Document Number: DSP0267

Date: 2016-08-23

Version: 1.0.0

Platform Level Data Model (PLDM) for Firmware Update Specification

Information for Work-in-Progress version:

IMPORTANT: This document is not a standard. It does not necessarily reflect the views of the DMTF or its members. Because this document is a Work in Progress, this document may still change, perhaps profoundly and without notice. This document is available for public review and comment until superseded.

Provide any comments through the DMTF Feedback Portal:
http://www.dmtf.org/standards/feedback

Supersedes: None

Document Class: Normative

Document Status: Work in Progress

Document Language: en-US

CONTENTS

Figures

Tables

Foreword

The *Update Specification* (DSP0267) 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 authors wish to acknowledge the following people.

Editor:

- Patrick Caporale – Lenovo

Contributors:

- James Smart – Broadcom Limited
- Hemal Shah – Broadcom Limited
- Bob Stevens – Dell
- Patrick Schoeller – Hewlett Packard Enterprise
- Edward Newman - Hewlett Packard Enterprise
- Tom Slaight – Intel Corporation
- Eliel Louzoun – Intel Corporation
- Scott Dunham – Lenovo
- Kaijie Guo – Lenovo
- Yuval Itkin – Mellanox Technologies
- Shai Lazmi – QLogic Corporation
- Rob Mapes – QLogic Corporation

Introduction

The *Platform Level Data Model (PLDM) Firmware Update Specification* defines messages and data structures for updating firmware or other code objects maintained within the firmware devices of a platform management subsystem. Additional functions related to the sequence of identifying and transferring the firmware, are also defined.

Document conventions

Typographical conventions

The following typographical conventions are used in this document:

- Document titles are marked in *italics*.

Platform Level Data Model (PLDM) for Firmware Update Specification

1 Scope

This specification defines messages and data structures for updating firmware or other objects maintained within the firmware devices of a platform management subsystem. Additional functions related to the sequence of identifying and transferring the component image, are also defined. This document does not specify the operation of PLDM messaging.

This specification is not a system-level requirements document. The mandatory requirements stated in this specification apply when a particular capability is implemented through PLDM messaging in a manner that is conformant with this specification. This specification does not specify whether a given system is required to implement that capability. For example, this specification does not specify whether a given system must support firmware updates over PLDM. However, if a system does support firmware updates over PLDM or other functions described in this specification, the specification defines the requirements to access and use those functions under PLDM.

Portions of this specification rely on information and definitions from other specifications, which are identified in clause 2. Two of these references are particularly relevant:

- DMTF DSP0240, *Platform Level Data Model (PLDM) Base Specification*, provides definitions of common terminology, conventions, and notations used across the different PLDM specifications as well as the general operation of the PLDM messaging protocol and message format.
- DMTF DSP0245, *Platform Level Data Model (PLDM) IDs and Codes Specification*, defines the values that are used to represent different type codes defined for PLDM messages.

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 DSP0236, *MCTP Base Specification 1.2.0*,
http://dmtf.org/sites/default/files/standards/documents/DSP0236_1.2.0.pdf

DMTF DSP0240, *Platform Level Data Model (PLDM) Base Specification 1.0*,
http://dmtf.org/sites/default/files/standards/documents/DSP0240_1.0.0.pdf

DMTF DSP0241, *Platform Level Data Model (PLDM) Over MCTP Binding Specification 1.0*,
http://dmtf.org/sites/default/files/standards/documents/DSP0241_1.0.pdf

DMTF DSP0245, *Platform Level Data Model (PLDM) IDs and Codes Specification 1.1.0*,
http://dmtf.org/sites/default/files/standards/documents/DSP0245_1.1.0.pdf

DMTF DSP0248, *Platform Level Data Model (PLDM) for Platform Monitoring and Control Specification 1.1.0*, http://dmtf.org/sites/default/files/standards/documents/DSP0248_1.1.0.pdf

DMTF DSP0249, *Platform Level Data Model (PLDM) State Sets Specification 1.0,* http://dmtf.org/sites/default/files/standards/documents/DSP0249_1.0.0.pdf

DMTF DSP0257, *Platform Level Data Model (PLDM) FRU Data Specification 1.0,* http://dmtf.org/sites/default/files/standards/documents/DSP0257_1.0.0.pdf

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

IETF STD63, *UTF-8, a transformation format of ISO 10646* http://www.ietf.org/rfc/std/std63.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

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
activation

a process in which the firmware device prepares the newly transferred component images to become the active running firmware components

3.2
auto-apply

a firmware device procedure which is implemented if the component image was being directly placed into the final memory destination in parallel while the component image was being transferred

3.3
automatic activation

a process whereby the firmware device automatically activates a transferred component image during the Apply stage of the firmware update process

3.4
AC Power Cycle Activation

a process whereby a complete removal of power to the entire PLDM subsystem is performed. A common example is a power supply AC cord removed from the system. This will cause all power inputs to the firmware device (including any auxiliary voltage inputs) to be removed. The firmware device will activate any pending firmware component images which indicated an AC Power Cycle as its activation method.

3.5
baseline transfer size

the minimum amount of data that can be requested by a firmware device in an individual command when transferring a component image

3.6

code image

a collection of bytes executed on a processor to perform a function

3.7

component classification

the general type of component. Values for this field are aligned with the Value Map from CIM_SoftwareIdentify.Classifications. Refer to Table 17 for values.

component comparison stamp

a value that can be used to determine if a given component is a higher or lower version then another component of the same identifier

3.8

component identifier

a vendor defined value which distinguishes between component images or firmware components which may have identical classifications but contain different code images

3.9

component image

a collection of bytes contained in a PLDM firmware update package associated with a firmware component of a firmware device. The collection of bytes is transferred to the firmware device and placed (perhaps in a modified form) into the non-volatile storage used by the firmware component.

3.10

component image set

one or more component images contained in a firmware update package that are associated with a particular firmware device

3.11

device identifier record

a set of descriptors used to identify a type of firmware device

3.12

DC Power Cycle Activation

a process whereby the firmware device has its power supply input removed. As most PLDM termini are contained within a solid state device such as an ASIC or FPGA, the power supply input to the firmware device is commonly a DC voltage. Auxiliary voltage inputs are typically not affected by a DC Power Cycle and may continue to be energized during the activation process. The firmware device will activate any pending firmware component images which indicated a DC Power Cycle as its activation method.

3.13

firmware

code stored within a local memory structure (such as a Flash NVRAM) which is executed by a firmware device microcontroller or host processor

3.14

firmware device

FD

a PLDM endpoint (terminus) which contains one or more microcontrollers which execute firmware. The firmware device interacts with the Firmware Update Agent to perform firmware updates of its resident firmware components. Typically this may be a PCI I/O device

3.15

firmware component

a logical entity representing a functional portion of a firmware device. It typically represents a code image resident on a firmware device and executed by a microcontroller on the firmware device or by a host processor to perform a particular function. It may represent both an active and pending code image for the function

3.16

firmware update package

a header describing the contents concatenated with one or more component images for one or more firmware devices

3.17

medium-specific reset

a process whereby a firmware device is reset. The reset type is dependent on the type of interface that the PLDM terminus within the firmware device uses to communicate. For example, a PCI device would have a medium-specific reset via a PCI-reset signal, while a SMBus device would have a SMBus-reset signal. The firmware device will activate any pending firmware component images which indicated a medium-specific reset as its activation method.

3.18

pending firmware component

a newly transferred component image has been delivered to the firmware device, but is not the actively running code image. The firmware component will report details on the pending image (such as version, date, and its activation methods). The applicable activation method must be performed for the pending image to become the actively running image.

3.19

self-contained activation

capability of a firmware device whereby the newly transferred component images can immediately become the actively running firmware component code image after receiving an activate command from the update agent

3.20

software bundle

one of the component classification values. This represents a single component image containing multiple code objects each of which would be known only be the firmware device. The layout of the code objects within the software bundle is not defined in this spec.

3.21

system reboot

a process whereby the PLDM subsystem, which may typically be contained within a platform that has a host operating system, is restarted. The firmware device will activate any pending firmware component images which indicated a system reboot as its activation method.

3.22

update agent

UA

a PLDM endpoint (terminus) which orchestrates passing component images from a firmware update package to a firmware device. Typically this agent is contained within a management controller

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
FD
Firmware Device

4.2
UA
Update Agent

5 Conventions

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

5.1 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.

5.2 Byte ordering

Unless otherwise specified, for all PLDM specifications byte ordering of multi-byte numeric fields or multibyte bit fields is "Little Endian" (that is, the lowest byte offset holds the least significant byte, and higher offsets hold the more significant bytes).

6 PLDM for firmware update overview

This specification describes the operation and format of request messages (also referred to as commands) and response messages for updating firmware components of a firmware device (FD) contained within a platform management subsystem. These messages are designed to be delivered using PLDM messaging. This specification also permits a subset of commands to be implemented by a firmware device which only supports the reporting of existing firmware component details, without the ability to perform a firmware update.

Traditionally, device firmware has been updated by a combination of update tools and binary files provided by individual device manufacturers. Those update tools normally operate inside a host operating system (Linux/Windows/DOS), whereby each device may have their own method provided by the device manufactures to update the firmware into flash chips on the device board. This specification identifies a common method to use PLDM messaging for transferring one or more component images to an FD within the PLDM subsystem and thereby avoiding the usage of host operating system based tools & utilities.

The basic format that is used for sending PLDM messages is defined in DSP0240. The format that is used for carrying PLDM messages over a particular transport or medium is given in companion documents to the base specification. For example, DSP0241 defines how PLDM messages are formatted and sent using MCTP as the transport. The Platform Level Data Model (PLDM) for Firmware Update Specification defines messages that support the following items and capabilities:

- Component Image Transfer
  - Component image transfer mechanism does not require FD specific logic in the UA
  - Regarding an individual firmware device, a Firmware update package may contain; a single combined component image (component classification of software bundle), a single component image for a single firmware component, or multiple component images for multiple firmware components that are applicable to the same firmware device
  - Transfer of a component image may be done sequentially or offset-based as directed by the FD
- Firmware Update Package to Firmware Device association
  - A mechanism to determine which type of FD a firmware update package is targeted at
  - A mechanism to distinguish between firmware update packages applicable to different instantiations of the same FD (e.g. planar vs. adapter)
  - A mechanism to identify the component image that is to be transferred based on device identifier records. A device identifier record may be based on PCI IDs, IANA ID, UUID, or a vendor specific ID.
- Activation Requirements Gathering
  - A mechanism to learn the activation requirements of the FD firmware components
  - This will allow more timely and coordinated activation of all firmware components in the system
  - Activation requirements must specify firmware device recovery times

6.1 Firmware update concepts

A firmware device (FD) is the minimum hardware unit that the PLDM-based firmware update is applied to and with which the Update Agent (UA) communicates with to accomplish the update. The firmware update package for an FD may contain an individual component image or a group of component images which is known as a component image set. This firmware update package is processed to update each firmware component of the FD during the PLDM update.

Each type of FD has a globally unique identity which can be used to distinguish it from other types of FDs. A device identifier record consisting of a set of device descriptors, which are typically based on Industry Standard definitions, may be used to describe an FD type. For example, the descriptors for PCI devices may include PCI Vendor ID and PCI Device ID.

Because an FD could be used in different instantiations (such as using the same device on an I/O adapter vs. on a system planar), which may require different firmware loads, a corresponding more specific set of device descriptors may be necessary to identify the type of FD intended for the update. For example, for PCI devices the additional descriptors such as PCI Subsystem Vendor ID and PCI Subsystem ID may be added to the identifier record used to match a firmware package to an FD.

Component Images that comprise the overall firmware update package each have a classification, identifier, an optional component comparison stamp, and version:

- Component Classification: identifies the function type of the component image, such as UEFI driver, port controller firmware, update SW, diagnostic code, firmware bundle, etc.
- Identifier: A unique value (per vendor) that distinguishes between component images which may have identical classifications but contain different code images
- Component Comparison Stamp: An optional vendor-assigned value that can be used to compare levels between the firmware component within the FD and the component image within the firmware update package. For example, an FD may use a value for this field in the format of MajorMinorRevisionPatch. When comparing Component Comparison Stamps the lower value is down-level compared to the other when performing an unsigned integer comparison between the two
- Version: Contains a string describing the component image version. The version string is provided by the FD manufacturer which should be representative of the contents of the firmware components.

6.2 Update Agent

The Update Agent (UA) is a function that is present within a PLDM subsystem that has the ability to discover firmware devices which are capable of performing a PLDM firmware update and subsequently transfer one or more component images to the device. Only one UA function is supported within a given PLDM subsystem.

6.3 PLDM firmware update packaging

The firmware update package provides the necessary information to be used with the PLDM Firmware Update commands.

To assist in performing an update over PLDM, the firmware update package shall contain a vendor header describing the contents of the firmware update package. The header shall include (refer to clause 7 for details of the header structure):

1) A Header Info Area describing the overall packaging version, date, and checksum
2) Device Identifier records to describe which FDs the update is intended for
3) Package contents information describing the component images contained within the package, including their classification, offset, size, and version

Prior to transferring the component images, the header can be parsed by the UA to identify the following:

- Determine if the firmware update package is applicable for updating a specific FD by comparing Device Identifier records in the package header to those obtained from the FD via the QueryDeviceIdentifiers command.
- Locate the component image for each firmware component if multiple components are contained in the firmware package. A bitmap of which packaged components are intended for which matched FDs is also contained in the header.

A firmware update package may contain one or more component images applicable to a single FD, The UA must advertise each component image individually and attempt to transfer each of the component images to the FD. The firmware update package header provides the information to be able to identify a component by comparing its identifier value, along with additional information such as the component classification.

6.4 Update flow overview

The flow diagram example below describes the high level process of how the UA updates a FD. This flow occurs after the UA has determined which FD(s) the firmware update package is intended for. If there is an error or timeout whereby the entire firmware update process is canceled, then the UA shall restart at the beginning of the update process by sending a RequestUpdate command to the FD.

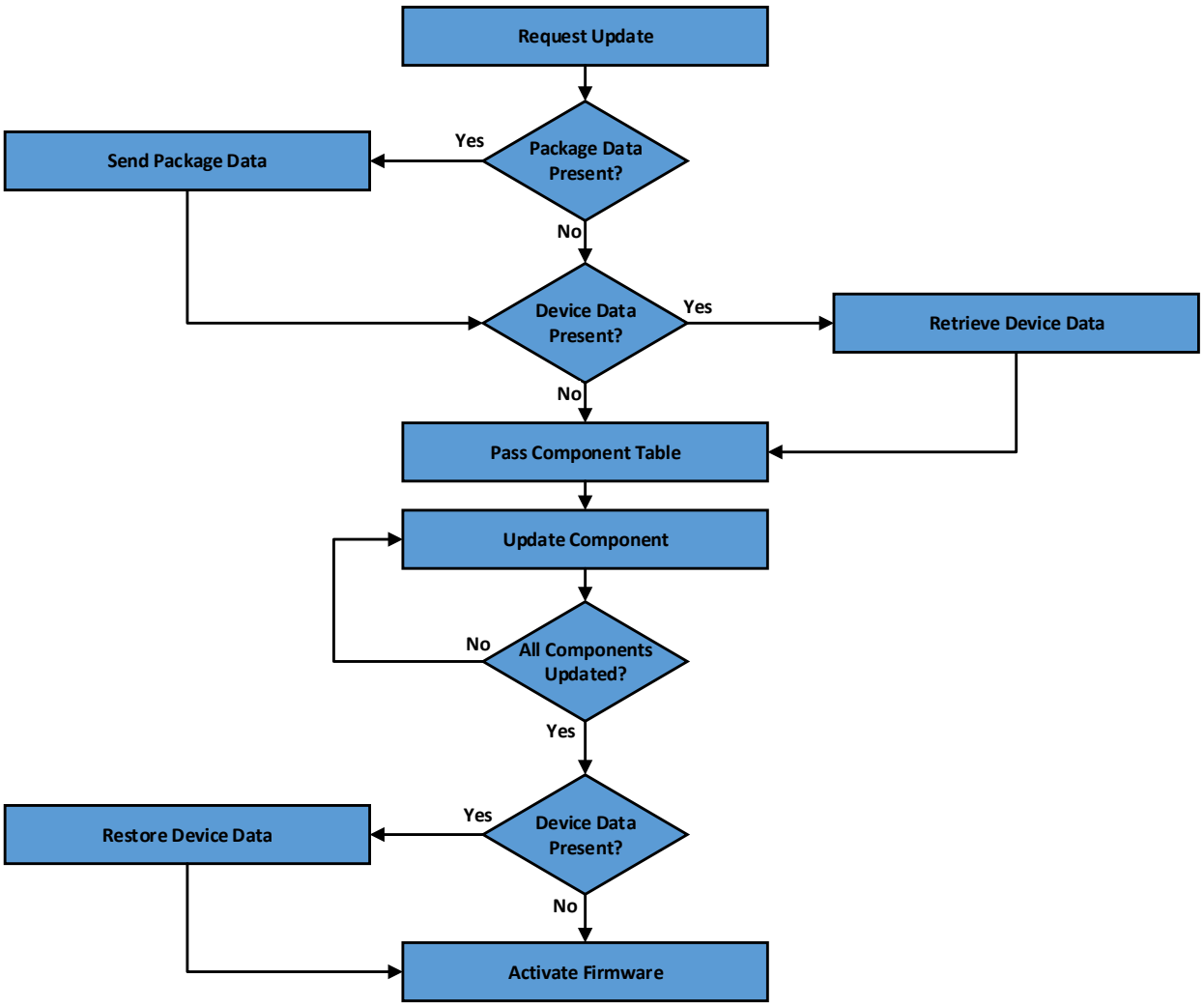


Figure 1 – High-level firmware update flow

As shown in Figure 1, updating an FD is divided into these general steps.

1) To initiate a firmware update, the UA sends the PLDM command RequestUpdate to an FD. The FD replies with a response indicating whether it is available for firmware update. The FD shall then enter an update mode that no longer permits another update request until the UA finishes or cancels the firmware update. During this firmware update mode, the device may or may not be able to provide normal service to the system depending on the capability of the device. The indication of this ability will be returned in the GetFirmwareParameters command.

2) The UA may optionally send data to the FD if the firmware update package contained information that must be sent to the FD prior to transferring component images.

3) The UA may also optionally retrieve FD device data which will be saved by the UA during the firmware update process and restored back to the FD after all component images have been transferred

4) The UA passes the component information table described in the firmware update package header to the FD, which includes the identifier, component comparison stamp, classification, and version information for each of the applicable component images. This is performed by issuing one or more PassComponentTable PLDM commands.

5) The UA processes each of the applicable component images in the firmware package one by one in the same sequence as is described in the firmware package header. The detailed steps of updating a component is described in clause 6.5.

6) If the UA retrieved any firmware device data from the FD prior to the start of the component image transfer, then it will restore the device data prior to proceeding to the activate firmware step.

7) After all component images have been successfully transferred, verified and applied into the firmware device's non-volatile storage, the UA will send the ActivateFirmware command to the FD to finish the firmware update sequence. The FD can return a maximum activation time required to perform the operation. Upon receiving the ActivateFirmware command, if self-contained activation is supported and requested by the UA, the FD should immediately enable the new component images which were transferred to become the actively running code image. The FD will then exit from update mode at the conclusion of the activation. The FD may not be able to provide normal service when activating firmware (as the endpoint may require a restart). The UA periodically sends "GetStatus" to the FD within the maximum activation time to detect when the activation is completed.

Note that for components that do not support self-contained activation, the ActivateFirmware command instructs the FD to perform FD-specific actions required to set all of the updated firmware components into a 'pending activation' state. The newly transferred component images will then become the actively running code images upon external activation (such as a medium specific reset or a host reboot). Non-self-contained activation can be scheduled for a later time via a procedure which is not defined within this specification.

8) The UA may send the CancelUpdate command at any time during the update process to the FD during firmware update, for example if an error is encountered. The FD will then exit update mode which completes the firmware update procedure. It is strongly recommended that the entire firmware update procedure is performed as a single sequence of events to avoid issues that may occur on the FD with partially updated firmware components.

9) If the UA is no longer able to communicate with the FD in order to cancel update mode, the FD itself must provide an internal timer to exit from update mode if no commands are received. Refer to PT1 in clause 6.10 of this document.

6.5 Detailed steps of updating a firmware component

The steps below define transactions required to update one firmware component. If there is any error or timeout during the transfer of a component image, the timing specifications defined within DSP0240 shall be followed for command response timeouts and retries. In addition, specific PLDM Firmware Update timing specifications are defined in clause 6.10 and must be followed.

1) The UA sends the UpdateComponent command, providing component Classification, Component version, Component Size, and Update Options to begin the process of updating a specific firmware component.

2) The FD proceeds to request the component image, by sending one or more RequestFirmwareData commands to the UA. The request command specifies a component image portion to be transferred via the offset and length fields in the RequestFirmwareData command. The UA will validate the request, and if within the boundary of the component image

defined by the firmware update package header, generate a successful response containing the component image portion requested by the FD.

The size of the component image portion requested shall:

- Must be equal to or larger than the baseline transfer size
- Not exceed the MaximumTransferSize value received in the RequestUpdate command.
- Not require the UA to add an amount of padding bytes which is greater than the baseline transfer size.

After a successful transmission of RequestFirmwareData, the FD sends the next RequestFirmwareData command to get the next portion of the component image. This step iterates until the FD receives all data transfers that are required for updating the firmware component, and signals the end of component image transfer to the Update Agent by the TransferComplete command. The UA will then proceed to the verification phase. The TransferComplete command may also be used by the FD to signal the detection of an error condition that terminates the data transfer of the component image.

3) The FD sends the TransferComplete command after the component image is successfully transferred and transitions to the VERIFY state to verify the payload transferred. The UA can optionally send the GetStatus command to query the completion status of the verification process asynchronously. The verify step may require a large amount of time depending on the FD and the operations it must perform to verify the firmware component.

4) Once the firmware component is verified as valid by FD-specific methods, the FD sends VerifyComplete command to the UA. The FD, upon sending the command, transitions to the APPLY state which applies the payload transferred into its non-volatile storage area. Note that some FDs may not have a separate apply step as the component image was being directly placed into the final memory destination in parallel while the component image was being requested. This can occur if the FD does not have a temporary memory location to store the transfer prior to committing the component image to the permanent memory location. In this case the FD must report this auto-apply mode of operation to the UA via the GetFirmwareParameters command, and the FD would send an ApplyComplete command immediately after the VerifyComplete command.

   The UA can optionally send the GetStatus command periodically to query the completion status of this step. The apply step may require a large amount of time depending on the FD and the operations it must perform to apply the firmware component.

   After component apply is complete, the FD may determine that the activation method for this firmware component is different than that reported previously in the GetFirmwareParameters command. This change in activation method can be indicated in the ApplyComplete command. The activation method provided in the ApplyComplete command does not supersede the Component Activation Methods override value that may be provided in the firmware package header. It is recommended that the FD temporarily disable any other management operations which may cause a reset of the device until this apply step is complete. Upon successful completion the FD sends the ApplyComplete command to the UA, and transitions to the READY XFER state.

5) If additional component images remain, the UA will continue to the next component image by sending another UpdateComponent command. Each component image must be transferred individually in the order which they were indexed within the firmware update package.

6) Once all applicable component images have been transferred, the UA shall send ActivateFirmware, and can optionally request activation for all firmware components that indicated support for Self-Contained activation. Activation of firmware components which

require a medium-specific reset, system reboot, or power cycle is not expected to be controlled by the UA and would likely be initiated by higher level systems management software having a broader view of the overall system state. However, the ActivateFirmware command informs the FD to do any preparation necessary to use the newly transferred component images at the next activation event.

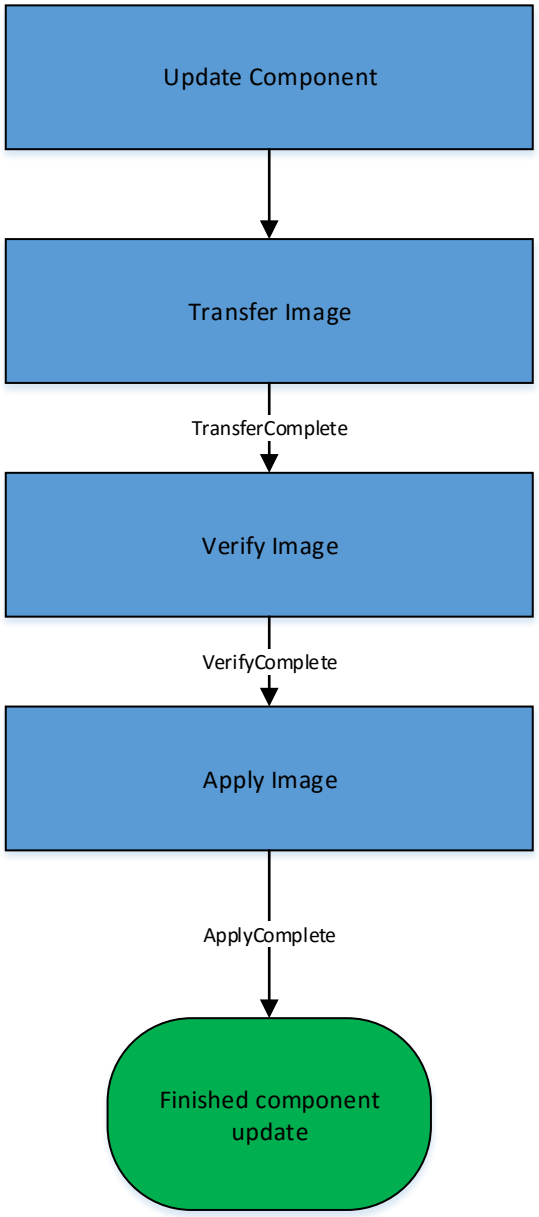
There are two additional commands that the UA can send to the FD during the update process:

1) The UA may send the CancelUpdateComponent command to cancel the update of the current component image being transferred. If the FD has currently requested a portion of component image data via the RequestFirmwareData command, the UA should first respond to any outstanding RequestFirmwareData commands received before sending its request to CancelUpdateComponent. Upon receiving this command, the FD remains in Update Mode and is capable of receiving another UpdateComponent command.

2) The UA may send the CancelUpdate command to cancel the entire firmware update process. Upon receiving the command, the FD returns to the Idle state and exits from update mode. It is strongly recommended that the entire firmware update procedure be performed as a single sequence of events and not cancelled by the UA. This specification does not describe or provide guidance on a recovery procedure if the FD operation is affected by a partially transferred image. After canceling the update, the FD may not be able to operate normally if only a portion of the firmware update has been completed.

Other timeouts or retries may occur and the timing specification defined within clause 6.10 must be followed.

Figure 2 shows the flow for updating a single firmware component.



572

573 **Figure 2 – Firmware component update flow**

574 6.6 Baseline transfer size

575 The baseline transfer size is the minimum amount of bytes that can be requested through the
576 RequestFirmwareData command by the FD. Both the FD and UA shall support the baseline transfer size.
577 The UA can advertise a higher value which it may support as indicated by the MaximumTransferSize
578 value in the RequestUpdate command. The baseline transfer size is 32 bytes.

579 6.7 Firmware component authentication

580 The entire firmware update package could also be signed and authenticated by the UA prior to executing
581 the PLDM Firmware update process, however this process is not within the scope of this specification and
582 is not defined. A higher level entity that delivers the PLDM firmware update package to the Update Agent
583 can add support for authentication.

Firmware components are required to be authenticated by the FD through methods defined by the FD manufacturer. It is recommend that the individual component images contain a signature which enhances the security of the firmware update. It is up to the FD to decide what level of authentication will be performed by the FD within the PLDM firmware update sequence during the verify process.

6.8 Type code

Refer to DSP0245 for a list of PLDM Type Codes in use. This specification requires the PLDM Type Code 000101b as defined in DSP0245.

6.9 Error completion codes

PLDM completion codes for firmware update that are beyond the scope of PLDM_BASE_CODES in DSP0240 are defined in the list below. The usage of individual error completion codes are defined within each of the PLDM command sections.

595 **Table 1 – PLDM firmware update completion codes**

Value	Name	Returned By	Description
Various	PLDM_BASE_CODES	FD & UA	Refer to DSP0240 for a full list of PLDM Base Code Completion values that are supported
0x80	NOT_IN_UPDATE_MODE	FD	Received PLDM firmware update command when the FD is not in update mode
0x81	ALREADY_IN_UPDATE_MODE	FD	Firmware device receives RequestUpdate when it's already in update mode.
0x82	DATA_OUT_OF_RANGE	UA	The requested component image section has an initial offset which is not contained within the image data, or the offset plus the length requested exceeds the range permitted by the UA.
0x83	INVALID_TRANSFER_LENGTH	UA	The length of the requested component image section exceeds the MaxImagePortion negotiated in the RequestUpdate command, or is less than the baseline transfer size
0x84	INVALID_STATE_FOR_COMMAND	FD	The FD is not in a state to expect this command.
0x85	INCOMPLETE_UPDATE	FD	One or more component transfers failed to complete
0x86	BUSY_IN_BACKGROUND	FD	The FD is performing critical background task and cannot execute the command.
0x87	CANCEL_PENDING	UA	Sent by the UA when it receives a RequestFirmwareData command after sending a CancelUpdate or CancelUpdateComponent command
0x88	NON_VOLATILE_WRITE_ERROR	FD	A write error has occurred during the apply component stage of the update
0x89	RETRY_REQUEST_FW_DATA	UA	The Update Agent has requested a retry of the RequestFirmwareData command as it needs more time to retrieve the section of firmware to transfer
0x8A	UNABLE_TO_INITIATE_UPDATE	FD	The Firmware Device is not able to enter into update mode to begin a transfer
0x8B	ACTIVATION_NOT_REQUIRED	FD	The firmware device already has enabled the firmware components to become the active running image on the next external activation, or the firmware components are already activated.
0x8C	SELF_CONTAINED_ACTIVATION_NOT_PERMITTED	FD	The firmware device does not permit Self-Contained activation and returns this code when the UA requests a self-contained activation
0x8D	SECURITY_RESTRICTION	FD	Returned by the FD when a component cannot be downgraded due to a security restriction

Value	Name	Returned By	Description
0x8E	RETRY_REQUEST_UPDATE	FD	The Firmware Device has requested a retry of the RequestUpdate command as it needs more time to prepare for a firmware update
0x8F	NO_DEVICE_DATA	FD	The Firmware Device has no firmware data that must be retrieved by the UA prior to the start of the component image transfers.

6.10 Timing specification

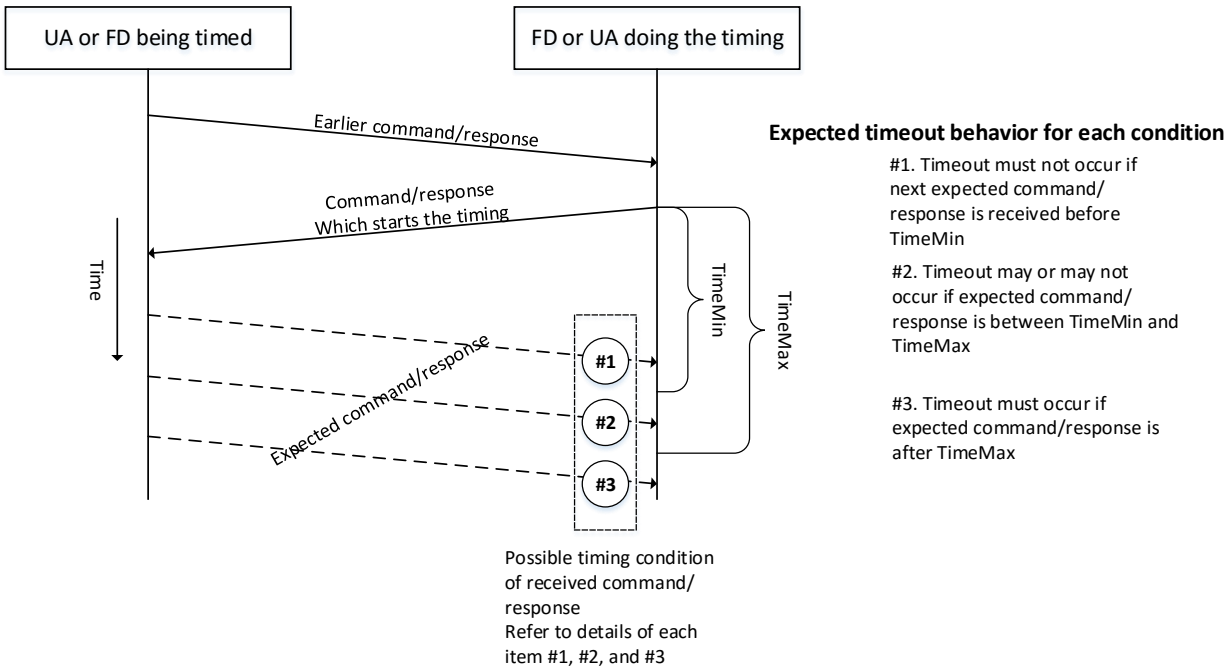
Table 2 below defines timing values that are specific to this document. The table below defines the timing parameters defined for the PLDM Firmware Update Specification. In addition, all timing parameters listed in DSP0240 for command timeouts and number of retries must also be followed. Figure 3 provides a visual representation example of how the minimum and maximum timing parameters should be implemented.

Table 2 – Timing specification

Timing specification	Applicable to UA or FD	Symbol	Min	Max	Description
PLDM Base Timing	UA & FD	PNx PTx			Refer to DSP0240 for the details on these timing values which are applicable to PLDM message timeouts where a response is not received by the UA or FD after sending a request
Number of request retries when a response is received that requires a retry	UA & FD	UAFD_T1	2		Total of three tries, minimum: the original try plus two retries.
Update mode idle timeout	FD	FD_T1	60 seconds	120 seconds	Amount of time before the FD must exit from update mode if no command is received from the Update Agent when it's expected, during the firmware update process. For example, the FD must wait a minimum of 60 seconds for the UA to send a PassComponentTable or UpdateComponent command
Retry request for firmware data	FD	FD_T2	1 second	5 seconds	Amount of time for the FD to wait before resending a RequestFirmwareData command after receiving a RETRY_REQUEST_FW_DATA code from the UA
Retry interval to send next cancel command	UA	UA_T1	500 milliseconds	5 seconds	Amount of time to wait before the UA sends an additional CancelUpdate or CancelUpdateComponent command.

Timing specification	Applicable to UA or FD	Symbol	Min	Max	Description
Request firmware data idle timeout	UA	UA_T2	60 seconds	90 seconds	Amount of time for the Update Agent to cancel the component update if no command is received from the FD when it's expected, during the component image transfer stage. For example, the UA must wait a minimum of 60 seconds for the FD to send another RequestFirmwareData command
State change timeout	UA	UA_T3	180 seconds	-	Amount of time for the Update Agent to wait before canceling the component update if the progress code in the GetStatus command remains unchanged.
Retry request for update	UA	UA_T4	1 second	5 seconds	Amount of time for the UA to wait before resending a RequestUpdate command after receiving a RETRY_REQUEST_UPDATE code from the FD

603



604

605 **Figure 3 – Timeout behavior diagram**

606 7 PLDM firmware package header

607 A header is required for the firmware update package to describe the supported devices and the
608 component images that the firmware update package contains.

The header supports the following:

- The package can be valid for multiple devices and allows for a method to describe each of the supported devices.

  This is useful for the case when a device manufacturer has a family of different devices that use the same component images.

- The package can be specific to a particular instantiation of the same device

  This allows for the case such as where the planar implementation and/or one or more adapter implementations of the same device use different packages. In this case the device subsystem IDs could be used to differentiate between the two firmware devices.

- One to N explicit component images

  The firmware update package can be used for a single monolithic image (component classification of Software Bundle) that contains 1 or more embedded code images. In this case it appears to the UA as if the package contains just one component image but is known by the FD to contain multiple bundled code images.. It can also be used for multiple separate components images, each of which has a vendor-specific component identifier to distinguish between its different components.

Figure 4 shows the entire firmware update package:

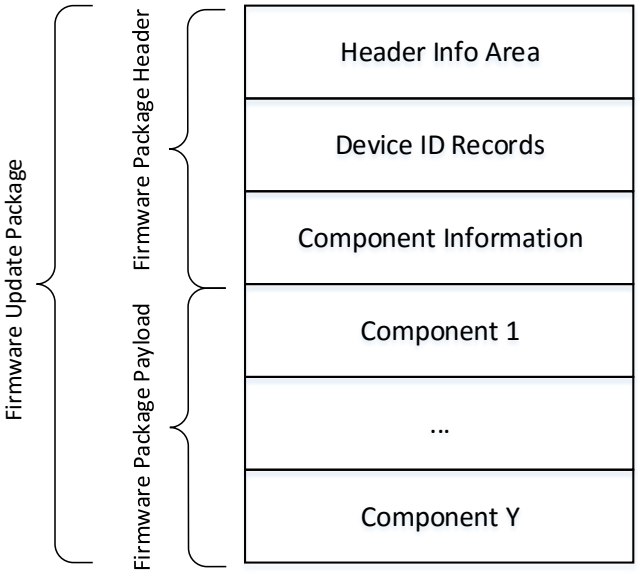


Figure 4 – PLDM firmware update package

Figure 5 shows the structures within the Firmware Package Header:

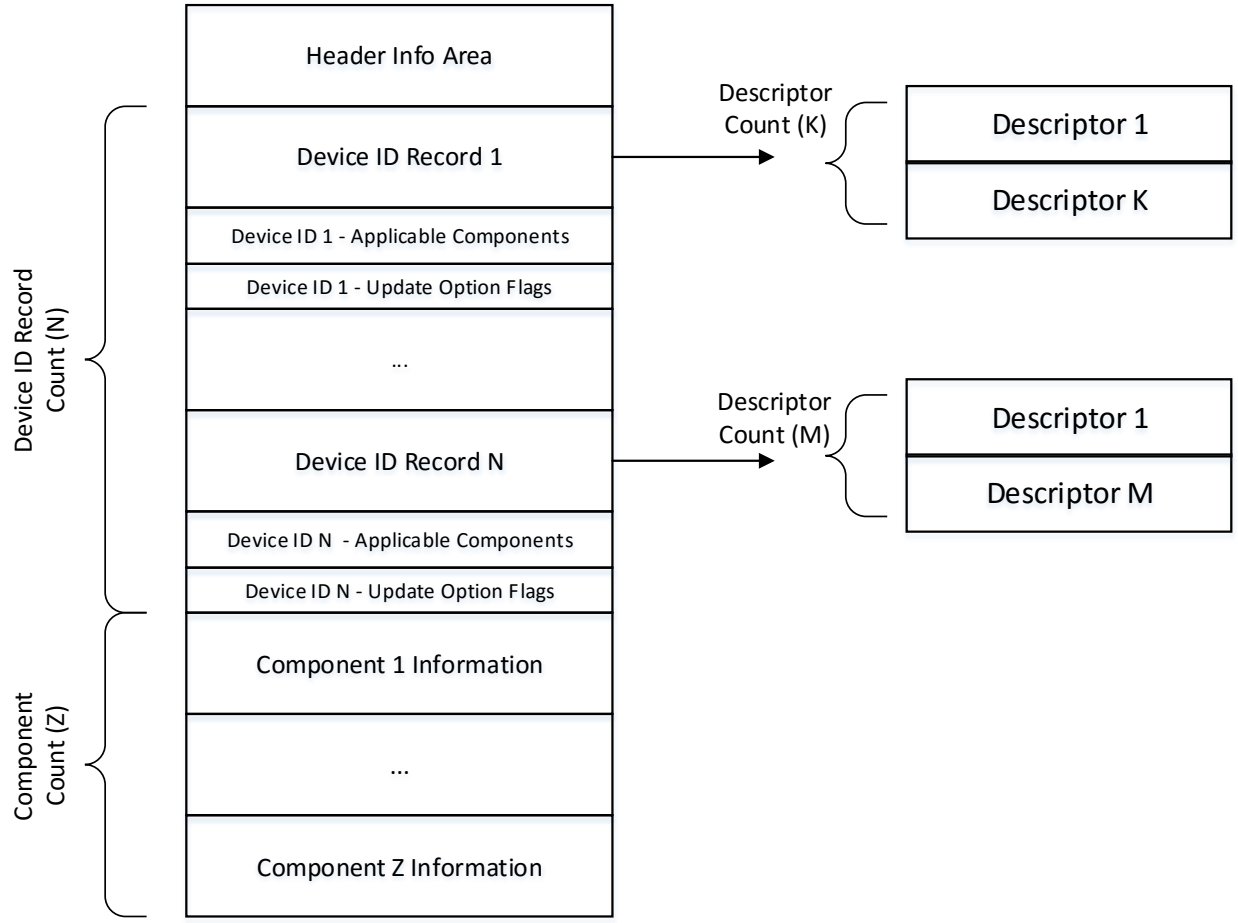


Figure 5 – PLDM firmware package header structure

The Header Information Area is used to validate the entire header.

The Device Identification Area is used to list the FDs that are supported by this firmware package and the component images associated with the device. The order of the devices within the Device Identification Area is of no significance and does not imply any order to the update of devices found to match.

The Component Information Area is used to list the individual component images (as can be seen within the package by the UA) and their size and location within the firmware payload file.

Individual components are required to be authenticated by the FD through methods defined by the FD manufacturer.

641 **Table 3 – PLDM firmware update header**

Package Header Information Byte ordering for entire header is Little Endian per Section 5.2	
Type	**Definition**
uint8[16]	**Package Header Identifier** Mandatory label which defines this object as a valid PLDM Firmware Update formatted header F018878CCB7D49439800A02F059ACA02 is the value to be used for this field which will identify the package as one that supports this PLDM Firmware Update specification
uint8	**Package Header Format Revision** The revision number of the header structure itself. Updated when any field in the PLDM Firmware Update Header changes. Current definition is value 0x01 All other values are Reserved
uint16	**Package Header Size** The count of all bytes in this header structure including the Package Header Information, Device Identification Area and Component Information Area.
enum8	**Package Version String Type** The type of string used in the Package Version String field Refer to Table 18 for values
uint8	**Package Version String Length** The length, in bytes, of the Package Version String not including the required null terminator.
Variable	**Package Version String** Package version information, up to 255 bytes, not including the null terminator. Contains a variable type string describing the version of this Firmware Update Package. The string shall have a null terminator as the last character in the string.
timestamp 104	**Package Release Date and Time** The date and time in which this package was released. Refer to the PLDM Base Specification for field format definition
uint16	**Component Bitmap Bit Length** The number of bits that will be used to represent the bitmap of Applicable Components for a matching device. The value must be a multiple of 8 and be large enough to contain a bit for each component in the package.
Firmware Device Identification Area	
Type	**Definition**
uint8	**Device ID record count** The count of Device Identifier records that are defined for this package. Each record contains a set of identifier descriptors and a component image bitmap indicating applicable firmware components in the package intended for the FD. If all descriptors contained in one of the records matches the record of identifiers returned from the FD via the QueryDeviceIdentifiers command then this package is applicable to the FD.

Firmware Device ID Records Area (this plus Descriptor area are repeated for each record)	
Type	**Definition**
uint16	**Record Length** The total length in bytes of this set of descriptor count field, applicable components field, device update option flags field, and all descriptors including this field
uint8	**Descriptor Count** The number of descriptors included in this record
bitfield32	**Device Update Option Flags** 32 bit field, each bit represents an update option. [31:1] – reserved [0] – Continue Component Updates after failure If set, the UA should always try to update any remaining components after an individual component update fails and the FD remains in the Update mode. This includes continuing after error response codes are received from the FD in the PassComponentTable command response.
uint16	**Firmware Device Package Data Length** The length in bytes of the Firmware Device Package Data field. If no data is provided in the firmware update package for the Firmware Device described by this portion of the header, then this length field should be set to 0x0.
Variable	**Firmware Device Package Data** An optional data field that can be provided within the Firmware Update Package which the UA will transfer to the FD during the firmware update process. The UA has no knowledge of what data is contained within this field, and will simply pass the contents of this field via the SendPackageData command.
Variable Bitfield	**Applicable Components** The size of this bitfield is based on the value contained in the Component Bitmap Bit Length field. Bitmap of which firmware components are applicable to FDs which match this Device Identifier record. A set bit N indicates the Nth (0-based) component in the payload (which is described by the Nth entry in the component information area of the package header) is applicable to this device. Since the Component Bitmap Bit Length field (a multiple of 8) may contain bit positions not associated with any component (if the number of components is not a multiple of 8), those bit positions will contain 0 and are located in the high order bit positions within the bitfield.
Descriptor Area (this is repeated for each descriptor)	
Type	**Definition**
Varies	**Descriptors** Refer to Table 4 for details of these fields and the values that can be selected
Component Information Area	
Type	**Definition**
uint16	**Component Count** Count of individual separately defined component images in this package

Individual Component Information Area (repeated for each component)	
Type	**Definition**
uint16	**Component Classification** FD Manufacturer selected value to indicate specific FD component Values for this field are aligned with the Value Map from CIM_SoftwareIdentify.Classifications Refer to Table 17 for values
uint16	**Component Identifier** FD Manufacturer selected unique value to distinguish between component images
uint32	**Component Comparison Stamp** When Component Options bit 1 is set, this field shall contain a FD Manufacturer selected value to use as a comparison value in determining if a firmware component is down-level or up-level. For the same component identifier, the greater of two component comparison stamps is considered up-level compared to the other when performing an unsigned integer comparison. FD manufacturers should choose the value for the comparison stamp in a manner that permits interim component versions such as patch releases. For example, a value for this field may follow the format of MajorMinorRevisionPatch where each subfield has a range of 0x00 to 0xFF. When Component Options bit 1 is not set, this field should use the value of 0xFFFFFFFF
enum8	**Component Version String Type** The type of strings used in the Component Version String Field Refer to Table 18 for values
uint8	**Component Version String Length** The length, in bytes, of the Component Version String not including the required null terminator.
Variable	**Component Version String** Component version information up to 255 bytes, not including the null terminator. Contains a variable type string describing the Vendor's version. The string shall have a null terminator as the last character in the string.
bitfield16	**Component Options** [15:2] – reserved [1] – Use the Component Comparison Stamp field for comparing this component against the component currently installed within the FD. If this bit is not set, the UA can only use the Component Version String information which does not provide a direct comparison method to determine whether the component is higher or lower than one which is currently installed within the FD. [0] - Force Update When set, this bit indicates to the UA that it should request a comparison override (update the firmware component even if the update would take the component to a lower or equal component comparison stamp than is currently active) in the SetUpdateOptions command for this component.
bitfield16	**Component Activation Methods Override** Provides the capability of the FD for firmware activation. This supersedes the values provided by the FD via the GetFirmwareParameters command. [15:6] – reserved [5]:-AC power cycle [4]:-DC power cycle [3]- System reboot [2]- Medium-specific reset [1]- Self-Contained (can be performed upon transmission of ActivateFirmware command) [0]- Automatic (becomes active as the Apply completes, or as download completes if the FD performs an auto-apply)

uint32	**Component location offset** Offset in Bytes from byte 0 of the Package header to where the Component image begins.
uint32	**Component size** Size in Bytes of the Component image
Package Header Checksum	
Type	**Definition**
uint32	**Package Header Checksum** The Integrity checksum of the PLDM Package Header. It is calculated starting at the first byte of the PLDM Firmware Update Header and includes all bytes of the package Header structure except for the bytes in this field. For this specification, CRC-32 algorithm with the polynomial x32 + x26 + x23 + x22 + x16 + x12 + x11 + x10 + x8 + x7 + x5 + x4 + x2 + 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.

Table 4 provides detailed information on the fields required for the device descriptor records. A device must have at least one descriptor in the package header, but typically will have additional descriptors that the UA will use to match against a FD. Each descriptor is comprised of three fields: (1) Type (2) Length (3) Value. The initial descriptor is restricted to one of three types, while additional descriptors can choose from a larger range of type values including a vendor defined type.

Table 4 – Descriptor definition

Initial Descriptor (This first descriptor and each of the three fields is mandatory – Type, Length, Value)	
Type	**Definition**
uint16	**Initial Descriptor Type** Indicates the type of the Initial descriptor. Refer to Table 5 for possible values. The initial descriptor for a device must be defined by one of the following (PCI Vendor ID, IANA Enterprise ID, or a UUID). If the FD uses Vendor Defined values as part of its implementation of this specification (for example to provide a vendor defined error code or component classification), then the initial descriptor must be set to either PCI Vendor ID or IANA Enterprise ID.
uint16	**Initial Descriptor length** Indicates the length, in bytes, of the Initial Identifier Data field in the descriptor. Refer to Table 5 for possible values.
varies	**Initial Descriptor Value** Payload containing the identifier value for the initial descriptor. Refer to Table 5 for details
Optional Additional Descriptors (repeated for each additional descriptor) **For each additional descriptor three fields are provided (Type, Length, Value)**	
Type	**Definition**
uint16	**Additional Descriptor Type** Indicates the type of the additional descriptor. Refer to Table 5 for possible values.
uint16	**Additional Descriptor length** Indicates the length, in bytes, of the Additional Identifier Data field in the descriptor. Refer to Table 5 for possible values.
varies	**Additional Descriptor Identifier Data** Payload containing the identifier value for the additional descriptors. Refer to Table 5 for details

Table 5 provides a list of available descriptor types that can be used by the firmware package header and FD devices. When the FD is a PCI device, there are four descriptors that are mandatory to be implemented.

Table 5 – Descriptor identifier table

Any one of the highlighted rows can be used for the Initial Device Descriptor		
Type	**Length**	**Value**
0x0000 – PCI Vendor ID	2 bytes	PCI Vendor ID assigned to the FD vendor. If the FD is a PCI device, this descriptor must be the initial descriptor.
0x0001 – IANA Enterprise ID	4 bytes	IANA Enterprise ID assigned to the FD vendor
0x0002 – UUID	16 bytes	UUID assigned to the FD Refer to PLDM Base Specification for UUID format. Version 1 format is recommended
0x0003 – PnP Vendor ID	3 bytes	PnP Vendor ID, in ASCII characters, assigned to the FD vendor. Refer to the PnP & ACPI Registry for more details. http://www.uefi.org/PNP_ACPI_Registry
0x0004 – ACPI Vendor ID	4 bytes	ACPI Vendor ID, in ASCII characters, assigned to the FD vendor. Refer to the PnP & ACPI Registry for more details. http://www.uefi.org/PNP_ACPI_Registry
0x0100 – PCI Device ID	2 bytes	PCI Device ID assigned by the FD vendor. If the FD is a PCI device, this descriptor must be provided.
0x0101 – PCI Subsystem Vendor ID	2 bytes	PCI Subsystem Vendor ID assigned to the FD vendor. If the FD is a PCI device, this descriptor must be provided.
0x0102 – PCI Subsystem ID	2 bytes	PCI Subsystem Device ID assigned by the FD vendor. If the FD is a PCI device, this descriptor must be provided.
0x0103 – PCI Revision ID	1 byte	PCI Revision ID assigned by the FD vendor
0x0104 – PnP Product Identifier	4 bytes	PnP Product Identifier, in ASCII characters, assigned to the FD vendor. Refer to the PnP & ACPI Registry for more details. http://www.uefi.org/PNP_ACPI_Registry
0x0105 – ACPI Product Identifier	4 bytes	ACPI Product Identifier, in ASCII characters, assigned by the FD vendor. Refer to the PnP & ACPI Registry for more details. http://www.uefi.org/PNP_ACPI_Registry
0xFFFF – Vendor Defined	Varies	See Table 6 If the FD or package header uses a Vendor Defined value then the initial descriptor must be set to either PCI Vendor ID or IANA Enterprise ID.

Table 6 provides details for the value field of a vendor defined descriptor.

653 **Table 6 – Vendor defined descriptor value definition**

Type	Definition
enum8	**Vendor Defined Descriptor Title String Type** The type of strings used in the Vendor Defined Descriptor Title String field Refer to Table 18 for values
uint8	**Vendor Defined Descriptor Title String Length** The length, in bytes, of the Vendor Defined Descriptor Title String not including the required null terminator.
Variable	**Vendor Defined Descriptor Title String** Vendor Defined Descriptor information up to 255 bytes, not including the null terminator. Contains a variable type string describing the Vendor's descriptor for the FD device. The string shall have a null terminator as the last character in the string.
Variable	**Vendor Defined Descriptor** Vendor-specific descriptor value. Value will be treated as binary data by the UA

654 ## 7.1 Package to firmware device association

655 The UA can associate a given firmware update package to an FD by using the following steps:

656 *FOR each FD that supports PLDM for Firmware Update*

657 *Retrieve FD identifier records via the QueryDeviceIdentifiers command*

658 *MATCH = FALSE; Start at First Device Identifier Record in the package header*

659 *WHILE ((MATCH==FALSE) AND (Device Identifier Record(s) remain in package))*

660 *Read Device Identifier Record from Package Header*

661 *IF all Package Device Identifier Record descriptors match FD descriptors*

662 *MATCH = TRUE; Selected Record = Current Record; Break;*

663 *Move to next Device Identifier Record in package header*

664 Note that all descriptors in a package Device Identifier Record must match those returned by the FD but
665 not vice-versa (the FD may return more descriptors than are indicated in the firmware package Device
666 Identifier record).

667 Each FD that generated a match can accept components from the firmware package.

668 # 8 Operational behaviors

669 This clause describes the operating states of the FD.

670 ## 8.1 State definitions

671 The following states are required to be implemented by the FD.

672 - **IDLE**

673 IDLE is the default state in which the firmware device must always start after an initialization. In
674 this state the FD is not performing any firmware update actions as it has not received a
675 RequestUpdate command from the UA.

- **LEARN COMPONENTS**

  After receiving the RequestUpdate command, the FD moves to this state while waiting to receive the PassComponentTable command from the UA. The FD will then learn the size, identifier, component comparison stamp, classification and version of the component images the UA intends to send.

- **READY XFER**

  After learning the component image information, the FD moves to this state to wait for the command initiating a component image transfer. This state is re-entered after each component image is transferred, verified and applied. The FD remains in this state after all firmware components have been applied as it waits for an activation command.

- **DOWNLOAD**

  After receiving the command to update a firmware component, the FD moves to this state to begin requesting the transfer of portions of the component image from the UA. When an entire component image has been transferred, the UA is informed and the FD moves to the VERIFY state.

- **VERIFY**

  In this state the FD performs a validation check of the firmware component, it is up to the FD to determine the method used for verification of the code image. Upon successful verification, the FD informs the UA and moves to the APPLY state

- **APPLY**

  In this state the FD writes the verified code image to the non-volatile storage area that will contain the code image within the device. When completed, the FD moves to the READY XFER state

- **ACTIVATE**

  The activation request from the UA occurs after all component images have been transferred, verified and applied. The FD performs activation of the firmware components it can activate internally (self-contained), and enables all other newly transferred code images to become the actively running code image on the next initialization. After activation the FD moves to the IDLE state

8.2 State machine

Table 7 describes the operating states, responses, and transitions between states that the FD must implement. The transition to the next state occurs after the FD performs the response action. Two commands, GetFirmwareParameters and QueryDeviceIdentifiers, are considered ‘inventory’ type commands and can be sent by the UA to the FD in any state.

Table 7 – Firmware device state machine

Current State	Trigger	Response	Next State
IDLE	RequestUpdate	Success	LEARN COMPONENTS
	RequestUpdate	Unable to Imitate Update or Retry Request Update	IDLE
	QueryDeviceIdentifiers	Success with Identifiers	IDLE
	GetFirmwareParameters	Success with fw info	IDLE
	GetStatus	Success with info	IDLE

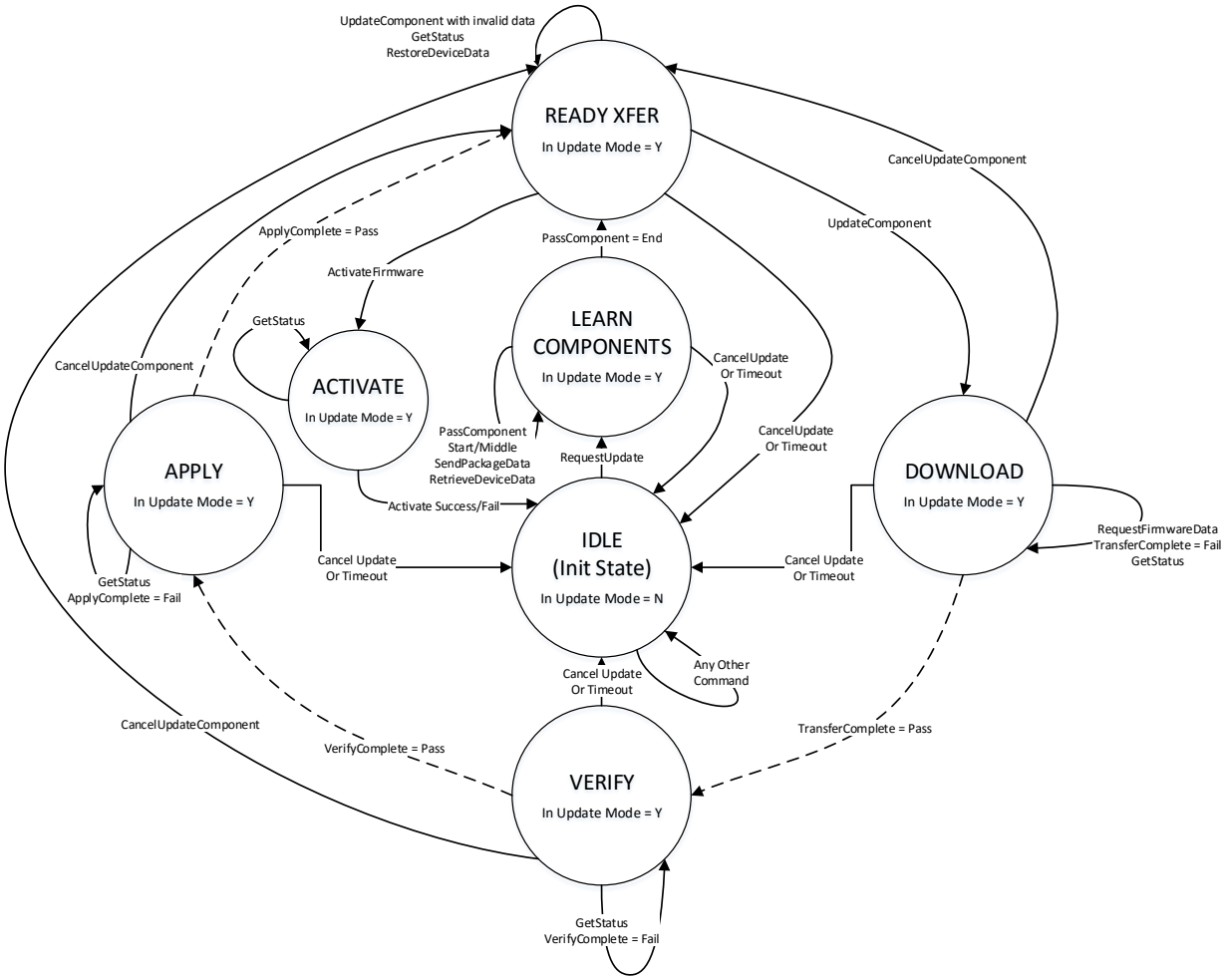
Current State	Trigger	Response	Next State
	Any other Update Command	Not in Update Mode	IDLE
LEARN COMPONENTS	FD_T1 timeout waiting for next command	None	IDLE
	SendPackageData	Success	LEARN COMPONENTS
	RetrieveDeviceData	Success	LEARN COMPONENTS
	PassComponentTable with TransferFlag set to Start or Middle	Success	LEARN COMPONENTS
	PassComponentTable with TransferFlag set to End or StartAndEnd	Success	READY XFER
	CancelUpdate	Success	IDLE
	Unknown Error	ERROR	IDLE
	Any other Update command	Invalid State Machine	LEARN COMPONENTS
READY XFER	FD_T1 timeout waiting for next command	None	IDLE
	RequestUpdate	Already In Update Mode	READY XFER
	GetFirmwareParameters	Success with fw info	READY XFER
	SetUpdateOptions	Success or Not Supported	READY XFER
	UpdateComponent with unsupported classification	Invalid Firmware Identity	READY XFER
	UpdateComponent with known incompatible version	Invalid Versions	READY XFER
	UpdateComponent with supported and acceptable parameters	Success	DOWNLOAD
	RestoreDeviceData	Success	READY XFER
	ActivateFirmware after all expected components have completed transfer, verify and apply	Success with Activation Delay Time	ACTIVATE
	ActivateFirmware prior to all expected components completed or expected RestoreDeviceData command was not received	Incomplete Update response	READY XFER
	CancelUpdate	Success	IDLE
	Unknown Error	ERROR	IDLE
	GetStatus	Success indicating READY XFER state	READY XFER
	Any other Update command	Invalid State Machine	READY XFER
DOWNLOAD	FD_T1 timeout waiting for response to	None	IDLE

Current State	Trigger	Response	Next State
	RequestFirmwareData		
	Ready to request next component image section	RequestFirmwareData	DOWNLOAD
	Receive RequestFirmwareData section	Process data	DOWNLOAD
	All necessary data received and processed for this component	TransferComplete with succesful result	VERIFY
	GetFirmwareParameters	Success with fw info	DOWNLOAD
	Corrupt data received	TransferComplete with failure result	DOWNLOAD
	Error response to RequestFirmwareData	TransferComplete with failure result	DOWNLOAD
	Retry response to RequestFirmwareData	Delay, then RequestFirmwareData For same component image section as prior request)	DOWNLOAD
	CancelUpdateComponent	Success	READY XFER
	CancelUpdate	Success	IDLE
	Unknown Error	ERROR	IDLE
	GetStatus while downloading	Download in progress	DOWNLOAD
	GetStatus after successful download	Download successful	DOWNLOAD
	Any other Update command	Invalid State Machine	DOWNLOAD
VERIFY	GetStatus while verifying	Verification in progress	VERIFY
	GetStatus after successful verify	Verification successful	VERIFY
	GetStatus after failure to verify	Verification failed	VERIFY
	Verify completes successfully	Send VerifyComplete command with successful result	APPLY
	Verify ended with failure	Send VerifyComplete command with failure result	VERIFY
	CancelUpdateComponent	Success	READY XFER
	CancelUpdate	Success	IDLE
	Unknown Error	ERROR	IDLE
	Any other Update command	Invalid State Machine	VERIFY
APPLY	GetStatus while applying	Apply in progress	APPLY
	GetStatus after successful apply	Apply successful	APPLY
	GetStatus after apply failure	Apply failed	APPLY
	Apply completes successfully	Send ApplyComplete command with successful result	READY XFER
	Apply ended with failure	Send ApplyComplete command with failure result	APPLY
	CancelUpdateComponent	Success	READY XFER
	CancelUpdate	Success	IDLE

Current State	Trigger	Response	Next State
	Unknown Error	ERROR	IDLE
	Any other Update command	Invalid State Machine	APPLY
ACTIVATE	Sets transferred component image to become active firmware component on next activation	Success	IDLE
	Self-contained activation option is requested for applicable components	Success with maximum activation time in response	IDLE
	GetStatus	Activate state	ACTIVATE
	Unknown Error	ERROR	IDLE
	Any other Update command	Invalid State Machine	ACTIVATE

8.3 State transition diagram

Figure 6 illustrates the state transitions the FD must implement. Each bubble represents a particular state as defined in Table 7. Upon initialization or a reboot/reset the FD must enter the IDLE state. The dashed lines represent state change transitions, not due to timeouts, which are initiated by the FD while the solid lines indicate transitions that are initiated by the UA.



716

717 **Figure 6 – Firmware device state transition diagram**

718 9 PLDM for firmware update command list

719 This clause provides the list of commands that are used by Update Agents and Firmware Devices that
720 implement PLDM Firmware Updates as defined in this specification. The command numbers for the
721 PLDM messages are given in Table 8.

722 If PLDM Firmware Update is implemented within a UA or FD, the Mandatory/Optional/Conditional (M/O/C)
723 requirements shown in Table 8 apply.

724 This specification permits the usage of only a limited number of supported commands for a Firmware
725 Device to provide inventory information only without the ability to update the components. This is known
726 as the ‘Inventory Only’ function of this specification.

727 **Table 8 – Firmware update commands requirements**

Command	Command Numbers	Command Support Requirements UA	Command Support Requirements FD	Command Support Requirements FD Inventory Only	Command Requestor (Initiator)	Reference
INVENTORY COMMANDS						
QueryDeviceIdentifiers	0x01	M	M	M	UA	See 10.1
GetFirmwareParameters	0x02	M	M	M	UA	See 10.2
Reserved	0x03-0x0F					
UPDATE COMMANDS						
RequestUpdate	0x10	M	M	O	UA	See 11.1
SendPackageData	0x11	M	O	O	--UA	See 11.2
RetrieveDeviceData	0x12	M	O	O	--UA	See 11.3
PassComponentTable	0x13	M	M	O	UA	See 11.4
UpdateComponent	0x14	M	M	O	UA	See 11.5
RequestFirmwareData	0x15	M	M	O	FD	See 11.6
TransferComplete	0x16	M	M	O	FD	See 11.7
VerifyComplete	0x17	M	M	O	FD	See 11.8
ApplyComplete	0x18	M	M	O	FD	See 11.9
RestoreDeviceData	0x19	M	O	O	UA	See 11.10
ActivateFirmware	0x1A	M	M	O	UA	See 11.11
GetStatus	0x1B	M	M	O	UA	See 11.12
CancelUpdateComponent	0x1C	M	M	O	UA	See 11.13
CancelUpdate	0x1D	M	M	O	UA	See 11.14

728 # 10 PLDM for firmware update – inventory commands

729 This clause describes the commands that are used by Update Agents and Firmware Devices that
730 implement the inventory commands which are defined in this specification. The command numbers for
731 the PLDM messages are given in Table 8.

732 ## 10.1 QueryDeviceIdentifiers command format

733 This command is used by the UA to obtain the firmware identifiers for the FD. The FD shall provide a
734 successful response message to this command even if it's not in update mode as long as it is able to
735 respond to commands.

736 **Table 9 – QueryDeviceIdentifiers command format**

Type	Request data
--	No request data
Type	**Response data**
enum8	**completionCode** value: { PLDM_BASE_CODES }
uint32	**Device Identifiers Length** Contains the length, in bytes, of the Descriptor Count field and all Descriptors including this field
uint8	**Descriptor Count** The total number of descriptors for the FD device
Varies	**Descriptors** Refer to Table 4 for details on the format and values for these fields

737 ## 10.2 GetFirmwareParameters command format

738 The UA sends GetFirmwareParameters command to acquire the component details such as classification
739 types and corresponding versions of the FD. The FD shall provide a successful response message to this
740 command even if it's not in update mode as long as it is able to respond to commands.

741 **Table 10 – GetFirmwareParameters command format**

Type	Request data
--	No request data
Type	**Response data**
enum8	**completionCode** value: { PLDM_BASE_CODES }
bitfield32	**Capabilities During Update** 32 bit field, containing capability of the firmware device Bit [31:8] – Reserved Bit [7:4] – Firmware Device Update Mode Restrictions Bit 4: 0 – No host OS environment restriction for update mode 1 – Firmware device unable to enter update mode if host OS environment is active Bit 7:5 -- Reserved Bit [3] – FIrmware Device Partial Updates 0: Firmware Device cannot accept a partial update and all components present on the FD must be updated 1: Firmware Device can support a partial update, and a partial component image set can be transferred to the FD. Bit [2] – Firmware Device Accessibility during Firmware Update 0: Device functionality is not reduced during Firmware Update 1: Device functionality will be reduced, perhaps becoming inaccessible, during Firmware Update Bit [1] – Component Update Failure Retry Capability 0: Device can have component updated again without exiting update mode and restarting transfer via RequestUpdate command 1: Device will not be able to update component again unless it exits update mode and the UA sends a new Request Update command Bit [0] – Component Update Failure Recovery Capability 0: Device will revert to previous component image upon a failure, timeout, or cancelation of the transfer 1: Device will not revert to previous component image upon a failure, timeout, or cancelation of the transfer. Therefore the current pending component version may be corrupt if the transfer does not complete. The UA must re-attempt a component transfer.
uint16	**Component Count** Number of firmware components which reside within the FD. Each one will have an entry in the following Component Parameter Table.
Variable	**Component Parameter Table** Table of component information entry for all of the updateable components both active and pending on the FD. Refer to Table 19 for details

742 **Table 11 – Component Parameter Table -- Entry Format**

Type	Data
uint16	**ComponentClassification** Vendor specific Component Classification information. Refer to Table 17for specific values special values: 0x00, 0xFFFF = reserved
uint16	**Component Identifier** FD Manufacturer selected unique value to distinguish between component images
uint8	**Component Classification Index** Used to distinguish identical components that have the same classification and identifier which can use the same component image but the images are stored in different locations in the FD
uint32	**Active Component Comparison Stamp** Optional Firmware component comparison stamp which is currently active. If the firmware component does not provide a component comparison stamp, this value should be set to 0x0
enum8	**Active Component Version String Type** The type of strings used in the Active Component Version String Field Refer to Table 18 for values
uint8	**Active Component Version String Length** The length, in bytes, of the Active Component Version String not including the required null terminator.
Variable	**Active Component Version String** Firmware component version, which is currently active, up to 255 bytes, not including the null terminator. Contains a variable type string describing the Vendor's version. The string shall have a null terminator as the last character in the string.
char[8]	**Active Component Release Date** The date corresponding to the component version level being reported – Format YYYYMMDD If the firmware component does not provide a date, this value should be set to "00000000"
uint32	**Pending Component Comparison Stamp** Optional Firmware component comparison stamp which is pending activation. This field is valid once the firmware device has received the ActivateFirmware command to prepare the firmware component for activation, but the activation method requires further action to enable the pending image to become the actively running code image. If no pending Firmware Component exists, this value should be set to 0x0
enum8	**Pending Component Version String Type** The type of strings used in the Pending Component Version String Field This field is valid once the firmware device has received the ActivateFirmware command to prepare the firmware component for activation, but the activation method requires further action to enable the pending image to become the actively running code image. Refer to Table 18 for values. If no pending Firmware Component exists, this value should be set to 0

Type	Data
uint8	**Pending Component Version String Length** The length, in bytes, of the Pending Component Version String not including the required null terminator. This field is valid once the firmware device has received the ActivateFirmware command to prepare the firmware component for activation, but the activation method requires further action to enable the pending image to become the actively running code image. If no pending Firmware Component exists, this value should be set to 0x0
Variable	**Pending Component Version String** Firmware component version, which is pending activation, up to 255 bytes, not including the null terminator. The version reported here should be the one that will become active on the next initialization or activation of the component. The pending component version value may be same as the active component version. Contains a variable type string describing the Vendor's version. The string shall have a null terminator as the last character in the string. This field is valid once the firmware device has received the ActivateFirmware command to prepare the firmware component for activation, but the activation method requires further action to enable the pending image to become the actively running code image. If no pending Firmware Component exists, this field is zero bytes in length
char[8]	**Pending Component Release Date** The date corresponding to the component version level being reported – Format YYYYMMDD This field is valid once the firmware device has received the ActivateFirmware command to prepare the firmware component for activation, but the activation method requires further action to enable the pending image to become the actively running code image. If no pending Firmware Component exists, this value should be set to "00000000"
bitfield16	**Component Activation Methods** Provides the capability of the FD for firmware activation [15:6] – reserved [5]:-AC power cycle [4]:-DC power cycle [3]- System reboot [2]- Medium-specific reset [1]- Self-Contained (can be performed upon transmission of ActivateFirmware command) [0]- Automatic (becomes active as the Apply completes, or as download completes if the FD performs an auto-apply)
Bitfield 32	**Capabilities During Update** 32 bit field, containing capability of the firmware component Bit [31:1] – Reserved Bit [0] – Firmware Device apply state functionality 0: Firmware Device will execute a full apply step when receiving the apply command which will include migrating the new component image to its final non-volatile storage destination. 1: Firmware Device performs an 'auto-apply' during transfer phase and apply step will be completed immediately

11 PLDM for firmware update – update commands

This clause describes the commands that are used by Update Agents and Firmware Devices that implement the firmware update capability as defined in this specification. The command numbers for the PLDM messages are given in Table 8.

11.1 RequestUpdate command format

This is the first PLDM command to initiate a firmware update for an FD.

The FD must enter update mode if command response indicates success. While the FD is in update mode, it shall not accept another RequestUpdate command. In this case, the FD must return the ALREADY_IN_UPDATE_MODE completion code.

If the FD is unable to enter update mode to begin a transfer due to other operations or the current operating environment it must return the UNABLE_TO_INITIATE_UPDATE completion code.

Table 12 -- RequestUpdate command format

Type	Request data
uint32	**MaximumTransferSize** Specifies the maximum size, in bytes, of the variable payload allowed to be requested by the FD via the **RequestFirmwareData** command that is contained within a PLDM message. This value must be equal to or greater than baseline transfer size. Refer to clause 6.6 for details on the baseline transfer size.
uint16	**Number of Components** Specifies the number of components that will be passed to the FD during the update. The FD can use this value to compare against the number of PassComponentTable commands received.
uint8	**Maximum Outstanding Transfer Requests** Specifies the number of outstanding RequestFirmwareData commands that can be sent by the FD. The minimum required value is ‘1’ which the UA shall support. It is optional for the UA to support a value higher then ‘1’ for this field.
Type	**Response data**
enum8	**completionCode** value: { PLDM_BASE_CODES, ALREADY_IN_UPDATE_MODE, UNABLE_TO_INITIATE_UPDATE, RETRY_REQUEST_UPDATE }
uint8	**Firmware Device Data Available** 0x00: The Firmware Device does not have device data 0x01: The Firmware Device has device data which must be obtained from the RetrieveDeviceData command prior to sending the UpdateComponent command. This data will be retained by the UA until it issues a RestoreDeviceData command to return the device data to the firmware device.

Error completion codes handling:

- ALREADY_IN_UPDATE_MODE: returned from the FD if the device is already in update mode. This may happens when the UA loses connection with the FD in the previous update operation due to an unexpected error. In this case, the UA may send CancelUpdate command requesting the FD to exit from update mode.
- UNABLE_TO_INITIATE_UPDATE: The FD is not able to enter update mode to begin the transfer. The FD shall remain in IDLE state.
- RETRY_REQUEST_UPDATE: The FD is not able to enter update mode immediately. The UA should resend the RequestUpdate command after UA_T4 as the FD needs more time to prepare to enter update mode. The FD shall remain in IDLE state.

11.2 SendPackageData command format

The UA sends this command to transfer optional data that the FD must receive during the firmware update process. This command is only used if the firmware update package contained content within the Firmware Device Package Data field.

Table 13 – SendPackageData command format

Type	Request data
uint16	**PackageData Length** The length in bytes of the PackageData field.
Variable	**PackageData** Firmware Device Package Data that is transferred by the UA to the FD. The data that is transferred in this field was provided by the Firmware Update Package via the Firmware Device Package Data field.
Type	**Response data**
enum8	**completionCode** value: { PLDM_BASE_CODES }

11.3 RetrieveDeviceData command format

The UA sends this command to acquire optional data that the FD must send to the UA prior to beginning the transfer of component images. This command is only used if the FD has indicated in the RequestUpdate command response that it has data that must be retrieved and restored by the UA. The Firmware Device Data retrieved by this command will be sent back to the FD through the RestoreDeviceData command after all component images have been transferred.

776 **Table 14 – RetrieveDeviceData command format**

Type	Request data
--	No request data
Type	**Response data**
enum8	**completionCode** value: { PLDM_BASE_CODES, NO_DEVICE_DATA }
uint16	**Firmware Device Data Length** The length in bytes of the Firmware Device Data field.
Variable	**Firmware Device Data** Firmware Device Data that will be retrieved by the UA prior to starting the component image transfers, and returned to the FD at the conclusion of the component image transfers. The UA has no knowledge of what data is contained within this field, and will simply pass the contents of this field via the RestoreDeviceData command.

777 Error completion codes handling:

778 • NO_DEVICE_DATA: returned from the FD if there is no firmware device data that needs to be
779 retrieved.

780 ## 11.4 PassComponentTable command format

781 PassComponentTable command is used to pass component information to the FD after the FD enters
782 update mode. The PassComponentTable command contains the component information table for a
783 specific component including Index, Classification, and Version.

784 If the firmware package contains more than one component, multiple PassComponentTable commands
785 are required to be sent by the UA (one for each component). The UA must pass the component table for
786 all applicable components listed in the firmware package header in ascending order of index.

787 By receiving the component table, the FD possesses the knowledge of which component(s) are going to
788 be updated. The UA must set the transferflag field to indicate whether the command represents the start,
789 middle, end, or both start and end of the table transfer. Upon receiving the end notification, this indicates
790 to the FD that the entire list has been sent and the FD should transition to the READY XFER state.

791 **Table 15 – PassComponentTable command format**

Type	Request data
enum8	**TransferFlag** The transfer flag that indicates what part of the Component Table this request represents Possible values: {Start = 0x1, Middle = 0x2, End = 0x4, StartAndEnd = 0x5}
uint16	**ComponentClassification** Vendor specific Component Classification information. Refer to Table 17 for specific values. special values: 0x00, 0xFFFF = reserved
uint16	**Component Identifier** FD Manufacturer selected unique value to distinguish between component images
uint8	**Component Classification Index** The Component Classification Index which was obtained from the GetFirmwareParameters command to indicate which firmware component the information contained within this command is applicable for.
uint32	**Component Comparison Stamp** FD Manufacturer selected value to use as a comparison value in determining if a firmware component is down-level or up-level. For the same component identifier, the greater of two component comparison stamps is considered up-level compared to the other when performing an unsigned integer comparison. FD manufacturers should choose the value for the comparison stamp in a manner that permits interim component versions such as patch releases. For example, a value for this field may follow the format of MajorMinorRevisionPatch where each subfield has a range of 0x00 to 0xFF.
enum8	**Component Version String Type** The type of strings used in the Component Version String Field Refer to Table 18for values
uint8	**Component Version String Length** The length, in bytes, of the Component Version String not including the required null terminator.
Variable	**Component Version String** Firmware component version information up to 255 bytes, not including the null terminator. Contains a variable type string describing the Vendor's version. The string shall have a null terminator as the last character in the string.
Type	**Response data**
enum8	**completionCode** value: { PLDM_BASE_CODES, NOT_IN_UPDATE_MODE }
enum8	**Component Response** The FD should reply back with initial compatibility with component provided by UA 0 – Component can be updated 1 – Component will not be updated – Non-zero Component Response Code must be provided All other values reserved

uint8	**Component Response Code** 0x00: No response code – used when component can be updated 0x01: Component comparison stamp is identical to the firmware component comparison stamp in the FD. Force update option flag (if supported by FD) will need to be set to update component. 0x02: Component comparison stamp is lower than the firmware component comparison stamp in the FD. Force update option flag (if supported by FD) will need to be set to update component. 0x03: Invalid component comparison stamp 0x04: Component has conflict with another component provided in a separate PassComponentTable command 0x05: Pre-requisites for this component have not been met 0x06: Component is not supported on FD 0x07: Security restrictions prevent component from being downgraded 0x08: Incomplete Component Image Set was received. The FD will reject each UpdateComponent command with response code of 0x08. 0x09: Reserved 0x0A: Component version string is identical to the firmware component version string in the FD. Force update option flag (if supported by FD) will need to be set to update component. This response code can be used only when component comparison stamp is not supported by the FD. 0x0B: Component version string is lower to the firmware component version string in the FD. Force update option flag (if supported by FD) will need to be set to update component. This response code can be used only when component comparison stamp is not supported by the FD. 0x0C – 0xCF - Reserved 0xD0-0xEF: Firmware Device Vendor defined component response code. When an FD device uses a vendor defined status code, it must also provide its Vendor ID information by using either the PCIe or IANA Vendor descriptor type. For details refer to Table 5. 0xF0 – 0xFF - Reserved

Error completion code handling:

- NOT_IN_UPDATE_MODE: Returned by the FD if it's not currently in update mode

11.5 UpdateComponent command format

The UA sends UpdateComponent command to request updating a specific firmware component.

Table 16 – UpdateComponent command format

Type	Request data
uint16	**ComponentClassification** Classification value provided by the Firmware Package Header information for the component to be transferred. Values for this field are aligned with the Value Map from CIM_SoftwareIdentity.Classifications. Refer to Table 17for values
uint16	**Component Identifier** FD Manufacturer selected unique value to distinguish between component images
uint8	**Component Classification Index** The Component Classification Index which was obtained from the GetFirmwareParameters command to indicate which firmware component should be updated

uint32	**Component Comparison Stamp** FD Manufacturer selected value to use as a comparison value in determining if a firmware component is down-level or up-level. For the same component identifier, the greater of two component comparison stamps is considered up-level compared to the other when performing an unsigned integer comparison. FD manufacturers should choose the value for the comparison stamp in a manner that permits interim component versions such as patch releases. For example, a value for this field may follow the format of MajorMinorRevisionPatch where each subfield has a range of 0x00 to 0xFF.
enum8	**Component Version String Type** The type of strings used in the Component Version String Field Refer to Table 18for values
uint8	**Component Version String Length** The length, in bytes, of the Component Version String not including the required null terminator.
Variable	**Component Version String** Firmware component version information up to 255 bytes, not including the null terminator. Contains a variable type string describing the Vendor's version. The string shall have a null terminator as the last character in the string.
uint32	**ComponentImageSize** Size in bytes of the component image
bitfield32	**UpdateOptionFlags** 32 bits field, each bit represents an update option that is reserved for future expansion. [31:1] – reserved [0] – Request Force Update of component – Can be used to inform the FD device to perform a transfer even if the component has a lower or equal component comparison stamp than what is currently installed. The UA will set this bit for any component which has the force update bit set in the Component Options field of the package header. Additionally, the UA could set the bit as instructed by commands used to provide the update package to the UA (these commands are out of scope for this spec).
Type	**Response data**
enum8	**completionCode** value: { PLDM_BASE_CODES, NOT_IN_UPDATE_MODE, SECURITY_RESTRICTION }
enum8	**Component Compatibility Response** The FD should reply back with initial compatibility with component provided by UA 0 – Component can be updated, and the FD will begin to request data via the RequestFirmwareData command 1 – Component will not be updated – Non-zero Component Response Code must be provided and the FD will not request any data for this component All other values reserved

uint8	**Component Compatibility Response Code** 0x00: No response code – used when component can be updated 0x01: Component comparison stamp is identical to the firmware component comparison stamp in the FD. Force update option flag (if supported by FD) will need to be set to update component. Can also be used if FD does not support force flag. 0x02: Component comparison stamp is lower than the firmware component comparison stamp in the FD. Force update option flag (if supported by FD) will need to be set to update component. Can also be used if FD does not support force flag. 0x03: Invalid component comparison stamp or version 0x04: Component has conflict with another component provided in a separate PassComponentTable command 0x05: Pre-requisites for this component have not been met 0x06: Component is not supported on FD 0x07: Security restrictions prevent component from being downgraded 0x08: Component cannot be updated as an Incomplete Component Image Set was received from the PassComponentTable commands. 0x09: Component information does not match details presented from PassComponentTable commands 0x0A: Component version string is identical to the firmware component version string in the FD. Force update option flag (if supported by FD) will need to be set to update component. Reason code can be used only when component comparison stamp is not supported by the FD. 0x0B: Component version string is lower to the firmware component version string in the FD. Force update option flag (if supported by FD) will need to be set to update component. Reason code can be used only when component comparison stamp is not supported by the FD. 0x0C – 0xCF - Reserved 0xD0-0xEF: Firmware Device Vendor defined component response code. When an FD device uses a vendor defined status code, it must also provide its Vendor ID information by using either the PCIe or IANA Vendor descriptor type. For details refer to Table 5. 0xF0 – 0xFF – Reserved
bitfield32	**Update Option Flags Result** If device fail to apply any of the option flags and the Completion Code to OPTION_FLAG_NOT_ACCEPTED, this field describes which flag(s) was not accepted by the FD, using the same bit flag definition as provided in UpdateOptionFlags in the request data field. A ‘1’ in the bit indicates the flag was not accepted
uint16	**Estimated Time Before Sending RequestFirmwareData** Amount of time the FD requires to prepare before sending the first RequestFirmwareData command. Measured in seconds. If this field contains a non-zero value, the UA should not begin any of the timers listed in Table 2 until after the amount of time present in this field has elapsed.

797

798 **Table 17 – Component classification values**

Value	Package Classification Type
0x0000	**Unknown**
0x0001	**Other**
0x0002	**Driver**
0x0003	**Configuration Software**
0x0004	**Application Software**
0x0005	**Instrumentation**
0x0006	**Firmware/BIOS**
0x0007	**Diagnostic Software**
0x0008	**Operating System**
0x0009	**Middleware**
0x000A	**Firmware**
0x000B	**BIOS/FCode**
0x000C	**Support/Service Pack**
0x000D	**Software Bundle**
0x8000-0xFFFF	**Reserved for Vendor Defined values by FD manufacturer.** When an FD device uses a vendor defined classification, it must also provide Vendor ID information by using either the PCIe or IANA Vendor descriptor type. For details refer to Table 5.

799

800 **Table 18 – Component version string type values**

Value	Package Classification Type
0	**Unknown**
1	**ASCII**
2	**UTF-8**
3	**UTF-16**
4	**UTF-16LE**
5	**UTF-16BE**

801 Error completion codes handling:

802 • NOT_IN_UPDATE_MODE: Returned by the FD if it's not currently in update mode.

803 • SECURITY_RESTRICTION: Returned by the FD when a component cannot be downgraded
804 due to a security restriction

11.6 RequestFirmwareData command format

In order for the FD to retrieve a clause of a component image, the FD sends RequestFirmwareData request message to the UA, specifying its offset and length. The UA will send a response message that includes the firmware payload specified by the offset and length from the request message. The length of the payload must be a value that is no smaller than the baseline transfer size and no greater than the MaximumTransferSize supported by the UA. The FD shall not request an offset and length values which would extend beyond the end of the component image by more than the baseline transfer size.

The length of the payload in the response message must match the length field specified in the request message, otherwise the FD shall drop the response data and resend the RequestFirmwareData command. If an error is encountered whereby the FD cannot continue to request data, the FD must send a TransferComplete command with the TransferResult field set to exception. The UA must then send a CancelUpdateComponent and can then attempt to restart the transfer or exit the update completely by sending CancelUpdate.

The FD can request the same data more than one time if it wants to perform an immediate verification of the data. The UA shall allow the FD to request data at any valid offset within the firmware data.

Table 19 – RequestFirmwareData command format

Type	Request data
uint32	**Offset** Offset of the component image segment within the current component being transferred
uint32	**Length** Size of the component image segment requested by the FD. This value must be set between the baseline transfer size, and the **MaximumTransferSize** value from the **RequestUpdate** command. Refer to clause 6.6 for details on the baseline transfer size.
Type	**Response data**
enum8	**completionCode** value: { PLDM_BASE_CODES, INVALID_TRANSFER_LENGTH, DATA_OUT_OF_RANGE, RETRY_REQUEST_FW_DATA, CANCEL_PENDING }
Variable	**ComponentImageSection** The payload contains the segment corresponding to the component image from Offset to (Offset + Length - 1). The UA must pad with 00s if the length requested extends past the end of the component image. The maximum amount of padding the UA must support is equal to the baseline transfer size. Any request from the FD which would require a larger amount of pad bytes must have its completion code set to DATA_OUT_OF_RANGE and no data is returned. Refer to clause 6.6 for details on the baseline transfer size.

Error completion codes handling:

- INVALID_TRANSFER_LENGTH: The length of the requested component image section exceeds the MaxTransferSize in the RequestUpdate command, or is less than the baseline transfer size.
- DATA_OUT_OF_RANGE: The requested component image segment offset exceeds the range of the component image, or would require the UA to pad the response with a number of bytes that is larger than the baseline transfer size. The FD can send another RequestFirmwareData command to attempt a retry with a different offset and length value.

- RETRY_REQUEST_FW_DATA: The requested component image section is not currently available from the UA. The UA requests that the firmware device retry this command after FD_T2 as it may be retrieving the component image data from an external source.
- CANCEL_PENDING: The requested component image section is not returned by the UA as it previously sent a CancelUpdate or CancelUpdateComponent command to the FD.

11.7 TransferComplete command format

The FD sends TransferComplete command to the UA once the FD has successfully downloaded all the data for the component image or determines the download has failed.

If the TransferResult of the request message indicates download is successful, the UA proceeds to the next step that verifies the firmware.

If the TransferResult of the request message indicates download has failed, the UA may retry on the failed component by sending another UpdateComponent command, or cancel the update by sending CancelUpdate command. The UA is required to send CancelUpdateComponent before retry.

Table 20 – TransferComplete command format

Type	Request data
uint8	**TransferResult** Use to indicate the result of the Download stage: 0x00: Transfer has completed with success 0x01: Reserved 0x02: Transfer has completed with error as the version of the image received does not match the version expected from the UpdateComponent command 0x03: Firmware Device has aborted the transfer 0x04 - 0x08: Reserved 0x09: Timeout occurred while performing action 0x0A: Generic Error has occurred 0x0B – 0x6F: Reserved 0x70 – 0x8F: Firmware Device Vendor defined status code. When an FD device uses a vendor defined status code, it must also provide Vendor ID information by using either the PCIe or IANA Vendor define type. For details refer to Table 4. 0x90 – 0xFF: Reserved When the FD has a result where multiple choices may be applicable, it should look to provide the most descriptive result code, which is applicable, in this field.
Type	**Response data**
enum8	**completionCode** value: { PLDM_BASE_CODES}

11.8 VerifyComplete command format

After the component image transfer finishes successfully, the FD transitions to the VERIFY state and performs a validation check against the component image that was received.

The time consumed on verification can be significant depending on the verification algorithm and hardware performance of the FD controller. The UA may send GetStatus commands to poll the state of verification from the FD controller

After the FD finishes verifying the component successfully (including that the image data represents the expected version that was to be transferred), it issues the VerifyComplete command and transitions to the

APPLY state. If the verification fails, the FD shall remain in the VERIFY state and issue VerifyComplete command indicating failed status of the verification. The UA must send a CancelUpdateComponent command if a verification failure occurs

Table 21 – VerifyComplete command format

Type	Request data
uint8	**Verify Result** Use to indicate the result of the Verify stage: 0x00: Verify has completed with success 0x01: Verify has completed with a verification failure – FD will not transition to APPLY state to apply the component 0x02: Verify has completed with error as the version of the image received does not match the version expected from the UpdateComponent command 0x03 - 0x08: Reserved 0x09: Timeout occurred while performing action – FD will not transition to APPLY state to apply the component 0x0A: Generic Error has occurred 0x0B – 0x8F: Reserved 0x90 - 0xAF: Firmware Device Vendor defined status code. When an FD device uses a vendor defined status code, it must also provide Vendor ID information by using either the PCIe or IANA Vendor define type. For details refer to Table 4. 0xB0 – 0xFF: Reserved When the FD has a result where multiple choices may be applicable, it should look to provide the most descriptive result code, which is applicable, in this field.
Type	**Response data**
enum8	**completionCode** value: { PLDM_BASE_CODES}

11.9 ApplyComplete command format

After firmware verification is successful, the FD transitions into the APPLY state and begins updating the component image into the storage location where the object resides. After the FD finishes applying the component successfully, it issues an ApplyComplete command indicating success and the FD transitions to the READY XFER state to be ready for the next component transfer. If the apply failed, the ApplyComplete command indicates the failure and the FD remains in the APPLY state.

For some firmware devices, the apply step was already occurring as the component was being transferred. For example, if the FD does not have a separate temporary storage region and the code being transferred was directly placed into the final non-volatile storage region. It is recommended but not required that an FD support a separate temporary storage region. When a FD uses this type of operation, it should report its behavior as ‘auto-apply’ via the GetFirmwareParameters.

Based on the newly applied component, if the FD determines that the activation method is different than what would be reported in the GetFirmwareParameters command prior to the component update, then the FD can set the appropriate bits in the Component Activation Methods Modification parameter.

869 **Table 22 – ApplyComplete command format**

Type	Request data
uint8	**Apply Result** Used to indicate the result of the Apply stage: 0x00: Apply has completed with success 0x01: Apply has completed with success and has modified its activation method. Values must be provided in the Component Activation Methods Modifications field 0x02: Apply has completed with a failure due to a memory write issue 0x03 - 0x08: Reserved 0x09: Timeout occurred while performing action 0x0A: Generic Error has occurred 0x03 – 0xAF: Reserved 0xB0 – 0xCF: Firmware Device Vendor defined status code. When an FD device uses a vendor defined status code, it must also provide Vendor ID information by using either the PCIe or IANA Vendor define type. For details refer to Table 4. 0xD0 – 0xFF: Reserved When the FD has a result where multiple choices may be applicable, it should look to provide the most descriptive result code, which is applicable, in this field.
bitfield16	**Component Activation Methods Modification** Field contain values when the Apply Result is set to 0x02. Otherwise, each bit shall be set to '0' Provides the capability of the FD for firmware activation. This supersedes the values provided by the FD via the GetFirmwareParameters command. It does NOT supersede the activation override methods that can be provided in the PLDM Firmware Update Header. [15:6] – reserved [5]:-AC power cycle [4]:-DC power cycle [3]- System reboot [2]- Medium-specific reset [1]- Self-Contained (can be performed upon transmission of ActivateFirmware command) [0]- Automatic (becomes active as the Apply completes, or as download completes if the FD performs an auto-apply)
Type	**Response data**
enum8	**completionCode** value: { PLDM_BASE_CODES}

870 ## 11.10 RestoreDeviceData command format

871 The UA sends this command to restore the data that was originally retrieved through the
872 RetrieveDeviceData command. This command is only used if the FD required some data to be retrieved
873 and restored by the UA.

874 **Table 23 – RestoreDeviceData command format**

Type	Request data
uint16	**Firmware Device Data Length** The length in bytes of the Firmware Device ata field.
Variable	**Firmware Device Data** Firmware Device Data that is transferred by the UA to the FD. The data that is transferred in this field was obtained by the UA through the RetrieveDeviceData command prior sending the component images to the FD..
Type	**Response data**
enum8	**completionCode** value: { PLDM_BASE_CODES }

875 ## 11.11 ActivateFirmware command format

876 After all firmware components in the FD have been transferred and applied, the UA sends this command
877 to inform the FD to prepare all successfully applied components to become active at the next activation.

878 The UA can also request activation of all components that have an activation method of 'Self-Contained'.

879 The FD must exit from update mode at the completion of this command.

880 The ActivationDelayTime in the response message indicates the maximum time in seconds to finish
881 activation if self-contained activation is requested. The FD controller may not be able to respond to
882 commands when activating firmware. The UA periodically sends "GetStatus" to the FD controller within
883 the maximum activation time to detect if the activation completes.

884 **Table 24 – ActivateFirmware command format**

Type	Request data
bool8	**Self-Contained Activation Request** True: FD must activate all self-contained components False: FD must not activate any self-contained components
Type	**Response data**
enum8	**completionCode** value: { PLDM_BASE_CODES, NOT_IN_UPDATE_MODE, INVALID_STATE_FOR_COMMAND, INCOMPLETE_UPDATE, ACTIVATION_NOT_REQUIRED, SELF_CONTAINED_ACTIVATION_NOT_PERMITTED }
uint16	**Estimated Time for SelfActivation** Amount of time the FD requires to perform a self-contained activation. Measured in seconds After sending this command, the UA should not begin any of the timers listed in Table 2 until after the amount of time present in this field has elapsed. If Self-Contained activation is not requested, this field should be set to zero.

885 Error completion codes handling:

886 • INCOMPLETE_UPDATE: Returned by the FD if it is able to determine that not all components
887 are updated completely, or the FD did not receive an expected RestoreDeviceData command.
888 The FD will remain in the READY XFER state, and will not perform activation.

889 • INVALID_STATE_FOR_COMMAND: The FD only expects this command in READY XFER
890 state.

891 • NOT_IN_UPDATE_MODE: Returned by the FD if it's not in the update mode

892

893 • ACTIVATION_NOT_REQUIRED: Returned by the FD if the new firmware components are
894 already pending activation (such as through a previous ActivateFirmware command), or the
895 activation method was ‘automatic’ and therefore the component was already activated at the
896 completion of the apply step.

897 • ACTIVATION_NOT_PERMITTED: Retuned by the FD if it does not support Self-Contained
898 activation and the Self-Contained Activation Request is set to True.

899 11.12 GetStatus command format

900 The UA sends this command to acquire the status of the FD.

901 **Table 25 – GetStatus command format**

Type	Request data
--	No request data
Type	**Response data**
enum8	**completionCode** value: { PLDM_BASE_CODES }
enum8	**CurrentState** Current State machine state of the FD. 0 – IDLE 1 – LEARN COMPONENTS 2 – READY XFER 3 – DOWNLOAD 4 – VERIFY 5 – APPLY 6 – ACTIVATE
enum8	**PreviousState** The previous different State machine state of the FD. 0 – IDLE 1 – LEARN COMPONENTS 2 – READY XFER 3 – DOWNLOAD 4 – VERIFY 5 – APPLY 6 – ACTIVATE
enum8	**AuxState** Used provide additional information to the UA to describe the current operation state of the FD while in one of the following states (Download, Verify, Apply, or Activate) 0 – Operation in progress 1 – Operation successful 2 – Operation failed – Must provide Error Code in AuxStateStatus field 3 – Value used when FD is in IDLE, Learn Components, or Ready Xfer state

uint8	**AuxStateStatus** 0x00: AuxState is In Progress or Success 0x01 - 0x08: Reserved 0x09: Timeout occurred while performing action 0x0A: Generic Error has occurred 0x02 – 0x6F: Reserved 0x70-0xEF: Firmware Device Vendor defined status code. When an FD device uses a vendor defined status code, it must also provide Vendor ID information by using either the PCIe or IANA Vendor define type. For details refer to Table 4. 0xF0 – 0xFF: Reserved
uint8	**ProgressPercent** Used when StateMachine is in the DOWNLOAD, VERIFY or APPLY state. Value range from 0 – 100 or 101This field is optional for an FD. If the FD does not support a progress code, the value returned must be 101 If this field is supported by the FD, the value provided in this field represents the percentage complete of the current action (DOWNLOAD, VERIFY, or APPLY). The value is initialized to 0 upon each transition of the StateMachine.
enum8	**ReasonCode** Used when StateMachine is in the IDLE state. Provides the reason for why the StateMachine entered the IDLE state. The value is retained until the next transition to IDLE has occurred which will then cause this field to be updated. 0 – Initialization of firmware device has occurred 1 -- ActivateFirmware command was received 2 – CancelUpdate command was received 3 – Timeout occurred when in LEARN COMPONENT state 4 – Timeout occurred when in READY XFER state 5 – Timeout occurred when in DOWNLOAD state 200-255: Firmware Device Vendor defined status code. When an FD device uses a vendor defined status code, it must also provide Vendor ID information by using either the PCIe or IANA Vendor define type. For details refer to Table 5.
bitfield32	**UpdateOptionFlagValues** 32 bits field, each bit represents the state of the update option flag that was requested via the SetUpdateOptions command [31:1 – reserved [0] – Request force update of component – Set to ‘1’ if the FD will perform a force update of the component

GetStatus is provided to poll the status of the FD controller. The timeout waiting for ProgressPercent change is defined by UA_T3. When the UA does not see a change in the ProgressPercent after waiting for UA_T3 time, then the UA can send CancelUpdateComponent command to cancel the component update

11.13 CancelUpdateComponent command format

During the firmware component transfer process, the UA may send this command to the FD. The FD, upon receiving this command must stop sending RequestFirmwareData commands to the UA, and cancel the current component update procedure. The FD controller must transition to the READY XFER state of update mode and be ready to accept another UpdateComponent command. The UA may attempt to resend the same component image to the UA.

It is strongly recommended that the entire firmware update procedure be performed as a single sequence of events and not cancelled by the UA. This specification does not describe or provide guidance on a

914 recovery procedure if the FD operation is affected by a partially transferred image. After canceling the
915 update, the FD may not be able to operate normally if only a portion of the firmware update has been
916 completed.

917 **Table 26 – CancelUpdateComponent command format**

Type	Request data
--	No request data
Type	**Response data**
enum8	**completionCode** value: { PLDM_BASE_CODES, NOT_IN_UPDATE_MODE, BUSY_IN_BACKGROUND }

918 Error completion codes handling:

919 • NOT_IN_UPDATE_MODE: returned by the FD if it's not currently in update mode.

920 • BUSY_IN_BACKGROUND: returned by the FD if there is a critical job in the background, and
921 cannot exit from update mode. The UA shall retry after UA_T1

922 11.14 CancelUpdate command format

923 This command signals to the FD that it should exit from update mode even if activation is required to
924 begin operating at the new firmware level. The UA should always attempt to complete the transfer of all
925 components and use this command only if it determines that there is no other method to continue with the
926 transfer process. The FD will provide a response field which indicates which components will be in a non-
927 functioning state upon exit of update mode and subsequent external activation, such as an initialization of
928 the FD. This will depend on the FD's capability to recover from failed component updates. The indication
929 will allow the UA to understand when a failed FD update results in a non-functioning component state
930 which may require recovery actions (outside the scope of this specification) to place the component into a
931 functioning state.

932 It is strongly recommended that the entire firmware update procedure be performed as a single sequence
933 of events and not cancelled by the UA. This specification does not describe or provide guidance on a
934 recovery procedure if the FD operation is affected by a partially transferred image. After canceling the
935 update, the FD may not be able to operate normally if only a portion of the firmware update has been
936 completed.

937 **Table 27 – CancelUpdate command format**

Type	Request data
--	No request data
Type	**Response data**
enum8	**completionCode** value: { PLDM_BASE_CODES, NOT_IN_UPDATE_MODE, BUSY_IN_BACKGROUND }
bool8	**Non-functioning component indication** True: one or more components will be in a non-functioning state upon the next activation. The non-functioning component bitmap field indicates which components will be non-functioning. False: all components will be functioning. GetFirmwareParameters can be used to determine the individual component version information
Bitfield64	**Non-functioning component bitmap** This field is valid only if the Non-functioning component indication field is set to True. Each bit n corresponds to the nth component passed in the PassComponentTable command. A set bit indicates the component will be in a non-functioning state upon the next activation.

Error completion codes handling:

- NOT_IN_UPDATE_MODE: returned by the FD if it's not in the update mode.
- BUSY_IN_BACKGROUND: returned by the FD if there are critical tasks already being performed by the device, and cannot exit from update mode. The UA shall retry within UA_T1 interval

12 Additional requirements

12.1 Transport protocol type supported

PLDM can support bindings over multiple interfaces, refer to DSP0245 for the complete list. This specification requires the transport protocol type to support asynchronous request/response messages which can be sent from either endpoint in order to support the full Firmware Update functionality. All transport protocol types can be supported for the two Inventory commands defined in Table 8.

12.2 Considerations for FD device manufacturers

This specification does not provide a direct recovery method for when the update process is interrupted by power loss, interface failures, or unplanned reboots. An FD device manufacturer can look to minimize the exposure to these types of events by implementing a dual bank approach for firmware components. By using a dual bank approach, the new component data being updated is placed into a 'backup' image location and the FD device would continue to use the actively running image location until an ActivateFirmware command has been received. At that point the FD device will enable the new image to become the active running image at the next activation. If a power loss or interruption occurred prior to receiving the ActivateFirmware command the FD device would continue to use actively running image and the UA can subsequently restart the firmware update process to update all components again.

ANNEX A
(informative)

Change log

Version	Date	Description
1.0.0	2016-08-23	Work in Progress

Bibliography

DMTF DSP4004, *DMTF Release Process 2.4,*
http://dmtf.org/sites/default/files/standards/documents/DSP4004_2.4.pdf