Document Number: DSP0257

Date: 2022-01-01

Version: 1.0.1

Platform Level Data Model (PLDM) for FRU Data Specification

Supersedes: 1.0.0

Document Type: Specification

Document Status: DMTF Standard

Document Language: en-US

CONTENTS

Tables

Figures

Foreword

The *Platform Level Data Model (PLDM) for FRU Data Specification* (DSP0257) was prepared by the Platform Management Components Intercommunications (PMCI) Working Group of the DMTF.

DMTF is a not-for-profit association of industry members dedicated to promoting enterprise and systems management and interoperability. For information about the DMTF, see http://www.dmtf.org.

Acknowledgments

The DMTF acknowledges the following individuals for their contributions to this document:

Editors

- Phil Chidester – Dell Technologies
- Eliel Louzoun – Intel Corp.

Contributors

- Alan Berenbaum – SMSC
- Bob Stevens – Dell Technologies
- Hoan Do – Broadcom Inc.
- Ed Klodnicki – IBM
- John Leung – Intel Corporation
- Hemal Shah – Broadcom Inc.
- Tom Slaight – Intel Corporation

Introduction

The *Platform Level Data Model (PLDM) FRU Data Specification* defines messages, data structures, and data types used for FRU (Field Replaceable Unit) Data access and representation. FRU Data typically includes the serial number, part number and manufacturer for a field replaceable unit.

Document conventions

Typographical conventions

The following typographical conventions are used in this document:

- Document titles are marked in *italics*.
- Important terms that are used for the first time are marked in *italics*.
- Terms include a link to the term definition in the "**Error! Reference source not found.**" clause, enabling easy navigation to the term definition.
- ABNF rules are in `monospaced font`.

ABNF usage conventions

Format definitions in this document are specified using ABNF (see RFC5234), with the following deviations:

- Literal strings are to be interpreted as case-sensitive Unicode characters, as opposed to the definition in RFC5234 that interprets literal strings as case-insensitive US-ASCII characters.

Reserved and unassigned values

Unless otherwise specified, any reserved, unspecified, or unassigned values in enumerations or other numeric ranges are reserved for future definition by the DMTF.

Unless otherwise specified, numeric or bit fields that are designated as reserved shall be written as 0 (zero) and ignored when read.

Other Conventions

See **Error! Reference source not found.** for other conventions

Platform Level Data Model (PLDM) for FRU Data Specification

1 Scope

DSP0257, *Platform Level Data Model for FRU Data Specification,* defines a FRU data format that provides platform asset information including part number, serial number and manufacturer. The FRU Record Table typically resides in a non-volatile memory accessible by the management controller and contains one or more FRU records. This document describes Platform Level Data Model (PLDM) data structures and commands for transferring FRU data between the components of a platform management subsystem.

This document meets the following objectives:

- Specifies PLDM representations of FRU Record Table and FRU record data forma
- Specifies a set of commands for transferring FRU record data information

2 Normative references

The following referenced documents are indispensable for the application of this document. For dated or versioned references, only the edition cited (including any corrigenda or DMTF update versions) applies. For references without a date or version, the latest published edition of the referenced document (including any corrigenda or DMTF update versions) applies.

ANSI/IEEE Standard 754-1985, *Standard for Binary Floating Point Arithmetic*

DMTF DSP0240, *Platform Level Data Model (PLDM) Base Specification 1.*2
http://www.dmtf.org/sites/default/files/standards/documents/DSP0240_1.2.pdf

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

DMTF DSP0248, *Platform Level Data Model (PLDM) for Platform Monitoring and Control Specification 1.0*, http://www.dmtf.org/sites/default/files/standards/documents/DSP0248_1.2.pdf

IETF RFC2781, *UTF-16, an encoding of ISO 10646*, February 2000, http://www.ietf.org/rfc/rfc2781.txt

IETF RFC3629, *UTF-8, a transformation format of ISO 10646*, November 2003,
http://www.ietf.org/rfc/rfc3629.txt

IETF RFC4122, *A Universally Unique Identifier (UUID) URN Namespace*, July 2005,
http://www.ietf.org/rfc/rfc4122.txt

IETF RFC4646, *Tags for Identifying Languages*, September 2006, http://www.ietf.org/rfc/rfc4646.txt

ISO 8859-1, *Final Text of DIS 8859-1, 8-bit single-byte coded graphic character sets -- Part 1: Latin alphabet No.1,* February 1998

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

3 Terms and definitions

In this document, some terms have a specific meaning beyond the normal English meaning. Those terms are defined in this clause.

The terms "shall" ("required"), "shall not," "should" ("recommended"), "should not" ("not recommended"), "may," "need not" ("not required"), "can" and "cannot" in this document are to be interpreted as described in ISO/IEC Directives, Part 2, Annex H. The terms in parenthesis are alternatives for the preceding term, for use in exceptional cases when the preceding term cannot be used for linguistic reasons. Note that ISO/IEC Directives, Part 2, Annex H specifies additional alternatives. Occurrences of such additional alternatives shall be interpreted in their normal English meaning.

The terms "clause," "subclause," "paragraph," and "annex" in this document are to be interpreted as described in ISO/IEC Directives, Part 2, Clause 5.

The terms "normative" and "informative" in this document are to be interpreted as described in ISO/IEC Directives, Part 2, Clause 3. In this document, clauses, subclauses, or annexes labeled "(informative)" do not contain normative content. Notes and examples are always informative elements.

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

Platform Descriptor Record

PDR

A set of data that is used to provide semantic information about sensors, effecters, monitored or controller entities, and functions and services within a PLDM implementation

PDRs are mostly used to support PLDM monitoring and control and platform events. This information also describes the relationships (associations) between sensor and control functions, the physical or logical entities that are being monitored or controlled, and the semantic information associated with those elements.

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

CIM

Common Information Model

4.2

EID

Endpoint ID

4.3

FRU

Field Replaceable Unit

4.4

IANA

Internet Assigned Numbers Authority

5 Conventions

Refer to DSP0240 for conventions, notations, and data types that are used across the PLDM specifications. The data types listed in Table 1 are also defined for use in this specification.

Table 1 – PLDM FRU Data Types

Data Type	Interpretation
ASCII	Characters are encoded using the 8-bit ISO8859-1 "ASCII + Latin1" character set encoding. ASCII strings are limited to a maximum of 255 bytes.
UTF-8	UTF-8 encoded string per RFC3629. UTF-8 defines a variable length for Unicode encoded characters where each individual character may require one to four bytes. UTF-8 encoded unicode strings are limited to a maximum of 255 bytes.
UTF-16	UTF-16 encoded string with Byte Order Mark (BOM) per RFC2781. UTF-16 defines an encoding for Unicode characters where each individual character requires two bytes. UTF-16 encoded unicode strings are limited to a maximum of 255 bytes.
UTF-16LE	UTF-16, "little endian" encoded string per RFC2781. UTF-16LE defines an encoding for Unicode characters where each individual character requires two bytes. UTF16LE encoded unicode strings are limited to a maximum of 255 bytes.
UTF-16BE	UTF-16, "big-endian" encoded string per RFC2781. UTF-16BE defines an encoding for Unicode characters where each individual character requires two bytes. UTF16BE encoded unicode strings are limited to a maximum of 255 bytes.

6 PLDM for FRU Data version

The version of this *Platform Level Data Model (PLDM) for FRU Data Specification* shall be 1.0.0 (major version number 1, minor version number 0, update version number 0, and no alpha version).

For the GetPLDMVersion command described in DSP0240, the version of this specification is reported using the encoding as 0xF1F0F000.

7 PLDM for FRU record data format

All PLDM FRU record data is represented by the fields in the following subclauses.

7.1 FRU Record Set Identifier

The FRU Record Set Identifier is a unique number that identifies the FRU record set.

7.2 FRU Record Type

The FRU Record Type identifies the FRU record and is defined in Table 4.

7.3 Number of FRU Fields

The Number of FRU fields indicated the number of fields that are included in a FRU record.

7.4 FRU Encoding Types

String values types for a specific FRU Record are defined in the Encoding Type field of the FRU. The Encoding Type shall apply for all FRU fields in a FRU Record with a sting format. The FRU Encoding Types are defined in Table 2.

All strings shall be preceded by a length variable where a length of zero indicates that the field is not used in this specific FRU. String lengths shall be in bytes. Strings are not null terminated and are limited to a 255-byte size. FRU record set identifiers and their associated record types shall be contiguous in the table.

7.5 FRU Field Type, Length and Value

All FRU fields are defined by a Type, Length and Value (TLV). The Type is defined in Table 5, which also defines the Field format and length ranges. The Field Value is defined by the manufacturer.

Table 2 specifies the format used for the PLDM FRU Data Format.

Table 2 – PLDM FRU Record Data Format

Size	Type	Field
2 bytes	uint16	**FRU Record Set Identifier**
1 byte	uint8	**FRU Record Type**
1 byte	uint8	**Number of FRU fields**
1 byte	uint8	**Encoding Type for FRU fields** **0 = Unspecified** **1 = ASCII** **2 = UTF8** **3 = UTF16** **4 = UTF16-LE** **5 = UTF16-BE** **6-255 = reserved**
1 byte	uint8	**FRU Field Type #1**
1 byte	uint8	**FRU Field Length #1**
Up to 255 bytes (see Table 5)	Determined by FRU Field Type (see Table 5)	**FRU Field #1 Value**
1 byte	uint8	**FRU Field #2 Type**
1 byte	uint8	**FRU Field #2 Length**
Up to 255 bytes (see Table 5)	Determined by FRU Field Type / Length (see Table 5)	**FRU Field #2 Value**
….	….	**……..**
1 byte	uint8	**FRU Field #n Type**
1 byte	uint8	**FRU Field #n Length**
Up to 255 bytes (see Table 5)	Determined by FRU Field Type / Length (see Table 5)	**FRU Field #n Value**

234 Table 3 specifies the format used for the PLDM FRU Record Table Format.

235 **Table 3 – PLDM FRU Record Data Table Format**

Field
FRU Record Data #1 (See Table 2)
FRU Record Data #2
FRU Record Data #3
….
FRU Record Data #n

236 Table 4 defines the FRU Record Types.

237 **Table 4 – FRU Record Type Definitions**

Record Type	Description
0	**Reserved**
1	**General FRU Record**
2 – 253	**Reserved**
254	**OEM FRU Record**
255	**Reserved**

238 Table 5 defines the General FRU record field type definitions.

239 **Table 5 – General FRU Record Field Type Definitions**

Field Type Number	Field Type Description	Field Format	Length
0	**Reserved**	**N/A**	**N/A**
1	**Chassis Type**	**String**	**1-255 bytes**
2	**Model**	**String**	**1-255 bytes**
3	**Part Number**	**String**	
4	**Serial Number**	**String**	
5	**Manufacturer**	**String**	
6	**Manufacture Date**	**Timestamp104**	**13 bytes**
7	**Vendor**	**String**	
8	**Name**	**String**	
9	**SKU**	**String**	
10	**Version**	**String**	
11	**Asset Tag**	**String**	
12	**Description**	**String**	
13	**Engineering Change Level**	**String**	
14	**Other Information**	**String**	

Field Type Number	Field Type Description	Field Format	Length
15	**Vendor IANA**	**uint32**	**4 bytes**
16 – 255	**Reserved**	**N/A**	

Table 6 defines the OEM FRU Record field type definitions.

When the record type is set to OEM = 254, then that record shall contain one field of field type 1 that contains the vendor IANA. Other field types 2-254 are defined by the OEM.

Table 6 – OEM FRU Record Field Type Definitions

Field Type Number	Field Type Description	Field Format
0	**Reserved**	**N/A**
1	**Vendor IANA**	**uint32**
2-254	**OEM specific field types**	**OEM specific**
255	**Reserved**	**N/A**

8 FRU Record Set PDR

The FRU Record Set PDR is used to describe characteristics of the PLDM FRU Record Set Data. The information can be used to locate a Terminus that holds FRU Record Set Data in order to access that data. The PDR also identifies the particular Entity that is associated with the FRU information.

The FRU Record Set PDR is defined in DSP0248.

9 PLDM for FRU Data Transfer

This clause defines the data representations and PLDM commands for FRU data transfer.

9.1 PLDM Representation of FRU Record Data

In the PLDM messages for FRU data transfers, the FRU Record Data representation is as shown in Table 7.

254 **Table 7 – PLDM Representation of FRU Record Data**

Byte	Type	Field
Variable	–	**FRU Record Data (one or more)** See Table 2 for the PLDM representation of PLDM FRU Record Data.

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 FRU Data by using the following algorithm:
		Let L be the total number of bytes in the FRU Record Data 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 FRU Record Data by using the following algorithm. In the algorithm, the receiver parses FRU Record 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 FRU Record Data including the pad and the integrity checksum.
		RemBytes = L;
		i = 0;
		while (RemBytes >= 8)
		{
		Process the i^{th} FRU Record Data in the FRU Record Table;
		RemBytes = RemBytes - 4 – Total length of i^{th} FRU Record Data including the formatted and unformed areas;
		i = i+1;
		}
		NumPadBytes = RemBytes modulo 4;

	uint32	**FRUDataStructureIntegrityChecksum** Integrity checksum on the FRU Data including the pad bytes (if any). It is calculated starting at the first byte of the PLDM representation of FRU Data. 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.

255

9.2 PLDM Commands for FRU Data Transfer

9.2.1 Overview

Table 8 defines the PLDM command codes defined in the following subclause for the PLDM for FRU data transfer. The PLDM FRU messages have their own PLDM message type, which is defined in DSP0245.

Table 8 – PLDM for FRU Data Transfer Command Codes

Command	Code Value	Requirement	Section
GetFRURecordTableMetadata	0x01	Mandatory	See 9.2.2.
GetFRURecordTable	0x02	Mandatory	See 9.2.3.
SetFRURecordTable	0x03	Conditional	See 9.2.4.
GetFRURecordByOption	0x04	Optional	See 9.2.5.

The requirements specified in Table 4 are relative to the services provided by the PLDM terminus.

9.2.2 Get FRURecordTableMetadata

The GetFRURecordTableMetadata command, described in Table 9, is used to get the FRU Record Table metadata information that includes the FRU Record major version, the FRU Record minor version, the size of the largest FRU Record data, total length of the FRU Record Table, total number of FRU Record Data structures, and the integrity checksum on the FRU Record Table data.

Table 9 – GetFRUTableMetadata Command Format

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

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

1	uint8	**FRUDATAMajorVersion** The major version of the FRU DATA specification with which the FRU Record Table. For an implementation compliant with this specification, the FRUDATAMajorVersion shall be set to 0x01.
2	uint8	**FRUDATAMinorVersion** The minor version of the FRU DATA specification with which the FRU Record Table. For an implementation compliant with this specification, the FRUDATAMinorVersion shall be set to 0x00.
3:6	uint32	**FRUTableMaximumSize** The maximum number of data bytes that can be stored in the FRU Record Table using the SetFRURecordTable command. A value of 0x00000000 in this field means that SetFRURecordTable command is not supported. A value of 0xffffffff in this field means unknown and cannot be specified.

Byte	Type	Request Data
7:10	uint32	**FRUTableLength** Total length of the FRU table in bytes
11:12	uint16	**Total number of Record Set Identifiers in table**
13:14	uint16	**Total number of records in table**
15:18	uint32	**FRU DATAStructureTableIntegrityChecksum (CRC-32)** Integrity checksum shall be computed on the FRU Record Table data as shown in Table 7 excluding pad bytes. See Table 7 for more information about this integrity checksum.

9.2.3 GetFRURecordTable

The GetFRURecordTable command, described in Table 10, is used to get the FRU Record Table data. This command is defined to allow the FRU Record 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 FRU Record Table, the response messages contain the non-overlapping contiguous portions of FRU Record Table as defined in Table 7. By combining the portions of FRU Record Table from the response messages, the entire FRU Record Table can be reconstructed.

Table 10 – GetFRURecordTable Command Format

Byte	Type	Request Data
0:3	uint32	**DataTransferHandle** A handle that is used to identify an FRU Record 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,
		FRU_DATA_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 FRU Record Table** This data is a portion of the overall FRU Record Table format shown in Table 3. The portion that is returned is determined by the combination of the DataTransferHandle and TransferOperationFlag fields passed in the request.

9.2.4 SetFRURecordTable

The SetFRURecordTable command, described in Table 11, is used to write the FRU Record Table. This command is defined to allow the FRU Record Table to be transferred using a sequence of one or more command/response messages. When more than one command is used to transfer the FRU Record Table, the request messages contain the non-overlapping contiguous portions of FRU Record Table as defined in Table 7. By combining the portions of FRU record table from the request messages, the entire FRU Record Table can be reconstructed.

Table 11 – SetFRURecordTable Command Format

Byte	Type	Request Data
0:3	uint32	**DataTransferHandle** A handle that is used to identify FRU Record 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 FRU Record Table** See Table 7 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

9.2.5 GetFRURecordByOption

The GetFRURecordByOption command, described in Table 12, is used to get the FRU records by record handle and length. This command is defined to allow the FRU Record Table to be transferred using a sequence of one or more command/response messages. When more than one command is used to transfer the FRU Record Table, the response messages contain the non-overlapping contiguous portions of FRU Record Data as defined in Table 7. By combining the portions of FRU Record Data from the response messages, the entire FRU Record Table can be reconstructed.

291 **Table 12 – GetFRURecordByOption Command Format**

Byte	Type	Request Data
0:3	uint32	**DataTransferHandle** A handle that is used to identify FRU Record Data transfer. This handle is ignored by the responder when the TransferOperationFlag is set to GetFirstPart.
4:5	uint16	**FRUTableHandle** A handle that is used to identify FRU DATA records.
6:7	uint16	**Record Set Identifier** Possible values: {All record sets=0x0000, Specific record set=0x0001 – 0xffff}
8	uint8	**Record Type** Possible values: {All record types=0x00, Specific record types=0x01 – 0xff}
9	uint8	**Field Type** Possible values: {All record field types=0x00, Specific field types=0x01 – 0xff} If field type is non-zero, the record type shall also be non-zero.
10	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,
		FRU_DATA_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	–	**FRU DATAStructureData** See Table 7 for the format.

292 # 10 PLDM for FRU Data Transfer Examples

293 This clause provides examples of PLDM communications using the PLDM commands defined in this
294 specification.

295 ## 10.1 Multipart Transfers

296 The commands defined in clause 9 for transferring FRU Record Table data support multipart transfers.
297 The Get* and Set* commands use flags and data transfer handles to perform multipart transfers. For a
298 data transfer for initiating a data transfer (or getting the first part of data) using a Get* command, the
299 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 and ignore the NextDataTransferHandle 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.

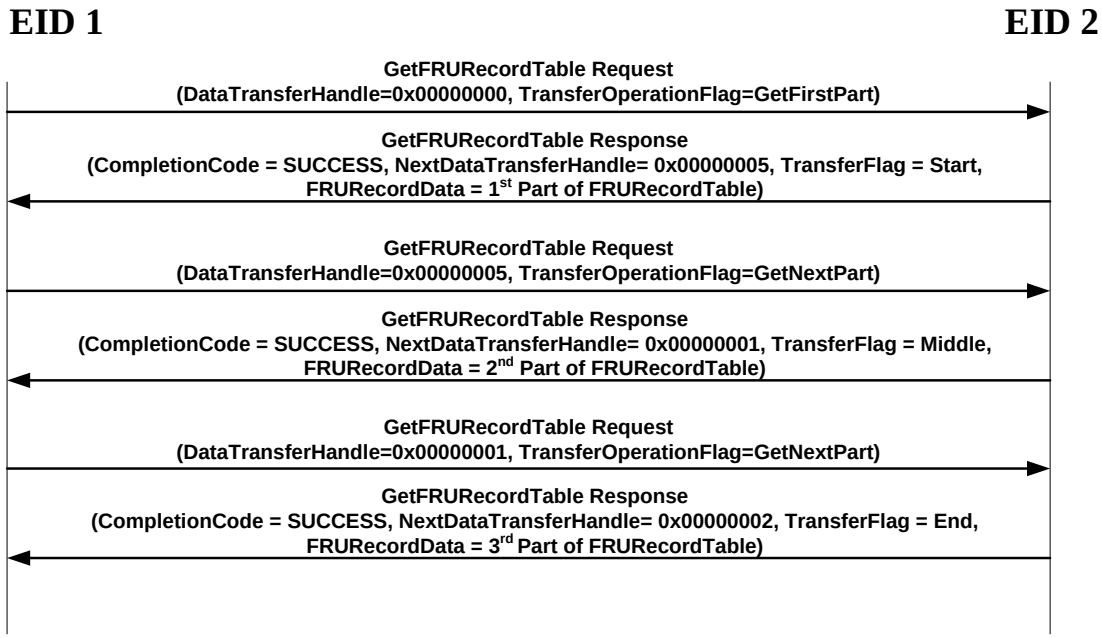

Figure 1 – Multipart FRU Record Table Transfer Using the GetFRURecordTable Command

10.2 FRU Record Table Transfer between Endpoints Example

In this example, the EID 1 sets the FRU Record Table on the EID 2. EID1 first queries the FRU Record Table metadata by using the GetFRURecordTableMetadata command. The response from the EID 2 to this command indicates that the EID 2 does not have the latest FRU Record Table. Upon finding that the EID 2 does not have the latest FRU Record Table, EID 1 transfers the FRU Record Table to EID 2 by using the SetFRURecordTable command. After transferring the latest FRU Record Table, EID 1 reads the FRU Record Table metadata on the EID 2 by using the GetFRURecordTableMetadata command to confirm that the FRU records were correctly set. This example can be used in a push model where EID 2 is maintaining a copy of the FRU Record Table provided by EID 1 and EID 1 pushes to EID 2 a copy of the FRU Record Table by using SetFRURecordTable command. Figure 2 shows the data transfer.

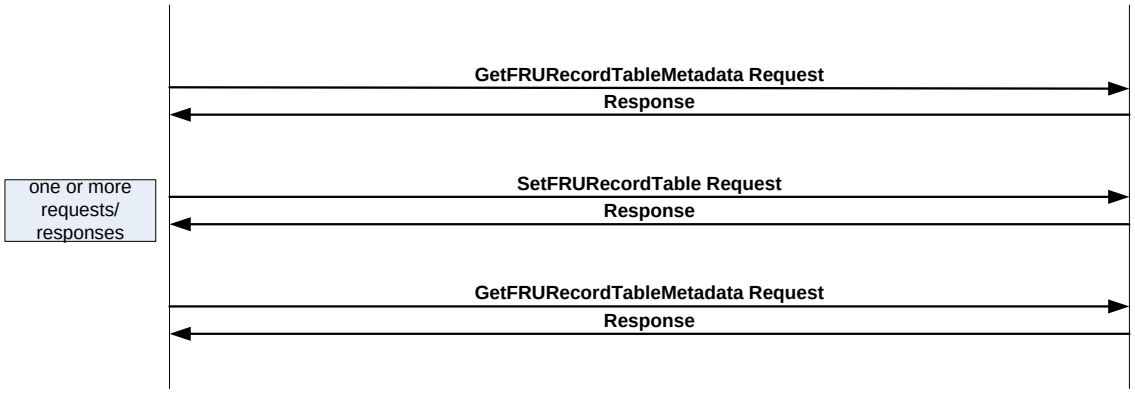

Figure 2 – Example of FRU Record Table Transfer Using the SetFRURecordTable Command

10.3 GetFRURecordByOption Examples

The following examples show three **GetFRURecordByOption** commands and their respective responses. Table 13 describes a sample FRU Record Table that has two FRU record set identifiers that include two FRU fields with part number and serial number FRU field types. The second record set identifier also contains an OEM FRU Record.

Table 13 – Sample FRU Record Table

Field	Value
FRU Record Set Identifier #1	**1030**
FRU Record Type #1	**1 = General FRU Record**
Number of FRU fields	**2**
Encoding Type for FRU fields	**1 = ASCII**
FRU Field #1 Type	**3 = Part Number**
FRU Field #1 Length	**6**
FRU Field #1 Value	**“123456”**
FRU Field #2 Type	**4 = Serial Number**
FRU Field #2 Length	**7**
FRU Field #2 Value	**“SN12345”**
FRU Record Set Identifier #2	**2040**
FRU Record Type #2	**1 = General FRU Record**
Number of FRU fields	**2**
Encoding Type for FRU fields	**1 = ASCII**
FRU Field #1 Type	**3 = Part Number**
FRU Field #1 Length	**6**

Field	Value
FRU Field #1 Value	“345678”
FRU Field #2 Type	0x04 = Serial Number
FRU Field #2 Length	0x07
FRU Field #2 Value	“SN34567”
FRU Record Set Identifier #2	2040
FRU Record Type #3	254 = OEM FRU Record
Number of FRU fields	2
Encoding Type for FRU fields	1 = ASCII
FRU Field #1 Type	1 = vendorIANA
FRU Field #1 Length	4
FRU Field #1 Value	3704
FRU Field #2 Type	2 = OEM
FRU Field #2 Length	6
FRU Field #2 Value	“Fusion”

EXAMPLE 1: This example returns all data for FRU record set identifier #1 = 1030.

In the **GetFRURecordByOption** command:

Record Set Identifier = 1030, **Record Type** =0 and **Field Type** = 0

This results in the data shown in Table 14.

Table 14 – Get FRU Record Set Identifier Response Data (Example 1)

Field	Value
FRU Record Set Identifier #1	1030
FRU Record Type #1	1 = General FRU Record
Number of FRU fields	2
Encoding Type for FRU fields	1 = ASCII
FRU Field #1 Type	3 = Part Number
FRU Field #1 Length	6
FRU Field #1 Value	“123456”
FRU Field #2 Type	4 = Serial Number
FRU Field #2 Length	7
FRU Field #2 Value	“SN12345”

EXAMPLE 2: This example returns all FRU Record type 1 records (get all General FRU Records).

In the **GetFRURecordByOption** command:

Record Set Identifier = 0, **Record Type** =1 and **Field Type** = 0

This results in the data shown in Table 15.

Table 15 – Get FRU Record Type Response Data (Example 2)

Field	Value
FRU Record Set Identifier #1	**1030**
FRU Record Type #1	**1 = General FRU Record**
Number of FRU fields	**2**
Encoding Type for FRU fields	**1 = ASCII**
FRU Field #1 Type	**3 = Part Number**
FRU Field #1 Length	**6**
FRU Field #1 Value	**“123456”**
FRU Field #2 Type	**4 = Serial Number**
FRU Field #2 Length	**7**
FRU Field #2 Value	**“SN12345”**
FRU Record Set Identifier #2	**2040**
FRU Record Type #2	**1 = General FRU Record**
Number of FRU fields	**2**
Encoding Type for FRU fields	**1 = ASCII**
FRU Field #1 Type	**3 = Part Number**
FRU Field #1 Length	**6**
FRU Field #1 Value	**“345678”**
FRU Field #2 Type	**4 = Serial Number**
FRU Field #2 Length	**7**
FRU Field #2 Value	**“SN34567”**

EXAMPLE 3: This example returns all FRU Record type / FRU field type = 4 fields (all General FRU Record serial number fields).

In the **GetFRURecordByOption** command:

Record Set Identifier = 0, **Record Type** = 1 and **Field Type** = 4

This results in the data shown in Table 16.

Table 16 – Get FRU Field Type Response Data (Example 3)

Field	Value
FRU Record Set Identifier #1	**1030**
FRU Record Type #1	**1 = General FRU Record**
Number of FRU fields	**1**
Encoding Type for FRU fields	**1 = ASCII**
FRU Field #2 Type	**4 = Serial Number**
FRU Field #2 Length	**7**
FRU Field #2 Value	**“SN12345”**
FRU Record Set Identifier #2	**2040**
FRU Record Type #2	**1 = General FRU Record**
Number of FRU fields	**1**
Encoding Type for FRU fields	**1 = ASCII**
FRU Field #2 Type	**4 = Serial Number**
FRU Field #2 Length	**7**
FRU Field #2 Value	**“SN34567”**

ANNEX A
(informative)
Notation and conventions

A.1 Notations

Examples of notations used in this document are as follows:

- 2:N — In field descriptions, this will typically be used to represent a range of byte offsets starting from byte two and continuing to and including byte N. The lowest offset is on the left; the highest is on the right.
- (6) — Parentheses around a single number can be used in message field descriptions to indicate a byte field that may be present or absent.
- (3:6) — Parentheses around a field consisting of a range of bytes indicates the entire range may be present or absent. The lowest offset is on the left; the highest is on the right.
- PCIe — Underlined, blue text is typically used to indicate a reference to a document or specification called out in "Normative references" clause or to items hyperlinked within the document.
- rsvd — This case-insensitive abbreviation is for “reserved.”
- [4] — Square brackets around a number are typically used to indicate a bit offset. Bit offsets are given as zero-based values (that is, the least significant bit [LSb] offset = 0).
- [7:5] — This notation indicates a range of bit offsets. The most significant bit is on the left; the least significant bit is on the right.
- `1b` — The lowercase "`b`" following a number consisting of `0`s and `1`s is used to indicate the number is being given in binary format.
- `0x12A` — A leading "`0x`" is used to indicate a number given in hexadecimal format.

ANNEX B
(informative)
Change log

Version	Date	Description
1.0.0	2011-10-26	DMTF Standard
1.0.1	2022-01-01	Fixed footer (Mantis 2977) Added Annex A.