for EFI Human Interface Infrastructure Specification*.

BiosVersion

The BIOS version number.

BiosReleaseDate

The BIOS release date.

BiosStartingAddress

The starting address of BIOS code space. Type **EFI_PHYSICAL_ADDRESS** is defined in **AllocatePages()** in the *EFI 1.10 Specification.*

BiosPhysicalDeviceSize

The maximum size in bytes of the physical device containing the BIOS. Type **EFI_EXP_BASE2_DATA** is defined in the *Intel® Platform Innovation Framework for EFI Data Hub Subclass Design Guide*.

BiosCharacteristics1

The BIOS characteristics supported by the system. Type **EFI_MISC_BIOS_CHARACTERISTICS** is defined in "Related Definitions" below.

BiosCharacteristics2

The BIOS characteristics supported by the system. Type **EFI_MISC_BIOS_CHARACTERISTICS_EXTENSION** is defined in "Related Definitions" below.

Description

This data record refers to the BIOS. This data record is a structure.

The type definition structure for **EFI_MISC_BIOS_VENDOR_DATA** is in SMBIOS 2.3.4, Type 0, in the following tables:

- Table 3.3.1.1
- Table 3.3.1.2.1
- Table 3.3.1.2.2

For this data record, **EFI_SUBCLASS_TYPE1_HEADER**.*RecordType* = **EFI_MISC_BIOS_VENDOR_RECORD_NUMBER**. Type **EFI_SUBCLASS_TYPE1_HEADER** is defined in the *Intel® Platform Innovation Framework for EFI Data Hub Subclass Design Guide*.

Related Definitions

```
//*******************************************************
// Record number
//*******************************************************
#define EFI_MISC_BIOS_VENDOR_RECORD_NUMBER         0x00000002


//*******************************************************
// EFI_MISC_BIOS_CHARACTERISTICS
//*******************************************************
typedef struct {
  UINT64 Reserved1                          :2;
  UINT64 Unknown                            :1;
  UINT64 BiosCharacteristicsNotSupported    :1;
  UINT64 IsaIsSupported                     :1;
  UINT64 McaIsSupported                     :1;
  UINT64 EisaIsSupported                    :1;
  UINT64 PciIsSupported                     :1;
  UINT64 PcmciaIsSupported                  :1;
  UINT64 PlugAndPlayIsSupported             :1;
  UINT64 ApmIsSupported                     :1;
  UINT64 BiosIsUpgradable                   :1;
  UINT64 BiosShadowingAllowed               :1;
  UINT64 VlVesaIsSupported                  :1;
  UINT64 EscdSupportIsAvailable             :1;
  UINT64 BootFromCdIsSupported              :1;
  UINT64 SelectableBootIsSupported          :1;
  UINT64 RomBiosIsSocketed                  :1;
  UINT64 BootFromPcmciaIsSupported          :1;
  UINT64 EDDSpecificationIsSupported        :1;
```

```
  UINT64 JapaneseNecFloppyIsSupported       :1;
  UINT64 JapaneseToshibaFloppyIsSupported   :1;
  UINT64 Floppy525_360IsSupported           :1;
  UINT64 Floppy525_12IsSupported            :1;
  UINT64 Floppy35_720IsSupported            :1;
  UINT64 Floppy35_288IsSupported            :1;
  UINT64 PrintScreenIsSupported             :1;
  UINT64 Keyboard8042IsSupported            :1;
  UINT64 SerialIsSupported                  :1;
  UINT64 PrinterIsSupported                 :1;
  UINT64 CgaMonoIsSupported                 :1;
  UINT64 NecPc98                            :1;
  UINT64 AcpiIsSupported                    :1;
  UINT64 UsbLegacyIsSupported               :1;
  UINT64 AgpIsSupported                     :1;
  UINT64 I20BootIsSupported                 :1;
  UINT64 Ls120BootIsSupported               :1;
  UINT64 AtapiZipDrivBootIsSupported        :1;
  UINT64 Boot1394IsSupported                :1;
  UINT64 SmartBatteryIsSupported            :1;
  UINT64 BiosBootSpecIsSupported            :1;
  UINT64 FunctionKeyNetworkBootIsSupported  :1;
  UINT64 Reserved                           :22;
} EFI_MISC_BIOS_CHARACTERISTICS;


//*******************************************************
// EFI_MISC_BIOS_CHARACTERISTICS_EXTENSION
//*******************************************************
typedef struct {
  UINT64 BiosReserved                       :16;
  UINT64 SystemReserved                     :16;
  UINT64 Reserved                           :32;
} EFI_MISC_BIOS_CHARACTERISTICS_EXTENSION;
```

BiosReserved

Reserved for future BIOS use. Must be initialized to zero.

SystemReserved

Reserved for future system use. Must be initialized to zero.

Reserved

Reserved for future use. Must be initialized to zero.

System Information (Type 1)

EFI_MISC_SYSTEM_MANUFACTURER_DATA

Summary

This data record refers to the system manufacturer.

Prototype

```
typedef struct {
  STRING_REF                    SystemManufacturer;
  STRING_REF                    SystemProductName;
  STRING_REF                    SystemVersion;
  STRING_REF                    SystemSerialNumber;
  EFI_GUID                      SystemUuid;
  EFI_MISC_SYSTEM_WAKEUP_TYPE   SystemWakeupType;
} EFI_MISC_SYSTEM_MANUFACTURER_DATA;
```

Parameters

SystemManufacturer

The system manufacturer. Type **STRING_REF** is defined in the *Intel® Platform Innovation Framework for EFI Human Interface Infrastructure Specification*.

SystemProductName

The system product name.

SystemVersion

The system version number.

SystemSerialNumber

The system serial number.

SystemUuid

The system Universal Unique ID (UUID) number. Type **EFI_GUID** is defined in **InstallProtocolInterface()** in the *EFI 1.10 Specification.*

SystemWakeupType

The event that caused the system to wake up. Type **EFI_MISC_SYSTEM_WAKEUP_TYPE** is defined in "Related Definitions" below.

Description

This data record refers to the system manufacturer. This data record is a structure.

The type definition structure for **EFI_MISC_SYSTEM_MANUFACTURER_DATA** is in SMBIOS 2.3.4, Type 1, Table 3.3.2.1.

For this data record, **EFI_SUBCLASS_TYPE1_HEADER**.*RecordType* = **EFI_MISC_SYSTEM_MANUFACTURER_RECORD_NUMBER**. Type **EFI_SUBCLASS_TYPE1_HEADER** is defined in the *Intel® Platform Innovation Framework for EFI Data Hub Subclass Design Guide*.

Related Definitions

```
//*******************************************************
// Record number
//*******************************************************
#define EFI_MISC_SYSTEM_MANUFACTURER_RECORD_NUMBER   0x00000003


//*******************************************************
// EFI_MISC_SYSTEM_WAKEUP_TYPE
//*******************************************************
typedef enum {
  EfiSystemWakeupTypeReserved          = 0,
  EfiSystemWakeupTypeOther             = 1,
  EfiSystemWakeupTypeUnknown           = 2,
  EfiSystemWakeupTypeApmTimer          = 3,
  EfiSystemWakeupTypeModemRing         = 4,
  EfiSystemWakeupTypeLanRemote         = 5,
  EfiSystemWakeupTypePowerSwitch       = 6,
  EfiSystemWakeupTypePciPme            = 7,
  EfiSystemWakeupTypeAcPowerRestored   = 8
} EFI_MISC_SYSTEM_WAKEUP_TYPE;
```

Baseboard Information (Type 2)

EFI_MISC_BASE_BOARD_MANUFACTURER_DATA

Summary

This data record refers to the baseboard manufacturer.

Prototype

```
typedef struct {
  STRING_REF                    BaseBoardManufacturer;
  STRING_REF                    BaseBoardProductName;
  STRING_REF                    BaseBoardVersion;
  STRING_REF                    BaseBoardSerialNumber;
  STRING_REF                    BaseBoardAssetTag;
  STRING_REF                    BaseBoardChassisLocation;
  EFI_BASE_BOARD_FEATURE_FLAGS  BaseBoardFeatureFlags;
  EFI_BASE_BOARD_TYPE           BaseBoardType;
  EFI_INTER_LINK_DATA           BaseBoardChassisLink;
  UINT32                        BaseBoardNumberLinks;
  EFI_INTER_LINK_DATA           LinkN;
} EFI_MISC_BASE_BOARD_MANUFACTURER_DATA;
```

Parameters

BaseBoardManufacturer

The baseboard manufacturer. Type **STRING_REF** is defined in the *Intel® Platform Innovation Framework for EFI Human Interface Infrastructure Specification*.

BaseBoardProductName

The baseboard product name.

BaseBoardVersion

The baseboard version number.

BaseBoardSerialNumber

The baseboard serial number.

BaseBoardAssetTag

The baseboard asset tag.

BaseBoardChassisLocation

The location of the baseboard within the chassis.

BaseBoardFeatureFlags

Flags that identify features contained on this baseboard. Type **EFI_BASE_BOARD_FEATURE_FLAGS** is defined in "Related Definitions" below.

BaseBoardType

Identifies the type of baseboard. Type **EFI_BASE_BOARD_TYPE** is defined in "Related Definitions" below.

BaseBoardChassisLink

A link to a chassis enclosure structure. Type **EFI_INTER_LINK_DATA** is defined in the *Intel® Platform Innovation Framework for EFI Data Hub Subclass Design Guide*.

BaseBoardNumberLinks

The number of links to other structures.

LinkN

A link to another structure. *N* is between 1 and *BaseBoardNumberLinks*.

Description

This data record refers to the baseboard manufacturer. This data record is a structure.

The type definition structure for **EFI_MISC_BASE_BOARD_MANUFACTURER_DATA** is in SMBIOS 2.3.4, Type 2.

For this data record, **EFI_SUBCLASS_TYPE1_HEADER**.*RecordType* = **EFI_MISC_BASE_BOARD_MANUFACTURER_RECORD_NUMBER**. Type **EFI_SUBCLASS_TYPE1_HEADER** is defined in the *Intel® Platform Innovation Framework for EFI Data Hub Subclass Design Guide*.

Related Definitions

```
//*******************************************************
// Record number
//*******************************************************
#define EFI_MISC_BASE_BOARD_MANUFACTURER_RECORD_NUMBER 0x00000004


//*******************************************************
// EFI_BASE_BOARD_FEATURE_FLAGS
//*******************************************************
typedef struct {
  UINT32 Motherboard             :1;
  UINT32 RequiresDaughterCard    :1;
  UINT32 Removable               :1;
  UINT32 Replaceable             :1;
  UINT32 HotSwappable            :1;
  UINT32 Reserved                :27;
} EFI_BASE_BOARD_FEATURE_FLAGS;
```

```
//*******************************************************
// EFI_BASE_BOARD_TYPE
//*******************************************************
typedef enum {
  EfiBaseBoardTypeUnknown                 = 1,
  EfiBaseBoardTypeOther                   = 2,
  EfiBaseBoardTypeServerBlade             = 3,
  EfiBaseBoardTypeConnectivitySwitch      = 4,
  EfiBaseBoardTypeSystemManagementModule = 5,
  EfiBaseBoardTypeProcessorModule         = 6,
  EfiBaseBoardTypeIOModule                = 7,
  EfiBaseBoardTypeMemoryModule            = 8,
  EfiBaseBoardTypeDaughterBoard           = 9,
  EfiBaseBoardTypeMotherBoard             = A,
  EfiBaseBoardTypeProcessorMemoryModule   = B,
  EfiBaseBoardTypeProcessorIOModule       = C,
  EfiBaseBoardTypeInterconnectBoard       = D
} EFI_BASE_BOARD_TYPE;
```

System/Chassis Enclosure (Type 3)

EFI_MISC_CHASSIS_MANUFACTURER_DATA

Summary

This data record refers to the chassis manufacturer.

Prototype

```
typedef struct {
  STRING_REF                    ChassisManufacturer;
  STRING_REF                    ChassisVersion;
  STRING_REF                    ChassisSerialNumber;
  STRING_REF                    ChassisAssetTag;
  EFI_MISC_CHASSIS_STATUS       ChassisType;
  EFI_MISC_CHASSIS_STATE        ChassisBootupState;
  EFI_MISC_CHASSIS_STATE        ChassisPowerSupplyState;
  EFI_MISC_CHASSIS_STATE        ChassisThermalState;
  EFI_MISC_CHASSIS_SECURITY_STATE ChassisSecurityState;
  UINT32                        ChassisOemDefined;
  UINT32                        ChassisHeight;
  UINT32                        ChassisNumberPowerCords;
  UINT32                        ChassisElementCount;
  UINT32                        ChassisElementRecordLength;
  EFI_MISC_ELEMENTS             ChassisElements;
} EFI_MISC_CHASSIS_MANUFACTURER_DATA;
```

Parameters

ChassisManufacturer

The chassis manufacturer. Type **STRING_REF** is defined in the *Intel® Platform Innovation Framework for EFI Human Interface Infrastructure Specification*.

ChassisVersion

The chassis version number.

ChassisSerialNumber

The chassis serial number.

ChassisAssetTag

The chassis asset tag.

ChassisType

The type of chassis. Type **EFI_MISC_CHASSIS_STATUS** is defined in "Related Definitions" below.

ChassisBootupState

The state of the enclosure when the system was last booted. Type **EFI_MISC_CHASSIS_STATE** is defined in "Related Definitions" below.

ChassisPowerSupplyState

The state of the enclosure's power supply when the system was last booted.

ChassisThermalState

The state of the enclosure's thermal state when the system was last booted.

ChassisSecurityState

The chassis security state when the system was last booted. Type **EFI_MISC_CHASSIS_SECURITY_STATE** is defined in "Related Definitions" below.

ChassisOemDefined

Original equipment manufacturer (OEM)– or BIOS-defined data.

ChassisHeight

Height in "U"s (1U=1.75"). A value of 0x00 indicates that the height is unspecified.

ChassisNumberPowerCords

The number of power cords associated with the chassis. A value of 0x00 indicates that the number of cords is unspecified.

ChassisElementCount

The number of element records that follow. A value of 0x00 indicates that no elements follow.

ChassisElementRecordLength

The length of each **EFI_MISC_ELEMENTS** structure. Type **EFI_MISC_ELEMENTS** is defined in "Related Definitions" below.

ChassisElements

Provides additional information about each element contained in the chassis. If present, then these *ChassisElementCount*s consist of *ChassisElements* of length **EFI_MISC_ELEMENTS**.

Description

This data record refers to the chassis manufacturer. This data record is a structure.

The type definition structure for **EFI_MISC_CHASSIS_MANUFACTURER_DATA** is in SMBIOS 2.3.4, Type 3, in the following tables:

- Table 3.3.4.1
- Table 3.3.4.2
- Table 3.3.4.3
- Table 3.3.4.4

For this data record, **EFI_SUBCLASS_TYPE1_HEADER**.*RecordType* = **EFI_MISC_CHASSIS_MANUFACTURER_RECORD_NUMBER**. Type **EFI_SUBCLASS_TYPE1_HEADER** is defined in the *Intel® Platform Innovation Framework for EFI Data Hub Subclass Design Guide*.

Related Definitions

```
//*******************************************************
// Record number
//*******************************************************
#define EFI_MISC_CHASSIS_MANUFACTURER_RECORD_NUMBER      0x00000005


//*******************************************************
// EFI_MISC_CHASSIS_STATUS
//*******************************************************
typedef struct {
  UINT32 EFI_MISC_CHASSIS_TYPE         :16;
  UINT32 ChassisLockPresent            :1;
  UINT32 Reserved                      :15;
} EFI_MISC_CHASSIS_STATUS;


//*******************************************************
// EFI_MISC_CHASSIS_TYPE
//*******************************************************
typedef enum {
  EfiMiscChassisTypeOther = 0x1,
  EfiMiscChassisTypeUnknown = 0x2,
  EfiMiscChassisTypeDeskTop = 0x3,
  EfiMiscChassisTypeLowProfileDesktop = 0x4,
  EfiMiscChassisTypePizzaBox = 0x5,
  EfiMiscChassisTypeMiniTower = 0x6,
  EfiMiscChassisTypeTower = 0x7,
  EfiMiscChassisTypePortable = 0x8,
  EfiMiscChassisTypeLapTop = 0x9,
  EfiMiscChassisTypeNotebook = 0xA,
  EfiMiscChassisTypeHandHeld = 0xB,
  EfiMiscChassisTypeDockingStation = 0xC,
  EfiMiscChassisTypeAllInOne = 0xD,
  EfiMiscChassisTypeSubNotebook = 0xE,
  EfiMiscChassisTypeSpaceSaving = 0xF,
  EfiMiscChassisTypeLunchBox = 0x10,
  EfiMiscChassisTypeMainServerChassis = 0x11,
  EfiMiscChassisTypeExpansionChassis = 0x12,
  EfiMiscChassisTypeSubChassis = 0x13,
  EfiMiscChassisTypeBusExpansionChassis = 0x14,
  EfiMiscChassisTypePeripheralChassis = 0x15,
  EfiMiscChassisTypeRaidChassis = 0x16,
  EfiMiscChassisTypeRackMountChassis = 0x17,
  EfiMiscChassisTypeSealedCasePc = 0x18,
  EfiMiscChassisMultiSystemChassis = 0x19
} EFI_MISC_CHASSIS_TYPE;
```

```
//*******************************************************
// EFI_MISC_CHASSIS_STATE
//*******************************************************
typedef enum {
  EfiChassisStateOther = 1,
  EfiChassisStateUnknown = 2,
  EfiChassisStateSafe = 3,
  EfiChassisStateWarning = 4,
  EfiChassisStateCritical = 5,
  EfiChassisStateNonRecoverable = 6
} EFI_MISC_CHASSIS_STATE;


//*******************************************************
// EFI_MISC_CHASSIS_SECURITY_STATE
//*******************************************************
typedef enum {
  EfiChassisSecurityStatusOther = 1,
  EfiChassisSecurityStatusUnknown = 2,
  EfiChassisSecurityStatusNone = 3,
  EfiChassisSecurityStatusExternalInterfaceLockedOut = 4,
  EfiChassisSecurityStatusExternalInterfaceLockedEnabled = 5
} EFI_MISC_CHASSIS_SECURITY_STATE;


//*******************************************************
// EFI_MISC_ELEMENTS
//*******************************************************
typedef struct {
  EFI_MISC_ELEMENT_TYPE    ChassisElementType;
  EFI_INTER_LINK_DATA      ChassisElementStructure;
  EFI_BASE_BOARD_TYPE      ChassisBaseBoard;
  UINT32                   ChassisElementMinimum;
  UINT32                   ChassisElementMaximum;
} EFI_MISC_ELEMENTS;
```

ChassisElementType

Specifies how to decode the other fields. If *ChassisElementType* **= 0**, then the *ChassisBaseBoard* field is valid and *ChassisElementStructure* is invalid. If *ChassisElementType* **= 1**, then vice versa. See below for the definition of **EFI_MISC_ELEMENT_TYPE**.

ChassisElementStructure

Specifies a link to a structure associated with this record. Only valid if *RecordType* **= 1**. Type **EFI_INTER_LINK_DATA** is defined in the *Intel® Platform Innovation Framework for EFI Data Hub Subclass Design Guide*.

ChassisBaseBoard

Specifies an **EFI_BASE_BOARD_TYPE**. Only valid if *RecordType* **= 0**. Type **EFI_BASE_BOARD_TYPE** is defined in "Related Definitions" in **EFI_MISC_BASE_BOARD_MANUFACTURER_DATA**.

ChassisElementMinimum

The minimum number of an element type that can be installed in the chassis for the chassis to properly operate. The valid range is 0x00 to 0xFE.

ChassisElementMaximum

The maximum number of an element type that can be installed in the chassis for the chassis to properly operate. The valid range is 0x01 to 0xFF.

```
//*******************************************************
// EFI_MISC_ELEMENT_TYPE
//*******************************************************
typedef struct {
  UINT32  RecordType  :1
  UINT32  Type        :7
  Uint32  Reserved    :24
} EFI_MISC_ELEMENT_TYPE;
```

RecordType

0 = **EFI_BASE_BOARD_TYPE**. Type **EFI_BASE_BOARD_TYPE** is defined in "Related Definitions" in **EFI_MISC_BASE_BOARD_MANUFACTURER_DATA**.

1 = Link to other structure. For example, *RecordType* could link to a system power supply (type 39) structure.

Type

The SMBIOS record type other than type 2 (baseboard information).

Reserved

Reserved for future use. Must be initialized to zero.

Port Connector Information (Type 8)

EFI_MISC_PORT_INTERNAL_CONNECTOR_DESIGNATOR_DATA

Summary

This data record refers to the connector designator that is internal to the chassis.

Prototype

```
typedef struct {
  STRING_REF                    PortInternalConnectorDesignator;
  STRING_REF                    PortExternalConnectorDesignator;
  EFI_MISC_PORT_CONNECTOR_TYPE  PortInternalConnectorType;
  EFI_MISC_PORT_CONNECTOR_TYPE  PortExternalConnectorType;
  EFI_MISC_PORT_TYPE            PortType;
  EFI_DEVICE_PATH_PROTOCOL      PortPath;
} EFI_MISC_PORT_INTERNAL_CONNECTOR_DESIGNATOR_DATA;
```

Parameters

PortInternalConnectorDesignator

The connector designator that is internal to the chassis. Type **STRING_REF** is defined in the *Intel® Platform Innovation Framework for EFI Human Interface Infrastructure Specification*.

PortExternalConnectorDesignator

The connector designator that is external to the chassis.

PortInternalConnectorType

The internal port connector type. Type **EFI_MISC_PORT_CONNECTOR_TYPE** is defined in "Related Definitions" below.

PortExternalConnectorType

The external port connector type.

PortType

The function of the port. Type **EFI_MISC_PORT_TYPE** is defined in "Related Definitions" below.

PortPath

The path to the physical device. Type **EFI_DEVICE_PATH** is defined in **LocateDevicePath()** in the *EFI 1.10 Specification.*

Description

This data record refers to the connector designator that is internal to the chassis. This data record is a structure.

The type definition structure for **EFI_MISC_PORT_INTERNAL_CONNECTOR_DESIGNATOR_DATA** is in SMBIOS 2.3.4, Type 8, in the following tables:

- Table 3.3.9.2
- Table 3.3.9.3

For this data record, **EFI_SUBCLASS_TYPE1_HEADER**.*RecordType* = **EFI_MISC_PORT_INTERNAL_CONNECTOR_DESIGNATOR_RECORD_NUMBER**. Type **EFI_SUBCLASS_TYPE1_HEADER** is defined in the *Intel® Platform Innovation Framework for EFI Data Hub Subclass Design Guide*.

Related Definitions

```
//*******************************************************
// Record number
//*******************************************************
#define EFI_MISC_PORT_INTERNAL_CONNECTOR_DESIGNATOR_RECORD_NUMBER
                                                    0x00000006

//*******************************************************
// EFI_MISC_PORT_CONNECTOR_TYPE
//*******************************************************
typedef enum {
  EfiPortConnectorTypeNone = 0x0,
  EfiPortConnectorTypeCentronics = 0x1,
  EfiPortConnectorTypeMiniCentronics = 0x2,
  EfiPortConnectorTypeProprietary = 0x3,
  EfiPortConnectorTypeDB25Male = 0x4,
  EfiPortConnectorTypeDB25Female = 0x5,
  EfiPortConnectorTypeDB15Male = 0x6,
  EfiPortConnectorTypeDB15Female = 0x7,
  EfiPortConnectorTypeDB9Male = 0x8,
  EfiPortConnectorTypeDB9Female = 0x9,
  EfiPortConnectorTypeRJ11 = 0xA,
  EfiPortConnectorTypeRJ45 = 0xB,
  EfiPortConnectorType50PinMiniScsi = 0xC,
  EfiPortConnectorTypeMiniDin = 0xD,
  EfiPortConnectorTypeMicriDin = 0xE,
  EfiPortConnectorTypePS2 = 0xF,
  EfiPortConnectorTypeInfrared = 0x10,
```

```
  EfiPortConnectorTypeHpHil = 0x11,
  EfiPortConnectorTypeUsb = 0x12,
  EfiPortConnectorTypeSsaScsi = 0x13,
  EfiPortConnectorTypeCircularDin8Male = 0x14,
  EfiPortConnectorTypeCircularDin8Female = 0x15,
  EfiPortConnectorTypeOnboardIde = 0x16,
  EfiPortConnectorTypeOnboardFloppy = 0x17,
  EfiPortConnectorType9PinDualInline = 0x18,
  EfiPortConnectorType25PinDualInline = 0x19,
  EfiPortConnectorType50PinDualInline = 0x1A,
  EfiPortConnectorType68PinDualInline = 0x1B,
  EfiPortConnectorTypeOnboardSoundInput = 0x1C,
  EfiPortConnectorTypeMiniCentronicsType14 = 0x1D,
  EfiPortConnectorTypeMiniCentronicsType26 = 0x1E,
  EfiPortConnectorTypeHeadPhoneMiniJack = 0x1F,
  EfiPortConnectorTypeBNC = 0x20,
  EfiPortConnectorType1394 = 0x21,
  EfiPortConnectorTypePC98 = 0xA0,
  EfiPortConnectorTypePC98Hireso = 0xA1,
  EfiPortConnectorTypePCH98 = 0xA2,
  EfiPortConnectorTypePC98Note = 0xA3,
  EfiPortConnectorTypePC98Full = 0xA4,
  EfiPortConnectorTypeOther = 0xFF
} EFI_MISC_PORT_CONNECTOR_TYPE;


//*******************************************************
// EFI_MISC_PORT_TYPE
//*******************************************************
typedef enum {
  EfiPortTypeNone = 0,
  EfiPortTypeParallelXtAtCompatible = 1,
  EfiPortTypeParallelPortPs2 = 2,
  EfiPortTypeParallelPortEcp = 3,
  EfiPortTypeParallelPortEpp = 4,
  EfiPortTypeParallelPortEcpEpp = 5,
  EfiPortTypeSerialXtAtCompatible = 6,
  EfiPortTypeSerial16450Compatible = 7,
  EfiPortTypeSerial16550Compatible = 8,
  EfiPortTypeSerial16550ACompatible = 9,
  EfiPortTypeScsi = A,
  EfiPortTypeMidi = B,
  EfiPortTypeJoyStick = C,
  EfiPortTypeKeyboard = D,
```

```
  EfiPortTypeMouse = E,
  EfiPortTypeSsaScsi = F,
  EfiPortTypeUsb = 10,
  EfiPortTypeFireWire = 11,
  EfiPortTypePcmciaTypeI = 12,
  EfiPortTypePcmciaTypeII = 13,
  EfiPortTypePcmciaTypeIII = 14,
  EfiPortTypeCardBus = 15,
  EfiPortTypeAccessBusPort = 16,
  EfiPortTypeScsiII = 17,
  EfiPortTypeScsiWide = 18,
  EfiPortTypePC98 = 19,
  EfiPortTypePC98Hireso = 1A,
  EfiPortTypePCH98 = 1B,
  EfiPortTypeVideoPort = 1C,
  EfiPortTypeAudioPort = 1D,
  EfiPortTypeModemPort = 1E,
  EfiPortTypeNetworkPort = 1F,
  EfiPortType8251Compatible = A0,
  EfiPortType8251FifoCompatible = A1,
  EfiPortTypeOther = FF
} EFI_MISC_PORT_TYPE;
```

System Slots (Type 9)

EFI_MISC_SYSTEM_SLOT_DESIGNATION_DATA

Summary

This data record refers to a system slot.

Prototype

```
typedef struct {
  STRING_REF                    SlotDesignation;
  EFI_MISC_SLOT_TYPE            SlotType;
  EFI_MISC_SLOT_DATA_BUS_WIDTH  SlotDataBusWidth;
  EFI_MISC_SLOT_USAGE           SlotUsage;
  EFI_MISC_SLOT_LENGTH          SlotLength;
  UINT16                        SlotId;
  EFI_MISC_SLOT_CHARACTERISTICS SlotCharacteristics;
  EFI_DEVICE_PATH_PROTOCOL      SlotDevicePath;
} EFI_MISC_SYSTEM_SLOT_DESIGNATION_DATA;
```

Parameters

SlotDesignation

The destination string of the slot. Type **STRING_REF** is defined in the *Intel® Platform Innovation Framework for EFI Human Interface Infrastructure Specification*.

SlotType

The type of slot. Type **EFI_MISC_SLOT_TYPE** is defined in "Related Definitions" below.

SlotDataBusWidth

The bit width of the slot. Type **EFI_MISC_SLOT_DATA_BUS_WIDTH** is defined in "Related Definitions" below.

SlotUsage

Indicates how the slot is currently used. Type **EFI_MISC_SLOT_USAGE** is defined in "Related Definitions" below.

SlotLength

The physical length of the slot. Type **EFI_MISC_SLOT_LENGTH** is defined in "Related Definitions" below.

SlotId

The logical slot identification. The ID varies depending upon the type of slot. Micro Channel Architecture (MCA) is 0x01 through 0x0F. EISA is 0x01 through 0x0F. PCI/AGP/PCI-X is 0x00xx through 0x00yy, where *xx* and *yy* are determined by the PCI IRQ routing table. PCMCIA is 0xyyxx, where *yy* is the socket number and *xx* is the adapter number.

SlotCharacteristics

The characteristics of the slot. Type **EFI_MISC_SLOT_CHARACTERISTICS** is defined in "Related Definitions" below.

SlotDevicePath

The path to the physical device. Type **EFI_DEVICE_PATH** is defined in **LocateDevicePath()** in the *EFI 1.10 Specification.*

Description

This data record refers to a system slot. This data record is a structure.

The type definition structure for **EFI_MISC_SYSTEM_SLOT_DESIGNATION_DATA** is in SMBIOS 2.3.4, Type 9, in the following tables:

- Table 3.3.10.1
- Table 3.3.10.2
- Table 3.3.10.3
- Table 3.3.10.4
- Table 3.3.10.5
- Table 3.3.10.6
- Table 3.3.10.7

For this data record, **EFI_SUBCLASS_TYPE1_HEADER**.*RecordType* = **EFI_MISC_SYSTEM_SLOT_DESIGNATION_RECORD_NUMBER**. Type **EFI_SUBCLASS_TYPE1_HEADER** is defined in the *Intel® Platform Innovation Framework for EFI Data Hub Subclass Design Guide*.

Related Definitions

```
//*******************************************************
// Record number
//*******************************************************
#define EFI_MISC_SYSTEM_SLOT_DESIGNATION_RECORD_NUMBER 0x00000007


//*******************************************************
// EFI_MISC_SLOT_TYPE
//*******************************************************
typedef enum {
  EfiSlotTypeOther = 0x1,
  EfiSlotTypeUnknown = 0x2,
  EfiSlotTypeIsa = 0x3,
  EfiSlotTypeMca = 0x4,
  EfiSlotTypeEisa = 0x5,
  EfiSlotTypePci = 0x6,
  EfiSlotTypePcmcia = 0x7,
```

```
  EfiSlotTypeVlVesa = 0x8,
  EfiSlotTypeProprietary = 0x9,
  EfiSlotTypeProcessorCardSlot = 0xA,
  EfiSlotTypeProprietaryMemoryCardSlot = 0xB,
  EfiSlotTypeIORiserCardSlot = 0xC,
  EfiSlotTypeNuBus = 0xD,
  EfiSlotTypePci66MhzCapable = 0xE,
  EfiSlotTypeAgp = 0xF,
  EfiSlotTypeApg2X = 0x10,
  EfiSlotTypeAgp4X = 0x11,
  EfiSlotTypePciX = 0x12,
  EfiSlotTypeAgp8x = 0x13,
  EfiSlotTypePC98C20 = 0xA0,
  EfiSlotTypePC98C24 = 0xA1,
  EfiSlotTypePC98E = 0xA2,
  EfiSlotTypePC98LocalBus = 0xA3,
  EfiSlotTypePC98Card = 0xA4
} EFI_MISC_SLOT_TYPE;


//*******************************************************
// EFI_MISC_SLOT_DATA_BUS_WIDTH
//*******************************************************
typedef enum {
  EfiSlotDataBusWidthOther = 1,
  EfiSlotDataBusWidthUnknown = 2,
  EfiSlotDataBusWidth8Bit = 3,
  EfiSlotDataBusWidth16Bit = 4,
  EfiSlotDataBusWidth32Bit = 5,
  EfiSlotDataBusWidth64Bit = 6,
  EfiSlotDataBusWidth128Bit = 7
} EFI_MISC_SLOT_DATA_BUS_WIDTH;


//*******************************************************
// EFI_MISC_SLOT_USAGE
//*******************************************************
typedef enum {
  EfiSlotUsageOther = 1,
  EfiSlotUsageUnknown = 2,
  EfiSlotUsageAvailable = 3,
  EfiSlotUsageInUse = 4
} EFI_MISC_SLOT_USAGE;
```

```
//***********************************************************
// EFI_MISC_SLOT_LENGTH
//***********************************************************
typedef enum {
  EfiSlotLengthOther = 1,
  EfiSlotLengthUnknown = 2,
  EfiSlotLengthShort = 3,
  EfiSlotLengthLong = 4
} EFI_MISC_SLOT_LENGTH;


//***********************************************************
// EFI_MISC_SLOT_CHARACTERISTICS
//***********************************************************
typedef struct {
  UINT32 CharacteristicsUnknown   :1;
  UINT32 Provides50Volts          :1;
  UINT32 Provides33Volts          :1;
  UINT32 SharedSlot               :1;
  UINT32 PcCard16Supported        :1;
  UINT32 CardBusSupported         :1;
  UINT32 ZoomVideoSupported       :1;
  UINT32 ModemRingResumeSupported :1;
  UINT32 PmeSignalSupported       :1;
  UINT32 HotPlugDevicesSupported  :1;
  UINT32 SmbusSignalSupported     :1;
  UINT32 Reserved                 :21;
} EFI_MISC_SLOT_CHARACTERISTICS;
```

Onboard Devices Information (Type 10)

EFI_MISC_ONBOARD_DEVICE_DATA

Summary

This data record refers to an onboard device that is present in the system.

Prototype

```
typedef struct {
  STRING_REF                        OnBoardDeviceDescription;
  EFI_MISC_ONBOARD_DEVICE_STATUS    OnBoardDeviceType;
  EFI_DEVICE_PATH_PROTOCOL          OnBoardDevicePath;
} EFI_MISC_ONBOARD_DEVICE_DATA;
```

Parameters

OnBoardDeviceDescription

The description of the onboard device. Type **STRING_REF** is defined in the *Intel® Platform Innovation Framework for EFI Human Interface Infrastructure Specification*.

OnBoardDeviceType

The type of device and its status. Type **EFI_MISC_ONBOARD_DEVICE_STATUS** is defined in "Related Definitions" below.

OnBoardDevicePath

The path to the physical device. Type **EFI_DEVICE_PATH** is defined in **LocateDevicePath()** in the *EFI 1.10 Specification.*

Description

This data record refers to an onboard device that is present in the system. This data record is a **UINT16**.

The type definition structure for **EFI_MISC_ONBOARD_DEVICE_DATA** is in SMBIOS 2.3.4, Type 10, Table 3.3.11.1.

For this data record, **EFI_SUBCLASS_TYPE1_HEADER**.*RecordType* = **EFI_MISC_ONBOARD_DEVICE_RECORD_NUMBER**. Type **EFI_SUBCLASS_TYPE1_HEADER** is defined in the *Intel® Platform Innovation Framework for EFI Data Hub Subclass Design Guide*.

Related Definitions

```
//*******************************************************
// Record number
//*******************************************************
#define EFI_MISC_ONBOARD_DEVICE_RECORD_NUMBER      0x00000008
```

```
//*******************************************************
// EFI_MISC_ONBOARD_DEVICE_STATUS
//*******************************************************
typedef struct {
  UINT32 DeviceType        :16;
  UINT32 DeviceEnabled     :1;
  UINT32 Reserved          :15;
} EFI_MISC_ONBOARD_DEVICE_STATUS;
```

Values for *DeviceType* are defined in **EFI_MISC_ONBOARD_DEVICE_TYPE** below.

```
//*******************************************************
// EFI_MISC_ONBOARD_DEVICE_TYPE
//*******************************************************
typedef enum {
  EfiOnBoardDeviceTypeOther = 1,
  EfiOnBoardDeviceTypeUnknown = 2,
  EfiOnBoardDeviceTypeVideo = 3,
  EfiOnBoardDeviceTypeScsiController = 4,
  EfiOnBoardDeviceTypeEthernet = 5,
  EfiOnBoardDeviceTypeTokenRing = 6,
  EfiOnBoardDeviceTypeSound = 7
} EFI_MISC_ONBOARD_DEVICE_TYPE;
```

OEM Strings (Type 11)

EFI_MISC_OEM_STRING_DATA

Summary

This data record refers to an OEM string.

Prototype

```
typedef struct {
  STRING_REF          OemStringRef[NumberOfInstallableLanguages];
} EFI_MISC_OEM_STRING_DATA;
```

Parameters

OemStringRef

The reference to a string. Type **STRING_REF** is defined in the *Intel® Platform Innovation Framework for EFI Human Interface Infrastructure Specification*.

NumberOfInstallableLanguages

As defined in **EFI_MISC_OEM_STRING_DATA**. If the record does not exist, then one language is assumed.

Description

This data record refers to an OEM string. This data record is a string.

The type definition structure for **EFI_MISC_OEM_STRING_DATA** is in SMBIOS 2.3.4, Type 11.

For this data record, **EFI_SUBCLASS_TYPE1_HEADER**.*RecordType* = **EFI_MISC_OEM_STRING_RECORD_NUMBER**. Type **EFI_SUBCLASS_TYPE1_HEADER** is defined in the *Intel® Platform Innovation Framework for EFI Data Hub Subclass Design Guide*.

Related Definitions

```
//*******************************************************
// Record number
//*******************************************************
#define EFI_MISC_OEM_STRING_RECORD_NUMBER          0x00000009
```

System Configuration Options (Type 12)

EFI_MISC_SYSTEM_OPTION_STRING_DATA

Summary

This data record refers to a system configuration option string.

Prototype

```
typedef struct {
  STRING_REF   SystemOptionStringRef[NumberOfInstallableLanguages];
} EFI_MISC_SYSTEM_OPTION_STRING_DATA;
```

Parameters

SystemOptionStringRef

The reference to a string. Type **STRING_REF** is defined in the *Intel® Platform Innovation Framework for EFI Human Interface Infrastructure Specification*.

NumberOfInstallableLanguages

As defined in **EFI_MISC_OEM_STRING_DATA**. If the record does not exist, then one language is assumed.

Description

This data record refers to a system configuration option string. This data record is a string.

The type definition structure for **EFI_MISC_SYSTEM_OPTION_STRING_DATA** is in SMBIOS 2.3.4, Type 12.

For this data record, **EFI_SUBCLASS_TYPE1_HEADER**.*RecordType* = **EFI_MISC_SYSTEM_OPTION_STRING_RECORD_NUMBER**. Type **EFI_SUBCLASS_TYPE1_HEADER** is defined in the *Intel® Platform Innovation Framework for EFI Data Hub Subclass Design Guide*.

Related Definitions

```
//*******************************************************
// Record number
//*******************************************************
#define EFI_MISC_SYSTEM_OPTION_STRING_RECORD_NUMBER  0x0000000A
```

BIOS Language Information (Type 13)

Number of Installable Languages

EFI_MISC_NUMBER_OF_INSTALLABLE_LANGUAGES_DATA

Summary

This data record refers to an OEM string.

Prototype

```
typedef struct {
  UINT16                    NumberOfInstallableLanguages;
  EFI_MISC_LANGUAGE_FLAGS   LanguageFlags;
  UINT16                    CurrentLanguageNumber;
} EFI_MISC_NUMBER_OF_INSTALLABLE_LANGUAGES_DATA;
```

Parameters

NumberOfInstallableLanguages

The number of languages supported. This parameter is one based.

LanguageFlags

The language flags. Type **EFI_MISC_LANGUAGE_FLAGS** is defined in "Related Definitions" below.

CurrentLanguageNumber

The current language number.

Description

This data record refers to an OEM string. This data record is a **UINT16**.

The type definition structure for **EFI_MISC_NUMBER_OF_INSTALLABLE_LANGUAGES_DATA** is in SMBIOS 2.3.4, Type 13.

For this data record, **EFI_SUBCLASS_TYPE1_HEADER**.*RecordType* = **EFI_MISC_NUMBER_OF_INSTALLABLE_LANGUAGES_RECORD_NUMBER**. Type **EFI_SUBCLASS_TYPE1_HEADER** is defined in the *Intel® Platform Innovation Framework for EFI Data Hub Subclass Design Guide*.

Related Definitions

```
//*******************************************************
// Record number
//*******************************************************
#define EFI_MISC_NUMBER_OF_INSTALLABLE_LANGUAGES_RECORD_NUMBER
                                                    0x0000000B

//*******************************************************
// EFI_MISC_LANGUAGE_FLAGS
//*******************************************************
typedef struct {
  UINT32 AbbreviatedLanguageFormat     :1;
  UINT32 Reserved                      :31;
} EFI_MISC_LANGUAGE_FLAGS;
```

AbbreviatedLanguageFormat

If *AbbreviatedLanguageFormat* is 0b, then the string is the two-character ISO 639 language name directly followed by the two-character ISO 3166 territory name.

If *AbbreviatedLanguageFormat* is 1b, then the string is in the form of "ISO 639 Language Name | ISO 3166 Territory Name | Encoding Method."

Reserved

Reserved for future use. Must be initialized to zero.

System Language String

EFI_MISC_SYSTEM_LANGUAGE_STRING_DATA

Summary

This data record refers to a language string.

Prototype

```
typedef struct {
  UINT16              LanguageId;
  STRING_REF          SystemLanguageString;
} EFI_MISC_SYSTEM_LANGUAGE_STRING_DATA;
```

Parameters

LanguageId

A number between 1 and *NumberOfInstallableLanguages*. See **EFI_MISC_OEM_STRING_DATA**.

If *AbbreviatedLanguageFormat* is 0b, then the string is the two-character ISO 639 language name directly followed by the two-character ISO 3166 territory name.

If *AbbreviatedLanguageFormat* is 1b, then the string is in the form of "ISO 639 Language Name | ISO 3166 Territory Name | Encoding Method."

See the definition of **EFI_MISC_LANGUAGE_FLAGS** in "Related Definitions" in **EFI_MISC_NUMBER_OF_INSTALLABLE_LANGUAGES_DATA** to detect if this field is in *AbbreviatedLanguageFormat*.

SystemLanguageString

Determines the format of string. Type **STRING_REF** is defined in the *Intel® Platform Innovation Framework for EFI Human Interface Infrastructure Specification*.

Description

This data record refers to a language string. One data record per language is supported. This data record is a string.

The type definition structure for **EFI_MISC_SYSTEM_LANGUAGE_STRING_DATA** is in SMBIOS 2.3.4, Type 13.

For this data record, **EFI_SUBCLASS_TYPE1_HEADER**.*RecordType* = **EFI_MISC_SYSTEM_LANGUAGE_STRING_RECORD_NUMBER**. Type **EFI_SUBCLASS_TYPE1_HEADER** is defined in the *Intel® Platform Innovation Framework for EFI Data Hub Subclass Design Guide*.

Related Definitions

```
//*************************************************
// Record number
//*************************************************
#define EFI_MISC_SYSTEM_LANGUAGE_STRING_RECORD_NUMBER  0x0000000C
```

Group Associations (Type 14)

Group Name

EFI_MISC_GROUP_NAME_DATA

Summary

This data record refers to a structure name that groups a set of subclass structures.

Prototype

```
typedef struct {
  STRING_REF                GroupName;
  UINT16                    NumberGroupItems;
  UINT16                    GroupId;
} EFI_MISC_GROUP_NAME_DATA;
```

Parameters

GroupName

A string name that groups a set of subclass structures. Type **STRING_REF** is defined in the *Intel® Platform Innovation Framework for EFI Human Interface Infrastructure Specification*.

NumberGroupItems

The number of group item sets. See **EFI_MISC_GROUP_ITEM_SET_DATA**.

GroupId

The ID of the group.

Description

This data record refers to a structure name that groups a set of subclass structures. The purpose of this record and the **EFI_MISC_GROUP_ITEM_SET_DATA** record is to allow an association of apparently diverse devices. An example could be an integrated circuit chip that contains both a voltage and a current probe. The integrated circuit could be given a group name with the voltage and temperature probes as separate **EFI_MISC_GROUP_ITEM_SET_DATA** records. This data record is a string.

The type definition structure for **EFI_MISC_GROUP_NAME_DATA** is in SMBIOS 2.3.4, Type 14, Offset 0x4.

For this data record, **EFI_SUBCLASS_TYPE1_HEADER**.*RecordType* = **EFI_MISC_GROUP_NAME_RECORD_NUMBER**. Type **EFI_SUBCLASS_TYPE1_HEADER** is defined in the *Intel® Platform Innovation Framework for EFI Data Hub Subclass Design Guide*.

Related Definitions

```
//*******************************************************
// Record number
//*******************************************************
#define EFI_MISC_GROUP_NAME_RECORD_NUMBER          0x0000000D
```

Group Item Set Element

EFI_MISC_GROUP_ITEM_SET_DATA

Summary

This data record refers to a group item set element.

Prototype

```
typedef struct {
  EFI_GUID                  SubClass;
  EFI_INTER_LINK_DATA       GroupLink;
  UINT16                    GroupId;
  UINT16                    GroupElementId;
} EFI_MISC_GROUP_ITEM_SET_DATA;
```

Parameters

SubClass

The data hub subclass that is associated with the *GroupLink*. Type **EFI_GUID** is defined in **InstallProtocolInterface()** in the *EFI 1.10 Specification.*

GroupLink

The *Producer*, *Instance*, and *SubInstance* of a data hub structure. Type **EFI_INTER_LINK_DATA** is defined in the *Intel® Platform Innovation Framework for EFI Data Hub Subclass Design Guide*.

GroupId

The ID of the group.

GroupElementId

A number between 1 and *NumberGroupItems* for the specified *GroupId*. See **EFI_MISC_GROUP_NAME_DATA**.

Description

This data record refers to a group item set element. Each set is identified by a group ID and each element is identified by an element ID. This data record is a structure.

The type definition structure for **EFI_MISC_GROUP_ITEM_SET_DATA** is in SMBIOS 2.3.4, Type 14, Offset 0x5 and 0x6.

For this data record, **EFI_SUBCLASS_TYPE1_HEADER**.*RecordType* = **EFI_MISC_GROUP_ITEM_SET_RECORD_NUMBER**. Type **EFI_SUBCLASS_TYPE1_HEADER** is defined in the *Intel® Platform Innovation Framework for EFI Data Hub Subclass Design Guide*.

Related Definitions

```
//*******************************************************
// Record number
//*******************************************************
#define EFI_MISC_GROUP_ITEM_SET_RECORD_NUMBER      0x0000000E
```

Built-in Pointing Device (Type 21)

EFI_MISC_POINTING_DEVICE_TYPE_DATA

Summary

This data record refers to a pointing device.

Prototype

```
typedef struct {
  EFI_MISC_POINTING_DEVICE_TYPE      PointingDeviceType;
  EFI_MISC_POINTING_DEVICE_INTERFACE PointingDeviceInterface;
  UINT16                             NumberPointingDeviceButtons;
  EFI_DEVICE_PATH_PROTOCOL           PointingDevicePath;
} EFI_MISC_POINTING_DEVICE_TYPE_DATA;
```

Parameters

PointingDeviceType

The type of device and its status. Type **EFI_MISC_POINTING_DEVICE_TYPE** is defined in "Related Definitions" below.

PointingDeviceInterface

The type of device interface. Type **EFI_MISC_POINTING_DEVICE_INTERFACE** is defined in "Related Definitions" below.

NumberPointingDeviceButtons

The number of buttons on the pointing device.

PointingDevicePath

The path to the physical device. Type **EFI_DEVICE_PATH** is defined in **LocateDevicePath()** in the *EFI 1.10 Specification.*

Description

This data record refers to a pointing device. This data record is an enumeration.

The type definition structure for **EFI_MISC_POINTING_DEVICE_TYPE_DATA** is in SMBIOS 2.3.4, Type 21, in the following tables:

- Table 3.3.22.1
- Table 3.3.22.2

For this data record, **EFI_SUBCLASS_TYPE1_HEADER**.*RecordType* = **EFI_MISC_POINTING_DEVICE_TYPE_RECORD_NUMBER**. Type **EFI_SUBCLASS_TYPE1_HEADER** is defined in the *Intel® Platform Innovation Framework for EFI Data Hub Subclass Design Guide*.

Related Definitions

```
//*************************************************
// Record number
//*************************************************
#define EFI_MISC_POINTING_DEVICE_TYPE_RECORD_NUMBER    0x0000000F


//*************************************************
// EFI_MISC_POINTING_DEVICE_TYPE
//*************************************************
typedef enum {
  EfiPointingDeviceTypeOther = 1,
  EfiPointingDeviceTypeUnknown = 2,
  EfiPointingDeviceTypeMouse = 3,
  EfiPointingDeviceTypeTrackBall = 4,
  EfiPointingDeviceTypeTrackPoint = 5,
  EfiPointingDeviceTypeGlidePoint = 6,
  EfiPointingDeviceTouchPad = 7,
  EfiPointingDeviceTouchScreen = 8,
  EfiPointingDeviceOpticalSensor = 9
} EFI_MISC_POINTING_DEVICE_TYPE;


//*************************************************
// EFI_MISC_POINTING_DEVICE_INTERFACE
//*************************************************
typedef enum {
  EfiPointingDeviceInterfaceOther = 1,
  EfiPointingDeviceInterfaceUnknown = 2,
  EfiPointingDeviceInterfaceSerial = 3,
  EfiPointingDeviceInterfacePs2 = 4,
  EfiPointingDeviceInterfaceInfrared = 5,
  EfiPointingDeviceInterfaceHpHil = 6,
  EfiPointingDeviceInterfaceBusMouse = 7,
  EfiPointingDeviceInterfaceADB = 8,
  EfiPointingDeviceInterfaceBusMouseDB9 = 0xA0,
  EfiPointingDeviceInterfaceBusMouseMicroDin = 0xA1,
  EfiPointingDeviceInterfaceUsb = 0xA2
} EFI_MISC_POINTING_DEVICE_INTERFACE;
```

Portable Battery (Type 22)

EFI_MISC_BATTERY_LOCATION_DATA

Summary

This data record refers to a portable battery.

Prototype

```
typedef struct {
  STRING_REF                          BatteryLocation;
  STRING_REF                          BatteryManufacturer;
  STRING_REF                          BatteryManufactureDate;
  STRING_REF                          BatterySerialNumber;
  STRING_REF                          BatteryDeviceName;
  STRING_REF                          BatterySbdsVersionNumber;
  STRING_REF                          BatterySbdsDeviceChemistry;
  EFI_MISC_BATTERY_DEVICE_CHEMISTRY   BatteryDeviceChemistry;
  EFI_EXP_BASE10_DATA                 BatteryDesignCapacity;
  EFI_EXP_BASE10_DATA                 BatteryDesignVoltage;
  UINT16                              BatteryMaximumError;
  UINT16                              BatterySbdsSerialNumber;
  EFI_MISC_BATTERY_SBDS_MANUFACTURE_DATE
                                      BatterySbdsManufacturingDate;
  UINT32                              BatteryOemSpecific;
} EFI_MISC_BATTERY_LOCATION_DATA;
```

Parameters

BatteryLocation

The portable battery location. Type **STRING_REF** is defined in the *Intel® Platform Innovation Framework for EFI Human Interface Infrastructure Specification*.

BatteryManufacturer

The portable battery manufacturer.

BatteryManufactureDate

The manufacture date of the portable battery.

BatterySerialNumber

The battery serial number.

BatteryDeviceName

The battery device name.

BatterySbdsVersionNumber

The Smart Battery Data Specification (SBDS) version supported by this battery.

BatterySbdsDeviceChemistry

The battery chemistry identifier. For this field to be valid, *BatteryDeviceChemistry* must be set to 0x02 or unknown.

BatteryDeviceChemistry

The battery device chemistry. Type **EFI_MISC_BATTERY_DEVICE_CHEMISTRY** is defined in "Related Definitions" below.

BatteryDesignCapacity

The battery design capacity in watt-hours. If the value is unknown, the field contains 0x00. Type **EFI_EXP_BASE10_DATA** is defined in the *Intel® Platform Innovation Framework for EFI Data Hub Subclass Design Guide*.

BatteryDesignVoltage

The design voltage in volts. If the value is unknown, the field is 0x00.

BatteryMaximumError

The maximum error in watt-hour data that is reported by the battery. Error is a percentage from 0 to 100. If the value is unknown, the field contains 0xFFFF.

BatterySbdsSerialNumber

The SBDS version supported by this battery.

BatterySbdsManufacturingDate

The battery's SBDS manufacture date. For this field to be valid, the *BatteryManufacturerDate* **STRING_REF** must be **NULL**. Type **EFI_MISC_BATTERY_SBDS_MANUFACTURE_DATE** is defined in "Related Definitions" below.

BatteryOemSpecific

OEM- or BIOS-vendor-specific information.

Description

This data record refers to a portable battery. This data record is a structure.

The type definition structure for **EFI_MISC_BATTERY_LOCATION_DATA** is in SMBIOS 2.3.4, Type 22, Table 3.3.23.1.

For this data record, **EFI_SUBCLASS_TYPE1_HEADER**.*RecordType* = **EFI_MISC_BATTERY_LOCATION_RECORD_NUMBER**. Type **EFI_SUBCLASS_TYPE1_HEADER** is defined in the *Intel® Platform Innovation Framework for EFI Data Hub Subclass Design Guide*.

Related Definitions

```
//*******************************************************
// Record number
//*******************************************************
#define EFI_MISC_BATTERY_LOCATION_RECORD_NUMBER  0x00000010


//*******************************************************
// EFI_MISC_BATTERY_DEVICE_CHEMISTRY
//*******************************************************
typedef enum {
  EfiBatteryDeviceChemistryTypeOther = 1,
  EfiBatteryDeviceChemistryTypeUnknown = 2,
  EfiBatteryDeviceChemistryTypeLeadAcid = 3,
  EfiBatteryDeviceChemistryTypeNickelCadmium = 4,
  EfiBatteryDeviceChemistryTypeNickelMetalHydride = 5,
  EfiBatteryDeviceChemistryTypeLithiumIon = 6,
  EfiBatteryDeviceChemistryTypeZincAir = 7,
  EfiBatteryDeviceChemistryTypeLithiumPolymer = 8
} EFI_MISC_BATTERY_DEVICE_CHEMISTRY;


//*******************************************************
// EFI_MISC_BATTERY_SBDS_MANUFACTURE_DATE
//*******************************************************
typedef struct  {
  UINT32 Date              :5;
  UINT32 Month             :4;
  UINT32 Year              :7;
  UINT32 Reserved          :16;
} EFI_MISC_BATTERY_SBDS_MANUFACTURE_DATE;
```

Date

A value in the range 1 to 31.

Month

A value in the range 1 to 12.

Year

A value in the range 0 to 127 biased by 1980.

Reserved

Reserved for future use. Must be initialized to zero.

System Reset (Type 23)

EFI_MISC_RESET_CAPABILITIES_DATA

Summary

This data record refers to the system reset.

Prototype

```
typedef struct {
  EFI_MISC_RESET_CAPABILITIES    ResetCapabilities;
  UINT16                         ResetCount;
  UINT16                         ResetLimit;
  UINT16                         ResetTimerInterval;
  UINT16                         ResetTimeout;
} EFI_MISC_RESET_CAPABILITIES_DATA;
```

Parameters

ResetCapabilities

The system reset capabilities. Type **EFI_MISC_RESET_CAPABILITIES** is defined in "Related Definitions" below.

ResetCount

The number of automatic system resets since the last intentional reset. A value of 0xFFFF indicates that the reset count is unknown.

ResetLimit

The maximum number of automatic resets that can be attempted. A value of 0xFFFF indicates that the limit is unknown.

ResetTimerInterval

The watchdog timer interval in minutes. If this timer is not reset within this value, then the system-reset timer is started. A value of 0xFFFF indicates that the interval is unknown.

ResetTimeout

The number of minutes before a reboot occurs. It occurs after a system power cycle, system reset, or an automatic reset. A value of 0xFFFF indicates that the interval is unknown.

Description

This data record refers to the system reset. This data record is a structure.

The type definition structure for **EFI_MISC_RESET_CAPABILITIES_DATA** is in SMBIOS 2.3.4, Type 23.

For this data record, **EFI_SUBCLASS_TYPE1_HEADER**.*RecordType* = **EFI_MISC_RESET_CAPABILITIES_RECORD_NUMBER**. Type **EFI_SUBCLASS_TYPE1_HEADER** is defined in the *Intel® Platform Innovation Framework for EFI Data Hub Subclass Design Guide*.

Related Definitions

```
//*******************************************************
// Record number
//*******************************************************
#define EFI_MISC_RESET_CAPABILITIES_RECORD_NUMBER    0x00000011


//*******************************************************
// EFI_MISC_RESET_CAPABILITIES
//*******************************************************
typedef struct {
  EFI_MISC_RESET_CAPABILITIES_TYPE  ResetCapabilities;
  UINT16                            ResetCount;
  UINT16                            ResetLimit;
  UINT16                            ResetTimerInterval;
  UINT16                            ResetTimeout;
} EFI_MISC_RESET_CAPABILITIES;
```

ResetCapabilities

Describes the types of resets that are supported. Type **EFI_MISC_RESET_CAPABILITIES_TYPE** is defined below.

ResetCount

The count of resets since the last intentional reset. A value of 0xFFFF indicates unknown.

ResetLimit

The number of consecutive times to attempt a system reset. A value of 0xFFFF indicates unknown.

ResetTimerInterval

The number of minutes to use for the watchdog timer. A value of 0xFFFF indicates unknown.

ResetTimeout

Indicates the number of minutes before a reboot is initiated. A value of 0xFFFF indicates unknown.

```
//*******************************************************
// EFI_MISC_RESET_CAPABILITIES_TYPE
//*******************************************************
typedef struct {
  UINT32  Status              :1;
  UINT32  BootOption          :2;
  UINT32  BootOptionOnLimit   :2;
  UINT32  WatchdogTimerPresent:1;
  UINT32  Reserved            :26;
} EFI_MISC_RESET_CAPABILITIES_TYPE;
```

Status

1 = Reset enabled by the user.

BootOption

Indicates the action to be taken following a watchdog reset:

- 00b: Reserved
- 01b: Operating system
- 10b: System utilities
- 11b: Do not reboot

BootOptionOnLimit

Indicates the action to be taken when the reset limit is reached:

- 00b: Reserved
- 01b: Operating system
- 10b: System utilities
- 11b: Do not reboot

WatchdogTimerPresent

1 = The watchdog timer is present.

Reserved

Reserved for future use. Must be initialized to zero

Hardware Security (Type 24)

EFI_MISC_HARDWARE_SECURITY_SETTINGS_DATA

Summary

This data record refers to the password and reset status of the system.

Prototype

```
typedef struct {
  EFI_MISC_HARDWARE_SECURITY_SETTINGS  HardwareSecuritySettings;
} EFI_MISC_HARDWARE_SECURITY_SETTINGS_DATA;
```

Parameters

HardwareSecuritySettings

The status of current security settings. Type **EFI_MISC_HARDWARE_SECURITY_SETTINGS** is defined in "Related Definitions" below.

Description

This data record refers to the password and reset status of the system. This data record is a bit map.

The type definition structure for **EFI_MISC_HARDWARE_SECURITY_SETTINGS_DATA** is in SMBIOS 2.3.4, Type 24, Offset 0x4.

For this data record, **EFI_SUBCLASS_TYPE1_HEADER**.*RecordType* = **EFI_MISC_HARDWARE_SECURITY_RECORD_NUMBER**. Type **EFI_SUBCLASS_TYPE1_HEADER** is defined in the *Intel® Platform Innovation Framework for EFI Data Hub Subclass Design Guide*.

Related Definitions

```
//*******************************************************
// Record number
//*******************************************************
#define EFI_MISC_HARDWARE_SECURITY_RECORD_NUMBER     0x00000012
```

```
//*******************************************************
// EFI_MISC_HARDWARE_SECURITY_SETTINGS
//*******************************************************
typedef struct  {
  UINT32 FrontPanelResetStatus         :2;
  UINT32 AdministratorPasswordStatus   :2;
  UINT32 KeyboardPasswordStatus        :2;
  UINT32 PowerOnPasswordStatus         :2;
  UINT32 Reserved                      :24;
} EFI_MISC_HARDWARE_SECURITY_SETTINGS;
```

FrontPanelResetStatus

The current status of the front panel reset:

- 00b: Disabled
- 01b: Enabled
- 10b: Not implemented
- 11b: Unknown

AdministratorPasswordStatus

The current status of the administrator password:

- 00b: Disabled
- 01b: Enabled
- 10b: Not implemented
- 11b: Unknown

KeyboardPasswordStatus

The current status of the keyboard password:

- 00b: Disabled
- 01b: Enabled
- 10b: Not implemented
- 11b: Unknown

PowerOnPasswordStatus

The current status of the power-on password:

- 00b: Disabled
- 01b: Enabled
- 10b: Not implemented
- 11b: Unknown

Reserved

Reserved for future use. Must be initialized to zero.

System Power Controls (Type 25)

System Power Controls (Type 25)

All fields are in Binary Coded Decimal (BCD). Any value out of range is ignored. The DMTF System Power Controls group contains a *Next Scheduled Power-on Time* that is specified as a number of seconds. Management software uses the date and time information that is specified in these structures to calculate the number of seconds from the current time to the specified power-on time. Out-of-range values are ignored.

Scheduled Power-On

EFI_MISC_SCHEDULED_POWER_ON_MONTH_DATA

Summary

This data record refers to the scheduled power-on.

Prototype

```
typedef struct {
  UINT16              ScheduledPoweronMonth;
  UINT16              ScheduledPoweronDayOfMonth;
  UINT16              ScheduledPoweronHour;
  UINT16              ScheduledPoweronMinute;
  UINT16              ScheduledPoweronSecond;
} EFI_MISC_SCHEDULED_POWER_ON_MONTH_DATA;
```

Parameters

ScheduledPoweronMonth

The month of the scheduled power-on. A valid number is 1 through 12.

ScheduledPoweronDayOfMonth

The day of the month of the scheduled power-on. Valid values are 1 through 31.

ScheduledPoweronHour

The hour of the scheduled power-on. Valid values are 0 through 23.

ScheduledPoweronMinute

The minute of the scheduled power-on. Valid values are 0 through 59.

ScheduledPoweronSecond

The second of the scheduled power-on. Valid values are 00 through 59.

Description

This data record refers to the scheduled power-on. This data record is a structure.

The type definition structure for **EFI_MISC_SCHEDULED_POWER_ON_MONTH_DATA** is in SMBIOS 2.3.4, Type 25, Table 3.3.26.1.

For this data record, **EFI_SUBCLASS_TYPE1_HEADER**.*RecordType* = **EFI_MISC_SCHEDULED_POWER_ON_MONTH_RECORD_NUMBER**. Type **EFI_SUBCLASS_TYPE1_HEADER** is defined in the *Intel® Platform Innovation Framework for EFI Data Hub Subclass Design Guide*.

Related Definitions

```
//*******************************************************
// Record number
//*******************************************************
#define EFI_MISC_SCHEDULED_POWER_ON_MONTH_RECORD_NUMBER
0x00000013
```

Voltage Probe (Type 26)

EFI_MISC_VOLTAGE_PROBE_DESCRIPTION_DATA

Summary

This data record refers to a voltage probe.

Prototype

```
typedef struct {
  STRING_REF                        VoltageProbeDescription;
  EFI_MISC_VOLTAGE_PROBE_LOCATION   VoltageProbeLocation;
  EFI_EXP_BASE10_DATA               VoltageProbeMaximumValue;
  EFI_EXP_BASE10_DATA               VoltageProbeMinimumValue;
  EFI_EXP_BASE10_DATA               VoltageProbeResolution;
  EFI_EXP_BASE10_DATA               VoltageProbeTolerance;
  EFI_EXP_BASE10_DATA               VoltageProbeAccuracy;
  EFI_EXP_BASE10_DATA               VoltageProbeNominalValue;
  EFI_EXP_BASE10_DATA               MDLowerNoncriticalThreshold;
  EFI_EXP_BASE10_DATA               MDUpperNoncriticalThreshold;
  EFI_EXP_BASE10_DATA               MDLowerCriticalThreshold;
  EFI_EXP_BASE10_DATA               MDUpperCriticalThreshold;
  EFI_EXP_BASE10_DATA               MDLowerNonrecoverableThreshold;
  EFI_EXP_BASE10_DATA               MDUpperNonrecoverableThreshold;
  UINT32                            VoltageProbeOemDefined;
} EFI_MISC_VOLTAGE_PROBE_DESCRIPTION_DATA;
```

Parameters

VoltageProbeDescription

A name describing the voltage probe. Type **STRING_REF** is defined in the *Intel® Platform Innovation Framework for EFI Human Interface Infrastructure Specification*.

VoltageProbeLocation

The location of the probe and its status. Type **EFI_MISC_VOLTAGE_PROBE_LOCATION** is defined in "Related Definitions" below.

VoltageProbeMaximumValue

The maximum voltage level in volts that is readable by the probe. A value of 0xFFFF FFFF indicates that the value is unknown. Type **EFI_EXP_BASE10_DATA** is defined in the *Intel® Platform Innovation Framework for EFI Data Hub Subclass Design Guide*.

VoltageProbeMinimumValue

The minimum voltage level in volts that is readable by the probe. A value of 0xFFFF FFFF indicates that the value is unknown.

VoltageProbeResolution

The resolution voltage level in volts that is readable by the probe. A value of 0xFFFF FFFF indicates that the value is unknown.

VoltageProbeTolerance

The tolerance of a value read by the probe in plus/minus volts. A value of 0xFFFF FFFF indicates that the value is unknown

VoltageProbeAccuracy

The accuracy of a value read by the probe in plus/minus one percent. A value of 0xFFFF FFFF indicates that the value is unknown.

VoltageProbeNominalValue

The nominal value in volts for the probe's reading. A value of 0xFFFF FFFF indicates that the value is unknown.

MDLowerNoncriticalThreshold

The lower noncritical threshold for this component

MDUpperNoncriticalThreshold

The upper noncritical threshold for this component.

MDLowerCriticalThreshold

The lower critical threshold for this component.

MDUpperCriticalThreshold

The upper critical threshold for this component.

MDLowerNonrecoverableThreshold

The lower nonrecoverable threshold for this component.

MDUpperNonrecoverableThreshold

The upper nonrecoverable threshold for this component.

VoltageProbeOemDefined

OEM- or BIOS-vendor-specific data.

Description

This data record refers to a voltage probe. This data record is a structure.

The type definition structure for **EFI_MISC_VOLTAGE_PROBE_DESCRIPTION_DATA** is in SMBIOS 2.3.4, Type 26, Table 3.3.27.1, Type 36.

For this data record, **EFI_SUBCLASS_TYPE1_HEADER**.*RecordType* = **EFI_MISC_VOLTAGE_PROBE_DESCRIPTION_RECORD_NUMBER**. Type **EFI_SUBCLASS_TYPE1_HEADER** is defined in the *Intel® Platform Innovation Framework for EFI Data Hub Subclass Design Guide*.

Related Definitions

```
//*******************************************************
// Record number
//*******************************************************
#define EFI_MISC_VOLTAGE_PROBE_DESCRIPTION_RECORD_NUMBER
                                                    0x00000014

//*******************************************************
// EFI_MISC_VOLTAGE_PROBE_LOCATION
//*******************************************************
typedef struct {
  UINT32 VoltageProbeSite        :5;
  UINT32 VoltageProbeStatus      :3;
  UINT32 Reserved                :24;
} EFI_MISC_VOLTAGE_PROBE_LOCATION;
```

VoltageProbeSite

The location of the probe within the chassis:

- 00000b: Reserved
- 00001b: Other
- 00010b: Unknown
- 00011b: Processor
- 00100b: Disk
- 00101b: Peripheral bay
- 00110b: System management module
- 00111b: Motherboard
- 01000b: Memory module
- 01001b: Processor module
- 01010b: Power unit
- 01011b: Add-in card
- 01100b–11111b: Reserved

VoltageProbeStatus

The last reading from the probe:

- 000b: Reserved
- 001b: Other
- 010b: Unknown
- 011b: OK
- 100b: Noncritical

- 101b: Critical
- 110b: Nonrecoverable
- 111b: Reserved

Reserved

Reserved for future use. Must be initialized to zero.

Cooling Device (Type 27)

EFI_MISC_COOLING_DEVICE_TEMP_LINK_DATA

Summary

This data record refers to a temperature probe.

Prototype

```
typedef struct {
  EFI_MISC_COOLING_DEVICE_TYPE   CoolingDeviceType;
  EFI_INTER_LINK_DATA            CoolingDeviceTemperatureLink;
  UINT16                         CoolingDeviceUnitGroup;
  EFI_EXP_BASE10_DATA            CoolingDeviceNominalSpeed;
  UINT32                         CoolingDeviceOemDefined;
} EFI_MISC_COOLING_DEVICE_TEMP_LINK_DATA;
```

Parameters

CoolingDeviceType

The type of cooling device and its status. Type **EFI_MISC_COOLING_DEVICE_TYPE** is defined in "Related Definitions" below.

CoolingDeviceTemperatureLink

The ID of a temperature probe. See Temperature Probe (Type 28). Type **EFI_INTER_LINK_DATA** is defined in the *Intel® Platform Innovation Framework for EFI Data Hub Subclass Design Guide*.

CoolingDeviceUnitGroup

The group of which this cooling device is a member. A value of 0x0000 indicates that the device is not a member of a group. A nonzero value indicates that multiple devices are in the same group and implies a redundant configuration.

CoolingDeviceNominalSpeed

The nominal value of the cooling device's rotational speed. A value of 0xFFFF FFFF indicates that the value is unknown. Type **EFI_EXP_BASE10_DATA** is defined in the *Intel® Platform Innovation Framework for EFI Data Hub Subclass Design Guide*.

CoolingDeviceOemDefined

OEM- or BIOS-specific data.

Description

This data record refers to a temperature probe. See Temperature Probe (Type 28). This data record is a structure.

The type definition structure for **EFI_MISC_COOLING_DEVICE_TEMP_LINK_DATA** is in SMBIOS 2.3.4, Type 27, Table 3.3.28.1.

For this data record, **EFI_SUBCLASS_TYPE1_HEADER**.*RecordType* = **EFI_MISC_COOLING_DEVICE_TEMP_LINK_RECORD_NUMBER**. Type **EFI_SUBCLASS_TYPE1_HEADER** is defined in the *Intel® Platform Innovation Framework for EFI Data Hub Subclass Design Guide*.

Related Definitions

```
//*******************************************************
// Record number
//*******************************************************
#define EFI_MISC_COOLING_DEVICE_TEMP_LINK_RECORD_NUMBER
                                                    0x00000015


//*******************************************************
// EFI_MISC_COOLING_DEVICE_TYPE
//*******************************************************
typedef struct {
  UINT32 CoolingDevice                  :5;
  UINT32 CoolingDeviceStatus            :3;
  UINT32 Reserved                       :24;
} EFI_MISC_COOLING_DEVICE_TYPE;
```

CoolingDevice

The type of cooling device:

- 00000b: Reserved
- 00001b: Other
- 00010b: Unknown
- 00011b: Fan
- 00100b: Centrifugal blower
- 00101b: Chip fan
- 00110b: Cabinet fan
- 00111b: Power supply fan
- 01000b: Heat pipe
- 01001b: Integrated refrigeration
- 01010b–10011b: Reserved
- 10100b: Active cooling
- 10101b: Passive cooling
- 10110b–11111b: Reserved

CoolingDeviceStatus

The last state of the device:

- 000b: Reserved
- 001b: Other
- 010b: Unknown
- 011b: OK
- 100b: Noncritical
- 101b: Critical
- 110b: Nonrecoverable
- 111b: Reserved

Reserved

Reserved for future use. Must be initialized to zero.

Temperature Probe (Type 28)

EFI_MISC_TEMPERATURE_PROBE_DESCRIPTION_DATA

Summary

This data record refers to a temperature probe.

Prototype

```
typedef struct {
  STRING_REF                    TemperatureProbeDescription;
  EFI_MISC_TEMPERATURE_PROBE_LOCATION
                                TemperatureProbeLocation;
  EFI_EXP_BASE10_DATA           TemperatureProbeMaximumValue;
  EFI_EXP_BASE10_DATA           TemperatureProbeMinimumValue;
  EFI_EXP_BASE10_DATA           TemperatureProbeResolution;
  EFI_EXP_BASE10_DATA           TemperatureProbeTolerance;
  EFI_EXP_BASE10_DATA           TemperatureProbeAccuracy;
  EFI_EXP_BASE10_DATA           TemperatureProbeNominalValue;
  EFI_EXP_BASE10_DATA           MDLowerNoncriticalThreshold;
  EFI_EXP_BASE10_DATA           MDUpperNoncriticalThreshold;
  EFI_EXP_BASE10_DATA           MDLowerCriticalThreshold;
  EFI_EXP_BASE10_DATA           MDUpperCriticalThreshold;
  EFI_EXP_BASE10_DATA           MDLowerNonrecoverableThreshold;
  EFI_EXP_BASE10_DATA           MDUpperNonrecoverableThreshold;
  UINT32                        TemperatureProbeOemDefined;
} EFI_MISC_TEMPERATURE_PROBE_DESCRIPTION_DATA;
```

Parameters

TemperatureProbeDescription

A name describing the temperature probe. Type **STRING_REF** is defined in the *Intel® Platform Innovation Framework for EFI Human Interface Infrastructure Specification*.

TemperatureProbeLocation

The location of the probe and its status. Type **EFI_MISC_TEMPERATURE_PROBE_LOCATION** is defined in "Related Definitions" below.

TemperatureProbeMaximumValue

The maximum temperature level in degrees Celsius that is readable by the probe. A value of 0xFFFF FFFF indicates that the value is unknown. Type **EFI_EXP_BASE10_DATA** is defined in the *Intel® Platform Innovation Framework for EFI Data Hub Subclass Design Guide*.

TemperatureProbeMinimumValue

The minimum temperature level in degrees Celsius that is readable by the probe. A value of 0xFFFF FFFF indicates that the value is unknown.

TemperatureProbeResolution

The resolution temperature level in degrees Celsius that is readable by the probe. A value of 0xFFFF FFFF indicates that the value is unknown.

TemperatureProbeTolerance

The tolerance of a value read by the probe in plus/minus degrees Celsius. A value of 0xFFFF indicates that the value is unknown.

TemperatureProbeAccuracy

The accuracy of a value read by the probe in plus/minus one percent. A value of 0xFFFF FFFF indicates that the value is unknown.

TemperatureProbeNominalValue

The nominal value in degrees Celsius for the probe's reading. A value of 0xFFFF FFFF indicates that the value is unknown.

MDLowerNoncriticalThreshold

The lower noncritical threshold for this component

MDUpperNoncriticalThreshold

The upper noncritical threshold for this component.

MDLowerCriticalThreshold

The lower critical threshold for this component.

MDUpperCriticalThreshold

The upper critical threshold for this component.

MDLowerNonrecoverableThreshold

The lower nonrecoverable threshold for this component.

MDUpperNonrecoverableThreshold

The upper nonrecoverable threshold for this component.

TemperatureProbeOemDefined

OEM- or BIOS-vendor-specific data.

Description

This data record refers to a temperature probe. This data record is a structure.

The type definition structure for **EFI_MISC_TEMPERATURE_PROBE_DESCRIPTION_DATA** is in SMBIOS 2.3.4, in the following types:

- Type 28, Table 3.3.29.1
- Type 36

For this data record, **EFI_SUBCLASS_TYPE1_HEADER**.*RecordType* = **EFI_MISC_TEMPERATURE_PROBE_DESCRIPTION_RECORD_NUMBER**. Type **EFI_SUBCLASS_TYPE1_HEADER** is defined in the *Intel® Platform Innovation Framework for EFI Data Hub Subclass Design Guide*.

Related Definitions

```
//*******************************************************
// Record number
//*******************************************************
#define EFI_MISC_TEMPERATURE_PROBE_DESCRIPTION_RECORD_NUMBER
                                                    0x00000016

//*******************************************************
// EFI_MISC_TEMPERATURE_PROBE_LOCATION
//*******************************************************
typedef struct {
  UINT32 TemperatureProbeSite          :5;
  UINT32 TemperatureProbeStatus        :3;
  UINT32 Reserved                      :24;
} EFI_MISC_TEMPERATURE_PROBE_LOCATION;
```

TemperatureProbeSite

The location of the probe within the chassis:

- 00000b: Reserved
- 00001b: Other
- 00010b: Unknown
- 00011b: Processor
- 00100b: Disk
- 00101b: Peripheral bay
- 00110b: System management module
- 00111b: Motherboard
- 01000b: Memory module
- 01001b: Processor module
- 01010b: Power unit
- 01011b: Add-in card
- 01100b: Front panel board
- 01101b: Back panel board
- 01110b: Power system board
- 01111b: Drive back plane
- 10000b–11111b: Reserved

TemperaureProbeStatus

The last temperature status read from the probe:

- 000b: Reserved
- 001b: Other
- 010b: Unknown
- 011b: OK
- 100b: Noncritical
- 101b: Critical
- 110b: Nonrecoverable
- 111b: Reserved

Reserved

Reserved for future use. Must be initialized to zero.

Electrical Current Probe (Type 29)

EFI_MISC_ELECTRICAL_CURRENT_PROBE_DESCRIPTION_DATA

Summary

This data record refers to an electrical current probe.

Prototype

```
typedef struct {
  STRING_REF                    ElectricalCurrentProbeDescription;
  EFI_MISC_ELECTRICAL_CURRENT_PROBE_LOCATION
                                ElectricalCurrentProbeLocation;
  EFI_EXP_BASE10_DATA           ElectricalCurrentProbeMaximumValue;
  EFI_EXP_BASE10_DATA           ElectricalCurrentProbeMinimumValue;
  EFI_EXP_BASE10_DATA           ElectricalCurrentProbeResolution;
  EFI_EXP_BASE10_DATA           ElectricalCurrentProbeTolerance;
  EFI_EXP_BASE10_DATA           ElectricalCurrentProbeAccuracy;
  EFI_EXP_BASE10_DATA           ElectricalCurrentProbeNominalValue;
  EFI_EXP_BASE10_DATA           MDLowerNoncriticalThreshold;
  EFI_EXP_BASE10_DATA           MDUpperNoncriticalThreshold;
  EFI_EXP_BASE10_DATA           MDLowerCriticalThreshold;
  EFI_EXP_BASE10_DATA           MDUpperCriticalThreshold;
  EFI_EXP_BASE10_DATA           MDLowerNonrecoverableThreshold;
  EFI_EXP_BASE10_DATA           MDUpperNonrecoverableThreshold;
  UINT32                        ElectricalCurrentProbeOemDefined;
} EFI_MISC_ELECTRICAL_CURRENT_PROBE_DESCRIPTION_DATA;
```

Parameters

ElectricalCurrentProbeDescription

A name describing the electrical current probe. Type **STRING_REF** is defined in the *Intel® Platform Innovation Framework for EFI Human Interface Infrastructure Specification*.

ElectricalCurrentProbeLocation

The location of the probe and its status. Type **EFI_MISC_ELECTRICAL_CURRENT_PROBE_LOCATION** is defined in "Related Definitions" below.

ElectricalCurrentProbeMaximumValue

The maximum electrical current level in amperes that is readable by the probe. A value of 0xFFFF FFFF indicates that the value is unknown. Type **EFI_EXP_BASE10_DATA** is defined in the *Intel® Platform Innovation Framework for EFI Data Hub Subclass Design Guide*.

ElectricalCurrentProbeMinimumValue

The minimum electrical current level in amperes that is readable by the probe. A value of 0xFFFF FFFF indicates that the value is unknown.

ElectricalCurrentProbeResolution

The resolution electrical current level in amperes that is readable by the probe. A value of 0xFFFF FFFF indicates that the value is unknown.

ElectricalCurrentProbeTolerance

The tolerance in plus/minus amperes of a value read by the probe. A value of 0xFFFF FFFF indicates that the value is unknown.

ElectricalCurrentProbeAccuracy

The accuracy in plus/minus one percent of a value read by the probe. A value of 0xFFFF FFFF indicates that the value is unknown.

ElectricalCurrentProbeNominalValue

The nominal value in amperes for the probe's reading. A value of 0xFFFF FFFF indicates that the value is unknown.

MDLowerNoncriticalThreshold

The lower noncritical threshold for this component

MDUpperNoncriticalThreshold

The upper noncritical threshold for this component.

MDLowerCriticalThreshold

The lower critical threshold for this component.

MDUpperCriticalThreshold

The upper critical threshold for this component.

MDLowerNonrecoverableThreshold

The lower nonrecoverable threshold for this component.

MDUpperNonrecoverableThreshold

The upper nonrecoverable threshold for this component.

ElectricalCurrentProbeOemDefined

OEM- or BIOS-vendor-specific data.

Description

This data record refers to an electrical current probe. This data record is a structure.

The type definition structure for **EFI_MISC_ELECTRICAL_CURRENT_PROBE_DESCRIPTION_DATA** is in SMBIOS 2.3.4, in the following types:

- Type 29, Table 3.3.30.1
- Type 36

For this data record, **EFI_SUBCLASS_TYPE1_HEADER**.*RecordType* = **EFI_MISC_ELECTRICAL_CURRENT_PROBE_DESCRIPTION_RECORD_NUMBER**. Type **EFI_SUBCLASS_TYPE1_HEADER** is defined in the *Intel® Platform Innovation Framework for EFI Data Hub Subclass Design Guide*.

Related Definitions

```
//*******************************************************
// Record number
//*******************************************************
#define EFI_MISC_ELECTRICAL_CURRENT_PROBE_DESCRIPTION_RECORD_NUMBER
                                                    0x00000017

//*******************************************************
// EFI_MISC_ELECTRICAL_CURRENT_PROBE_LOCATION
//*******************************************************
typedef struct {
  UINT32 ElectricalCurrentProbeSite    :5;
  UINT32 ElectricalCurrentProbeStatus  :3;
  UINT32 Reserved                      :24;
} EFI_MISC_ELECTRICAL_CURRENT_PROBE_LOCATION;
```

ElectricalCurrentProbeSite

The location of the probe within the chassis:

- 00000b: Reserved
- 00001b: Other
- 00010b: Unknown
- 00011b: Processor
- 00100b: Disk
- 00101b: Peripheral bay
- 00110b: System management module
- 00111b: Motherboard
- 01000b: Memory module
- 01001b: Processor module
- 01010b: Power unit
- 01011b: Add-in card
- 01100b–11111b: Reserved

ElectricalCurrentProbeStatus

The last current status read by the probe:

- 000b: Reserved
- 001b: Other
- 010b: Unknown
- 011b: OK

- 100b: Noncritical
- 101b: Critical
- 110b: Nonrecoverable
- 111b: Reserved

Reserved

Reserved for future use. Must be initialized to zero.

Out-of-Band Remote Access (Type 30)

EFI_MISC_REMOTE_ACCESS_MANUFACTURER_DESCRIPTION_DATA

Summary

This data record refers to a manufacturer of the out-of-band access facility.

Prototype

```
typedef struct {
  STRING_REF                    RemoteAccessManufacturerNameDescription;
  EFI_MISC_REMOTE_ACCESS_CONNECTIONS     RemoteAccessConnections;
} EFI_MISC_REMOTE_ACCESS_MANUFACTURER_DESCRIPTION_DATA;
```

Parameters

RemoteAccessManufacturerNameDescription

A name describing the manufacturer of the out-of-band access facility. Type **STRING_REF** is defined in the *Intel® Platform Innovation Framework for EFI Human Interface Infrastructure Specification*.

RemoteAccessConnections

The current remote access connections. Type **EFI_MISC_REMOTE_ACCESS_CONNECTIONS** is defined in "Related Definitions" below.

Description

This data record refers to a manufacturer of the out-of-band access facility. This data record is a structure.

The type definition structure for **EFI_MISC_REMOTE_ACCESS_MANUFACTURER_DESCRIPTION_DATA** is in SMBIOS 2.3.4, Type 30.

For this data record, **EFI_SUBCLASS_TYPE1_HEADER**.*RecordType* = **EFI_MISC_REMOTE_ACCESS_MANUFACTURER_DESCRIPTION_RECORD_NUMBER**. Type **EFI_SUBCLASS_TYPE1_HEADER** is defined in the *Intel® Platform Innovation Framework for EFI Data Hub Subclass Design Guide*.

Related Definitions

```
//*******************************************************
// Record number
//*******************************************************
#define EFI_MISC_REMOTE_ACCESS_MANUFACTURER_DESCRIPTION_RECORD_NUMBER
                                                    0x00000018
```

```
//*******************************************************
// EFI_MISC_REMOTE_ACCESS_CONNECTIONS
//*******************************************************
typedef struct  {
  UINT32 InboundConnectionEnabled           :1;
  UINT32 OutboundConnectionEnabled          :1;
  UINT32 Reserved                           :30;
} EFI_MISC_REMOTE_ACCESS_CONNECTIONS;
```

InboundConnectionEnabled

1 = The facility is allowed to initiate outbound connections to receive incoming connections for the purpose of remote operations or problem management.

OutboundConnectionEnabled

1 = The facility is allowed to initiate outbound connections to contact an alert management facility when critical connections occur.

Reserved

Reserved for future use. Must be initialized to zero.

Boot Integrity Services (BIS) Entry Point (Type 31)

EFI_MISC_BIS_ENTRY_POINT_DATA

Summary

This data record refers to the Boot Integrity Services (BIS) entry point address.

Prototype

```
typedef struct {
  EFI_PHYSICAL_ADDRESS                  BisEntryPoint;
} EFI_MISC_BIS_ENTRY_POINT_DATA;
```

Parameters

BisEntryPoint

The physical address of the BIS entry point. Type **EFI_PHYSICAL_ADDRESS** is defined in **AllocatePages()** in the *EFI 1.10 Specification.*

Description

This data record refers to the BIS entry point address. This data record is a structure.

The type definition structure for **EFI_MISC_BIS_ENTRY_POINT_DATA** is in SMBIOS 2.3.4, Type 31.

For this data record, **EFI_SUBCLASS_TYPE1_HEADER**.*RecordType* = **EFI_MISC_BIS_ENTRY_POINT_RECORD_NUMBER**. Type **EFI_SUBCLASS_TYPE1_HEADER** is defined in the *Intel® Platform Innovation Framework for EFI Data Hub Subclass Design Guide*.

Related Definitions

```
//*******************************************************
// Record number
//*******************************************************
#define EFI_MISC_BIS_ENTRY_POINT_RECORD_NUMBER       0x00000019
```

System Boot Information (Type 32)

EFI_MISC_BOOT_INFORMATION_STATUS_DATA

Summary

This data record refers to the system boot information status.

Prototype

```
typedef struct {
  EFI_MISC_BOOT_INFORMATION_STATUS_TYPE  BootInformationStatus;
  UINT8                                  BootInformationData[9];
} EFI_MISC_BOOT_INFORMATION_STATUS_DATA;
```

Parameters

BootInformationStatus

The system boot information status. Type **EFI_MISC_BOOT_INFORMATION_STATUS_TYPE** is defined in "Related Definitions" below.

BootInformationData

OEM- or product-specific implementations.

Description

This data record refers to the system boot information status. This data record is a structure.

The type definition structure for **EFI_MISC_BOOT_INFORMATION_STATUS_DATA** is in SMBIOS 2.3.4, Type 32.

For this data record, **EFI_SUBCLASS_TYPE1_HEADER**.*RecordType* = **EFI_MISC_BOOT_INFORMATION_STATUS_RECORD_NUMBER**. Type **EFI_SUBCLASS_TYPE1_HEADER** is defined in the *Intel® Platform Innovation Framework for EFI Data Hub Subclass Design Guide*.

Related Definitions

```
//*******************************************************
// Record number
//*******************************************************
#define EFI_MISC_BOOT_INFORMATION_STATUS_RECORD_NUMBER  0x0000001A
```

```
//*******************************************************
// EFI_MISC_BOOT_INFORMATION_STATUS_TYPE
//*******************************************************
typedef enum {
  EfiBootInformationStatusNoError = 0,
  EfiBootInformationStatusNoBootableMedia = 1,
  EfiBootInformationStatusNormalOSFailedLoading = 2,
  EfiBootInformationStatusFirmwareDetectedFailure = 3,
  EfiBootInformationStatusOSDetectedFailure = 4,
  EfiBootInformationStatusUserRequestedBoot = 5,
  EfiBootInformationStatusSystemSecurityViolation = 6,
  EfiBootInformationStatusPreviousRequestedImage = 7,
  EfiBootInformationStatusWatchdogTimerExpired = 8,
  EfiBootInformationStatusStartReserved = 9,
  EfiBootInformationStatusStartOemSpecific = 128,
  EfiBootInformationStatusStartProductSpecific = 192
} EFI_MISC_BOOT_INFORMATION_STATUS_TYPE;
```

NOTE

The last three entries specify the start of a range for reserved, OEM-specific, or product-specific entries respectively.

Management Device (Type 34)

EFI_MISC_MANAGEMENT_DEVICE_DESCRIPTION_DATA

Summary

This data record refers to a management device.

Prototype

```
typedef struct {
  STRING_REF                          ManagementDeviceDescription;
  EFI_MISC_MANAGEMENT_DEVICE_TYPE     ManagementDeviceType;
  UINTN                               ManagementDeviceAddress;
  EFI_MISC_MANAGEMENT_DEVICE_ADDRESS_TYPE
                                      ManagementDeviceAddressType;
} EFI_MISC_MANAGEMENT_DEVICE_DESCRIPTION_DATA;
```

Parameters

ManagementDeviceDescription

A name describing the management device. Type **STRING_REF** is defined in the *Intel® Platform Innovation Framework for EFI Human Interface Infrastructure Specification*.

ManagementDeviceType

The type of management device. Type **EFI_MISC_MANAGEMENT_DEVICE_TYPE** is defined in "Related Definitions" below.

ManagementDeviceAddress

The management device address.

ManagementDeviceAddressType

The type of address presented in *ManagementDeviceAddress*. Type **EFI_MISC_MANAGEMENT_DEVICE_ADDRESS_TYPE** is defined in "Related Definitions" below.

Description

This data record refers to a management device. This data record is a structure.

The type definition structure for **EFI_MISC_MANAGEMENT_DEVICE_DESCRIPTION_DATA** is in SMBIOS 2.3.4, Type 34, in the following tables:

- Table 3.3.35.1
- Table 3.3.35.2

For this data record, **EFI_SUBCLASS_TYPE1_HEADER**.*RecordType* = **EFI_MISC_MANAGEMENT_DEVICE_DESCRIPTION_RECORD_NUMBER**. Type **EFI_SUBCLASS_TYPE1_HEADER** is defined in the *Intel® Platform Innovation Framework for EFI Data Hub Subclass Design Guide*.

Related Definitions

```
//*********************************************************
// Record number
//*********************************************************
#define EFI_MISC_MANAGEMENT_DEVICE_DESCRIPTION_RECORD_NUMBER
                                                        0x0000001B

//*********************************************************
// EFI_MISC_MANAGEMENT_DEVICE_TYPE
//*********************************************************
typedef enum {
  EfiManagementDeviceTypeOther = 1,
  EfiManagementDeviceTypeUnknown = 2,
  EfiManagementDeviceTypeLm75 = 3,
  EfiManagementDeviceTypeLm78 = 4,
  EfiManagementDeviceTypeLm79 = 5,
  EfiManagementDeviceTypeLm80 = 6,
  EfiManagementDeviceTypeLm81 = 7,
  EfiManagementDeviceTypeAdm9240 = 8,
  EfiManagementDeviceTypeDs1780 = 9,
  EfiManagementDeviceTypeMaxim1617 = A,
  EfiManagementDeviceTypeGl518Sm = B,
  EfiManagementDeviceTypeW83781D = C,
  EfiManagementDeviceTypeHt82H791 = D
} EFI_MISC_MANAGEMENT_DEVICE_TYPE;


//*********************************************************
// EFI_MISC_MANAGEMENT_DEVICE_ADDRESS_TYPE
//*********************************************************
typedef enum {
  EfiManagementDeviceAddressTypeOther = 1,
  EfiManagementDeviceAddressTypeUnknown = 2,
  EfiManagementDeviceAddressTypeIOPort = 3,
  EfiManagementDeviceAddressTypeMemory = 4,
  EfiManagementDeviceAddressTypeSmbus = 5
} EFI_MISC_MANAGEMENT_DEVICE_ADDRESS_TYPE;
```

Management Device Component (Type 35)

EFI_MISC_MANAGEMENT_DEVICE_COMPONENT_DESCRIPTION_DATA

Summary

This data record refers to a management device component.

Prototype

```
typedef struct {
  STRING_REF                ManagementDeviceComponentDescription;
  EFI_INTER_LINK_DATA       ManagementDeviceLink;
  EFI_INTER_LINK_DATA       ManagementDeviceComponentLink;
} EFI_MISC_MANAGEMENT_DEVICE_COMPONENT_DESCRIPTION_DATA;
```

Parameters

ManagementDeviceComponentDescription

A name describing the management device. Type **STRING_REF** is defined in the *Intel® Platform Innovation Framework for EFI Human Interface Infrastructure Specification*.

ManagementDeviceLink

The ID of a management device. See Management Device (Type 34) for management device structures. Type **EFI_INTER_LINK_DATA** is defined in the *Intel® Platform Innovation Framework for EFI Data Hub Subclass Design Guide*.

ManagementDeviceComponentLink

The ID of a component device like a voltage probe. See Voltage Probe (Type 26) through Electrical Current Probe (Type 29) for device component structures.

Description

This data record refers to a management device component. This data record is a structure.

The type definition structure for **EFI_MISC_MANAGEMENT_DEVICE_COMPONENT_DESCRIPTION_DATA** is in SMBIOS 2.3.4, Type 35.

For this data record, **EFI_SUBCLASS_TYPE1_HEADER**.*RecordType* = **EFI_MISC_MANAGEMENT_DEVICE_COMPONENT_DESCRIPTION_RECORD_NUMBER**. Type **EFI_SUBCLASS_TYPE1_HEADER** is defined in the *Intel® Platform Innovation Framework for EFI Data Hub Subclass Design Guide*.

Related Definitions

```
//*******************************************************
// Record number
//*******************************************************
#define EFI_MISC_MANAGEMENT_DEVICE_COMPONENT_DESCRIPTION_RECORD_NUMBER
                                                        0x0000001C
```

IPMI Device Information (Type 38)

EFI_MISC_IPMI_INTERFACE_TYPE_DATA

Summary

This data record refers to the Intelligent Platform Management Interface (IPMI) type.

Prototype

```
typedef struct {
  EFI_MISC_IPMI_INTERFACE_TYPE    IpmiInterfaceType;
  EFI_MISC_IPMI_SPECIFICATION_REVISION
                                  IpmiSpecificationRevision;
  UINT16                          IpmiI2CSlaveAddress;
  UINT16                          IpmiNvDeviceAddress;
  UINT64                          IpmiBaseAddress;
  EFI_DEVICE_PATH_PROTOCOL        IpmiDevicePath;
} EFI_MISC_IPMI_INTERFACE_TYPE_DATA;
```

Parameters

IpmiInterfaceType

The IPMI Baseboard Management Controller (BMC) interface type. Type **EFI_MISC_IPMI_INTERFACE_TYPE** is defined in "Related Definitions" below.

IpmiSpecificationRevision

The revision of the *Intelligent Platform Management Interface Specification.* Type **EFI_MISC_IPMI_SPECIFICATION_REVISION** is defined in "Related Definitions" below.

IpmiI2CSlaveAddress

The I2C slave address of this BMC.

IpmiNvDeviceAddress

The bus ID of the nonvolatile storage device. A value of 0xFFFF indicates that there is no storage device.

IpmiBaseAddress

The I/O or memory-mapped base address of the BMC. If the least significant bit is a 1, then the address is I/O; otherwise, the address is memory mapped.

IpmiDevicePath

The path to the physical device.

Description

This data record refers to the IPMI type. This data record is a structure.

The type definition structure for **EFI_MISC_IPMI_INTERFACE_TYPE_DATA** is in SMBIOS 2.3.4, Type 38, Table 3.3.39.1.

For this data record, **EFI_SUBCLASS_TYPE1_HEADER**.*RecordType* = **EFI_MISC_IPMI_INTERFACE_TYPE_RECORD_NUMBER**. Type **EFI_SUBCLASS_TYPE1_HEADER** is defined in the *Intel® Platform Innovation Framework for EFI Data Hub Subclass Design Guide*.

Related Definitions

```
//*******************************************************
// Record number
//*******************************************************
#define EFI_MISC_IPMI_INTERFACE_TYPE_RECORD_NUMBER 0x0000001D


//*******************************************************
// EFI_MISC_IPMI_INTERFACE_TYPE
//*******************************************************
typedef enum {
  EfiIpmiOther = 0,
  EfiIpmiKcs = 1,
  EfiIpmiSmic = 2,
  EfiIpmiBt = 3,
} EFI_MISC_IPMI_INTERFACE_TYPE;


//*******************************************************
// EFI_MISC_IPMI_SPECIFICATION_REVISION
//*******************************************************
typedef struct {
  UINT16 IpmiSpecLeastSignificantDigit :4;
  UINT16 IpmiSpecMostSignificantDigit  :4;
  UINT16 Reserved                      :8;
} EFI_MISC_IPMI_SPECIFICATION_REVISION;
```

IpmiSpecLeastSignificantDigit

The least significant digit of the IPMI specification that this record referenced.

IpmiSpecMostSignificantDigit

The most significant digit of the IPMI specification that this record referenced.

Reserved

Reserved for future use. Should be initialized to zero.

System Power Supply (Type 39)

EFI_MISC_POWER_SUPPLY_UNIT_GROUP_DATA

Summary

This data record refers to a system power supply.

Prototype

```
typedef struct {
  UINT16                        PowerUnitGroup;
  STRING_REF                    PowerSupplyLocation;
  STRING_REF                    PowerSupplyDeviceName;
  STRING_REF                    PowerSupplyManufacturerName;
  STRING_REF                    PowerSupplySerialNumber;
  STRING_REF                    PowerSupplyAssetTag;
  STRING_REF                    PowerSupplyModelPartNumber;
  STRING_REF                    PowerSupplyRevisionLevel;
  UINT16                        PowerSupplyMaxPowerCapacity;
  EFI_MISC_POWER_SUPPLY_CHARACTERISTICS
                                PowerSupplyCharacteristics;
  EFI_INTER_LINK_DATA           PowerSupplyInputVoltageProbeLink;
  EFI_INTER_LINK_DATA           PowerSupplyCoolingDeviceLink;
  EFI_INTER_LINK_DATA           PowerSupplyInputCurrentProbeLink;
} EFI_MISC_POWER_SUPPLY_UNIT_GROUP_DATA;
```

Parameters

PowerUnitGroup

The group of which this power supply is a member. A value of 0x0000 indicates that the device is not a member of a group. A nonzero value indicates that multiple devices are in the same group and implies a redundant configuration.

PowerSupplyLocation

A name indicating the power supply's location. Type **STRING_REF** is defined in the *Intel® Platform Innovation Framework for EFI Human Interface Infrastructure Specification*.

PowerSupplyDeviceName

The power supply's device name.

PowerSupplyManufacturerName

The power supply manufacturer's name.

PowerSupplySerialNumber

The power supply's serial number.

PowerSupplyAssetTag

The power supply's asset tag.

PowerSupplyModelPartNumber

The power supply's model part number.

PowerSupplyRevisionLevel

The revision of the power supply.

PowerSupplyMaxPowerCapacity

The maximum sustained power output in watts. The DMTF specification is in milliwatts. A value of 0x8000 indicates that the value is unknown.

PowerSupplyCharacteristics

The characteristics and status of the power supply. Type **EFI_MISC_POWER_SUPPLY_CHARACTERISTICS** is defined in "Related Definitions" below.

PowerSupplyInputVoltageProbeLink

The ID of the input voltage probe. See Voltage Probe (Type 26) for input voltage probe structures. Type **EFI_INTER_LINK_DATA** is defined in the *Intel® Platform Innovation Framework for EFI Data Hub Subclass Design Guide*.

PowerSupplyCoolingDeviceLink

The ID of a cooling device. See Cooling Device (Type 27) for cooling device structures.

PowerSupplyInputCurrentProbeLink

The ID of the input current probe. See Electrical Current Probe (Type 29) for input current probe structures.

Description

This data record refers to a system power supply. This data record is structure.

The type definition structure for **EFI_MISC_POWER_SUPPLY_UNIT_GROUP_DATA** is in SMBIOS 2.3.4, Type 39, Table 3.3.40.1.

For this data record, **EFI_SUBCLASS_TYPE1_HEADER**.*RecordType* = **EFI_MISC_POWER_SUPPLY_UNIT_GROUP_RECORD_NUMBER**. Type **EFI_SUBCLASS_TYPE1_HEADER** is defined in the *Intel® Platform Innovation Framework for EFI Data Hub Subclass Design Guide*.

Related Definitions

```
//*******************************************************
// Record number
//*******************************************************
#define EFI_MISC_POWER_SUPPLY_UNIT_GROUP_RECORD_NUMBER 0x0000001E
```

```
//*******************************************************
// EFI_MISC_POWER_SUPPLY_CHARACTERISTICS
//*******************************************************
typedef struct {
  UINT32 PowerSupplyHotReplacable       :1;
  UINT32 PowerSupplyPresent             :1;
  UINT32 PowerSupplyIsUnplugged         :1;
  UINT32 InputVoltageRangeSwitch        :4;
  UINT32 PowerSupplyStatus              :3;
  UINT32 PowerSupplyType                :4;
  UINT32 Reserved                       :18;
} EFI_MISC_POWER_SUPPLY_CHARACTERISTICS;
```

PowerSupplyHotReplacable

Indicates that the power supply can be replaced while AC is connected to the chassis.

PowerSupplyPresent

Indicates that the power supply is present in the system.

PowerSupplyIsUnplugged

Indicates that the power supply is currently removed from the system.

InputVoltageRangeSwitch

- 0000b: Reserved
- 0001b: Other
- 0010b: Unknown
- 0011b: Manual
- 0100b: Auto switch
- 0101b: Wide range
- 0110b: Not applicable
- 0111b–1111b: Reserved

PowerSupplyStatus

Indicates the status of the power supply:

- 000b: Reserved
- 001b: Other
- 010b: Unknown
- 011b: OK
- 100b: Noncritical
- 101b: Critical
- 110b: Reserved
- 111b: Reserved

PowerSupplyType

The type of power supply being described:

- 0000b: Reserved
- 0001b: Other
- 0010b: Unknown
- 0011b: Linear
- 0100b: Switching
- 0101b: Battery
- 0110b: UPS
- 0111b: Converter
- 1000b: Regulator
- 1001b–1111b: Reserved

Reserved

Reserved for future use. Must be initialized to zero.

SMBIOS Structure Encapsulation (Types 0x80 to 0xFF)

EFI_MISC_SMBIOS_STRUCT_ENCAPSULATION_DATA

Summary

This data record carries one complete SMBIOS structure.

Prototype

```
typedef struct {
  EFI_SMBIOS_STRUCTURE_HDR   Header;
  UINT8                      RawData[1];
} EFI_MISC_SMBIOS_STRUCT_ENCAPSULATION_DATA;
```

Parameters

Header

The SMBIOS structure header. The *Type* and *Length* fields must correspond to the desired SMBIOS structure. *Handle* is ignored. Type **EFI_SMBIOS_STRUCTURE_HDR** is defined in "Related Definitions" below.

RawData

The raw data to be included in this SMBIOS structure.

Description

This data record carries one complete SMBIOS structure. Unlike other records, this record must contain the entire SMBIOS structure including the structure type, length, and handle. The structure's *Type* and *Length* must contain the values corresponding to the corresponding SMBIOS structure. The *Handle* field will be ignored by the consumers. As in SMBIOS structures, *Length* represents the length of the formatted part of the structure. If the SMBIOS structure requires any strings, raw strings (not string tokens) must be included in this record after the formatted data, just like the actual SMBIOS structures. The length of this data record depends upon the *Length* field and the length of the unformatted data. A consumer of this record can determine the length by using the *RecordSize* field in the data hub record header (**EFI_DATA_RECORD_HEADER**, which is defined in the *Intel® Platform Innovation Framework for EFI Data Hub Specification*) or by parsing the unformatted data such as SMBIOS table parsers.

This record is intended to be used for OEM-type SMBIOS structures. According to the *SMBIOS Specification*, structure types 0x80 to 0xFF are available for OEM usage. Any driver that creates an SMBIOS table based on the data hub records will copy the contents of **EFI_MISC_SMBIOS_STRUCT_ENCAPSULATION_DATA** to the SMBIOS table and update the handle number. This driver is allowed to reject a record of this type if the *Type* is less than 0x80. Therefore, producers should not use this type of record for generating standard SMBIOS structures in the range of 0 to 0x7F.

For this data record, **EFI_SUBCLASS_TYPE1_HEADER**.*RecordType* = **EFI_MISC_SMBIOS_STRUCT_ENCAP_RECORD_NUMBER**. Type **EFI_SUBCLASS_TYPE1_HEADER** is defined in the *Intel® Platform Innovation Framework for EFI Data Hub Subclass Design Guide*.

Related Definitions

```
//*******************************************************
// Record number
//*******************************************************
#define EFI_MISC_SMBIOS_STRUCT_ENCAP_RECORD_NUMBER 0x0000001F


//*******************************************************
// EFI_SMBIOS_STRUCTURE_HDR
//*******************************************************
//
// The SMBIOS structure header
//

#pragma pack(1)
typedef struct {
  UINT8      Type;
  UINT8       Length;
  UINT16      Handle;
} EFI_SMBIOS_STRUCTURE_HDR;
#pragma pack()
```

4 Examples

Legacy Miscellaneous Record Numbers

Last PCI Bus Number

Data Example: The last PCI bus enumerated is 7.

```
LastPciBus = 0x07;
```

BIOS Information (Type 0) Record Numbers

BIOS Vendor

Data Example: The BIOS vendor is AMI.

```
BiosVendor = String reference to "AMI",0;
```

BIOS Version

Data Example: The BIOS version is production release 10.

```
BiosVersion = String reference to "P10",0;
```

BIOS Release Date

Data Example: The BIOS release date is January 31, 2001.

```
BiosReleaseDate = String reference to "01/31/2001",0;
```

BIOS Starting Address

Data Example: The BIOS start of code space is FFFE0000 physical.

```
BiosStartingAddress = 0xFFFE0000;
```

BIOS Physical Device Size

Data Example: Physical size is 256 KB.

```
BiosPhysicalDeviceSize.Value  = 0x04;
BiosPhysicalDeviceSize.Exponent  = 0x04;
```

BIOS Characteristics

Data Example: BIOS supports PCI; is upgradeable; can be shadowed; and supports the following: boot from CD, selectable boot, print screen, 8042, serial, and parallel.

```
BiosCharacteristics1 = 0000 0000 0000 0000   0000 0000 0000 0000
                       0011 1100 0000 0001   1001 1000 1000 0000b;
```

BIOS Characteristics Extension Bytes

Data Example: The BIOS does not support any additional characteristics in the extension bytes.

```
BiosCharacteristics2 = 0000 0000 0000 0000   0000 0000 0000 0000b;
```

System Information (Type 1) Record Numbers

System Manufacturer

Data Example: The system manufacturer is Fly-by-Night.

```
SystemManufacturer = String reference to "Fly-by-Night",0;
```

System Product Name

Data Example: The system product name is Crash-n-Burn.

```
SystemProductName = String reference to "Crash-n-Burn",0;
```

System Version

Data Example: The system version is 1001.

```
SystemVersion =String reference to "1001",0;
```

System Serial Number

Data Example: The system's serial number is 1.

```
SystemSerialNumber = String reference to "01",0;
```

System UUID

Data Example: The system UUID is DEAD0000-1111-2222-3333-444455556666.

```
SystemUuid =  0xDEAD000011112222333344445556666;
```

System Wake-up Type

Data Example: The system woke up from PCI PME#.

```
SystemWakeupType = EfiSystemWakeupTypePciPme;
```

Baseboard Information (Type 2) Record Numbers

Baseboard Manufacturer

Data Example: The system's product name is Crispy PC.

```
BaseBoardManufacturer = String reference to "Crispy PC",0;
```

Baseboard Product Name

Data Example: The baseboard's product name is Krakatoa.

```
BaseBoardProductName = String reference to "Krakatoa",0;
```

Baseboard Version

Data Example: The baseboard version is 9999.

```
BaseBoardVersion = Sting reference to "9999",0;
```

Baseboard Serial Number

Data Example: The baseboard's serial number is DEAD987.

```
BaseBoardSerialNumber = String reference to "DEAD987",0;
```

Baseboard Asset Tag

Data Example: The baseboard's asset tag is 1234.

```
BaseBoardAssetTag = String reference to "1234",0;
```

Baseboard Chassis Location

Data Example: The chassis location is the motherboard.

```
BaseBoardChassisLocation = String reference to "Motherboard",0;
```

Baseboard Feature Flags

Data Example: The baseboard is a motherboard and is replacable.

```
BaseBoardFeatureFlags = 0x09;
```

Baseboard Type

Data Example: The baseboard is a motherboard.

```
BaseBoardType = EfiBaseBoardTypeMotherBoard;
```

Baseboard Chassis Link

Data Example: The *Instance* number of the chassis enclosure (type 3) that contains this baseboard is 1.

```
BaseBoardChassisLink = 0x01;
```

Baseboard Number of Links

Data Example: This structure does not have any links to other structures.

```
BaseBoardNumberLinks = 0x00;
```

Baseboard LinkN

Data Example: There are no other links.

```
LinkN = 0x00;
```

System/Chassis Enclosure (Type 3) Record Numbers

Chassis Manufacturer

Data Example: The chassis manufacturer is Slice-n-Dice.

```
ChassisManufacturer = String reference to "Slice-n-Dice",0;
```

Chassis Version

Data Example: The chassis version is No Fingers.

```
ChassisVersion = String reference to "No Fingers",0;
```

Chassis Serial Number

Data Example: The chassis's serial number is 123.

```
ChassisSerialNumber = String reference to "123",0;
```

Chassis Asset Tag Number

Data Example: The chassis's asset tag number is 444.

```
ChassisAssetTag = String reference to "444",0;
```

Chassis Type

Data Example: The chassis type is pizza box.

```
ChassisType = EfiMiscChassisTypePizzaBox;
```

Chassis Boot-up State

Data Example: The chassis boot-up state is safe.

```
ChassisBootupState = EfiChassisStateSafe;
```

Chassis Power Supply State

Data Example: The chassis's power supply state is safe.

```
ChassisPowerSupplyState = EfiChassisStateSafe;
```

Chassis Thermal State

Data Example: The chassis's thermal state is safe.

```
ChassisThermalState = EfiChassisStateSafe;
```

Chassis Security State

Data Example: The chassis's security state is unknown.

```
ChassisSecurityState = EfiChassisSecurityStatusUnknown;
```

Chassis OEM Defined

Data Example: The chassis's OEM defined field is null.

```
ChassisOemDefined = 0x00;
```

Chassis Height

Data Example: The chassis is 2 "U"s high.

```
ChassisHeight = 0x02;
```

Chassis Number of Power Cords

Data Example: One power cord is associated with chassis.

```
ChassisNumberPowerCords = 0x01;
```

Chassis Element Count

Data Example: No element records follow.

```
ChassisElementCount = 0x00;
```

Chassis Element Record Length

Data Example: The length of the chassis element record is the size of **EFI_MISC_ELEMENTS**.

```
ChassisElementRecordLength = sizeof (EFI_MISC_ELEMENTS);
```

Chassis Elements

Data Example: In the first example, the *ChassisBaseBoard* field is valid and the *ChassisElementStructure* field is invalid. The baseboard is a motherboard, and the number of motherboards must be 1.

```
ChassisElements.ChassisElementType.RecordType = 0;
ChassisElements.ChassisBaseBoard = EfiBaseBoardTypeMotherBoard;
ChassisElements.ChassisElementMinimum = 1;
ChassisElements.ChassisElementMaximum = 1;
```

Data Example: The *ChassisBaseBoard* field is invalid and the *ChassisElementStructure* field is valid. The *ChassisElementType.Type* field refers to a type 28 record number, which is 0x1C and a temperature probe. This instance is the third instance of a type 28 record. There must be a minimum of one type 28 record and a maximum of five.

```
ChassisElements.ChassisElementType.RecordType = 1;
ChassisElements.ChassisElementType.Type = 28;
ChassisElements.ChassisElementStructure = 3;
ChassisElements.ChassisElementMinimum = 1;
ChassisElements.ChassisElementMaximum = 5;
```

Port Connector Information (Type 8) Record Numbers

Port Internal Connector Reference Designator

Data Example: The internal connection is at J4C1.

```
PortInternalConnectorDesignator = String reference to "J4C1",0;
```

Port External Connector Reference Designator

Data Example: The external port connector's reference designator is "Parallel Port."

```
PortExternalConnectorReferenceDesignator = String reference to
"Parallel Port",0;
```

Port Internal Connector Type

Data Example: The internal port is a 25-pin dual inline connector.

```
PortInternalConnectorType = EfiPortConnectorType25PinDualInline;
```

Port External Connector Type

Data Example: The internal port is a 25-pin DB male connector.

```
PortExternalConnectorType = EfiPortConnectorTypeDB25Male;
```

Port Type

Data Example: The port type is ECP/EPP parallel port.

```
PortType = EfiPortTypeParallelPortEcpEpp;
```

System Slots (Type 9) Record Numbers

Slot Designation

Data Example: The slot designation is "PCI 3."

```
SlotDesignation = String reference to "PCI 3",0;
```

Slot Type

Data Example: The slot type is PCI.

```
SlotType = EfiSlotTypePci;
```

Slot Data Bus Width

Data Example: The slot width is 32 bits.

```
SlotDataBusWidth = EfiSlotDataBusWidth32Bit;
```

Slot Current Usage

Data Example: The slot is in use.

```
SlotUsage = EfiSlotUsageInUse;
```

Slot Length

Data Example: The slot length is short.

```
SlotLength = EfiSlotLengthShort;
```

Slot ID

Data Example: The ID of the PCI slot is 31 decimal.

```
SlotId = 0x1F;
```

Slot Characteristics

Data Example: The slot provides both 5.0 and 3.3 volts and PME#.

```
SlotCharacteristics = 0000 0000 0000 0000   0000 0001 0000 0110b;
```

Onboard Devices Information (Type 10) Record Numbers

Onboard Device Description

Data Example: Video is onboard and VGA.

```
OnBoardDeviceDescription = String reference to "VGA Video",0;
```

Onboard Device Type

Data Example: Video is onboard and enabled.

```
OnBoardDeviceType = EfiOnBoardDeviceTypeVideo | 0x800;
```

OEM Strings (Type 11) Record Numbers

Number of OEM Strings

Data Example: There are five OEM strings.

```
NumberOfInstallableLanguages = 0x05;
```

OEM Strings

Data Example: "This string is OEM string #1." Note that the Number of OEM Strings example implies that there are five OEM strings records, each with a unique *OemStringId* between 1 and 5 inclusively.

```
OemStringId = 0x01;
OemStringRef = String reference to "This string is OEM string
#1",0;
```

System Configuration Options (Type 12) Record Numbers

Number of System Configuration Option Strings

Data Example: There are three option strings.

```
NumberOfInstallableLanguages = 0x03;
```

System Configuration Option Strings

Data Example: "This string is option string #2." Note that the Number of System Configuration Option Strings example implies that there are three System Configuration Option Strings records, each with a unique *SystemOptionStringId* between 1 and 3 inclusively.

```
SystemOptionStringId = 0x02;
SystemOptionStringRef = String reference to "This string is
option string #2",0;
```

BIOS Language Information (Type 13) Record Numbers

Number of Installable Languages

Data Example: Five languages are supported. The order is French, French Canadian, English, German, and Spanish.

```
NumberOfInstallableLanguages = 0x05;
```

Language Flags

Data Example: The BIOS uses an abbreviated language format.

```
LanguageFlags = 0000 0000 0000 0000  0000 0000 0000 0001b;
```

Current Language Number

Data Example: The current language is number 2 (1 based), which is French Canadian. This value must be smaller than **EFI_MISC_NUMBER_OF_INSTALLABLE_LANGUAGES_DATA**.

```
CurrentLanguageNumber = 2;
```

Language String

Data Example: "This string is Language string #2." Note that the Number of Installable Languages example implies that there are 5 language strings records, each with a unique *LanguageId* between 1 and 5 inclusively. The Current Language Number example indicates that the current language has a *LanguageId* of 2, which is French Canadian in this example. Both formats are shown.

```
LanguageId = 0x02;
SystemLanguageString =
String reference to "fr|CA|iso8859-1",0;  /* Long format */
SystemLanguageString =
String reference to "frCA",0;  /* Abbreviated format */
```

Group Associations (Type 14) Record Numbers

Group Name

Data Example: The group name is "Chassis Cooling System."

```
Instance = 0x14;
GroupName =  String reference to "Chassis Cooling System",0;
```

Number of Items in Group

Data Example: There are two items in the group.

```
Instance = 0x14;
NumberGroupItems = 0x02;
```

Group Item Set

Data Example: Both processor number 1 (*GroupElementId* 0x01) and onboard SCSI (*GroupElementId* 0x02) are cooled by same fan (*GoupElementId* 0x03) and grouped together by group device (*GroupID* 0x14). There are three group item sets, one per device.

The processor is the one identified by an *Instance* 0x37.

Onboard SCSI is the one identified by an *Instance* 0x28E.

The fan is the one identified by an *Instance* 0x27, *SubInstance* 0x03.

```
DeviceOne.GroupId = 0x14;
DeviceOne.GroupElementID = 0x01;
DeviceOne.GroupLink.SubClassNumber = EFI_PROCESSOR_SUBCLASS;
DeviceOne.GroupLink.ProducerName = GUID processor;
DeviceOne.GroupLink.GroupId = 0x37;
DeviceOne.GroupLink.GroupElementId = 0x01;

DeviceTwo.GroupId = 0x14;
DeviceTwo.GroupElementID = 0x02;
DeviceTwo.GroupLink.SubClassNumber = EFI_MISC_SUBCLASS;
DeviceTwo.GroupLink.ProducerName = GUID Block IO;
DeviceTwo.GroupLink.GroupId = 0x28E;
DeviceTwo.GroupLink.GroupElementId = 0x00;

DeviceThree.GroupId = 0x14;
DeviceThree.GroupElementID = 0x03;
DeviceThree.GroupLink.SubClassNumber = EFI_MISC_SUBCLASS;
DeviceThree.GroupLink.ProducerName= GUID fan;
DeviceThree.GroupLink.GoupId = 0x27;
DeviceThree.GroupLink.GroupElementId = 0x03;
```

Built-in Pointing Device (Type 21) Record Numbers

Pointing Device Type

Data Example: The pointing device is a mouse.

```
PointingDeviceType = EfiPointingDeviceTypeMouse;
```

Pointing Device Interface

Data Example: The device is on a PS/2* port.

```
PointingDeviceInterface = EfiPointingDeviceInterfacePs2;
```

Number of Pointing Device Buttons

Data Example: The mouse has two buttons.

```
NumberPointingDeviceButtons = 0x02;
```

Portable Battery (Type 22) Record Numbers

Portable Battery Location

Data Example: The battery is in the case.

```
BatteryLocation = String reference to "In the case",0;
```

Portable Battery Manufacturer

Data Example: The battery is made by Shocking Corp.

```
BatteryManufacturer = String reference to "Shocking Corp",0;
```

Portable Battery Manufacture Date

Data Example: Use the SBDS field instead.

```
BatteryManufactureDate = 0;
```

Portable Battery Serial Number

Data Example: Use the SBDS field instead.

```
BatterySerialNumber = 0;
```

Portable Battery Device Name

Data Example: The battery name is "The Bomb."

```
BatteryDeviceName = String reference to "The Bomb",0;
```

Portable Battery SBDS Version Number

Data Example: The battery supports SBDS version 1.0.

```
BatterySbdsVersionNumber = String reference to "1.0",0;
```

Portable Battery SBDS Device Chemistry

Data Example: The SBDS information is invalid because the *BatteryDeviceChemistry* field is known. See the Portable Battery Device Chemistry example.

```
BatterySbdsDeviceChemistry = 0;
```

Portable Battery Device Chemistry

Data Example: The battery is Lithium ion.

```
BatteryDeviceChemistry = EfiBatteryDeviceChemistryTypeLithiumIon;
```

Portable Battery Design Capacity

Data Example: The battery is good for 1 watt-hour.

```
BatteryDesignCapacity.Value = 0x01;
BatteryDesignCapacity.Exponent = 0x00;
```

Portable Battery Design Voltage

Data Example: The battery is 3.0 volts.

```
BatteryDesignVoltage.Value = 0x03;
BatteryDesignVoltage.Exponent = 0x00;
```

Portable Battery Maximum Error in Battery Data

Data Example: The battery data has a maximum error of 10 percent. This value is denoted by 10 decimal or 0x0A.

```
BatteryMaximumError = 0x0A;
```

Portable Battery SBDS Serial Number

Data Example: The SBDS serial number is 0x1234.

```
BatterySbdsSerialNumber = 0x1234;
```

Portable Battery SBDS Manufacture Date

Data Example: The battery was made on March 10, 2001. The day is 10 decimal or 01010b. The month is 3 decimal or 0011b. The year is 2001 minus 1980 or 21 decimal or 0010101b.

```
BatterySbdsManufacturingDate =
                        0000 0000 0000 0000  0010 1010 0110 1010b ;
```

Portable Battery OEM Specific

Data Example: There is no OEM-specific data.

```
BatteryOemSpecific = 0x0;
```

System Reset (Type 23) Record Numbers

Reset Capabilities

Data Example: Reset was enabled by user, the operating system handles resets on error conditions, and the watchdog timer is enabled.

```
ResetCapabilities = 0000 0000 0000 0000   0000 0000 0010 1011b;
```

Reset Count

Data Example: No automatic system resets occurred.

```
ResetCount = 0x0;
```

Reset Limit

Data Example: The operating system can try a maximum of three resets before a fatal error.

```
ResetLimit = 0x03;
```

Reset Timer Interval

Data Example: The watchdog timer is set for 5 minutes.

```
ResetTimerInterval = 0x05;
```

Reset Timeout

Data Example: Do not reboot for 3 minutes after any reset.

```
ResetTimeout = 0x03;
```

Hardware Security (Type 24) Record Numbers

Hardware Security Settings

Data Example: All passwords are enabled. Front panel reset is disabled.

```
HardwareSecuritySettings =
                    0000 0000 0000 0000    0000 0000 0101 0100b;
```

System Power Controls (Type 25) Record Numbers

System Power Controls (Type 25) Record Numbers

All structures and all examples are in Binary Coded Decimal (BCD). Out-of-range values are ignored.

Scheduled Power-on Month

Data Example: Ignore the month of a scheduled power-on.

```
ScheduledPoweronMonth = 0xFF;
```

Scheduled Power-on Day-of-Month

Data Example: Power on the fifteenth of each month.

```
ScheduledPoweronDayOfMonth = 0x15;
```

Scheduled Power-on Hour

Data Example: Power on at 3:45 AM.

```
ScheduledPoweronHour = 0x03;
```

Scheduled Power-on Minute

Data Example: Power on at 3:45 AM.

```
ScheduledPoweronMinute = 0x45;
```

Scheduled Power-on Second

Data Example: Power on at 3:45:00 AM.

```
ScheduledPoweronSecond = 0x00;
```

Voltage Probe (Type 26) Record Numbers

Voltage Probe Description

Data Example: The voltage probe is on the processor.

```
VoltageProbeDescription = String reference to "Processor voltage
probe",0;
```

Voltage Probe Location and Status

Data Example: The processor probe and voltage are okay.

```
VoltageProbeLocation = 0000 0000 0000 0000   0000 0000 0110
0011b;
```

Voltage Probe Maximum Value

Data Example: The probe can read up to 15.0 volts.

```
VoltageProbeMaximumValue.Value = 0x0f;
VoltageProbeMaximumValue.Exponent = 0x00;
```

Voltage Probe Minimum Value

Data Example: The probe can read up to -10.0 volts.

```
VoltageProbeMinimumValue.Value = 0x8001;
VoltageProbeMinimumValue.Exponent = 0x01;
```

Voltage Probe Resolution

Data Example: The probe resolves voltage down to 1 millivolt.

```
VoltageProbeResolution.Value = 0x01;
VoltageProbeResolution.Exponent = 0x8003;
```

Voltage Probe Tolerance

Data Example: The probe is accurate to 20 millivolts.

```
VoltageProbeTolerance.Value = 0x02;
VoltageProbeTolerance.Exponent = 0x8002;
```

Voltage Probe Accuracy

Data Example: Based upon the position of the probe, it is accurate to 1 percent.

```
VoltageProbeAccuracy.Value = 0x01;
VoltageProbeAccuracy.Exponent = 0x00;
```

Voltage Probe Nominal Value

Data Example: The nominal value is 5.0 volts.

```
VoltageProbeNominalValue.Value = 0x05;
VoltageProbeNominalValue.Exponent = 0x00;
```

Management Device Lower Noncritical Threshold (Type 26)

Data Example: The voltage probe's lower noncritical threshold is 0 volts.

```
MDLowerNoncriticalThreshold.Value = 0x00;
MDLowerNoncriticalThreshold.Exponent = 0x00;
```

Management Device Upper Noncritical Threshold (Type 26)

Data Example: The voltage probe's upper noncritical threshold is 5 volts.

```
MDUpperNoncriticalThreshold.Value = 0x05;
MDUpperNoncriticalThreshold.Exponent = 0x00;
```

Management Device Lower Critical Threshold (Type 26)

Data Example: The voltage probe's lower critical threshold is -5.0 volts.

```
MDLowerCriticalThreshold.Value = 0x8005;
MDLowerCriticalThreshold.Exponent = 0x00;
```

Management Device Upper Critical Threshold (Type 26)

Data Example: The voltage probe's upper critical threshold is 10 volts.

```
MDUpperCriticalThreshold.Value = 0x01;
MDUpperCriticalThreshold.Exponent = 0x01;
```

Management Device Lower Nonrecoverable Threshold (Type 26)

Data Example: The voltage probe's lower nonrecoverable threshold is -10 volts.

```
MDLowerNonrecoverableThreshold.Value = 0x8001;
MDLowerNonrecoverableThreshold.Exponent = 0x0001;
```

Management Device Upper Nonrecoverable Threshold (Type 26)

Data Example: The voltage probe's upper nonrecoverable threshold is 15 volts.

```
MDUpperNonrecoverableThreshold.Value = 0x0f;
MDUpperNonrecoverableThreshold.Exponent = 0x00;
```

Voltage Probe OEM Defined

Data Example: No OEM bit fields are defined.

```
VoltageProbeOemDefined = 0x0;
```

Cooling Device (Type 27) Record Numbers

Cooling Device Type and Status

Data Example: The cooling device is a fan and the temperature is okay.

```
CoolingDeviceType = 0000 0000 0000 0000   0000 0000 0110 0011b;
```

Cooling Device Temperature Link

Data Example: This cooling device, which has an *Instance* of 0x13, is associated with a temperature probe structure set that has an *Instance* of 0x88.

```
ThisDevice.Instance = 0x13;
ThisDevice.CoolingDeviceTemperatureLink.ProducerName = GUID Temp
probe;
ThisDevice.CoolingDeviceTemperatureLink.Instance = 0x88;
ThisDevice.CoolingDeviceTemperatureLink.SubInstance = 0x01;
```

Cooling Device Unit Group

Data Example: The fan is the sole cooling device for the given area.

```
CoolingDeviceUnitGroup = 0x0;
```

Cooling Device Nominal Speed

Data Example: The fan's normal rotational speed is unknown.

```
CoolingDeviceNominalSpeed.Value = 0xFFFF;
CoolingDeviceNominalSpeed.Exponent = 0xFFFF;
```

Cooling Device OEM Defined

Data Example: There are no OEM-defined fields.

```
CoolingDeviceOemDefined = 0x0;
```

Temperature Probe (Type 28) Record Numbers

Temperature Probe Description

Data Example: The temperature probe is in the processor.

```
TemperatureProbeDescription = String reference to "Processor
Temperature Probe",0;
```

Temperature Probe Location and Status

Data Example: The probe is on the processor and temperature is noncritical.

```
TemperatureProbeLocation =
                    0000 0000 0000 0000   0000 0000 1000 0011b;
```

Temperature Probe Maximum Value

Data Example: The maximum probe value is 100 degrees Celsius.

```
TemperatureProbeMaximumValue.Value = 0x01;
TemperatureProbeMaximumValue.Exponent = 0x02;
```

Temperature Probe Minimum Value

Data Example: The minimum probe value is 0 degrees Celsius.

```
TemperatureProbeMinimumValue.Value = 0x00;
TemperatureProbeMinimumValue.Exponent = 0x00;
```

Temperature Probe Resolution

Data Example: The probe resolves to 1 degree Celsius.

```
TemperatureProbeResolution.Value = 0x01;
TemperatureProbeResolution.Exponent = 0x00;
```

Temperature Probe Tolerance

Data Example: The probe tolerance is 1 degree Celsius.

```
TemperatureProbeTolerance.Value = 0x01;
TemperatureProbeTolerance.Exponent = 0x00;
```

Temperature Probe Accuracy

Data Example: The probe only gives 5 percent accuracy.

```
TemeratureProbeAccuracy.Value = 0x05;
TemeratureProbeAccuracy.Exponent = 0x00;
```

Temperature Probe Nominal Value

Data Example: The nominal value is 70 degrees Celsius.

```
TemperatureProbeNominalValue.Value = 0x07;
TemperatureProbeNominalValue.Exponent = 0x01;
```

Management Device Lower Noncritical Threshold (Type 28)

Data Example: The device is a fan with a lower noncritical threshold of 15 degrees Celsius or 0x0F.

```
MDLowerNoncriticalThreshold.Value = 0x0F;
MDLowerNoncriticalThreshold.Exponent = 0x00;
```

Management Device Upper Noncritical Threshold (Type 28)

Data Example: The device is a fan with an upper noncritical threshold of 75 degrees Celsius or 0x4B.

```
MDUpperNoncriticalThreshold.Value = 0x4B;
MDUpperNoncriticalThreshold.Exponent = 0x00;
```

Management Device Lower Critical Threshold (Type 28)

Data Example: The device is a fan with a lower critical threshold of 10 degrees Celsius.

```
MDLowerCriticalThreshold.Value = 0x01;
MDLowerCriticalThreshold.Exponent = 0x01;
```

Management Device Upper Critical Threshold (Type 28)

Data Example: The device is a fan with an upper noncritical threshold of 80 degrees Celsius.

```
MDUpperCriticalThreshold.Value = 0x08;
MDUpperCriticalThreshold.Exponent = 0x01;
```

Management Device Lower Nonrecoverable Threshold (Type 28)

Data Example: The device is a fan with a lower nonrecoverable threshold of 5 degrees Celsius.

```
MDLowerNonrecoverableThreshold.Value = 0x05;
MDLowerNonrecoverableThreshold.Exponent = 0x00;
```

Management Device Upper Nonrecoverable Threshold (Type 28)

Data Example: The device is a fan with an upper nonrecoverable threshold of 85 degrees Celsius or 0x55.

```
MDUpperNonrecoverableThreshold.Value = 0x55;
MDUpperNonrecoverableThreshold.Exponent = 0x00;
```

Temperature Probe OEM Defined

Data Example: There are no OEM-defined values.

```
TemperatureProbeOemDefined = 0x0;
```

Electrical Current Probe (Type 29) Record Numbers

Electrical Current Probe Description

Data Example: The current probe is on the IDE disk.

```
ElectricalCurrentProbeDescription =  String reference to "IDE
Disk Current Probe",0;
```

Electrical Current Probe Location and Status

Data Example: The probe is for the IDE disk and its status is okay.

```
ElectricalCurrentProbeLocation =
                    0000 0000 0000 0000    0000 0000 0110 0100b;
```

Electrical Current Probe Maximum Value

Data Example: The maximum current is 3 amperes.

```
ElectricalCurrentProbeMaximumValue.Value = 0x03;
ElectricalCurrentProbeMaximumValue.Exponent = 0x01;
```

Electrical Current Probe Minimum Value

Data Example: The minimum current is 0 amperes.

```
ElectricalCurrentProbeMinimumValue.Value = 0x00;
ElectricalCurrentProbeMinimumValue.Exponent = 0x00;
```

Electrical Current Probe Resolution

Data Example: The probe's resolution is 50 milliamperes.

```
ElectricalCurrentProbeResolution.Value = 0x05;
ElectricalCurrentProbeResolution.Exponent = 0x8002;
```

Electrical Current Probe Tolerance

Data Example: The probe's tolerance is 10 milliamperes.

```
ElectricalCurrentProbeTolerance.Value = 0x01;
ElectricalCurrentProbeTolerance.Exponent = 0x8002;
```

Electrical Current Probe Accuracy

Data Example: The probe's accuracy is 10 percent.

```
ElectricalCurrentProbeAccuracy.Value = 0x01;
ElectricalCurrentProbeAccuracy.Exponent = 0x01;
```

Electrical Current Probe Nominal Value (Type 29)

Data Example: The nominal value is 100 milliamperes.

```
ElectricalCurrentProbeNominalValue.Value = 0x01;
ElectricalCurrentProbeNominalValue.Exponent = 0x80014;
```

Management Device Lower Noncritical Threshold (Type 29)

Data Example: The probe's lower noncritical threshold is 0.5 amperes.

```
MDLowerNoncriticalThreshold.Value = 0x05;
MDLowerNoncriticalThreshold.Exponent = 0x8001;
```

Management Device Upper Noncritical Threshold (Type 29)

Data Example: The probe's upper noncritical threshold is 2.5 amperes.

```
MDUpperNoncriticalThreshold.Value = 0x19;
MDUpperNoncriticalThreshold.Exponent = 0x8001;
```

Management Device Upper Critical Threshold (Type 29)

Data Example: The probe's upper critical threshold is 2.75 amperes. 275 decimal is 0x113.

```
MDUpperCriticalThreshold.Value = 0x0113;
MDUpperCriticalThreshold.Exponent = 0x8002;
```

Management Device Lower Critical Threshold (Type 29)

Data Example: The probe's lower critical threshold is 0.25 amperes.

```
MDLowerCriticalThreshold.Value = 0x19;
MDLowerCriticalThreshold.Exponent = 0x8002;
```

Management Device Lower Nonrecoverable Threshold (Type 29)

Data Example: The probe's lower nonrecoverable threshold is 0.1 amperes.

```
MDLowerNonrecoverableThreshold.Value = 0x01;
MDLowerNonrecoverableThreshold.Exponent = 0x8001;
```

Management Device Upper Nonrecoverable Threshold (Type 29)

Data Example: The probe's nonrecoverable threshold is 2.90 amperes. 29 decimal is 0x1d.

```
MDUpperNonrecoverableThreshold.Value = 0x1d;
MDUpperNonrecoverableThreshold.Exponent = 0x8001;
```

Electrical Current Probe OEM Defined

Data Example: No OEM fields are defined.

```
ElectricalCurrentProbeOemDefined = 0x0;
```

Out-of-Band Remote Access (Type 30) Record Numbers

Remote Access Manufacturer Name Description

Data Example: The manufacturer is Big Brother.

```
RemoteAccessManufacturerNameDescription = String reference to
"Big Brother",0;
```

Remote Access Connections

Data Example: Only outbound connections are enabled.

```
RemoteAccessConnecions =
                        0000 0000 0000 0000   0000 0000 0000 0010b;
```

Boot Integrity Services (BIS) Entry Point (Type 31) Record Numbers

BIS Entry Point Address

Data Example: The BIS entry point is 0xFFFE1234.

```
BisEntryPoint = 0xFFFE1234;
```

System Boot Information (Type 32) Record Numbers

Boot Information Status

Data Example: There is no bootable media.

```
BootInformationStatus = EfiBootInformationStatusNoBootableMedia;
```

Boot Information Data

Data Example: The OEM defined byte 0 with a value of 0x55.

```
BootInformationData[0] = 0x55;
BootInformationData[1] = 0x0;
BootInformationData[2] = 0x0;
BootInformationData[3] = 0x0;
BootInformationData[4] = 0x0;
BootInformationData[5] = 0x0;
BootInformationData[6] = 0x0;
BootInformationData[7] = 0x0;
BootInformationData[8] = 0x0;
```

Management Device (Type 34) Record Numbers

Management Device Description

Data Example: The device's description is "Keeping tab on you."

```
ManagementDeviceDescription = String reference to "Keeping tab on
you",0;
```

Management Device Type

Data Example: The management type is National Semiconductor* LM 75.

```
ManagementDeviceType = EfiManagementDeviceTypeLm75;
```

Management Device Address

Data Example: The device address is 3.

```
ManagementDeviceAddress = 0x03;
```

Management Device Address Type

Data Example: The device is a System Management Bus (SMBus) device.

```
ManagementDeviceAddressType =
EfiManagementDeviceAddressTypeSmbus;
```

Management Device Component (Type 35) Record Numbers

Management Device Component Description

Data Example: The management device component is "Freezer Fan."

```
ManagementDeviceComponentDescription = String reference to
"Freezer Fan",0;
```

Management Device Link

Data Example: The device *Instance* is 0x12E.

```
ThisDevice.ManagementDeviceLink.ProducerName = GUID Fan
management;
ThisDevice.ManagementDeviceLink.Instance = 0x12E;
ThisDevice.ManagementDeviceLink.SubInstance = 0x00;
```

Management Device Component Link

Data Example: The fan device component *Instance* is 0x12F and *SubInstance* is 0x02 (second fan).

```
ThisDevice.ManagementDeviceComponentLink.ProducerName = GUID fan;
ThisDevice.ManagementDeviceComponentLink.Instance = 0x12F;
ThisDevice.ManagementDeviceComponentLink.SubInstance = 0x02;
```

IPMI Device Information (Type 38) Record Numbers

IPMI Interface Type

Data Example: The interface type is keyboard controller style.

```
IpmiInterfaceType = EfiIpmiKcs;
```

IPMI Specification Revision

Data Example: The IPMI specification is revision 1.2.

```
IpmiSpecificationRevision = 0000 0000 0001 0010b;
```

IPMI I2C Slave Address

Data Example: The I2C slave address is 2.

```
IpmiI2CSlaveAddress = 0x2;
```

IPMI NV Storage Device Address

Data Example: There is no storage device.

```
IpmiNvDeviceAddress = 0xFFFF;
```

IPMI Base Address

Data Example: The base address is 0x68.

```
IpmiBaseAddress = 0x68;
```

System Power Supply (Type 39) Record Numbers

Power Supply Group

Data Example: The power supply is not a part of a group and is not redundant.

```
PowerSupplyGroup = 0x0;
```

Power Supply Location

Data Example: The power supply is in the chassis.

```
PowerSupplyLocation = String reference to "In the chassis",0;
```

Power Supply Device Name

Data Example: The power supply's device name is "Shocker."

```
PowerSupplyDeviceName = String reference to "Shocker",0;
```

Power Supply Manufacturer Name

Data Example: The manufacturer is "Zap."

```
PowerSupplyManufacturer = String reference to "Zap",0;
```

Power Supply Serial Number

Data Example: The power supply's serial number is 00001.

```
PowerSupplySerialNumber = String reference to "00001",0;
```

Power Supply Asset Tag

Data Example: The power supply's asset tag does not exist.

```
PowerSupplyAssetTag = 0;
```

Power Supply Model Part Number

Data Example: The power supply's model part number is 666.

```
PowerSupplyModelPartNumber = String reference to "666",0;
```

Power Supply Revision Level

Data Example: The power supply's revision number is 13.7.

```
PowerSupplyRevisionLevel = String reference to "13.7",0;
```

Power Supply Maximum Power Capacity

Data Example: The maximum power supply wattage is 3 watts.

```
PowerSupplyMaxPowerCapacity.Value = 0x03;
PowerSupplyMaxPowerCapacity.Exponent = 0x00;
```

Power Supply Characteristics

Data Example: The power supply is present, the input voltage range is manual switching, the power supply is okay, and the power supply type is switching.

```
PowerSupplyCharacteristics =
                          0000 0000 0000 0000   0001 0001 1001 1010;
```

Power Supply Input Voltage Probe Link

Data Example: The power supply's input voltage probe *Instance* is 0x3E7.

```
ThisDevice.PowerSupplyInputVoltageProbeLink.ProducerName = GUID
Voltage probe;
ThisDevice.PowerSupplyInputVoltageProbeLink.Instance = 0x3E7;
ThisDevice.PowerSupplyInputVoltageProbeLink.SubInstance = 0x00;
```

Power Supply Cooling Device Link

Data Example: The power supply's cooling device *Instance* is 0x489 and the *SubInstance* is 0x02 (second fan).

```
ThisDevice.PowerSupplyCoolingDeviceLink.ProducerName = GUID fan;
ThisDevice.PowerSupplyCoolingDeviceLink.Instance = 0x489;
ThisDevice.PowerSupplyCoolingDeviceLink.SubInstance = 0x02;
```

Power Supply Input Current Probe Link

Data Example: The power supply's input current probe *Instance* is 0x1033.

```
ThisDevice.PowerSupplyInputCurrentProbeLink.ProducerName = GUID
current probe;
ThisDevice.PowerSupplyInputCurrentProbeLink.Instance = 0x1033;
ThisDevice.PowerSupplyInputCurrentProbeLink.SubInstance = 0x00;
```

SMBIOS Structure Encapsulation (Types 0x80 to 0xFF) Record Numbers

SMBIOS Header

Data Example: Need to describe SMBIOS OEM-specific structure type 0x80 that conveys information about an OEM server management card. The structure prototype is defined below:

```
typedef struct {
  EFI_SMBIOS_STRUCTURE_HDR        Hdr;
  UINT16                          CardVersion;
  UINT8                           CardNameStrRef;
} EFI_SMBIOS_TYPE80_FORMATTED_DATA;
```

The formatted data is followed on one double **NULL**-terminated string that represents the name of the card.

Therefore, the data producer sets *Hdr.Type* is to **0x80**.

Hdr.Length is initialized to **sizeof (EFI_SMBIOS_TYPE80_FORMATTED_DATA)** by the producer.

Hdr.Handle is left uninitialized.

Raw Data

Data Example: The card version is 3. Therefore, the producer initializes *CardVersion* to 3. *CardNameStrRef* is 0x1 in accordance with the *SMBIOS Specification.*

The card is named "The Greatest." The formatted data is followed by the actual string "The Greatest," followed by two **NULL** bytes. The raw data can be represented by the following sequence of bytes:

```
0x3, 0x1, ' T', 'h', 'e', 'G', 'r', 'e', 'a', 't', 'e', 's', 't',
0x0, 0x0
```

The producer can choose to publish the string in a language that corresponds to the system language as described by the global variable *Lang*.

In any case, the raw data is opaque to a driver that consumes this record to construct an SMBIOS table.