Document Number: DSP0246

Date: 2009-12-11

Version: 1.0.1

Platform Level Data Model (PLDM) for SMBIOS Data Transfer Specification

Document Type: Specification

Document Status: DMTF Standard

Document Language: E

CONTENTS

Figures

Tables

Foreword

The *Platform Level Data Model (PLDM) for SMBIOS Data Transfer Specification* (DSP0246) was prepared by the Platform Management Components Intercommunications (PMCI) Working Group.

DMTF is a not-for-profit association of industry members dedicated to promoting enterprise and systems management and interoperability.

Introduction

This specification describes Platform Level Data Model (PLDM) data structures and commands for transferring SMBIOS data between the components of a platform management hardware subsystem. This document specifies PLDM representations of SMBIOS structure table and SMBIOS structures, and a set of commands for transferring SMBIOS structure table and SMBIOS structure data.

Platform Level Data Model (PLDM) for SMBIOS Data Transfer Specification

1 Scope

DSP0134, *System Management BIOS (SMBIOS) Reference Specification,* defines BIOS extensions that provide platform asset information such as BIOS version, processor speed/type, and memory capacity. The SMBIOS structure table typically resides in the system memory and contains one or more SMBIOS structures. This document describes Platform Level Data Model (PLDM) data structures and commands for transferring SMBIOS data between the components of a platform management hardware subsystem.

This document meets the following objectives:

- Specifies PLDM representations of SMBIOS structure table and SMBIOS structures
- Specifies a set of commands for transferring SMBIOS structure table and SMBIOS structure data

2 Normative References

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

DMTF DSP0134, *System Management BIOS (SMBIOS) Reference Specification 2.6*, http://www.dmtf.org/standards/published_documents/DSP0134_2.6.pdf

DMTF DSP0240, *Platform Level Data Model (PLDM) Base Specification 1.0*, http://www.dmtf.org/standards/published_documents/DSP0240_1.0.pdf

DMTF DSP0245, *Platform Level Data Model (PLDM) IDs and Codes 1.0*, http://www.dmtf.org/standards/published_documents/DSP0245_1.0.pdf

ISO/IEC Directives, Part 2, *Rules for the structure and drafting of International Standards,* http://isotc.iso.org/livelink/livelink.exe?func=ll&objId=4230456&objAction=browse&sort=subtype

OMG, *Unified Modeling Language (UML) from the Open Management Group (OMG),* http://www.uml.org/

3 Terms and Definitions

Refer to DSP0240 for terms and definitions that are used across the PLDM specifications. For the purposes of this document, the following additional terms and definitions apply.

3.1

System Management BIOS
SMBIOS

BIOS extensions that provide platform asset information such as BIOS version, processor speed/type, and memory capacity as specified in DSP0134

3.2

SMBIOS structure

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

3.3

SMBIOS structure table

the table that contains the SMBIOS structures

3.4

SMBIOS table

See 3.3.

4 Symbols and Abbreviated Terms

Refer to DSP0240 for symbols and abbreviated terms that are used across the PLDM specifications. For the purposes of this document, the following additional symbols and abbreviated terms apply.

4.1

BIOS

Basic Input Output System

4.2

SMBIOS

System Management BIOS

5 Conventions

Refer to DSP0240 for conventions, notations, and data types that are used in the PLDM specifications.

6 SMBIOS Overview

The *Platform Level Data Model (PLDM) for SMBIOS Data Transfer Specification* defines BIOS extensions that provide platform asset information such as BIOS version, processor speed/type, and memory capacity. The SMBIOS structure table resides in the system memory and contains one or more SMBIOS structures. Each SMBIOS structure begins with a (type, length, and handle) header. The SMBIOS structures are not ordered and searching for a specific structure requires parsing the SMBIOS structure table.

The SMBIOS structure table data is important for the instrumentation because it is used in the following ways:

- Platform asset information available in the SMBIOS structure table can be used to populate the instances of CIM classes that provide physical asset and hardware inventory information to the remote management console using the WBEM infrastructure.
- The information available in the SMBIOS structure table can be used for system health monitoring.
- The event log information, if available in the SMBIOS structure table, can be used to access the event log and perform system event monitoring.

7 PLDM for SMBIOS Data Transfer Overview

A Management Controller may wish to utilize the data in the SMBIOS structure table as a data source for providing platform inventory information via a CIM-based external interface. Depending on the implementation, additionally providing this information when the system is in low power states may require maintaining multiple copies of SMBIOS structure table data within a platform and keeping the copies consistent between the MC and the SMBIOS table information that is accessed by the system software and BIOS. There is thus a need for a platform-level data model (PLDM) for SMBIOS data transfer that can be used between the system firmware (BIOS) and a management controller, and between management controllers. Following are the design characteristics for the PLDM for SMBIOS data transfer:

- The PLDM defines commands to obtain the SMBIOS structure table metadata information, such as versioning information, checksum information, table length, number of SMBIOS structures, and maximum structure size.
- The PLDM preserves the SMBIOS structure format for the data transfer. By maintaining the SMBIOS structure format at the PLDM level, the need to parse the SMBIOS structure data for the PLDM data transfer is avoided.
- The SMBIOS structure table or SMBIOS structures can be large. The SMBIOS structure table data or SMBIOS structure data may not fit in a single PLDM message. The PLDM defines commands that allow the transfer of entire SMBIOS structure table or SMBIOS structures using either a single request/response or multiple requests/responses.
- The PLDM supports both pull and push models for the SMBIOS structure table data transfer and SMBIOS structure data transfer. In the push model, the SMBIOS structure table transfer is initiated by the sender without being explicitly requested by the receiving entity. In the pull model, the transfer of the SMBIOS structure table is requested by a receiving entity. The BIOS initiating the transfer of its SMBIOS structure table to a management controller is an example of the push model. A management controller sending read requests to BIOS telling it to provide SMBIOS structure table data is an example of the pull model.
- The PLDM defines a data integrity check to protect the SMBIOS structure table data transfer and SMBIOS structure data transfer.
- The PLDM defines commands to read SMBIOS structure data by type or by handle to enable reading a subset of structures from the SMBIOS structure table.
- The PLDM does not define commands to update or write a subset of structures from the SMBIOS structure table as it typically requires reading the entire table, followed by writing the subset of structures, and updating the SMBIOS table integrity checksum that covers the entire SMBIOS structure table.
- The PLDM does not define commands (read or write) to transfer partial SMBIOS structure or elements of an SMBIOS structure.

8 PLDM for SMBIOS Data Transfer

This section defines the data structures and commands for SMBIOS data transfer.

8.1 PLDM Representation of SMBIOS Structure Table

In the PLDM messages for SMBIOS data transfers, an SMBIOS structure representation is the same as described in the SMBIOS specification (DSP0134). Each SMBIOS structure has a formatted section and an optional unformed section as defined in DSP0134. The formatted section begins with a 4-byte header: Type (1 byte), Length (1 byte), and Handle (2 bytes). The unformed section is used to pass variable length structures (for example, text strings). Each SMBIOS structure is terminated by double null (0000h).

Table 1 shows the SMBIOS structure representation at the PLDM level.

Table 1 – PLDM Representation of an SMBIOS Structure

Byte	Type	Field
0	uint8	**Type** as defined in DSP0134
1	uint8	**Length** (L bytes) as defined in DSP0134
2:3	uint16	**Handle** as defined in DSP0134
4:L-1	–	The formatted area of the structure
Variable	–	Variable bytes of unformed area of the structure terminated by double null (0000h) as defined in DSP0134

The SMBIOS structure table data consists of multiple SMBIOS structures. When a set of one or more SMBIOS structures (up to the entire SMBIOS structure table data) is transferred using PLDM messages, the PLDM representation shown in Table 2 is used.

Table 2 – PLDM Representation of SMBIOSStructureData

Byte	Type	Field
Variable	–	**SMBIOS structures (one or more)** See Table 1 for the PLDM representation of an SMBIOS structure.

Variable	uint8[ ]	**Pad**
		0 to 3 number of pad bytes. The value stored in each pad byte is 0x00.
		The transmitter can compute the number of pad bytes from the SMBIOSStructureData by using the following algorithm:
		Let L be the total number of bytes in the SMBIOSStructureData excluding the pad and the integrity checksum.
		if (L modulo 4 == 0) then NumPadBytes = 0; else NumPadBytes = 4 – L modulo 4;
		The receiver can compute the number of pad bytes from the SMBIOSStructureData by using the following algorithm. In the algorithm, the receiver parses SMBIOS structure data until the remaining bytes are less than 8. When it reaches that stage, the remaining bytes contain the pad bytes and four bytes of data integrity checksum.
		Let L be the total number of bytes in the SMBIOSStructureData including the pad and the integrity checksum.
		RemBytes = L;
		i = 0;
		while (RemBytes >= 8)
		{
		Process the i^{th} SMBIOS structure in the SMBIOSStructureData;
		RemBytes = RemBytes - 4 – Total length of i^{th} SMBIOS structure including the formatted and unformed areas;
		i = i+1;
		}
		NumPadBytes = RemBytes modulo 4;

	uint32	**SMBIOSStructureDataIntegrityChecksum** Integrity checksum on the SMBIOS structure data including the pad bytes (if any). It is calculated starting at the first byte of the PLDM representation of SMBIOSStructureData. For this specification, the CRC-32 algorithm with the polynomial $x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^{8} + x^{7} + x^{5} + x^{4} + x^{2} + x + 1$ (same as the one used by IEEE 802.3) shall be used for the integrity checksum computation. The CRC computation involves processing a byte at a time with the least significant bit first.

8.2 PLDM Commands for SMBIOS Data Transfer

Table 3 defines the PLDM command codes defined in this section for the PLDM for SMBIOS data transfer.

Table 3 – PLDM for SMBIOS Data Transfer Command Codes

Command	Code Value	Requirement	Section
GetSMBIOSStructureTableMetadata	0x01	Mandatory	See 8.2.1.
SetSMBIOSStructureTableMetadata	0x02	Conditional[2]	See 8.2.2.
GetSMBIOSStructureTable	0x03	Conditional[1]	See 8.2.3.
SetSMBIOSStructureTable	0x04	Conditional[1]	See 8.2.4.
GetSMBIOSStructureByType	0x05	Optional	See 8.2.5.
GetSMBIOSStructureByHandle	0x06	Optional	See 8.2.6.

[1] At least one of these two commands must be supported by the requester and the responder for a compliant PLDM for SMBIOS data transfer implementation.

[2] If an implementation is transferring SMBIOS structure table data using the SetSMBIOSStructureTable command, it shall support the SetSMBIOSStructureTableMetadata command.

The requirements specified in Table 3 are relative to the services provided by the PLDM terminus.

The following sections define the PLDM commands for SMBIOS data transfer.

8.2.1 GetSMBIOSStructureTableMetadata

The GetSMBIOSStructureTableMetadata command, described in Table 4, is used to get the SMBIOS structure table metadata information that includes the SMBIOS major version, the SMBIOS minor version, the size of the largest SMBIOS structure, total length of the SMBIOS structure table, total number of SMBIOS structures, and the integrity checksum on the SMBIOS structure table data.

Table 4 – GetSMBIOSStructureTableMetadata Command Format

Byte	Type	Request Data
–	–	No Request Data
Byte	**Type**	**Response Data**

0	enum8	**CompletionCode**
		Possible values:
		{
		PLDM_BASE_CODES,
		NO_SMBIOS_STRUCTURE_TABLE_METADATA=0x83
		}

1	uint8	**SMBIOSMajorVersion** The major version of the SMBIOS specification with which the SMBIOS structure table complies
2	uint8	**SMBIOSMinorVersion** The minor version of the SMBIOS specification with which the SMBIOS structure table complies

3:4	uint16	**MaximumStructureSize** Size of the largest SMBIOS structure, in bytes, including the structure's formatted area and unformed area
5:6	uint16	**SMBIOSStructureTableLength** Total length of the SMBIOS structure table, in bytes
7:8	uint16	**NumberOfSMBIOSStructures** Total number of SMBIOS structures present in the SMBIOS structure table
9:12	uint32	**SMBIOSStructureTableIntegrityChecksum (CRC-32)** Integrity checksum on the SMBIOS structure table data as shown in Table 2 excluding pad bytes. See Table 2 for more information about this integrity checksum.

216 8.2.2 SetSMBIOSStructureTableMetadata

217 The SetSMBIOSStructureTableMetadata command, described in Table 5, is used to set the SMBIOS
218 structure table metadata information that includes the SMBIOS major version, the SMBIOS minor version,
219 the size of the largest SMBIOS structure, total length of the SMBIOS structure table, total number of
220 SMBIOS structures, and the integrity checksum on the SMBIOS structure table data.

221 **Table 5 – SetSMBIOSStructureTableMetadata Command Format**

Byte	Type	Request Data
0	uint8	**SMBIOSMajorVersion** The major version of the SMBIOS specification with which the SMBIOS structure table complies
1	uint8	**SMBIOSMinorVersion** The minor version of the SMBIOS specification with which the SMBIOS structure table complies
2:3	uint16	**MaximumStructureSize** Size of the largest SMBIOS structure, in bytes, including the structure's formatted area and unformed area
4:5	uint16	**SMBIOSStructureTableLength** Total length of the SMBIOS structure table, in bytes
6:7	uint16	**NumberOfSMBIOSStructures** Total number of SMBIOS structures present in the SMBIOS structure table
8:11	uint32	**SMBIOSStructureTableIntegrityChecksum (CRC-32)** Integrity checksum on the SMBIOS structure table data as shown in Table 2 excluding pad bytes. See Table 2 for more information about this integrity checksum.
Byte	**Type**	**Response Data**
0	enum8	**CompletionCode** Possible value: { PLDM_BASE_CODES}

8.2.3 GetSMBIOSStructureTable

The GetSMBIOSStructureTable command, described in Table 6, is used to get the SMBIOS structure table data. This command is defined to allow the SMBIOS structure table data to be transferred using a sequence of one or more command/response messages. When more than one command is used to transfer the SMBIOS structure table data, the response messages contain the non-overlapping contiguous portions of SMBIOS structure table data as defined in Table 2. By combining the portions of SMBIOS structure table data from the response messages, the entire SMBIOS structure table data can be reconstructed.

Table 6 – GetSMBIOSStructureTable Command Format

Byte	Type	Request Data
0:3	uint32	**DataTransferHandle** A handle that is used to identify an SMBIOS structure table data transfer. This handle is ignored by the responder when the TransferOperationFlag is set to GetFirstPart.
4	enum8	**TransferOperationFlag** The operation flag that indicates whether this is the start of the transfer Possible values: {GetNextPart=0x00, GetFirstPart=0x01}
Byte	**Type**	**Response Data**

0	enum8	**CompletionCode**
		Possible values:
		{
		PLDM_BASE_CODES,
		INVALID_DATA_TRANSFER_HANDLE=0x80,
		INVALID_TRANSFER_OPERATION_FLAG=0x81,
		SMBIOS_STRUCTURE_TABLE_UNAVAILABLE=0x85
		}

1:4	uint32	**NextDataTransferHandle** A handle that is used to identify the next portion of the transfer
5	enum8	**TransferFlag** The transfer flag that indicates what part of the transfer this response represents Possible values: {Start=0x01, Middle=0x02, End=0x04, StartAndEnd = 0x05}
Variable	–	**Portion of SMBIOSStructureData** See Table 2 for the format.

8.2.4 SetSMBIOSStructureTable

The SetSMBIOSStructureTable command, described in Table 7, is used to push the SMBIOS structure table data. This command is defined to allow the SMBIOS structure table data to be transferred using a sequence of one or more command/response messages. When more than one command is used to transfer the SMBIOS structure table data, the request messages contain the non-overlapping contiguous portions of SMBIOS structure table data as defined in Table 2. By combining the portions of SMBIOS structure table data from the request messages, the entire SMBIOS structure table data can be reconstructed.

Table 7 – SetSMBIOSStructureTable Command Format

Byte	Type	Request Data
0:3	uint32	**DataTransferHandle** A handle that is used to identify SMBIOS structure table transfer. This handle is ignored by the responder when the TransferFlag is set to Start or StartAndEnd.
4	enum8	**TransferFlag** The transfer flag that indicates what part of the transfer this request represents Possible values: {Start=0x01, Middle=0x02, End=0x04, StartAndEnd = 0x05}
Variable	–	**Portion of SMBIOSStructureData** See Table 2 for the format.
Byte	**Type**	**Response Data**

0	enum8	**CompletionCode**
		Possible values:
		{
		PLDM_BASE_CODES,
		INVALID_DATA_TRANSFER_HANDLE=0x80,
		INVALID_TRANSFER_FLAG=0x82,
		INVALID_DATA_INTEGRITY_CHECK=0x84
		}

1:4	uint32	**NextDataTransferHandle** A handle that is used to identify the next portion of the transfer

8.2.5 GetSMBIOSStructureByType

The GetSMBIOSStructureByType command, described in Table 8, is used to get the SMBIOS structures of a specific type. This command is defined to allow the SMBIOS structure data to be transferred using a sequence of one or more command/response messages. When more than one command is used to transfer the SMBIOS structure data, the response messages contain the non-overlapping contiguous portions of SMBIOS structure data as defined in Table 2. By combining the portions of SMBIOS structure data from the response messages, the entire SMBIOS structure data can be reconstructed.

Table 8 – GetSMBIOSStructureByType Command Format

Byte	Type	Request Data
0:3	uint32	**DataTransferHandle** A handle that is used to identify SMBIOS structure data transfer. This handle is ignored by the responder when the TransferOperationFlag is set to GetFirstPart.
4	enum8	**TransferOperationFlag** The operation flag that indicates whether this is the start of the transfer Possible values: {GetNextPart=0x00, GetFirstPart=0x01}
5	uint8	**Type** Specifies the type of the SMBIOS structures

6:7	uint16	**StructureInstanceID** A handle that is used to identify an instance of an SMBIOS structure of the specified type Special values: 0xFFFF – All instances of the specified type
Byte	**Type**	**Response Data**

0	enum8	**CompletionCode**
		Possible values:
		{
		PLDM_BASE_CODES,
		INVALID_DATA_TRANSFER_HANDLE=0x80,
		INVALID_TRANSFER_OPERATION_FLAG=0x81,
		NO_SMBIOS_STRUCTURES=0x86,
		INVALID_SMBIOS_STRUCTURE_TYPE=0x87,
		INVALID_SMBIOS_STRUCTURE_INSTANCE_ID=0x89
		}

1:4	uint32	**NextDataTransferHandle** A handle that is used to identify the next portion of the transfer
5	enum8	**TransferFlag** The transfer flag that indicates what part of the transfer this response represents Possible values: {Start=0x01, Middle=0x02, End=0x04, StartAndEnd = 0x05}
Variable	–	**SMBIOSStructureData** See Table 2 for the format.

8.2.6 GetSMBIOSStructureByHandle

The GetSMBIOSStructureByHandle command, described in Table 9, is used to get the SMBIOS structure by a specific handle. This command is defined to allow the SMBIOS structure data to be transferred by using a sequence of one or more command/response messages. When more than one command is used to transfer the SMBIOS structure data, the response messages contain the non-overlapping contiguous portions of SMBIOS structure data as defined in Table 2. By combining the portions of SMBIOS structure data from the response messages, the entire SMBIOS structure data can be constructed.

Table 9 – GetSMBIOSStructureByHandle Command Format

Byte	Type	Request Data
0:3	uint32	**DataTransferHandle** A handle that is used to identify SMBIOS structure data transfer. This handle is ignored by the responder when the TransferOperationFlag is set to GetFirstPart.
4	enum8	**TransferOperationFlag** The operation flag that indicates whether this is the start of the transfer Possible values: {GetNextPart=0x00, GetFirstPart=0x01}
5:6	uint16	**Handle** Specifies the handle of the SMBIOS structure

Byte	Type	Response Data

0	enum8	**CompletionCode**
		Possible values:
		{
		PLDM_BASE_CODES,
		INVALID_DATA_TRANSFER_HANDLE=0x80,
		INVALID_TRANSFER_OPERATION_FLAG=0x81,
		INVALID_SMBIOS_STRUCTURE_HANDLE=0x88
		}

1:4	uint32	**NextDataTransferHandle** A handle that is used to identify the next portion of the transfer
5	enum8	**TransferFlag** The transfer flag that indicates what part of the transfer this response represents Possible values: {Start=0x01, Middle=0x02, End=0x04, StartAndEnd = 0x05}
Variable	–	**SMBIOSStructureData** See Table 2 for the format.

8.3 PLDM for SMBIOS Data Transfer Version

The version of this PLDM for SMBIOS data transfer specification shall be 1.0.1 (major version number 1, minor version number 0, update version number 1, and no alpha version).

For the GetPLDMVersion command described in DSP0240, the version of this specification is reported using the encoding as: `0xF1F0F100`.

9 PLDM for SMBIOS Data Transfer Examples

This section provides examples of PLDM communications using the PLDM commands defined in this specification.

9.1 Multipart Transfers

The commands defined in Section 8 for transferring SMBIOS structure table data or SMBIOS structure data support multipart transfers. The Get* and Set* commands use flags and data transfer handles to perform multipart transfers. A data transfer handle uniquely identifies the next part of the transfer. The data transfer handle values are implementation specific. For example, an implementation can use memory addresses or sequence numbers as data transfer handles. Following are some requirements for using TransferOperationFlag, TransferFlag, and DataTransferHandle for a given data transfer:

- For initiating a data transfer (or getting the first part of data) using a Get* command, the TransferOperationFlag shall be set to GetFirstPart in the request of the Get* command.
- For transferring a part other than the first part of data by using a Get* command, the TransferOperationFlag shall be set to GetNextPart and the DataTransferHandle shall be set to the NextDataTransferHandle that was obtained in the response of the previous Get* command for this data transfer.
- The TransferFlag specified in the request of a Set* command or the response of a Get* command has the following meanings:
  - Start, which is the first part of the data transfer

  - Middle, which is neither the first nor the last part of the data transfer
  - End, which is the last part of the data transfer
  - StartAndEnd, which is the first and the last part of the data transfer

- The requester shall consider a data transfer complete when the TransferFlag in the response of a Get* command is set to End or StartAndEnd.
- The responder shall consider a data transfer complete when the TransferFlag in the request of a Set* command is set to End or StartAndEnd.

The following two examples show how the multipart transfers can be performed using the generic mechanism defined in the commands.

EXAMPLE 1: In this example, the MC maintains a copy of the SMBIOS structure table provided by the BIOS. The BIOS pushes a copy of its SMBIOS structure table to the MC by using the SetSMBIOSStructureTable command. Figure 1 shows the flow of the data transfer.

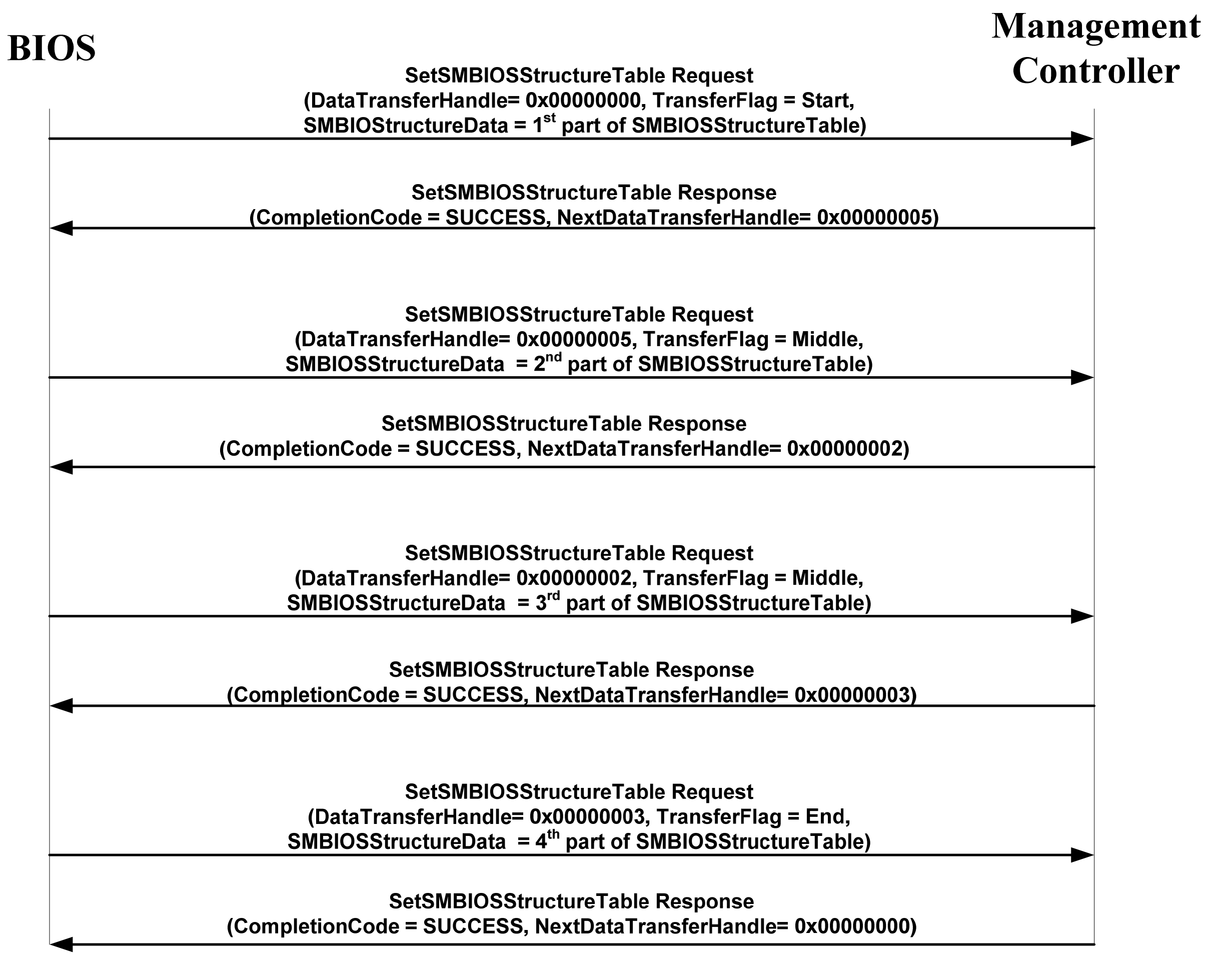

BIOS
Management Controller
SetSMBIOSStructureTable Request (DataTransferHandle= 0x00000000, TransferFlag = Start, SMBIOStructureData = 1st part of SMBIOSStructureTable)
SetSMBIOSStructureTable Response (CompletionCode = SUCCESS, NextDataTransferHandle= 0x00000005)
SetSMBIOSStructureTable Request (DataTransferHandle= 0x00000005, TransferFlag = Middle, SMBIOSStructureData = 2nd part of SMBIOSStructureTable)
SetSMBIOSStructureTable Response (CompletionCode = SUCCESS, NextDataTransferHandle= 0x00000002)
SetSMBIOSStructureTable Request (DataTransferHandle= 0x00000002, TransferFlag = Middle, SMBIOSStructureData = 3rd part of SMBIOSStructureTable)
SetSMBIOSStructureTable Response (CompletionCode = SUCCESS, NextDataTransferHandle= 0x00000003)
SetSMBIOSStructureTable Request (DataTransferHandle= 0x00000003, TransferFlag = End, SMBIOSStructureData = 4th part of SMBIOSStructureTable)
SetSMBIOSStructureTable Response (CompletionCode = SUCCESS, NextDataTransferHandle= 0x00000000)

Figure 1 – Multipart SMBIOS Structure Table Transfer Using the SetSMBIOSStructureTable Command

EXAMPLE 2: In this example, the MC reads the SMBIOS structure table from the BIOS by using the GetSMBIOSStructureTable command. This example shows a pull model where the MC obtains a copy of the SMBIOS structure table from the BIOS. Figure 2 shows the flow of the data transfer.

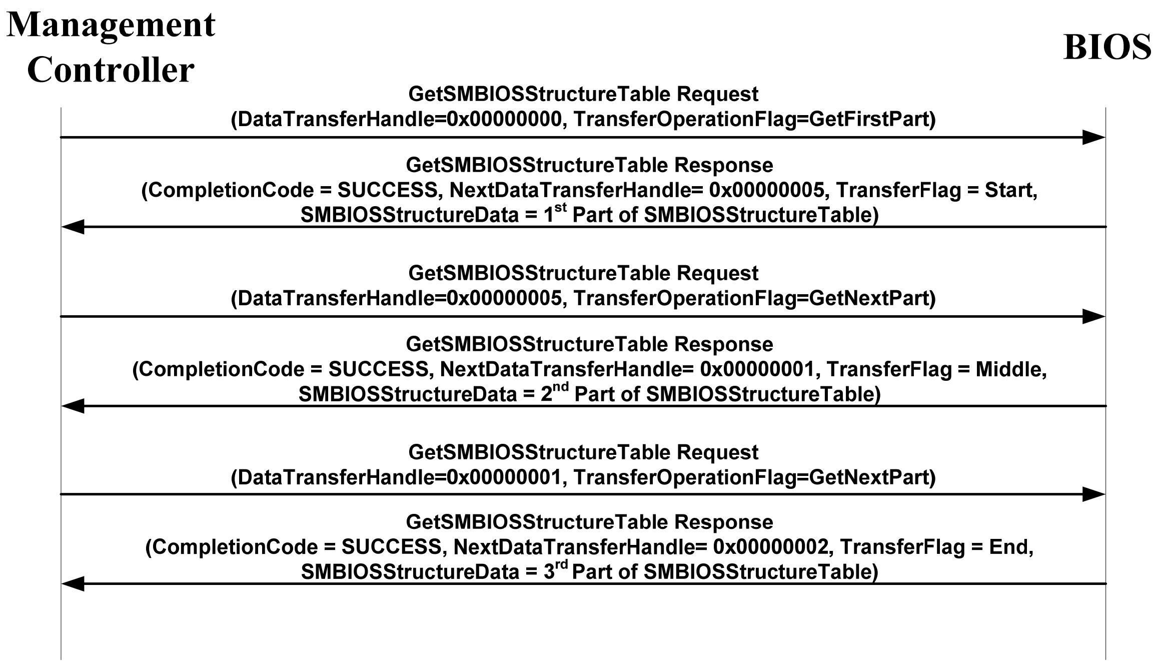
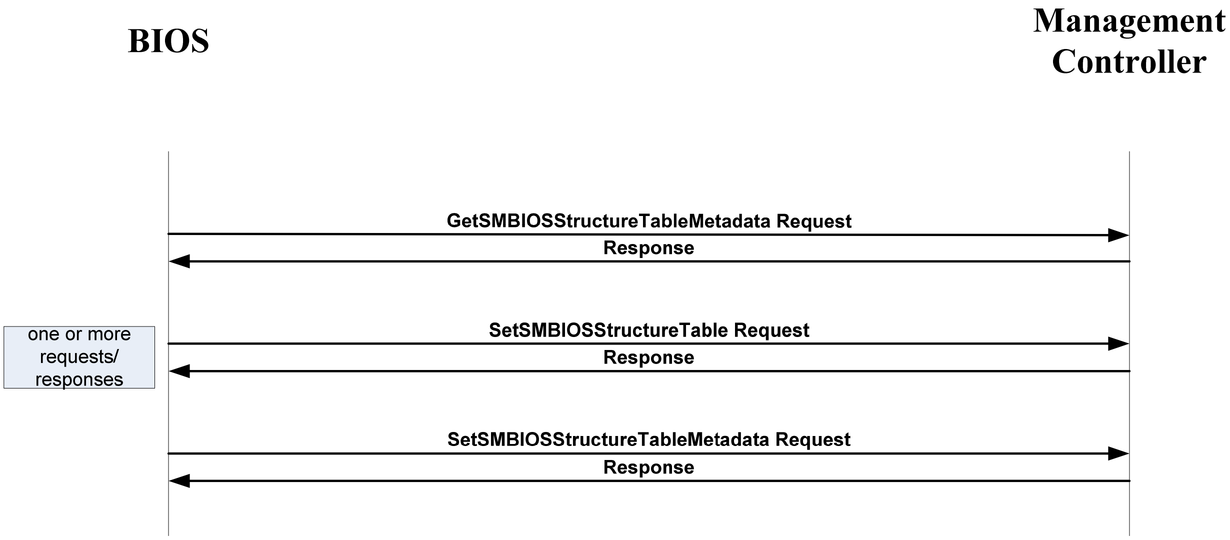

Management Controller
BIOS
GetSMBIOSStructureTable Request
(DataTransferHandle=0x00000000, TransferOperationFlag=GetFirstPart)
GetSMBIOSStructureTable Response
(CompletionCode = SUCCESS, NextDataTransferHandle= 0x00000005, TransferFlag = Start,
SMBIOSStructureData = 1st Part of SMBIOSStructureTable)
GetSMBIOSStructureTable Request
(DataTransferHandle=0x00000005, TransferOperationFlag=GetNextPart)
GetSMBIOSStructureTable Response
(CompletionCode = SUCCESS, NextDataTransferHandle= 0x00000001, TransferFlag = Middle,
SMBIOSStructureData = 2nd Part of SMBIOSStructureTable)
GetSMBIOSStructureTable Request
(DataTransferHandle=0x00000001, TransferOperationFlag=GetNextPart)
GetSMBIOSStructureTable Response
(CompletionCode = SUCCESS, NextDataTransferHandle= 0x00000002, TransferFlag = End,
SMBIOSStructureData = 3rd Part of SMBIOSStructureTable)

Figure 2 – Multipart SMBIOS Structure Table Transfer Using the GetSMBIOSStructureTable Command

9.2 SMBIOS Table Transfer from BIOS to MC Example

EXAMPLE: In this example, the BIOS sets the SMBIOS table on the MC. The BIOS first queries the SMBIOS table metadata by using the GetSMBIOSStructureTableMetadata command. The response from the MC to this command indicates that the MC does not have the latest SMBIOS structure table. Upon finding that the MC does not have the latest SMBIOS structure table, the BIOS transfers the SMBIOS structure table to the MC by using the SetSMBIOSStructureTable command. After transferring the latest SMBIOS structure table, the BIOS sets up the SMBIOS structure table metadata on the MC by using the SetSMBIOSStructureTableMetadata command. This example can be used in a push model where the MC is maintaining a copy of the SMBIOS structure table provided by the BIOS and the BIOS pushes to the MC a copy of the SMBIOS structure table by using SetSMBIOSStructureTable command. Figure 3 shows the data transfer.


BIOS
Management Controller
GetSMBIOSStructureTableMetadata Request
Response
one or more requests/ responses
SetSMBIOSStructureTable Request
Response
SetSMBIOSStructureTableMetadata Request
Response

311

312 **Figure 3 – Example of SMBIOS Table Transfer Using the SetSMBIOSStructureTable Command**

313

314

ANNEX A
(Informative)

Change Log

Version	Date	Description
1.0.0	2009/4/23	DMTF Standard Release
1.0.1	2009/12/11	Erratum version to clarify that the integrity checksum in the metadata does not include pad bytes.