JEDEC STANDARD

Universal Flash Storage (UFS)

Version 2.1

JESD220C

(Revision of JESD220B, September 2013)

MARCH 2016

JEDEC SOLID STATE TECHNOLOGY ASSOCIATION

NOTICE

JEDEC standards and publications contain material that has been prepared, reviewed, and approved through the JEDEC Board of Directors level and subsequently reviewed and approved by the JEDEC legal counsel.

JEDEC standards and publications are designed to serve the public interest through eliminating misunderstandings between manufacturers and purchasers, facilitating interchangeability and improvement of products, and assisting the purchaser in selecting and obtaining with minimum delay the proper product for use by those other than JEDEC members, whether the standard is to be used either domestically or internationally.

JEDEC standards and publications are adopted without regard to whether or not their adoption may involve patents or articles, materials, or processes. By such action JEDEC does not assume any liability to any patent owner, nor does it assume any obligation whatever to parties adopting the JEDEC standards or publications.

The information included in JEDEC standards and publications represents a sound approach to product specification and application, principally from the solid state device manufacturer viewpoint. Within the JEDEC organization there are procedures whereby a JEDEC standard or publication may be further processed and ultimately become an ANSI standard.

No claims to be in conformance with this standard may be made unless all requirements stated in the standard are met.

Inquiries, comments, and suggestions relative to the content of this JEDEC standard or publication should be addressed to JEDEC at the address below, or refer to www.jedec.org under Standards and Documents for alternative contact information.

Published by
©JEDEC Solid State Technology Association 2016
3103 North 10th Street
Suite 240 South
Arlington, VA 22201-2107

PRICE: Contact JEDEC

UNIVERSAL FLASH STORAGE (UFS)

Contents

Figures

Tables

Foreword

This standard has been prepared by JEDEC. The purpose of this standard is definition of a UFS Universal Flash Storage electrical interface and a UFS memory device. This standard defines a unique UFS feature set and includes the feature set of eMMC standard as a subset. This standard references several other standard specifications by MIPI (M-PHY and UniPro specifications) and INCITS T10 (SBC, SPC and SAM draft standards) organizations.

Introduction

The UFS electrical interface is a universal serial communication bus which can be utilized for different types of applications. It's based on MIPI M-PHY specification as physical layer for optimized performance and power. The UFS architectural model references the INCITS T10 SAM model for ease of adoption.

The UFS device is a universal data storage and communication media. It is designed to cover a wide area of applications as smart phones, cameras, organizers, PDAs, digital recorders, MP3 players, internet tablets, electronic toys, etc.

UNIVERSAL FLASH STORAGE (UFS)

(From JEDEC Board Ballot JCB-16-10, formulated under the cognizance of the JC-64.1 Subcommittee on Electrical Specifications and Command Protocols, Item 133.00B)

1 SCOPE

This standard specifies the characteristics of the UFS electrical interface and the memory device. Such characteristics include (among others) low power consumption, high data throughput, low electromagnetic interference and optimization for mass memory subsystem efficiency. The UFS electrical interface is based on an advanced differential interface by MIPI M-PHY specification which together with the MIPI UniPro specification forms the interconnect of the UFS interface. The architectural model is referencing the INCITS T10 (SCSI) SAM standard and the command protocol is based on INCITS T10 (SCSI) SPC and SBC standards.

2 NORMATIVE REFERENCE

The following normative documents contain provisions that, through reference in this text, constitute provisions of this standard. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply. However, parties to agreements based on this standard are encouraged to investigate the possibility of applying the most recent editions of the normative documents indicated. For undated references, the latest edition of the normative document referred to applies.

[MIPI-M-PHY], *MIPI Alliance Specification for M-PHY®, Version 3.00.00*

[MIPI-UniPro], *MIPI Alliance Specification for Unified Protocol (UniPro℠), Version 1.6.00*

[MIPI-DDB], *MIPI Alliance Specification for Device Descriptor Block (DDB), Version 1.0*

[SAM], *INCITS T10 draft standard: SCSI Architecture Model – 5 (SAM–5), Revision 05*, 19 May 2010

[SPC], *INCITS T10 draft standard: SCSI Primary Commands – 4 (SPC-4), Revision 27*, 11 October 2010

[SBC], *INCITS T10 draft standard: SCSI Block Commands – 3 (SBC–3), Revision 24*, 05 August 2010

[*JESD8-12A*], *1.2 V +/- 0.1V (Normal Range) and 0.8 - 1.3 V (Wide Range) Power Supply Voltage and Interface Standard for Nonterminated Digital Integrated Circuits*

[HBM-MM], *JEDEC Recommended ESD Target Levels for HBM/MM Qualification, JEP155A.01, March 2012*

[CDM], *JEDEC Recommended ESD-CDM Target Levels, JEP157, October 2009*

[HMAC-SHA], *Eastlake, D. and T. Hansen, US Secure Hash Algorithms (SHA and HMAC-SHA), RFC 4634, July 2006.*

3 TERMS AND DEFINITIONS

For the purposes of this standard, the terms and definitions given in the document included in clause 2, Normative Reference, and the following apply.

Application Client: An entity that is the source of SCSI commands and task management function requests in the host.

Byte: An 8-bit data value with most significant bit labeled as bit 7 and least significant bit as bit 0.

Command Descriptor Block: The structure used to communicate commands from an application client to a device server. A CDB may have a fixed length of up to 16 bytes or a variable length of between 12 and 260 bytes.

Device ID: The bus address of a UFS device.

Device Server: An entity in the device that processes SCSI commands and task management functions.

Doubleword: A 32-bit data value with most significant bit labeled as bit 31 and least significant bit as bit 0.

Dword: 32-bit data value, a Doubleword.

Gigabyte: 1,073,741,824 or 2^{30} bytes.

Host: An entity or a device with the characteristics of a primary computing device that includes one or more SCSI initiator devices.

Initiator device: Within a transaction, the originator of a SCSI command request message to a target device.

Kilobyte: 1024 or 2^{10} bytes.

Logical Unit: A logical unit is an internal entity of a bus device that performs a certain function or addresses a particular space or configuration within a bus device.

Logical Unit Number: A numeric value that identifies a logical unit within a device

Megabyte: 1,048,576 or 2^{20} bytes.

Quadword: A 64-bit data value with most significant bit labeled as bit 63 and least significant bit as 0.

Segment: A specified number of sequentially addressed bytes representing a data structure or section of a data structure.

Segment ID: A 16-bit value that represents an index into a table or an address of a segment descriptor or simply an absolute value that is an element of an absolute address

SCSI Request Block: A data packet that contains a multi-byte SCSI command and additional contextual information needed to carry out the command operation. A SCSI Request Block is built by the host and is targeted at a particular bus device.

Target device: Within a transaction, the recipient of a SCSI command request message from an initiator device.

Task: A task is a SCSI command which includes all transactions to complete all data transfers and a status response that will satisfy the requirements of the requested services of the command.

Transaction: A UFS primitive action which results in transmission of serial data packets between a target device and initiator device.

3 Terms and Definitions (cont'd)

Terabyte: 1.099.511.627.776 or 2^{40} bytes.

UFS Protocol Information Unit: Information transfer (communication) between a UFS host and device is done through messages which are called UFS Protocol Information Units. These messages are UFS defined data structures that contain a number of sequentially addressed bytes arranged as various information fields.

Unit: A bus device

Unit Attention: A condition of a bus device utilizing the SCSI protocol where it needs to be serviced before it can continue processing requests and responses.

Word: A 16-bit data value with most significant bit labeled as bit 15 and least significant bit as bit 0.

3.1 Acronyms

CDB	Command Descriptor Block
CPort	A CPort is a Service Access Point on the UniPro Transport Layer (L4) within a Device that is used for Connection-oriented data transmission
DMA	Direct Memory Access
DSC	Digital Still Camera
FFU	Field Firmware Update
GB	Gigabyte
HCI	Host Controller Interface
IID	Initiator ID
KB	Kilobyte
LUN	Logical Unit Number
MB	Megabyte
MIPI	Mobile Industry Processor Interface
MP3	MPEG-2 Audio Layer 3
NA	Not applicable
NU	Not used
PDU	Protocol Data Unit
PLL	Phase-Locked Loop
PMP	Portable media player
PSA	Production State Awareness
PWM	Pulse Width Modulation
RFU	Reserved for future use
RPMB	Replay Protected Memory Block

3.1 Acronyms (cont'd)

SBC	SCSI Block Commands
SID	Segment ID
SDU	Service Data Unit
SPC	SCSI Primary Commands
TB	Terabyte
T_PDU	MIPI Unipro Protocol Data Unit
T_SDU	MIPI Unipro protocol Service Data Unit
UFS	Universal Flash Storage
UMPC	Ultra-Mobile PC
UniPro	Unified Protocol
UPIU	UFS Protocol Information Unit
UTP	UFS Transport Protocol

3.2 Conventions

This standard follows some conventions used in SCSI documents since it adopts several SCSI standards.

A binary number is represented in this standard by any sequence of digits consisting of only the Western-Arabic numerals 0 and 1 immediately followed by a lower-case b (e.g., 0101b). Spaces may be included in binary number representations to increase readability or delineate field boundaries (e.g., 0 0101 1010b).

A hexadecimal number is represented in this standard by any sequence of digits consisting of only the Western-Arabic numerals 0 through 9 and/or the upper-case English letters A through F immediately followed by a lower-case h (e.g., FA23h). Spaces may be included in hexadecimal number representations to increase readability or delineate field boundaries (e.g., B FD8C FA23h).

A decimal number is represented in this standard by any sequence of digits consisting of only the Western-Arabic numerals 0 through 9 not immediately followed by a lower-case b or lower-case h (e.g., 25).

A range of numeric values is represented in this standard in the form "a to z", where a is the first value included in the range, all values between a and z are included in the range, and z is the last value included in the range (e.g., the representation "0h to 3h" includes the values 0h, 1h, 2h, and 3h).

When the value of the bit or field is not relevant, x or xx appears in place of a specific value.

The first letter of the name of a Flag is a lower-case f (e.g., fMyFlag).

The first letter of the name of a parameter included in a Descriptor or the first letter of the name of an Attribute is:

- a lower-case b if the parameter or the Attribute size is one byte (e.g., bMyParameter),
- a lower-case w if the parameter or the Attribute size is two bytes (e.g., wMyParameter),
- a lower-case d if the parameter or the Attribute size is four bytes (e.g., dMyParameter),
- a lower-case q if the parameter or the Attribute size is eight bytes (e.g., qMyParameter).

3.3 Keywords

Several keywords are used to differentiate levels of requirements and options, as follow:

Can - A keyword used for statements of possibility and capability, whether material, physical, or causal (*can* equals *is able to*).

Expected - A keyword used to describe the behavior of the hardware or software in the design models assumed by this standard. Other hardware and software design models may also be implemented.

Ignored - A keyword that describes bits, bytes, quadlets, or fields whose values are not checked by the recipient.

Mandatory - A keyword that indicates items required to be implemented as defined by this standard.

May - A keyword that indicates a course of action permissible within the limits of the standard (*may* equals *is permitted*).

Must - The use of the word *must* is deprecated and shall not be used when stating mandatory requirements; *must* is used only to describe unavoidable situations.

Obsolete - A keyword indicating that an item was defined in prior standards but has been removed from this standard.

Optional - A keyword that describes features which are not required to be implemented by this standard. However, if any optional feature defined by the standard is implemented, it shall be implemented as defined by the standard.

Reserved - A keyword used to describe objects—bits, bytes, and fields—or the code values assigned to these objects in cases where either the object or the code value is set aside for future standardization. Usage and interpretation may be specified by future extensions to this or other standards. A reserved object shall be zeroed or, upon development of a future standard, set to a value specified by such a standard. The recipient of a reserved object shall not check its value. The recipient of a defined object shall check its value and reject reserved code values.

Shall - A keyword that indicates a mandatory requirement strictly to be followed in order to conform to the standard and from which no deviation is permitted (*shall* equals *is required to*). Designers are required to implement all such mandatory requirements to assure interoperability with other products conforming to this standard.

Should - A keyword used to indicate that among several possibilities one is recommended as particularly suitable, without mentioning or excluding others; or that a certain course of action is preferred but not necessarily required; or that (in the negative form) a certain course of action is deprecated but not prohibited (*should* equals *is recommended that*).

Will - The use of the word *will* is deprecated and shall not be used when stating mandatory requirements; *will* is only used in statements of fact.

3.4 Abbreviations

etc. - And so forth (Latin: et cetera)

e.g. - For example (Latin: exempli gratia)

i.e. - That is (Latin: id est)

4 INTRODUCTION

Universal Flash Storage (UFS) is a simple, high performance, mass storage device with a serial interface. It is primarily for use in mobile systems, between host processing and mass storage memory devices. The following is a summary of the UFS device features.

4.1 General Features

- Target performance
  - High speed GEARs (1)
    - Support for GEAR1 is mandatory
    - Support for GEAR2 is mandatory
    - Support for GEAR3 is optional
- Target host applications
  - Mobile phone, UMPC, DSC, PMP, MP3 and any other applications that require mass storage, bootable mass storage, and external card
- Target device types
  - External card
  - Embedded device
    - Mass storage and bootable mass storage
  - Future expansion of device class types
    - I/O devices, camera, wireless, … , etc.
- Topology
  - One device per UFS port.
- UFS Layering
  - UFS Command Set Layer (UCS)
    - Simplified SCSI command set based on SBC and SPC. UFS will not modify these SBC and SPC Compliant commands. Option for defining UFS Native command and future extension exist.
  - UFS Transport Protocol Layer (UTP)
    - JEDEC to define the supported protocol layer, i.e., UTP for SCSI. This does not exclude the support of other protocol in UFS Transport Protocol Layer.
  - UFS Interconnect Layer (UIC)
    - MIPI UniPro^{SM} [MIPI-UniPro] is adopted for data link layer
    - MIPI M-PHY ^{SM} [MIPI-M-PHY] is adopted for physical layer

NOTE 1 See 6.4.1 for details.

4.2 Interface Features

- Three power supplies
  - VCCQ power supply: 1.2 V (nominal)
  - VCCQ2 power supply: 1.8 V (nominal)
  - VCC power supply: 1.8 V/3.3 V (nominal)
- Signaling as defined by [MIPI-M-PHY]
  - 400 mVp/240 mVp (not terminated),
  - 200 mVp/120 mVp (terminated)
- 8b10b line coding, as defined by MIPI M-PHY
- High reliability – BER under 10^{-10}
- Two signaling schemes
  - Low-speed mode with PWM signaling scheme
  - High-Speed burst mode
- Multiple gears defined for both Low-Speed and High-Speed mode

4.3 Functional Features

UFS functional features are NAND management features. These include

- Similar functional features as eMMC
- Boot Operation Mode
- Multiple logical units with configurable characteristics
- Replay Protected Memory Block (RPMB)
- Reliable write operation
- Background operations
- Secure operations, Purge and Erase to enhance data security
- Write Protection options, including Permanent and Power-On Write Protection
- Signed access to a Replay Protected Memory Block
- HW Reset Signal
- Task management operations
- Power management operations

5 UFS ARCHITECTURE OVERVIEW

5.1 UFS Top Level Architecture

Figure 5-1 shows the Universal Flash Storage (UFS) top level architecture.

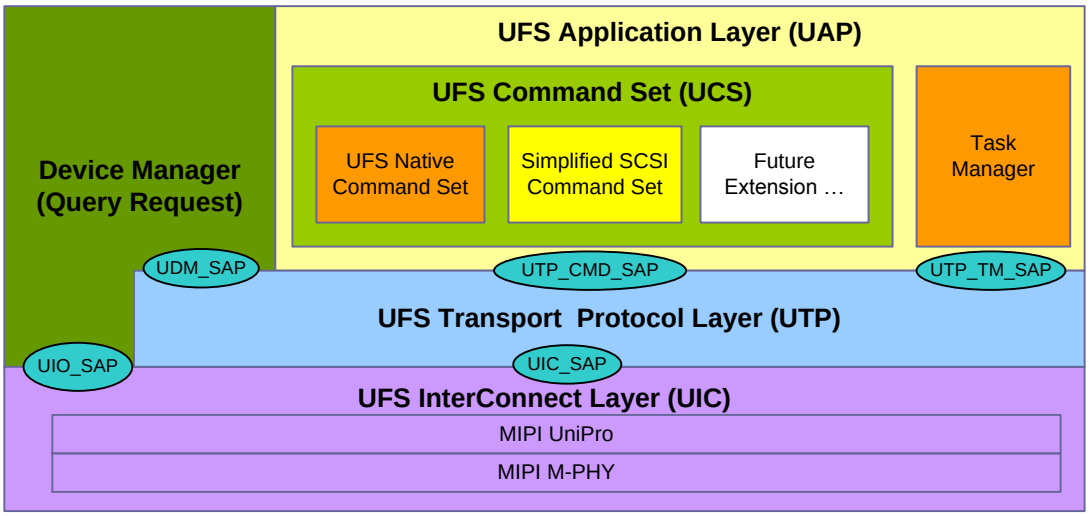


Figure 5-1 — UFS Top Level Architecture

UFS communication is a layered communication architecture. It is based on SCSI SAM architectural model [SAM].

5.1.1 Application Layer

The application layer consists of the UFS command set (UCS), the device manager and the Task Manager. The UCS will handle the normal commands like read, write, and so on. UFS may support multiple command sets. UFS is designed to be protocol agnostic. The command set for this version UFS standard is based on SCSI command set. In particular, a simplified SCSI command set was selected for UFS. UFS Native command set can be supported when it is needed to extend the UFS functionalities.

The Task Manager handles commands meant for command queue control. The Device Manager will provide device level control like Query Request and lower level link-layer control.

5.1.2 UFS Device Manager

The device manager has the following two responsibilities:

- Handling device level operations.
- Managing device level configurations.

Device level operations include functions such as device power management, settings related to data transfer, background operations enabling, and other device specific operations.

Device level configuration is managed by the device manager by maintaining and storing a set of descriptors. The device manager handles commands like query request which allow to modify or retrieve configuration information of the device.

5.1.3 Service Access Points

As seen from the diagram the device manager interacts with lower layers using the following two service access points

- UDM_SAP
- UIO_SAP

UDM_SAP is the service access point exposed by the UTP for the device manager to allow handling of device level operations and configurations. For example the handling of query request for descriptors would be done using this service access point. Figure 5-2 depicts the usage of the service access point.

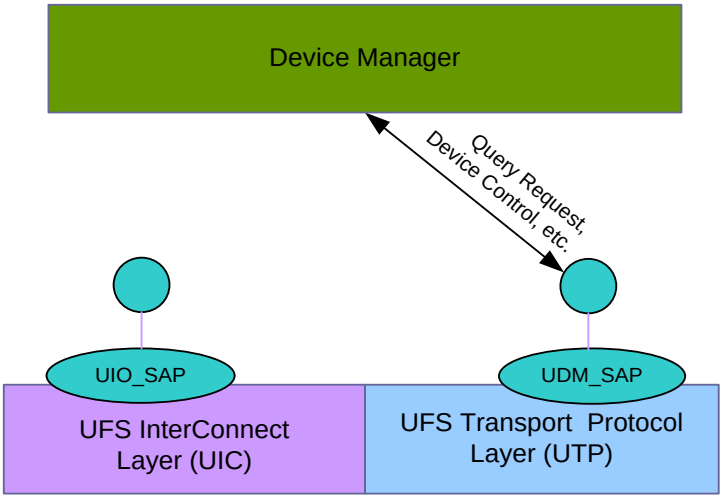


Figure 5-2 — Usage of UDM_SAP

UIO_SAP is the service access point exposed by the UIC layer for the device manager to trigger the reset of the UIC layer and to transfer requests and responses related to UIC management functions. Figure 5-3 depicts the usage of the service access point.

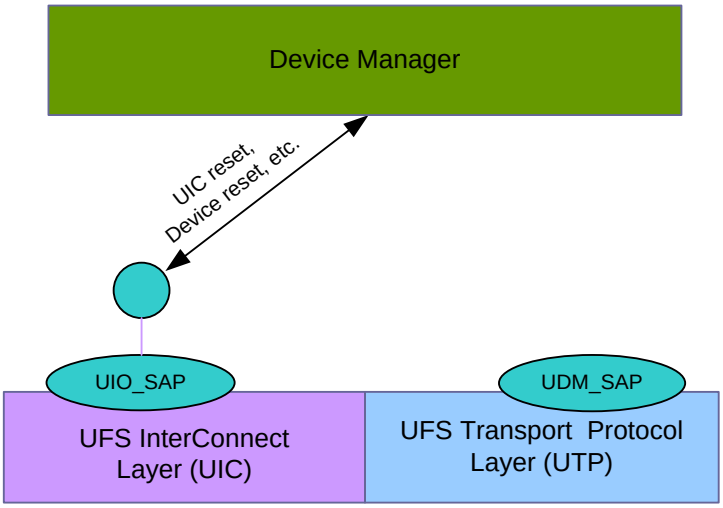


Figure 5-3 — Usage of UIO_SAP

5.1.4 UIO_SAP

UIO_SAP is the service access point exposed by the UIC layer. In UniPro, UIO_SAP corresponds to DME_SAP. The DME_SAP provides service primitives including one for resetting the entire UniPro protocol stack and one for UFS device reset, etc.

- DME_RESET : It is used when the UniPro stack has to be reset.
- DME_ENDPOINTRESET: It is used when UFS host wants the UFS device to perform a reset.

For the detailed internal mechanism, refer the UniPro specification [MIPI-UniPro] released by MIPI (MIPI is Mobile Industry Processor Interface).

5.1.5 UDM_SAP

UDM_SAP is the service access point exposed by the UTP layer to the Device Manager for UFS device level functions. UDM_SAP corresponds to the Query Request and Query Response functions defined by the UFS UTP layer.

For further details refer to the following subclauses: 10.9.9, Query Function transport protocol services, 10.7.8, QUERY REQUEST UPIU, and 10.7.9, QUERY RESPONSE UPIU.

5.1.7 UFS Transport Protocol Layer

The UFS Transport Protocol (UTP) layer provides services for the higher layer . UPIU is “UFS Protocol Information Unit” which is exchanged between UTP layers of UFS host and UFS device. For example, if host side UTP receives the request from application layer or Device Manager, UTP generates a UPIU for that request and transports the generated UPIU to the peer UTP in UFS device side. The UTP layer provides the following three access points.

1) UFS Device Manager Service Access Point (UDM_SAP) to perform the device level management like descriptor access.
2) UTP Command Service Access Point (UTP_CMD_SAP) to transport commands.
3) UTP Task Management Service Access Point (UTP_TM_SAP) to transport task-management function like “abort task” function.

5.1.8 UFS Interconnect Layer

The lowest layer is UFS Interconnect Layer (UIC) which handles connection between UFS host and UFS device. UIC consists of MIPI UniPro and MIPI M-PHY. The UIC provides two service access points to upper layer. The UIC Service Access Point (UIC_SAP) to transport UPIU between UFS host and UFS device. The UIC_SAP corresponds to T_SAP in UniPro. The UIC IO control Service Access Point (UIO_SAP) to manage UIC. The UIO_SAP corresponds to DME_SAP in UniPro.

5.1.9 UFS Topology

This version of the standard assumes that only one device is connected to a UFS port. Other topologies may be defined in future versions of the standard.

5.2 UFS System Model

Figure 5-4 shows an example of UFS system. It shows how a UFS host is connected to a UFS device, the position of UFS host controller and its related UFS HCI interface.

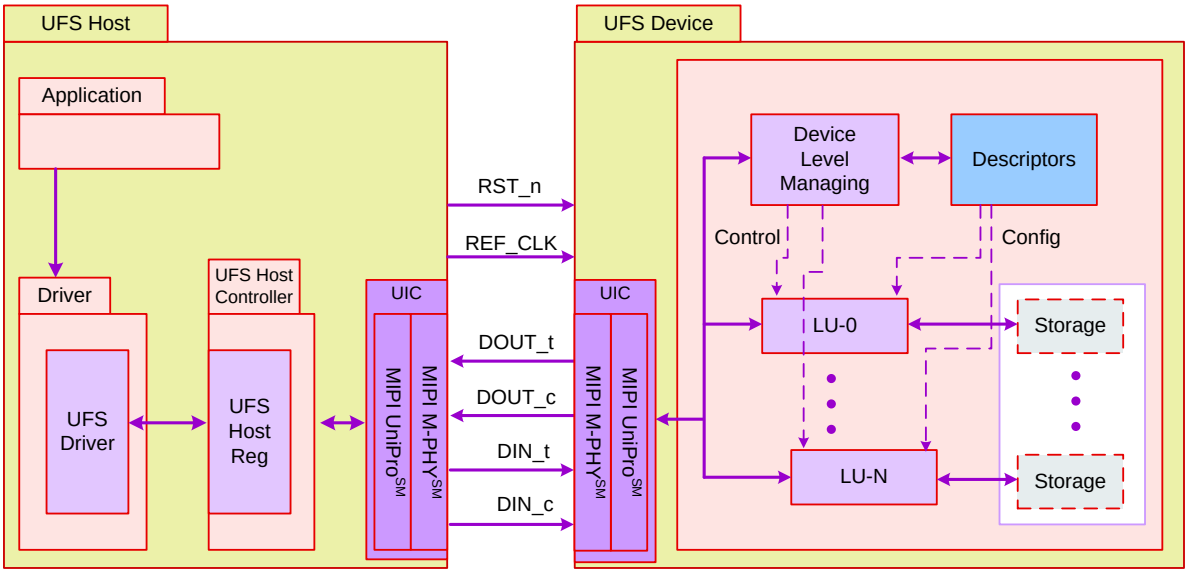


Figure 5-4 — UFS System Model

The UFS host consists of the application which wishes to communicate with the UFS device. It communicates with the device using the UFS driver. The UFS driver is meant for managing the UFS host controller through the UFS HCI (UFS Host Controller Interface). The UFS HCI is basically a set of registers exposed by the host controller.

Figure 5-4 also indicates the UFS interface between the UFS host and the UFS device. The UFS Interconnect (UIC) Layer consists of MIPI UniPro and MIPI M-PHY. The physical layer M-PHY is differential, dual simplex PHY that includes TX and RX pairs.

Potential UFS devices can be memory card (full size and micro size), embedded bootable mass storage devices, IO devices, etc. A UFS device is comprised of multiple logical units, a device manager and descriptors. The device manager performs device level functions like power management while the logical unit performs functions like read, write etc. The descriptors are meant for storage of configuration related information.

5.3 System Boot and Enumeration

The system boot from a bootable UFS device will initiate after power up when the UFS InterConnect Layer (MIPI M-PHY and UniPro) has completed its boot sequence. The boot code can be read from the appropriate boot logical unit, or as desired, boot ROM code can re-configure MIPI M-PHY and UniPro to appropriate setting before reading the boot code.

Multiple boot logical units may be available in a UFS device. However, only one boot logical unit will be active at power-up. Appropriate descriptors are available to configure the boot process.

During boot, accesses to boot logical unit are supported via SCSI commands.

5.4 UFS Interconnect (UIC) Layer

UFS interconnect layer is composed by MIPI UniPro, which provides basic transfer capabilities to the upper layer (UTP), and MIPI M-PHY, adopted as UFS physical layer.

5.4.1 UFS Physical Layer Signals

The UFS physical layer defines the physical portion of the UFS interface that connects UFS device and UFS host. This is based on MIPI M-PHY specification. UFS interface can support multiple lanes in each direction. Each lane consists of a differential pair. Basic configuration is based on one transmit lane and one receive lane.

Optionally, a UFS device may support two downstream lanes and two upstream lanes. An equal number of downstream and upstream lanes shall be provided in each link.

Table 5-1 summarizes the signals required for a UFS device. Only the single lane, per direction, per link, configuration is shown. See clause 6, UFS Electrical: Clock, Reset, Signals and Supplies, and clause 8, UFS UIC Layer: MIPI M-PHY, for full details about UFS signals.

Table 5-1 — UFS Signals

Name	Type	Description
REF_CLK	Input	Reference clock Relatively low speed clock common to all UFS devices in the chain, used as a reference for the PLL in each device.
DIN_t DIN_c	Input	Downstream lane input Differential input true and complement signal pair.
DOUT_t DOUT_c	Output	Upstream lane output Differential output true and complement signal pair.
RST_n	Input	Reset UFS Device hardware reset signal

5.4.2 MIPI UniPro

In UFS, UniPro is responsible for management of the link, including the PHY.

NOTE Device management is outside the scope of the interconnect layer and is the responsibility of the upper layers.

The basic interface to the interconnect layer is UniPro definition of a CPort. CPort is used for all data transfer as well as all control and configuration messages. In general, multiple CPorts can be supported on a device and the number of CPorts is implementation dependent.

Traffic sent over UniPro link can be classified as TC0 or TC1 traffic class with TC1 as higher priority traffic class. This version of UFS standard only uses a single CPort and TC0 traffic class.

UFS takes advantage of the basic types of UniPro services. These include data transfer service, and config/control/status service.

For more details, please refer to clause 9, UIC layer: MIPI UniPro, and MIPI UniPro specification [MIPI-UniPro].

5.4 UFS Interconnect (UIC) Layer (cont'd)

5.4.3 MIPI UniPro Related Attributes

In general the UniPro related attributes, values and use of them are defined in the MIPI UniPro specification. The attributes may be generic for all UniPro applications and thus out of scope of this document. Following attributes are defined in this standard specifically for UFS application as indicated in Table 5-2.

Table 5-2 — ManufacturerID and DeviceClass Attributes

Attribute	AttributeID(1)	Value	Description
DME_DDBL1_ManufacturerID	0x5003		MIPI manufacturer ID. MIPI MID shall be used in this Attribute also for UFS applications. The ID can be requested from MIPI.
DME_DDBL1_DeviceClass	0x5002	Memory = 0x02 Host = 0x03	UniPro DeviceClass ID for UFS application
NOTE 1 Reference MIPI Alliance Specification for Device Descriptor Block [MIPI-DDB]			

5.5 UFS Transport Protocol (UTP) Layer

As mentioned previously, the Transport Layer is responsible for encapsulating the protocol into the appropriate frame structure for the interconnect layer. UFS is protocol agnostic and thus any protocol will need the appropriate translation layer. For this version of UFS standard, this is UTP (UFS Transport Protocol) layer.

In this version of the standard, all accesses are supported only through SCSI, however additional API/service/extension may be added in future versions to introduce new features or address specific requirements.

A design feature of UTP is to provide a flexible packet architecture that will assist the UFS controller in directing the encapsulated command, data and status packets into and out of system memory. The intention is to allow the rapid transmittal of data between the host system memory and the UFS device with minimal host processor intervention. Once the data structures are set up in host memory and the target device is notified, the entire commanded transaction can take place between the UFS device and the host memory. The means by which the UFS controller transfers data into and out of host memory is via a hardware and/or firmware mechanism that is beyond the scope of this document. See the UFS controller standard for further information.

A second feature of the UTP design is that once a device receives a command request notification from the host, the device will control the pacing and state transitions needed to satisfy the data transfers and status completion of the request. The idea here is that the device knows its internal condition and state and when and how to best transfer the data that makes up the request. It is not necessary for the host system or controller to continuously poll the device for "ready" status or for the host to estimate when to start a packet transfer. The device will start the bus transactions when it determines its conditions and status are optimal. This approach cuts down on the firmware and logic needed within the host to communicate with a device. It also affords the maximum possible throughput with the minimum number of bus transactions needed to complete the operation.

5.5 UFS Transport Protocol (UTP) Layer (cont'd)

5.5.1 Architectural Model

The SCSI Architecture Model [SAM] is used as the general architectural model for UTP. The SAM architecture is a client-server model or more commonly a request-response architecture.

5.5.1.1 Client-Server Model

A client-server transaction is represented as a procedure call with inputs supplied by the application client on the Initiator device. The procedure call is processed by the server and returns outputs and a procedure call status. Client-server relationships are not symmetrical. A client only originates requests for service. A server only responds to such requests.

Initiator device and Target device are mapped into UFS physical network devices. An Initiator device may request processing of a command or a task management function through a request directed to the Target device. Device service requests are used to request the processing of commands and task manager requests are used to request the processing of task management functions.

Target device is a UFS device. A UFS device will contain one or more Logical Units. A Logical Unit is an independent processing entity within the device.

An Initiator request is directed to a single Logical Unit within a device. A Logical Unit will receive and process the client command or request. Each Logical Unit has an address within the Target device called a Logical Unit Number (LUN).

Communication between the Initiator device and Target device is divided into a series of messages. These messages are formatted into UFS Protocol Information Units (UPIU) as defined within this standard. There are a number of different UPIU types defined. All UPIU structures contain a common header area at the beginning of the data structure (lowest address). The remaining fields of the structure vary according to the type of UPIU.

A Task is a command or sequence of actions that perform a requested service. A Logical Unit contains a task queue that will support the processing of one or more Tasks. The Task queue is managed by the Logical Unit. A unique Task Tag is generated by the Initiator device when building the Task. This Task Tag is used by the Target device and the Initiator device to distinguish between multiple Tasks. All transactions and sequences associated with a particular Task will contain that Task Tag in the transaction associated data structures.

Command structures consist of Command Descriptor Blocks (CDB) that contain a command opcode and related parameters, flags and attributes. The description of the CDB content and structure are defined in detail in the [SAM], [SBC] and [SPC] INCITS T10 draft standards.

5.5.1.1 Client-Server Model (cont'd)

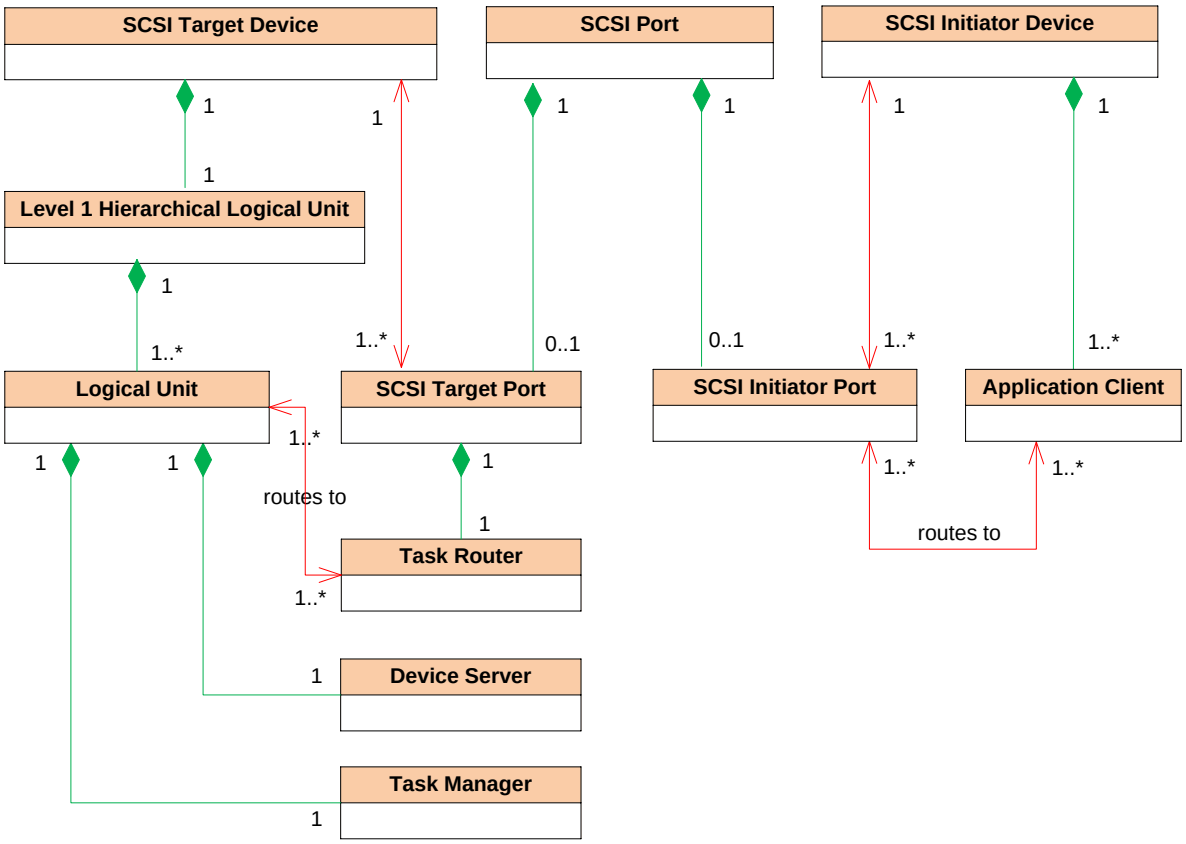


Figure 5-5 — SCSI Domain Class Diagram

5.5.1.1 Client-Server Model (cont'd)

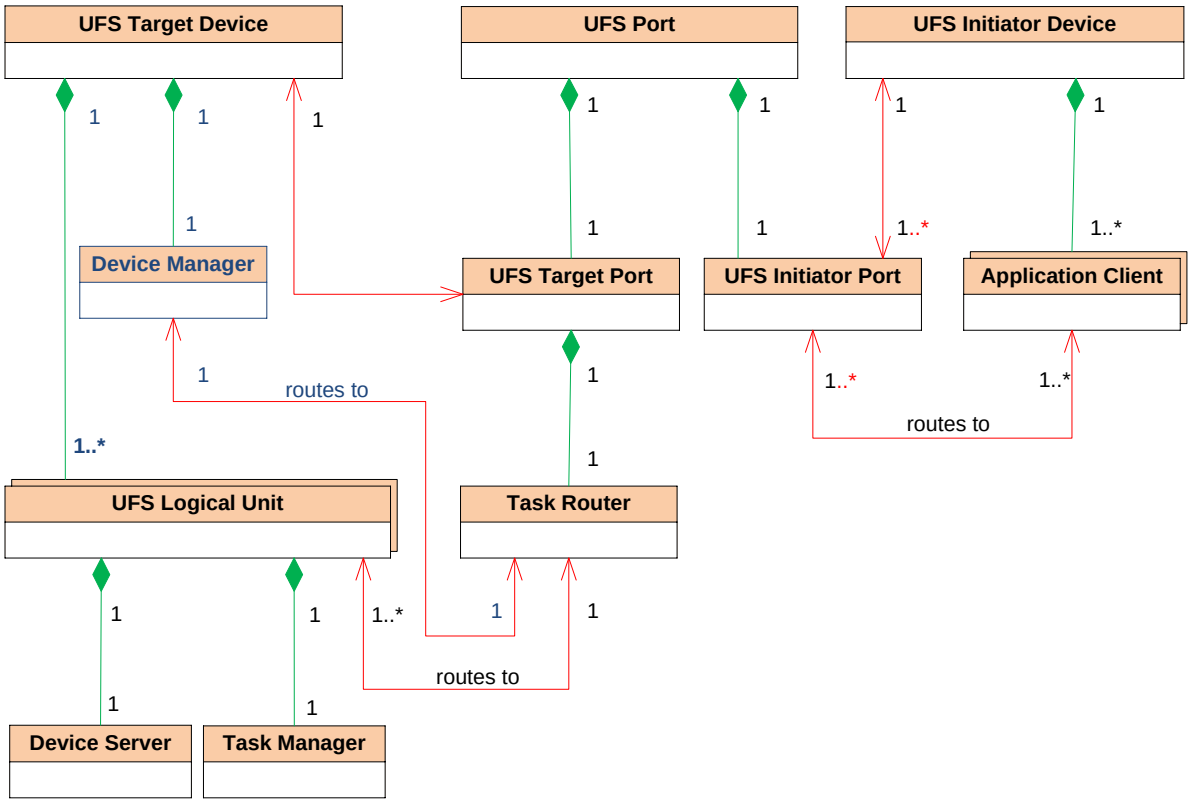


Figure 5-6 — UFS Domain Class Diagram

5.5.1.2 CDB, Status, Task Management

UTP adopts Command Descriptor Block (CDB) format for commands, device status data hierarchy and reporting method, and task management functions of outstanding commands, described in [SAM]. Regardless of the command protocol to be delivered by UTP, SCSI CDB, Status and Task Management Functions should be adopted uniformly in UFS devices.

5.5.1.3 Nexus

The nexus represents the relationship among the Initiator, Target, Logical Unit and Command (Task)

- Nexus notation: I_T_L_Q nexus; where I =Initiator, T = Target, L = Logical Unit, Q = Command

There shall be at least one initiator device in the UFS definition. There shall only one target device, the UFS device. There shall be one or more logical units in a UFS device. The command identifier (i.e., the Q in an I_T_L_Q nexus) is assigned by the Initiator device to uniquely identify one command in the context of a particular I_T_L nexus, allowing more than one command to be outstanding for the I_T_L nexus at the same time.

An overlapped command occurs when a task manager or a task router detects the use of a duplicate I_T_L_Q nexus in a command before that I_T_L_Q nexus completes its command lifetime (see [SAM]).

Concurrent overlapped commands are not allowed in UFS. Each command shall have an unique Task Tag. The UFS device is not required to detect overlapped commands.

5.5.1.4 SCSI Command Model

All command requests originate from application clients in an Initiator device. An application Client requests the processing of a command with the following procedure call:

- Service Response = Execute Command (IN (I_T_L_Q Nexus, CDB, Task Attribute, [Data-In Buffer Size], [Data-Out Buffer], [Data-Out Buffer Size], [CRN], [Command Priority]), OUT ([Data-In Buffer], [Sense Data], [Sense Data Length], Status, [Status Qualifier]))

Parameter fields in the UTP Command, Response, Ready-to-Transfer, Data-Out, Data-In UPIU headers contain the requisite information for the input and output arguments of the Execute Command procedure call in compliance with [SAM].

5.5.1.5 SCSI Task Management Functions

An application client requests the processing of a task management function with the following procedure call:

- Service Response = Function name (IN (Nexus), OUT ([Additional Response Information])

Parameters fields in the UTP Task Management Request and UTP Task Management Response headers contain the requisite information for the input and output arguments of the Task Management Function procedure call in compliance with [SAM].

5.6 UFS Application and Command Layer

UFS interface is designed to be protocol agnostic interface. However, as mentioned previously, SCSI has been selected as the baseline protocol layer for this standard. Descriptors are available to identify and select the appropriate protocol for UFS interface.

The primary functions of the Command Set Layer are to establish a method of data exchange between the UFS host and UFS device and to provide fundamental device management capability. SCSI SBC and SPC commands are the baseline for UFS. UFS will not modify the SBC and SPC Compliant commands. The goal is to maximize re-use and leverage of the software codebase available on platforms (PC, netbook, MID) that are already supporting SCSI.

Options are available to define UFS Native commands and extension as needed.

UFS SCSI command set includes:

1. SBC compliant commands [SBC]:
   - FORMAT UNIT
   - READ (6) and READ (10)
   - READ CAPACITY (10)
   - REQUEST SENSE
   - SEND DIAGNOSTIC
   - UNMAP
   - WRITE (6) and WRITE (10)
2. SPC compliant commands [SPC]:
   - INQUIRY
   - REPORT LUNS
   - READ BUFFER (optional)
   - TEST UNIT READY
   - WRITE BUFFER
   - SECURITY PROTOCOL IN and SECURITY PROTOCOL OUT
3. SCSI operational commands for UFS applications and compatible with existing SCSI driver
   - MODE SELECT (10) and MODE SENSE (10)
   - PRE-FETCH (10)
   - START STOP UNIT
   - SYNCHRONIZE CACHE (10)
   - VERIFY (10)
4. Value-added optional commands for UFS:
   - READ (16), WRITE(16), PRE-FETCH (16), SYNCHRONIZE CACHE (16), and READ CAPACITY(16).

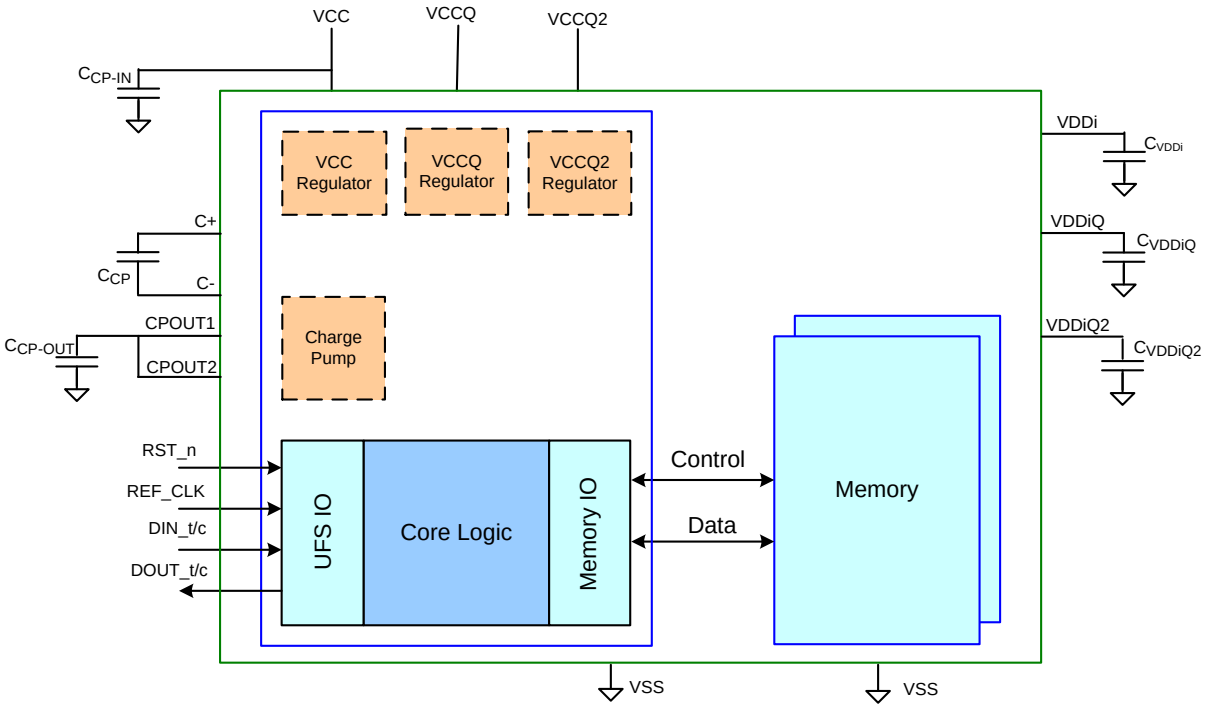
     NOTE These commands support logical units with larger capacities having an 8-byte LBA field.

Refer to clause 11, UFS Application (UAP) Layer – SCSI Commands, for more details.

6 UFS ELECTRICAL: CLOCK, RESET, SIGNALS AND SUPPLIES

6.1 UFS Signals

Figure 6-1 represents a conceptual drawing of UFS device. Utilization of internal regulators and connection of those to different parts of the sub-system may differ per implementation.



NOTE 1 The memory core power supply may be connected to VCC power supply ball, or the VCC regulator output, while it is connected to the charge pump output if VCC =1.8 V and the memory requires 3.3 V core power supply.

NOTE 2 The memory IO may consume power from any power supply: VCC, VCCQ or VCCQ2.

NOTE 3 C_{CP-IN}, C_{CP} and C_{CP-OUT} may be required only when internal charge pump is used.

Figure 6-1 — UFS Device Block Diagram

6.1 UFS Signals (cont'd)

Table 6-1 — Signal Name and Definitions

Name	Type	Description
VCC	Supply	Supply voltage for the memory devices
VCCQ	Supply	Supply voltage used typically for the memory controller and optionally for the PHY interface, the memory IO, and any other internal very low voltage block
VCCQ2	Supply	Supply voltage used typically for the PHY interface and the memory controller and any other internal low voltage block
VDDiQ (1)	Input	Input terminal to provided bypass capacitor for VCCQ internal regulator
VDDiQ2 (1)	Input	Input terminal to provide bypass capacitor for VCCQ2 internal regulator
VDDi (1)	Input	Input terminal to provide bypass capacitor for VCC internal regulator
VSS	Supply	Ground
RST_n	Input	Input hardware reset signal. This is an active low signal
REF_CLK	Input	Input reference clock. When not active, this signal should be pull-down or driven low by the host SoC.
Differential input signals into UFS device from the host		
DIN_t or DIN0_t (2) DIN_c or DIN0_c (2)	Input	Downstream data lane 0. DIN_t is the positive node of the differential signal.
DIN1_t (2), DIN1_c (2)	Input	Downstream data lane 1.
Differential output signals from the UFS device to the host		
DOUT_t or DOUT0_t (3) DOUT_c or DOUT0_c (3)	Output	Upstream data lane 0. DOUT_t is the positive node of the differential signal.
DOUT1_t (3), DOUT1_c (3)	Output	Upstream data lane 1.
C+	Input	Optional charge pump capacitor, positive terminal. For more information, please refer to 6.5
C-	Input	Optional charge pump capacitor, negative terminal. For more information, please refer to 6.5
CPOUT1, CPOUT2	Input	Optional Charge pump output capacitor terminal. For more information, please refer to 6.5

NOTE 1 If there is no internal regulator requiring output capacitor then VDDi pins should be internally connected as follows: VDDi to VCC, VDDiQ to VCCQ, and VDDiQ2 to VCCQ2.

NOTE 2 DIN0_t/_c and DIN1_t/_c apply if the device has two downstream lanes.

NOTE 3 DOUT0_t/_c and DOUT1_t/_c apply if the device has two upstream lanes.

NOTE 4 It is recommended to apply [HBM-MM] and [CDM] to all signals described in this table. This standard does not require use of the Machine Model for ESD qualification.

6.2 Reset Signal

To meet the requirements of the JEDEC Standard [JESD8-12A], the RST_n signal voltages shall be within the specified ranges for VCCQ in Table 6-2.

Table 6-2 — Reset Signal Electrical Parameters

Parameter	Symbol	Min	Max	Unit	Notes
Input HIGH voltage	VIH	0.65*VCCQ	VCCQ+0.3	V	For VCCQ as defined in Table 6-3
Input LOW voltage	VIL	VSS-0.3	0.35*VCCQ	V	For VCCQ as defined in Table 6-3
Input Capacitance	Cin		10	pF	
Input leakage Current	Ilkg		10	µA	

6.3 Power Supplies

Table 6-3 — UFS Power Supply Parameters

Parameter	Symbol	Min	Max	Unit	Notes
VCC DC operating range	VCC	2.7	3.6	V	
		1.70	1.95	V	
VCCQ DC operating range	VCCQ	1.1	1.3	V	1
VCCQ2 DC operating range	VCCQ2	1.70	1.95	V	
Supply Voltage power up timing for 3.3 V	tPRUH		35	ms	2
Supply Voltage power up timing for 1.8 V	tPRUL		25	ms	2
Supply Voltage power up timing for 1.2 V	tPRUV		20	ms	2
VCC internal regulator capacitor	C_{VDDi}	1		µF	
VCCQ internal regulator capacitor	C_{VDDiQ}	1		µF	
VCCQ2 internal regulator capacitor	C_{VDDiQ2}	1		µF	

NOTE 1 See [JESD8-12A].

NOTE 2 Power up timing starts when the supply voltage crosses 300 mV and ends when it reaches the minimum operating value.

6.3 Power Supplies (cont'd)

Figure 6-2 shows tPRUH, tPRUL and tPRUV timings for VCC = 3.3 V.

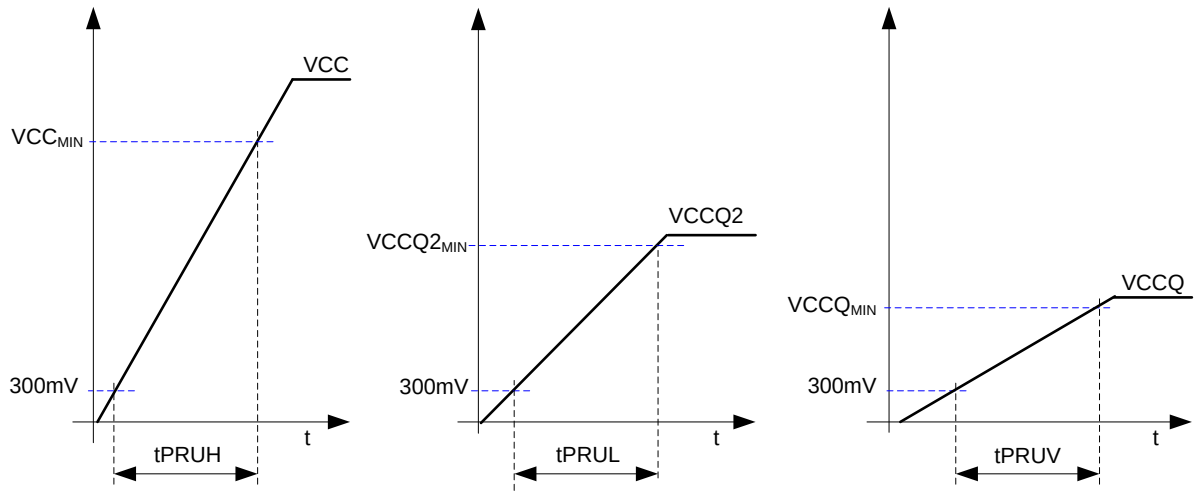


Figure 6-2 — Supply voltage power up timings

Table 6-4 defines the valid voltage configurations for an UFS device. UFS device shall support at least one of the valid voltage configurations, and it may optionally support more than one of them.

Table 6-4 — Voltage configurations for UFS

Power Supplies		VCCQ	VCCQ2
		1.1 V - 1.3 V	1.7 V - 1.95 V
VCC	2.7 V - 3.6 V	Valid	
	1.7 V - 1.95 V	Valid	

6.4 Reference Clock

The M-PHY specification defines the reference clock optional for the State Machine Type I [MIPI M-PHY]. As the PWM signaling is self-clocked the reference clock is not required for the data latching. Therefore, UFS devices shall be able to operate without reference clock in LS-MODE (LINE-CFG, SLEEP and PWM-BURST).

Still existence of the reference clock may be utilized to enable lower BER and faster HS-MODE PLL/DLL locking. Thus a UFS device shall implement a square wave single ended reference clock input and it requires the presence of a reference clock with the characteristics described in this section when operating in HS-MODE (STALL and HS-BURST). In order to avoid potential race conditions, it is recommended that such reference clock is already present when requesting a power mode change into Fast_Mode or FastAuto_Mode.

6.4 Reference Clock (cont'd)

Table 6-5 — Reference Clock parameters

Parameter	Symbol	Min	Max	Unit	Notes
Frequency	f_{ref}	19.2 26 38.4 52		MHz	1
Frequency Error	f_{ERROR}	-150	+150	ppm	
Input High Voltage	V_{IH}	0.65 * VCCQ		V	2
Input Low Voltage	V_{IL}		0.35 * VCCQ	V	2
Input Clock Rise Time	t_{IRISE}		2	ns	3
Input Clock Fall Time	t_{IFALL}		2	ns	3
Duty Cycle	t_{DC}	45	55	%	4
Phase Noise	N		-66	dBc	5
Noise Floor Density	$N_{density}$		-140	dBc/Hz	6
Input Impedance	RL_{RX}	100		kΩ	7
	CL_{RX}		5	pF	

NOTE 1 HS-BURST rates A and B are achieved with integer multipliers of f_{ref}.

NOTE 2 Figure 6-3 shows the input levels $V_{IL,MAX}$ to $V_{IH,MIN}$.

NOTE 3 Clock rise time and clock fall time shall be measured from 20% to 80% of the window defined by $V_{IL,MAX}$ to $V_{IH,MIN}$, see Figure 6-3.

NOTE 4 Clock duty cycle shall be measured at the crossings of the REF_CLK signal with the midpoint V_{MID}, defined as: $V_{MID} = (V_{IL,MAX} + V_{IH,MIN}) / 2$, see Figure 6-3.

NOTE 5 Integrated single side band phase noise from 50kHZ to 10MHz. This parameter refers to the random jitter only.

NOTE 6 White noise floor. This parameter refers to the random jitter only.

NOTE 7 RL_{RX} and CL_{RX} include Rx package and Rx input impedance.

bRefClkFreq attribute indicates to the device the frequency of the REF_CLK signal, and its default value corresponds to 26 MHz.

UFS device can operate in HS-MODE only if bRefClkFreq attribute indicates the correct REF_CLK frequency value. bRefClkFreq attribute can be written only if both sub-links are in LS-MODE.

The reference clock is not required and it may be turned off when both SUB-LINKs have reached and are operating in one of the following M-PHY states:

- LS-MODE (LINE-CFG, SLEEP or PWM-BURST state)
- HIBERN8 state

The reference clock shall be turned ON and stably running before initiation of the state transition to STALL from a LS-MODE (LINE-CFG or SLEEP) or from the HIBERN8 state.

6.4 Reference Clock (cont'd)

Figure 6-3 shows clock rise time and fall time measurements.

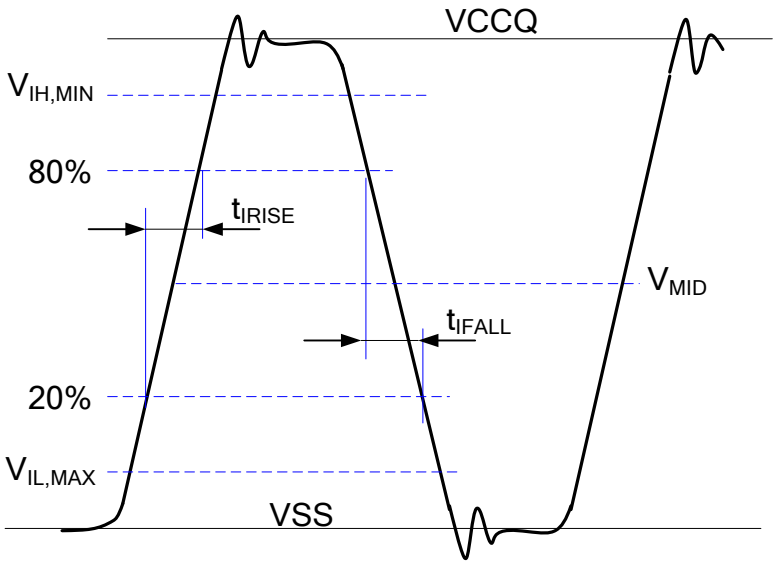


Figure 6-3 — Clock input levels, rise time, and fall time

6.4.1 HS Gear Rates

Table 6-6 defines the data rate values for the two rate series with respect REF_CLK frequency value (fref).

Table 6-6 — HS-BURST Rates

HS-GEAR	Rate A-series	Rate B-series		Rate A-series (from [MIPI-M-PHY])	Rate B-series[4] (from [MIPI-M-PHY])	Unit
	f_{ref}	f_{ref}		f_{ref}	f_{ref}	
	19.2 / 38.4 / 26 / 52	19.2 / 38.4	26 / 52	19.2 / 38.4 / 26 / 52		MHz
HS-GEAR1	1248[2]	1459.2	1456.0	1248	1457.6	Mbps
HS-GEAR2	2496	2918.4	2912.0	2496	2915.2	Mbps
HS-GEAR3[3]	4992	5836.8	5824.0	4992	5830.4	Mbps

NOTE 1 "Mbps" indicates 1000000 bit per sec.

NOTE 2 1248Mbps with fref = 38.4 MHz may be obtained using a prescaler. fref * M /P, M = 65 (PLL multiplier), P = 2 (Prescaler).

NOTE 3 Support for HS-GEAR3 is optional.

NOTE 4 The B-series rates shown are not integer multiples of common reference frequencies 19.2 MHz or 26 MHz, but are within the tolerance range of 2000 ppm.

6.4 Reference Clock (cont'd)

6.4.2 Host Controller requirements for reference clock generation

Table 6-7 — Host controller reference clock parameters

Parameter	Symbol	Min	Max	Unit	Notes
DC Output High Voltage	V_{OH}	0.75 * VCCQ		V	1, 2
DC Output Low Voltage	V_{OL}		0.25 * VCCQ	V	1, 2
Output Clock Rise Time	t_{ORISE}		2	ns	3
Output Clock Fall Time	t_{OFALL}		2	ns	3
Test Load Impedance	RL_{Test}	100		kΩ	1, 4, 5
	CL_{Test}	20		pF	1, 4, 5

NOTE 1 Output load resistive and capacitance component are defined as 20 pF shunted by 100 KΩ.

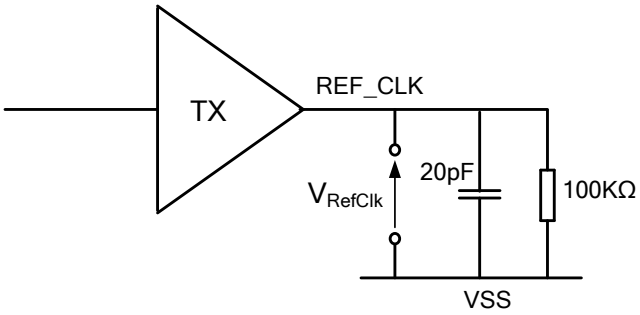


Figure 6-4 — Test Load Impedance

NOTE 2 Following graph shows Output driver and Input receiver levels. REF_CLK driver AC voltage (e.g., ring back) shall be kept inside Output Voltage limit.

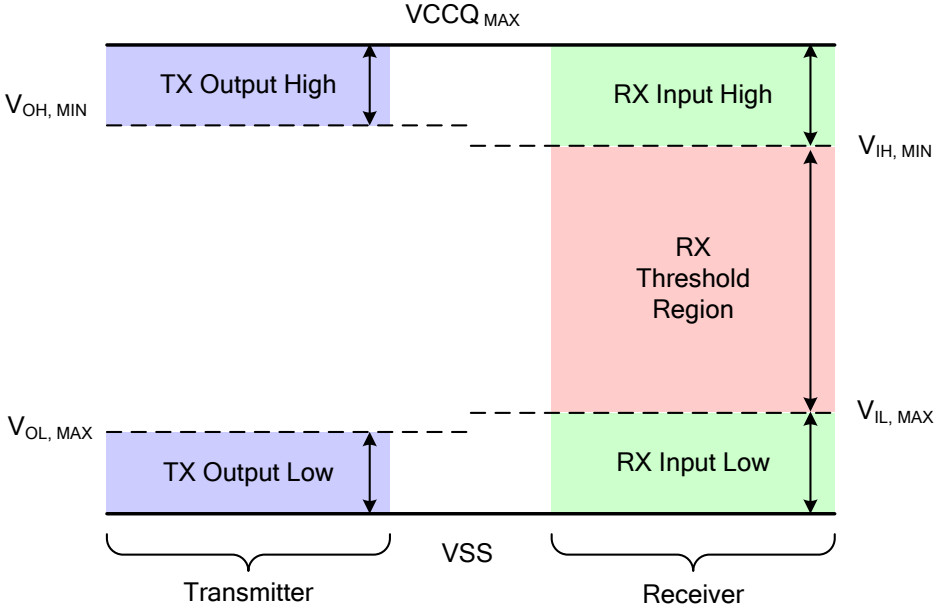


Figure 6-5 — Output driver and Input receiver levels

6.4.2 Host Controller requirements for reference clock generation (cont'd)

Table 6-7 — Host controller reference clock parameters (cont'd)

NOTE 3 Clock rise time and clock fall time shall be measured from 20% to 80% of the window defined by $V_{OL,MAX}$ and $V_{OH,MIN}$.

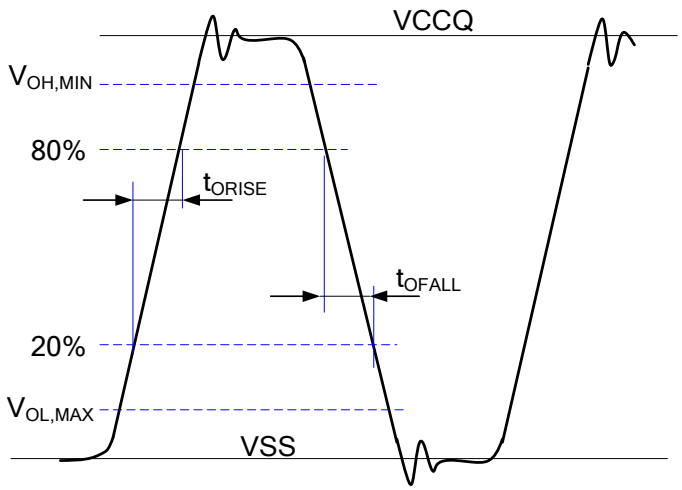


Figure 6-6 — Clock output levels, rise time and fall time

NOTE 4 Including transmitter output, Tx package, interconnect, Rx package and Rx input impedance

NOTE 5 The Test Load Impedance is placed at the output of the driver, with the shortest interconnect.

6.5 External Charge Pump Capacitors (Optional)

In order to produce memory devices that can accommodate a low voltage core supply (VCC=1.8 V) an internal charge pump circuit may be required.

Charge pump circuit requires extra-sized passive components. An optional usage of external charge pump capacitors is provided.

Figure 6-1 shows the electrical connections required in case of charge pump implementation that uses external capacitors.

Table 6-8 provides description of the capacitors to be used.

Table 6-8 — Charge pump capacitors description

Capacitor Name	Min	Typ	Max	Description
C_{CP-IN}	TBD	4.7uF	TBD	When charge pump is used, this capacitor is used as the charge pump input bypass capacitor. When charge pump is not used this capacitor is used as a bypass capacitor for the memory.
C_{CP-OUT}	TBD	4.7uF	TBD	Charge pump output bypass capacitor
C_{CP}	TBD	0.22uF	TBD	Charge pump flying capacitor

The charge pump capacitors are optional for UFS devices. Table 6-9 specifies name and description of the balls for connecting external charge pump capacitors.

6.6 External Charge Pump Capacitors (Optional) (cont'd)

Table 6-9 — Charge pump related ball names

Ball Name	Description
C+	C_{CP} capacitor's positive terminal
C-	C_{CP} capacitor's negative terminal
CPOUT1, CPOUT2	Charge pump output capacitor (2 balls)[1]

NOTE 1 Two CPOUT balls are required to reduce inductance, improve ripple and transient response

NOTE 2 The given capacitors shall be placed close to the memory device to minimize the inductance. As a guideline for package design, it is recommended to place the CP related balls close to each other and close to the edge of the package.

6.6 Absolute Maximum DC Ratings

Stresses greater than those listed in Table 6-10 may cause permanent damage to the device. This is a stress rating only, and functional operation of the device at these or any other conditions above those indicated in the operational sections of this standard is not implied. Exposure to absolute maximum rating conditions for extended periods may affect reliability.

Table 6-10 — Absolute maximum DC ratings

Parameter	Symbol	Min	Max	Unit	Notes
Voltage on M-PHY signals		- 0.2	1.6	V	1
Voltage on REF_CLK, RST_n signals		- 0.2	1.6	V	1
VCC supply voltage	VCC	- 0.6	4.6	V	1
VCCQ supply voltage	VCCQ	- 0.2	1.6	V	1
VCCQ2 supply voltage	VCCQ2	- 0.2	2.4	V	1
Storage Temperature	T_{STG}	-40	85	°C	

NOTE 1 Voltage relative to VSS.

7 RESET, POWER-UP AND POWER-DOWN

7.1 Reset

Following sub-sections define the means for resetting the UFS device or a layer of it.

7.1.1 Power-on Reset

A power-on reset is obtained switching the VCCQ, VCCQ2 and VCC power supplies off and back on. The UFS device shall have its own power-on detection circuitry which puts the UFS device and all the different layers of it into a defined state after the power-on.

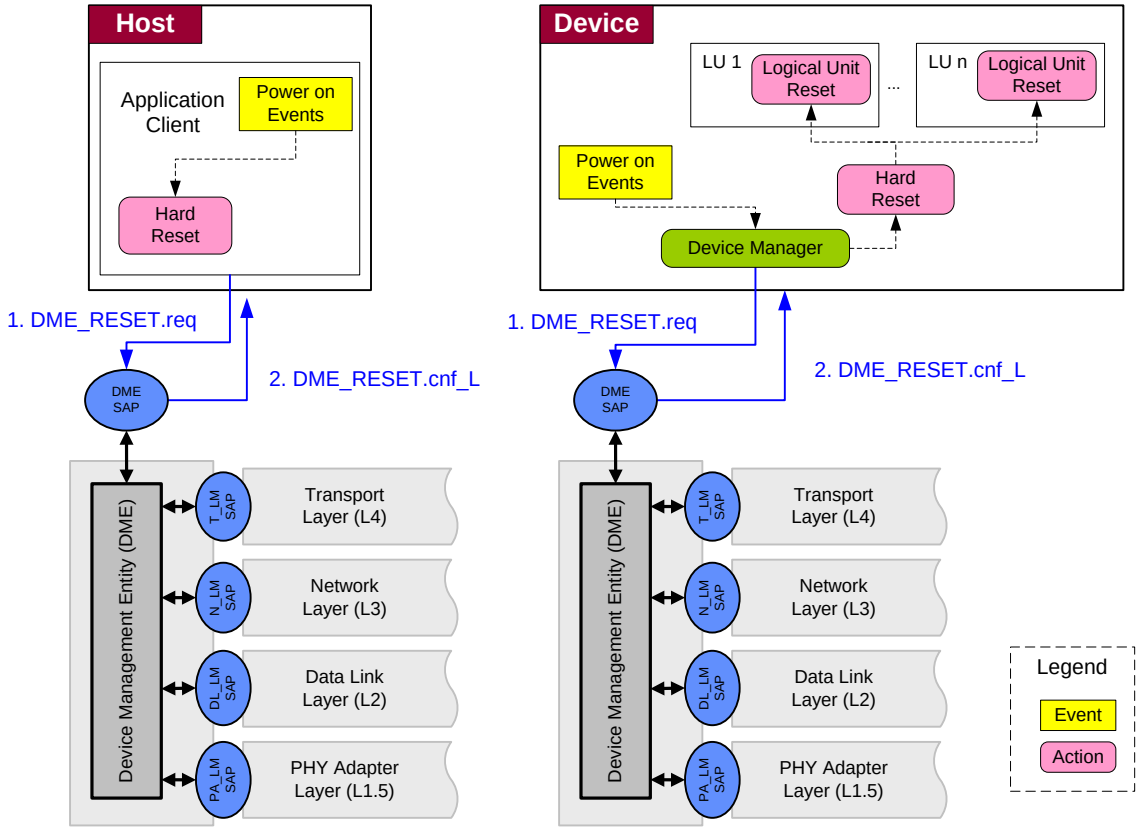


Figure 7-1 — Power-on Reset

7.1.2 Hardware Reset

A dedicated hardware reset signal is defined for the UFS device.

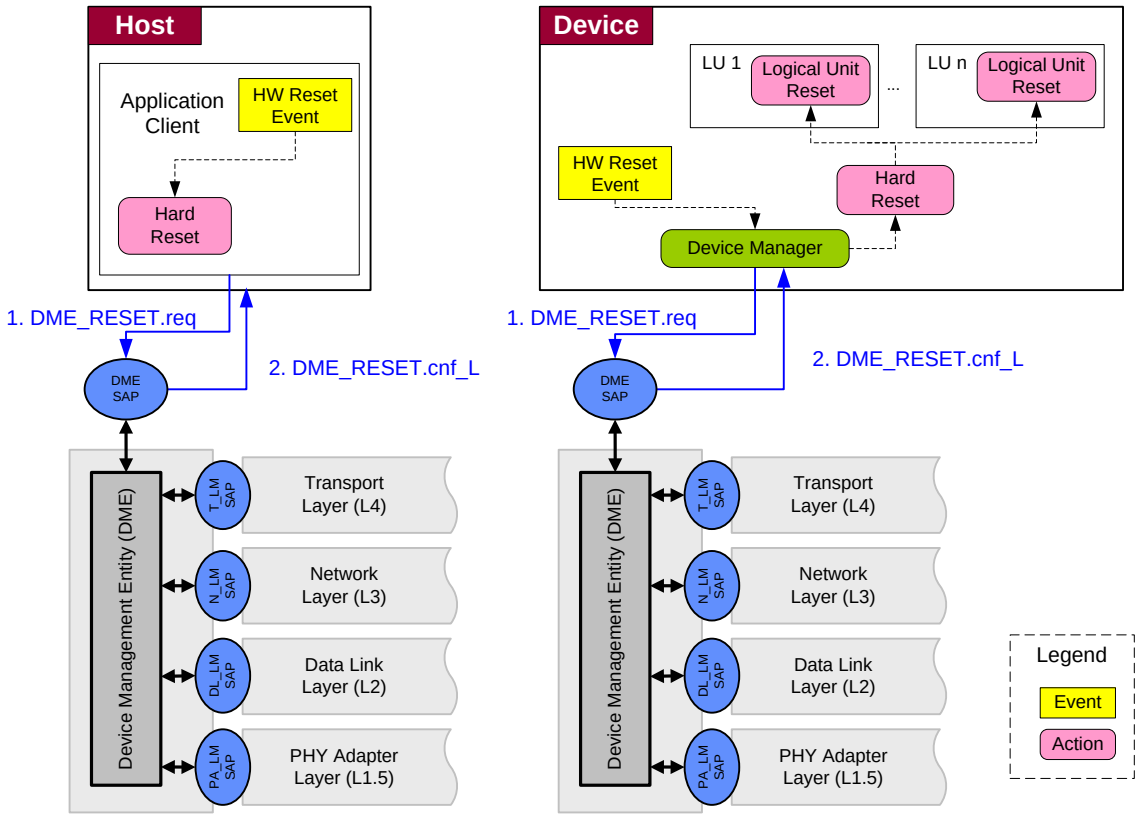


Figure 7-2 — Hardware Reset

Figure 7-3 shows the hardware reset AC timings.

Figure 7-3 — Reset AC timings

Table 7-1 — Reset timing parameters

Symbol	Comment	Min	Max	Unit
tRSTW	RST_n Pulse Width	1		µs
tRSTH	RST_n High Period (Interval)	1		µs
tRSTF	RST_n filter	100		ns

The reset signal is active low. The UFS device shall not detect 100 ns or less of positive or negative RST_n pulse. The UFS device shall detect more than or equal to 1us of positive or negative RST_n pulse width.

7.1.3 EndPointReset

The EndPointReset feature is defined in the MIPI UniPro specification.

Function call from Host Application Client to Host UniPro via DME_SAP:DME_ENDPOINTRESET.req = 1

Device Manager receives the EndPointReset function call from the device UniPro via DME_SAP and executes the EndPointReset function.

A UFS device shall completely reset itself on reception of an EndPointReset: UFS Flags (except Power on reset UFS Flags), UFS Attributes (except Power on reset UFS Attributes), and UniPro attributes are reset to their default value and the UniPro link startup is initiated.

The device may need to be configured again since attributes are reset to their default value. Further, downloading the boot code from the UFS device is optional, and is based on system-level conditions.

A UFS Host should ignore the reception of an EndPointReset from a UFS device.

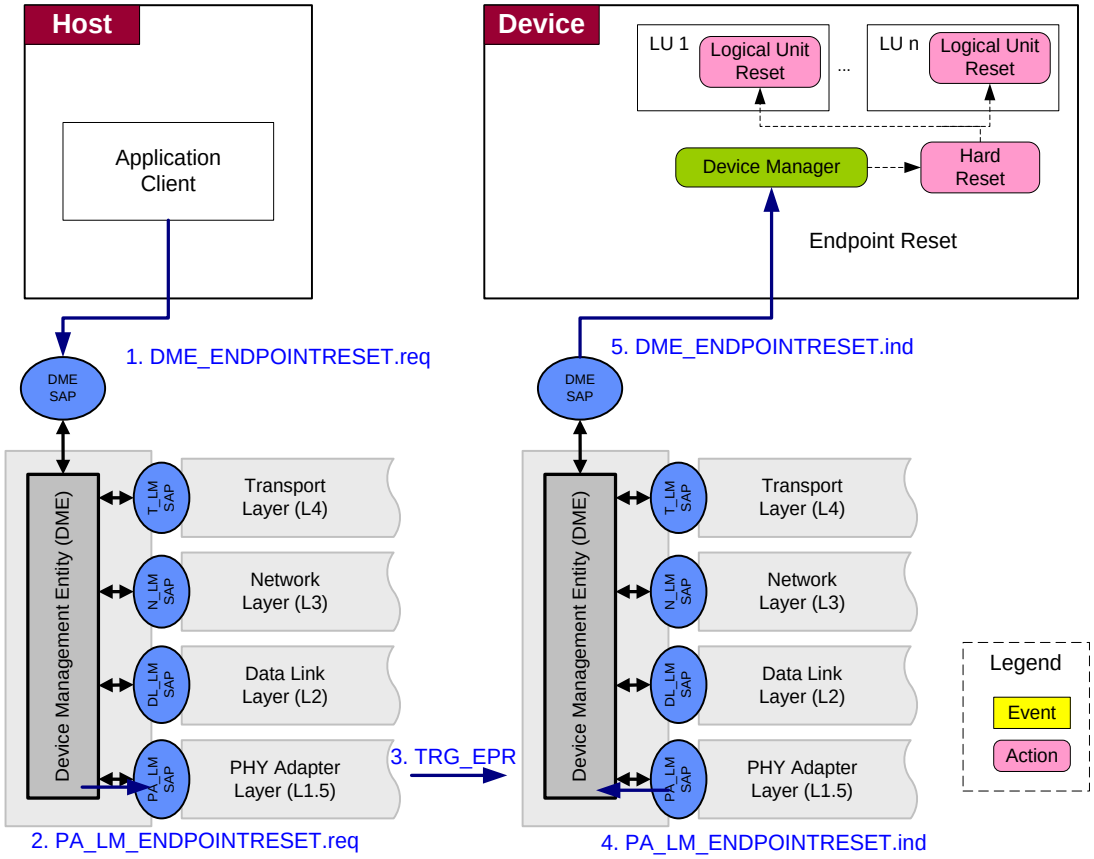


Figure 7-4 — EndPointReset

7.1.4 Logical Unit Reset

The Logical Unit Reset feature is defined in the SCSI Architectural Model [SAM]. This reset is triggered via the SCSI Task Management features described in 0.

LU reset (LU 0, ... , Maximum LU specified by bMaxNumberLU):

Function Call from Host Application Client to Host UTP via UTP_TM_SAP: Task management LOGICAL UNIT RESET (IN (I_T_L Nexus)).

LU Task manager shall receive the function call from device UTP via UTP_TM_SAP and executes the LU reset function.

NOTE The Logical Unit Reset does not set the device parameters to their default value, therefore it is not recommended to use Logical Unit Reset to prepare the UFS device for a system boot.

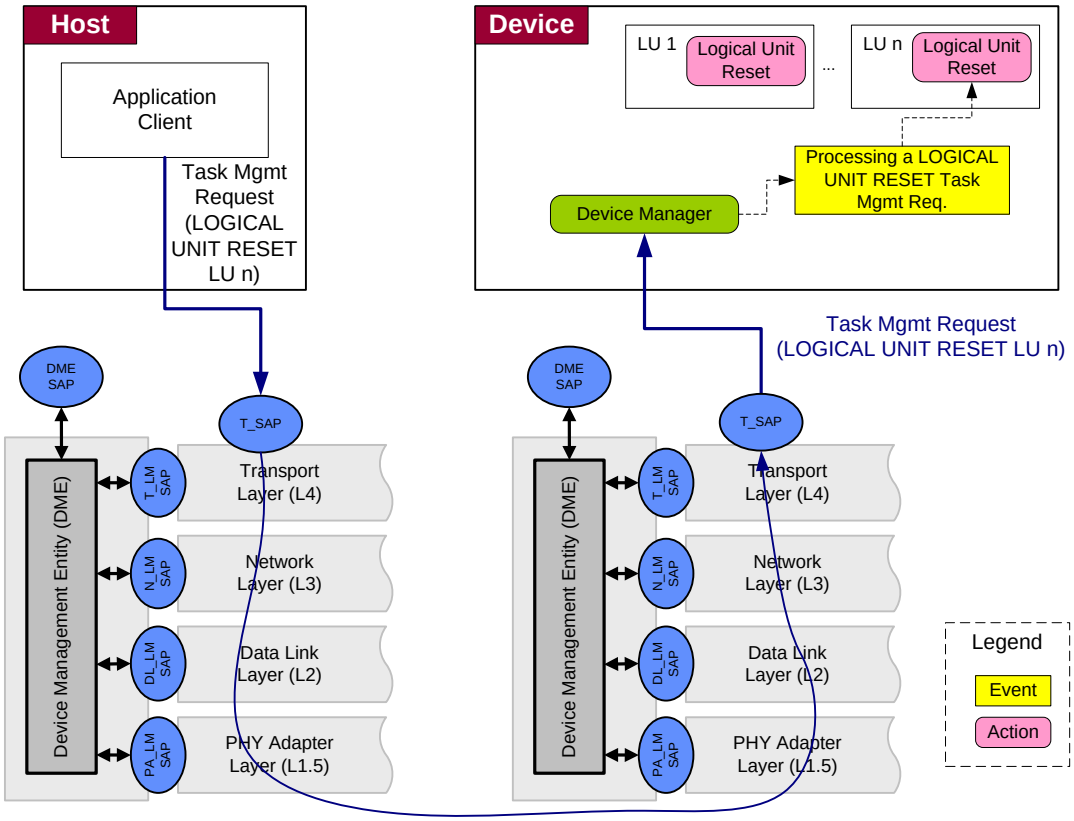


Figure 7-5 — Logical Unit Reset

7.1.5 Host UniPro Warm Reset

A UniPro Warm Reset event in the host is an indirect cause for a UFS device reset . See [MIPI-UniPro] for details.

Host System resets its own UniPro stack: UniPro Stack reset activity in host system side also UniPro Stack reset in the UFS Device side via the DME_LINKLOST.ind message. In case UFS device will receive such DME_LINKLOST.ind message from host system it shall start process of re-initializing its own UniPro stack. In addition also all UFS device level activity has been aborted, task queue lists in all logical units shall be cleared and UFS power mode shall return to Sleep mode or Active mode depending on bInitPowerMode.

7.1.6 Summary of Resets and Device Behavior

Table 7-2 and Table 7-3 summarize the different types of reset and the UFS device behavior related to them.

Table 7-2 — Reset States

Reset Type	Initiator Device	Current Power Mode	Power Mode after Reset		Boot Process(2)
			bInitPowerMode = 00h	bInitPowerMode = 01h	
Power-on	Host	Any	Sleep (1)	Active	Enabled
HW Reset	Host	Any	Sleep (1)	Active	Enabled
EndPointReset	Host	Any	Sleep (1)	Active	Enabled
LU Reset	Host	Active or Idle	Maintain the current power mode	Maintain the current power mode	Disabled
Host UniPro Warm Reset	Host	Any	Sleep (1)	Active	Enabled

NOTE 1 At the end of the device initialization, the power mode transitions from Active to Pre-Sleep and then Sleep (after an implementation specific time).

NOTE 2 The column “Boot process” shows after which type of reset the system can execute the boot process as described in 13.1, UFS Boot. The boot process is enabled if the reset event restores the UFS device to the default state: all parameters are set to the default value, queue are empty, etc.

Table 7-3 — UniPro Attributes, UFS Attributes and UFS Flags reset

Reset Type	UniPro Stack and Attributes	Volatile and Set Only Attributes and Flags (1)	Power on reset Attributes and Flags (1)	Logical Unit Queue
Power-on	Reset	Reset	Reset	Reset (all logical units)
HW Reset	Reset	Reset	Reset	Reset (all logical units)
EndPointReset	Reset	Reset	Not affected	Reset (all logical units)
LU Reset	Not affected	Not affected	Not affected	Reset (addressed logical unit)
Host UniPro Warm Reset	Reset	Reset	No affected	Reset (all logical units)

NOTE 1 See Table 14-24 and Table 14-26 for the definition of Flags and Attributes write access properties.

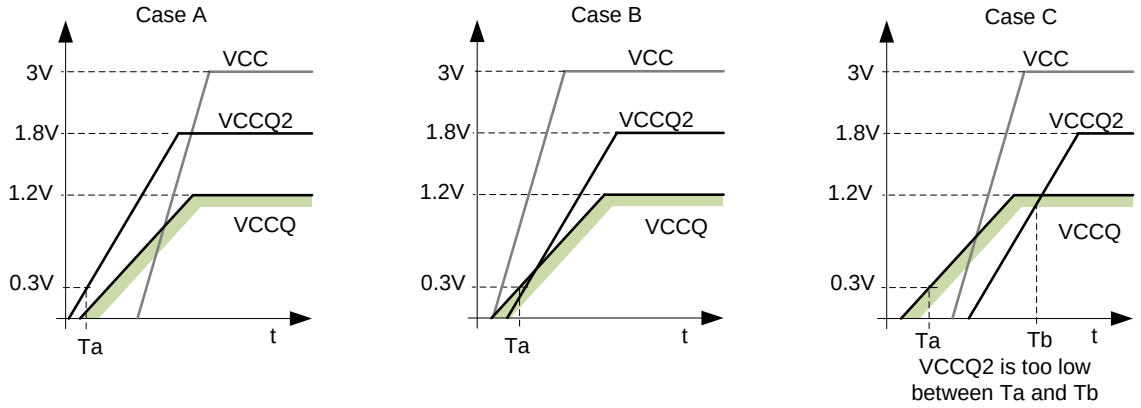
NOTE 2 Values of Attributes and Flags with “Write once” or “Persistent” access property are kept after power cycle or any type of reset event.

7.2 Power up ramp

During power up, VCC and VCCQ2 should be applied as described in the following.

- Ta is the point where VCCQ or VCCQ2 power supply first reaches 300 mV.
- After Ta is reached, VCCQ2 should be greater than VCCQ - 200 mV.
- VCC can be ramped up independently from VCCQ value or VCCQ2 value.
- While powering on the device,
  - RST_n signal should be kept low
  - REF_CLK signal should be between VSS and VCCQ.

Figure 7-6 shows three power up ramp examples: case A and case B meet the requirement, while case C violates it in the time interval from Ta to Tb (VCCQ2 is lower VCCQ - 200 mV).



NOTE 1 The green band represents the voltage range between VCCQ-200 mV and VCCQ.

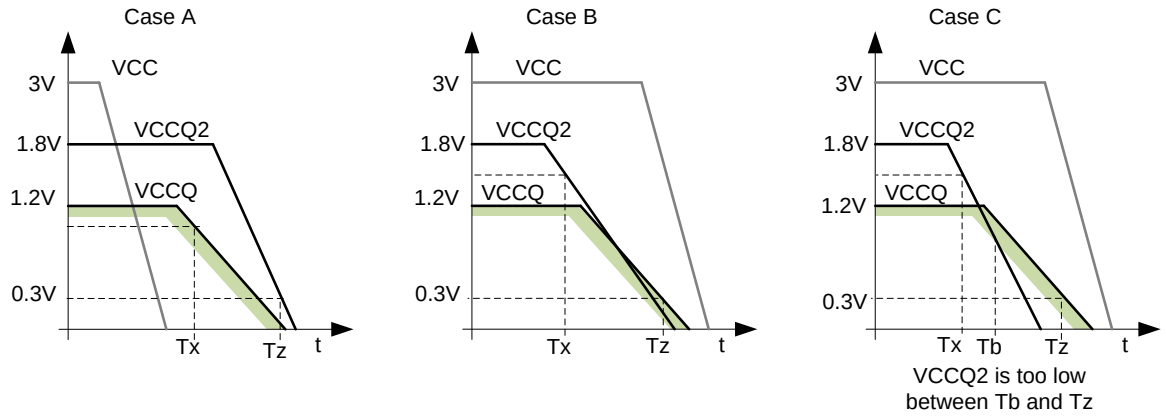
Figure 7-6 — Power up ramps

7.3 Power off ramp

During power off, VCC and VCCQ2 should be removed as described as follows:

- Tx is the point where VCCQ or VCCQ2 power supply decreases under its minimum operating condition value specified (see Table 6-3).
- Tz is the point where VCCQ and VCCQ2 power supplies are below 300 mV.
- VCCQ2 should be greater than VCCQ - 200 mV between Tx and Tz.
- VCC can be ramped down independently from VCCQ value or VCCQ2 value.
- While powering off the device, RST_n signal and REF_CLK signal should be between VSS and VCCQ.

Figure 7-7 shows three power down ramp examples: case A and case B meet the requirement, while case C violates it in the time interval from Tb to Tz.



NOTE 1 The green band represents the voltage range between VCCQ-200 mV and VCCQ.

Figure 7-7 — Power off ramps

The requirements described in this paragraph may not be met only in case of a sudden power off event. Uncontrolled power off should be avoided.

A violation of the power off ramp requirement should not result in any corruption of stored data.

7.4 UFS Device Power Modes and LU Power Condition

7.4.1 Device Power Modes

The device supports multiple power modes, which are controlled by the START STOP UNIT command and some attributes. The device power mode is independent of the bus state of the upstream or downstream links, which are controlled independently.

In order to minimize power consumption in a variety of operating environments, UFS devices will support four basic power modes. One where the device is working, one where it is awaiting the next instruction, one where it has been put to sleep until the host wants it, and a final mode where it can be turned off completely. These four power modes will cover the need for the host to control the power consumed by the device, while still maintaining appropriate responsiveness from the device. There are also three transitional modes needed to facilitate the change from one mode to the next.

While in active mode processing instructions, there are several possible power scenarios. UFS devices can be expected to be battery powered. However, they may be plugged directly into a power source to recharge those batteries. During those times, a larger current may be available, and large amounts of data may be processed at the same time. There is also the possibility that the device is attached to a mobile device with a failing battery, in which case minimal power consumption is a requirement. Finally, there is the possibility that the host would know nothing of the device with which it is paired, and the device would need to be configured to operate within the host's current requirements.

In order to support these varied scenarios, UFS can support up to sixteen active configurations, each with its own current profile. The host can choose from either pre-defined or user-defined current profiles to deliver the highest performance possible. The following seven power modes are defined: Active, Idle, Pre-Active, UFS-Sleep, Pre-Sleep, UFS-PowerDown, Pre-PowerDown. The details of the system are described in the following sections.

7.4.1.1 Active Power Mode

In the Active power mode, the device is responding to a command or performing a background operation. In general, the M-PHY® interface may be in either STALL or HS-BURST state (if in high-speed operation), or SLEEP or PWM-BURST (if in low-speed operation).

The maximum power consumption in Active is determined by the bActiveICCLevel attribute, and there are sixteen different current consumption levels. The maximum current consumption associated with each level for the three power supplies is described in the Power Parameters Descriptor by:

- wActiveICCLevelsVCC[15:0] parameter for VCC,
- wActiveICCLevelsVCCQ[15:0] parameter for VCCQ,
- wActiveICCLevelsVCCQ2[15:0] parameter for VCCQ2.

For example, when the bActiveICCLevel attribute is set to N, the maximum current consumed on VCC is specified by wActiveICCLevelsVCC[N], the maximum current consumed on VCCQ is specified by wActiveICCLevelsVCCQ[N], and the maximum current consumed on VCCQ2 is specified by wActiveICCLevelsVCCQ2[N].

The assumption is that the current consumption levels are ordered in terms of performance: that is, that level 0 is lower performance than level 1, which is lower than level 2, and so on until level 15 which corresponds to the highest performance. The host can then read current consumption values associated with each level in the Power Parameters descriptor, and choose the highest performance levels which fits within its current limitations on each power supply.

7.4.1.1 Active Power Mode (cont'd)

Valid values for the bActiveICCLevel are from "00h" to "0Fh", other values are reserved and should not be set.

UFS devices should primarily use settings of "06h" and "0Ch", for normal (battery) and high (plugged in) power operating modes. See vendor datasheet for the maximum current consumption for: those two Active ICC levels, for the Sleep power mode and the PowerDown power mode.

The bInitActiveICCLevel parameter in the Device Descriptor allows the user to configure the Active ICC level after power on or reset.

The bInitPowerMode parameter in the Device Descriptor defines the power mode to which the device shall transition to after completing the initialization phase (fDeviceInit cleared to zero).

Active Mode can be entered from the Powered On mode or the Pre-Active mode after the completion of all setup necessary to handle commands.

The following power mode may be: Idle, Pre-Sleep, or Pre-PowerDown.

All supported commands are available in Active Mode.

7.4.1.2 Idle Power Mode

The Idle power mode is reached when the device is not executing any operation. In general, the M-PHY® interface may be in STALL, SLEEP or HIBERN8 state. If background operations are continuing, the device should be considered Active power mode.

This mode can only be entered from an Active power mode, and the following state is always the Active power mode. The receipt of any command will transition the device into Active power mode.

7.4.1.3 Pre-Active Power Mode

The Pre-Active power mode is a transitional mode associated with Active power mode. The power consumed will be no more than that consumed in Active power mode. The device shall remain in this power mode until all of the preparation needed to accept commands has been completed.

Pre-Active power mode can be entered from Pre-Sleep, Sleep, Pre-PowerDown, or PowerDown. The following power mode is the Active power mode.

While in Pre-Active power mode:

a) the Device well known logical unit may successfully complete only: START STOP UNIT command and REQUEST SENSE command; other commands may be terminated with CHECK CONDITION status, with the sense key set to NOT READY, with the additional sense code set to LOGICAL UNIT IS IN PROCESS OF BECOMING READY, see 7.4.1.9 for further details;

b) a REQUEST SENSE command shall terminated with GOOD status and provide pollable sense data with the sense key set to NO SENSE, and the additional sense code set to LOGICAL UNIT TRANSITIONING TO ANOTHER POWER CONDITION.

7.4.1.4 UFS-Sleep Power Mode

The UFS-Sleep power mode allows to reduce considerably the power consumption of the device.

VCC power supply can be removed in this state.

The UFS-Sleep power mode is entered from Pre-Sleep power mode.

While in UFS-Sleep power mode:

a) the Device well known logical unit may successfully complete only: START STOP UNIT command and REQUEST SENSE command; other commands may be terminated with CHECK CONDITION status, with the sense key set to NOT READY and the additional sense code set to LOGICAL UNIT NOT READY, INITIALIZING COMMAND REQUIRED, see 7.4.1.9 for further details;

b) a REQUEST SENSE command shall be terminated with GOOD status and provide pollable sense data with the sense key set to NOT READY and the additional sense code set to LOGICAL UNIT NOT READY, INITIALIZING COMMAND REQUIRED.

It is recommended to put the link in HIBERN8 state, although it is actually under host control and can come up and down independently of the UFS power mode.

VCC power supply should be restored before issuing START STOP UNIT command to request transition to Active power mode or PowerDown power mode.

7.4.1.5 Pre-Sleep Power Mode

The Pre-Sleep Mode is a transitional mode associated with UFS-Sleep entry. The power consumed will be no more than that consumed in Active power mode. Pre-Sleep can be entered from Active power mode.

The device will automatically advance to Sleep power mode once any outstanding operations and management activities have been completed.

The device will transition from Pre-Sleep power mode to Pre-Active power mode if START STOP UNIT command with POWER CONDITION = 1h is issued.

While in Pre-Sleep power mode:

a) the Device well known logical unit may successfully complete only: START STOP UNIT command, REQUEST SENSE command and task management functions; other commands may be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, see 7.4.1.9 for further details;

b) a REQUEST SENSE command shall be terminated with GOOD status and provide pollable sense data with the sense key set to NO SENSE and the additional sense code set to LOGICAL UNIT TRANSITIONING TO ANOTHER POWER CONDITION.

7.4.1.6 UFS-PowerDown Power Mode

The UFS-PowerDown power mode is the maximum power saving mode. All volatile data may be lost, and VCC or all power supplies can be removed.

This mode is automatically entered from the Pre-PowerDown power mode, at the completion of the power mode transition.

While in UFS-PowerDown power mode:

a) the Device well known logical unit may successfully complete only: START STOP UNIT command and REQUEST SENSE command; other commands may be terminated with CHECK CONDITION status, with the sense key set to NOT READY and the additional sense code set to LOGICAL UNIT NOT READY, INITIALIZING COMMAND REQUIRED, see 7.4.1.9 for further details;

b) a REQUEST SENSE command shall be terminated with GOOD status and provide pollable sense data with the sense key set to NOT READY, and the additional sense code set to LOGICAL UNIT NOT READY, INITIALIZING COMMAND REQUIRED.

7.4.1.7 Pre-PowerDown Power Mode

The Pre-PowerDown power mode is a transitional mode associated with UFS-PowerDown entry. The power consumed will be no more than that consumed in Active power mode. Pre-PowerDown can be entered from Active or Sleep.

The device will automatically advance to PowerDown power mode once any outstanding operations and management activities have been completed.

The device will transition to Pre-Active mode if START STOP UNIT command with POWER CONDITION field set to 1h is issued.

The following power mode may be PowerDown or Pre-Active.

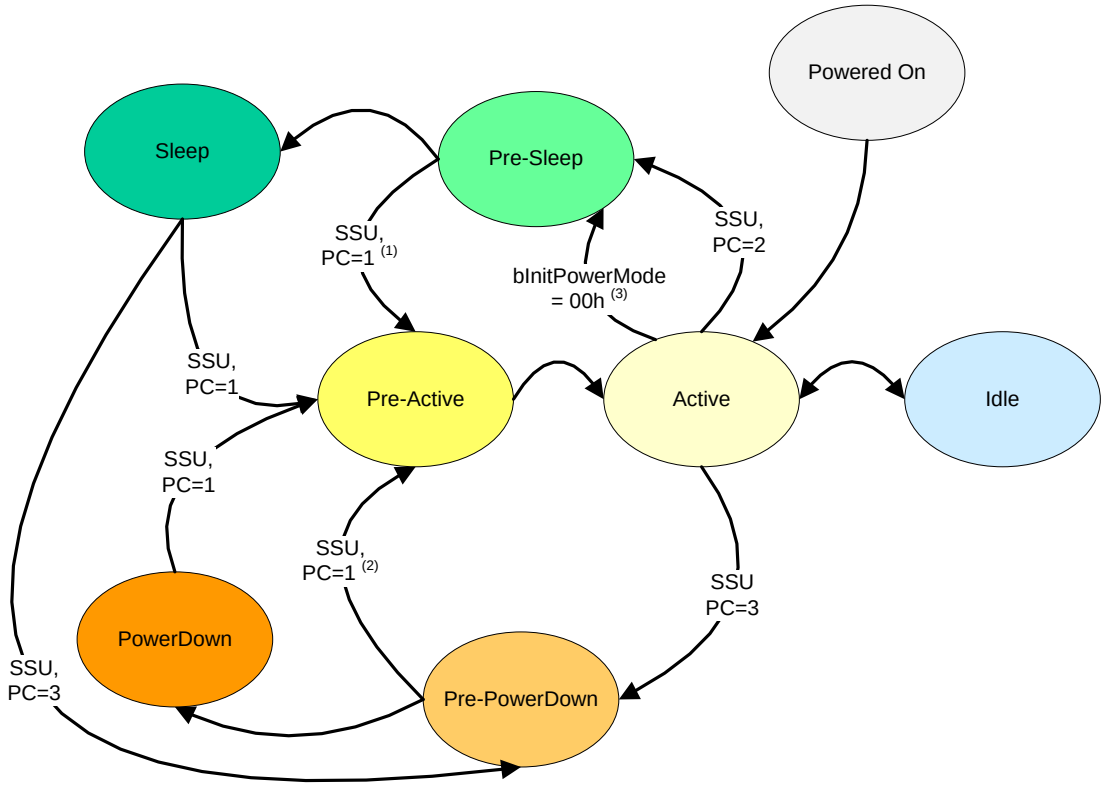
While in Pre-PowerDown power mode:

a) the Device well known logical unit may successfully complete only: START STOP UNIT command, REQUEST SENSE command and task management functions; other commands may be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, see 7.4.1.9 for further details;

b) a REQUEST SENSE command shall be terminated with GOOD status and provide pollable sense data with the sense key set to NO SENSE and the additional sense code set to LOGICAL UNIT TRANSITIONING TO ANOTHER POWER CONDITION.

7.4.1.8 Power Mode State Machine

The relationship amongst the different power modes is shown in Figure 7-8.



(1) This transition may occur only if the SSU command that caused the transition to Pre-Sleep had IMMED set to one.
(2) This transition may occur only if the SSU command that caused the transition to Pre-PowerDown had IMMED set to one.
(3) This automatic transition shall occur at the end of device initialization if bInitPowerMode = 00h.

Figure 7-8 — Power Mode State Machine

7.4.1.8.1 Transitions from Powered On Power Mode

The device shall enter in Powered On when: the power supplies are applied, after hardware reset, EndPointReset or Host UniPro Warm Reset.

Transition from Powered_On to Active

This transition shall occur when the device is ready to begin power on initialization.

7.4.1.8.2 Transitions from Pre-Active Power Mode

Transition from Pre-Active to Active

This transition shall occur when the device meets the requirements for being in Active power mode.

7.4.1.8.3 Transitions from Active Power Mode

Transition from Active to Idle

This transition may occur when the device completes any ongoing operations.

Transition from Active to Pre-Sleep

This transition shall occur

- at the end of the device initialization and if the bInitPowerMode parameter is set to "00h", or
- if the device server processes a START STOP UNIT command with the POWER CONDITION field set to 2h.

Transition from Active to Pre-PowerDown

This transition shall occur if the device server processes a START STOP UNIT command with the POWER CONDITION field set to 3h.

7.4.1.8.4 Transitions from Idle Power Mode

Transition from Idle to Active

This transition shall occur if the device processes a request that requires to be in Active power mode.

7.4.1.8.5 Transitions from Pre-Sleep Power Mode

Transition from Pre-Sleep to Pre-Active

This transition shall occur if the START STOP UNIT command that caused the transition to Pre-Sleep power mode had IMMED set to one, and when the device server processes a START STOP UNIT command with the POWER CONDITION field set to 1h.

Transition from Pre-Sleep to Sleep

This transition shall occur when the device meets the requirements for being in Sleep power mode.

7.4.1.8.6 Transitions from UFS-Sleep Power Mode

Transition from UFS-Sleep to Pre-Active

This transition shall occur if the device server processes a START STOP UNIT command with the POWER CONDITION field set to 1h.

Transition from UFS-Sleep to Pre-PowerDown

This transition shall occur if the device server processes a START STOP UNIT command with the POWER CONDITION field set to 3h.

7.4.1.8.7 Transitions from Pre-PowerDown Power Mode

Transition from Pre-PowerDown to Pre-Active

This transition shall occur if the START STOP UNIT command that caused the transition to Pre-PowerDown power mode had IMMED set to one, and when the device server processes a START STOP UNIT command with the POWER CONDITION field set to 1h.

Transition from Pre-PowerDown to PowerDown

This transition shall occur when the device meets the requirements for being in UFS-PowerDown power mode.

7.4.1.8.8 Transitions from UFS-PowerDown Power Mode

Transition from UFS-PowerDown to Pre-Active

This transition shall occur if the device server processes a START STOP UNIT command with the POWER CONDITION field set to 1h.

7.4.1.8.9 SCSI command and UPIU transactions

The current power mode may be retrieved reading the bCurrentPowerMode attribute.

bCurrentPowerMode is the only attribute the device is required to return in any power mode. If the device is not in Active power mode or Idle power mode, a QUERY REQUEST UPIU to access descriptors, flags, or attributes other than bCurrentPowerMode may fail.

By setting the IMMED bit to one during the START STOP UNIT command, the device can be instructed to respond at the entrance to the transitional mode, once the command is received.

The effects of concurrent power mode changes requested by START STOP UNIT commands with the IMMED bit set to one are vendor specific.

A START STOP UNIT command with the IMMED bit set to zero causing a transition to Active, Sleep, or PowerDown power modes shall not complete with GOOD status until the device reaches the power mode specified by the command.

Table 7-4 summarizes which SCSI commands and UPIU transactions are allowed for each power mode.

Table 7-4 — Allowed SCSI commands and UPIU for each Power Mode

Power Mode	SCSI Commands	UPIU Transactions
Active	Any commands	Any UPIU
Idle	Any commands	Any UPIU
Pre-Active	START STOP UNIT, REQUEST SENSE	COMMAND UPIU, RESPONSE UPIU, REJECT UPIU, DATA IN UPIU, QUERY REQUEST UPIU, QUERY RESPONSE UPIU
UFS-Sleep	START STOP UNIT, REQUEST SENSE	COMMAND UPIU, RESPONSE UPIU, REJECT UPIU, DATA IN UPIU, QUERY REQUEST UPIU, QUERY RESPONSE UPIU
Pre-Sleep	START STOP UNIT, REQUEST SENSE, Task Managem. Fun.	COMMAND UPIU, RESPONSE UPIU, REJECT UPIUDATA IN UPIU, QUERY REQUEST UPIU, QUERY RESPONSE UPIU, TASK MANAGEMENT REQUEST UPIU, TASK MANAGEMENT RESPONSE UPIU
UFS-PowerDown	START STOP UNIT, REQUEST SENSE	COMMAND UPIU, RESPONSE UPIU, REJECT UPIU, DATA IN UPIU, QUERY REQUEST UPIU, QUERY RESPONSE UPIU
Pre-PowerDown	START STOP UNIT, REQUEST SENSE, Task Managem. Fun.	COMMAND UPIU, RESPONSE UPIU, REJECT UPIU, DATA IN UPIU, QUERY REQUEST UPIU, QUERY RESPONSE UPIU, TASK MANAGEMENT REQUEST UPIU, TASK MANAGEMENT RESPONSE UPIU

7.4.1.9 Responses to SCSI commands

Table 7-5 defines the Device well known logical unit response to a START STOP UNIT command for a given power mode. It is assumed that the IMMED bit in START STOP UNIT commands is set to zero.

Table 7-5 — Device Well Known Logical Unit Responses to SSU command

Current Power Mode	PC	STATUS	SENSE KEY	ASC, ASCQ
Pre-Active	1h	GOOD (1)	-	-
	Others	CHECK CONDITION	NOT READY	LOGICAL UNIT NOT READY, START STOP UNIT COMMAND IN PROGRESS
Active	1h, 2h, 3h	GOOD (1)	-	-
	Others	CHECK CONDITION	ILLEGAL REQUEST	INVALID FIELD IN CDB
Pre-Sleep	2h	GOOD (1)	-	-
	Others	CHECK CONDITION	NOT READY	LOGICAL UNIT NOT READY, START STOP UNIT COMMAND IN PROGRESS
UFS-Sleep	1h, 2h, 3h	GOOD (1)	-	-
	Others	CHECK CONDITION	ILLEGAL REQUEST	INVALID FIELD IN CDB
Pre-PowerDown	3h	GOOD (1)	-	-
	Others	CHECK CONDITION	NOT READY	LOGICAL UNIT NOT READY, START STOP UNIT COMMAND IN PROGRESS
UFS-PowerDown	1h, 3h	GOOD (1)	-	-
	Others	CHECK CONDITION	ILLEGAL REQUEST	INVALID FIELD IN CDB

NOTE 1 The START STOP UNIT command may not terminate with GOOD status for condition not due to CDB content.

Table 7-6 summarizes the response that the Device well known logical unit may provide to a command other than START STOP UNIT for various device power modes.

7.4.1.9 Responses to SCSI commands (cont'd)

Table 7-6 — Device Well Known Logical Unit Responses to commands other than SSU

Power Mode	Command	STATUS	SENSE KEY	ASC, ASCQ
Pre-Active	REQUEST SENSE	GOOD (1)	-	-
	Others (1)	CHECK CONDITION	NOT READY	LOGICAL UNIT IS IN PROCESS OF BECOMING READY
Pre-Sleep, Pre-PowerDown.	REQUEST SENSE	GOOD (1)	-	-
	Others (1)	CHECK CONDITION	ILLEGAL REQUEST	-
UFS-Sleep, UFS-PowerDown	REQUEST SENSE	GOOD (1)	-	-
	Others (1)	CHECK CONDITION	NOT READY	LOGICAL UNIT NOT READY, INITIALIZING COMMAND REQUIRED
NOTE 1 Rows identified with "Others" define Device well known logical unit response to command other than START STOP UNIT command and REQUEST SENSE command.				

Table 7-7 defines the pollable sense data for various device power modes.

Table 7-7 — Pollable Sense Data for each Power Modes

Power Mode	SENSE KEY	ASC, ASCQ
Pre-Active, Pre-Sleep, Pre-PowerDown	NO SENSE	LOGICAL UNIT TRANSITIONING TO ANOTHER POWER CONDITION
UFS-PowerDown, UFS-Sleep	NOT READY	LOGICAL UNIT NOT READY, INITIALIZING COMMAND REQUIRED

7.4.2 Power Management Command: START STOP UNIT

When the START STOP UNIT command is sent to a logical unit, it can be used to enable or disable that logical unit, flush all cached logical blocks to the medium (for logical units that contain cache), or load or eject the medium.

When the START STOP UNIT command is sent to the UFS Device well-known logical unit (W-LUN = 50h), it can be used to select the device power mode.

Table 7-8 — START STOP UNIT command

Bit / Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (1Bh)							
1	Reserved							IMMED
2	Reserved							
3	Reserved				POWER CONDITION MODIFIER (Reserved = 0000b)			
4	POWER CONDITION				Reserved	NO_FLUSH	LOEJ = 0b	START
5	CONTROL (00h)							

7.4.2 Power Management Command: START STOP UNIT (cont'd)

The POWER CONDITION field selects the desired mode. If the command is sent to a logical unit other than the Device well known logical unit, the POWER CONDITION field may be ignored.

Table 7-9 — START STOP UNIT fields

IMMED	No Flush	Power Condition	Start	LUN or WLUN / LUN Field in UPIU	Action
0	-	-	-	-	Response is sent after change is complete.
1	-	-	-	-	Response is sent immediately after command decode
-	0	-	-	-	Dynamic data should be flushed to non-volatile storage
-	1	-	-	-	No requirements regarding dynamic data
-	-	0h	0	00h to N-1 (1) / 00h to N-1 (1)	Stop the designated LU.
-	-	0h	1	00h to N-1 (1) / 00h to N-1 (1)	Start the designated LU.
-	-	1h	0	50h / D0h	Cause a transition to the Active power mode
-	-	2h	0	50h / D0h	Cause a transition to the UFS-Sleep power mode
-	-	3h	0	50h / D0h	Cause a transition to the UFS-PowerDown power mode

NOTE 1 The value of N is indicated by bMaxNumberLU parameter in the Geometry Descriptor.

7.4.3 Power Mode Control

Table 7-10 defines a series of attributes used to control the active current levels and power modes.

Table 7-10 — Attribute for Power Mode Control

Attribute Name	Size	Type	Description
bCurrentPowerMode	1 byte	Read only	Current device Power Mode Status 00h: Idle power mode 10h: Pre-Active power mode 11h: Active power mode 20h: Pre-Sleep power mode 22h: UFS-Sleep power mode 30h: Pre-PowerDown power mode 33h: UFS-PowerDown power mode Others: Reserved
bActiveICCLevel	1 byte	Read / Volatile	bActiveICCLevel defines the maximum current consumption allowed during Active mode. 00h: Lowest Active ICC level … 0Fh: Highest Active ICC level Others: Reserved Valid range from 00h to 0Fh.

Table 7-11 shows the Device Descriptor parameters that specify the power mode and the Active ICC level after power on or reset. See 14.1.4.3, Configuration Descriptor, for details about device configuration.

Table 7-11 — Device Descriptor parameters

Parameter Name	Size	Description
bInitActiveICCLevel	1 byte	Initial Active ICC Level bInitActiveICCLevel defines the bActiveICCLevel value after power on or reset. Valid range from 00h to 0Fh.
bInitPowerMode	1 byte	Initial Power Mode bInitPowerMode defines the Power Mode after device initialization or hardware reset 00h: UFS-Sleep Mode 01h: Active Mode Others: Reserved

7.4.3 Power Mode control (cont'd)

Table 7-12 defines the parameters of the Power Parameters Descriptor. Each parameter is composed by sixteen elements, the size of each element is two bytes, and it is structured as shown in Table 7-13.

Table 7-12 — Power Parameters Descriptor fields

Parameter Name	Size	Type	Description
wActiveICCLevelsVCC[15:0]	32 bytes	Read Only	Active ICC Levels for VCC Maximum peak current consumed from VCC in each of the sixteen current consumption levels defined for the Active mode.
wActiveICCLevelsVCCQ [15:0]	32 bytes	Read Only	Active ICC Levels for VCCQ Maximum peak current consumed from VCCQ in each of the sixteen current consumption levels defined for the Active mode.
wActiveICCLevelsVCCQ2 [15:0]	32 bytes	Read Only	Active ICC Levels for VCCQ2 Maximum peak current consumed from VCCQ2 in each of the sixteen current consumption levels defined for the Active mode.

Table 7-13 — Format for Power Parameter element

Field Name	Bit Range	Description
Unit	bit [15:14]	00b:nA 01b:uA 10b:mA 11b: A
-	bit [13:10]	Reserved (0000b)
Value	bit [9: 0]	The maximum current expected in each current consumption level

7.4.4 Logical Unit Power Condition

Each logical unit may be in active power condition and stopped power condition. See [SPC] and [SBC] for the definition of these two logical unit power conditions.

All logical units shall be in the active power condition after power on or any type of reset event.

Transition from active power condition to stopped power condition shall occur if the device server processes a START STOP UNIT command with the START bit set to zero and the POWER CONDITION field set to 0h.

Transition from stopped power condition to active power condition shall occur if the device server processes a START STOP UNIT command with the START bit set to one and the POWER CONDITION field set to 0h.

START STOP UNIT command to change the logical unit power condition should be issued only if the device is in Active power mode or Idle power mode.

A request to move to the stopped power condition should be made only when the logical unit command queue is empty.

A transition in the device power mode state shall not change the logical unit power condition.

Table 7-14 defines the logical unit responses to SCSI commands for various device power modes, assuming that the logical unit is in active power condition. See 7.4.1.9 for details about Device well known logical unit.

Table 7-14 — Logical Unit Response to SCSI command

Power Mode	COMMAND	STATUS	SENSE KEY	ASC, ASCQ
Pre-Active, Pre-Sleep, UFS-Sleep Pre-PowerDown UFS-PowerDown	Any	CHECK CONDITION	NOT READY	-

If the logical unit is in the stopped power condition, then the device server shall

- provide pollable sense data with sense key set to NOT READY and the additional sense code set to LOGICAL UNIT NOT READY, INITIALIZING COMMAND REQUIRED;
- terminate each media access command or TEST UNIT READY command with CHECK CONDITION status with the sense key set to NOT READY and the additional sense code set to LOGICAL UNIT NOT READY, INITIALIZING COMMAND REQUIRED.

8 UFS UIC LAYER: MIPI M-PHY

8.1 Termination

The M-TX shall be terminated as defined in section “Termination Scheme” of the M-PHY specification [MIPI-M-PHY].

The M-RX shall include switchable differential termination. By default the M-RX termination shall be off in PWM-BURST state and may be turned on setting the proper MIPI Attribute. The termination shall be on by default in HS-BURST state and may be turned off. There shall be no termination in SLEEP and STALL states. During DISABLE and HIBERNATE states M-TX drives High-Z while M-RX terminates the lane by a 'Dif-Z keeper'. Dif-Z keeper means M-RX drives a weak differential zero on the lane.

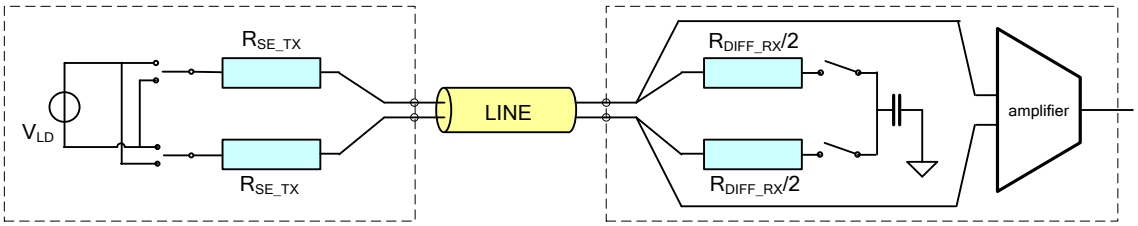


Figure 8-1 — Simplified example for I/O termination

M-RX of a SUBLINK in a LINK may have different termination settings.

The supported termination settings are defined in the Capability Attributes in Table 8-1 and Table 8-2. The termination is controlled via the Configuration Attributes. The receiver termination resistance is defined in the M-PHY specification.

Timings of the termination enable and disable are defined in the M-PHY specification.

8.2 Drive Levels

The M-PHY specification defines two different drive amplitudes: the large amplitude (LA) and the small amplitude (SA). The UFS interface utilizes both of these drive amplitudes as defined in the M-PHY specification.

Every M-TX in every LINK will start communication with LA after power up or reset. Option to switch to SA mode shall be supported via a Configuration Attribute.

SUBLINKS in a LINK shall communicate with same amplitude.

8.3 PHY State machine

The UFS interface shall implement the Type I state machine.

The M-PHY specification defines two different signaling schemes for low-speed mode (LS-MODE): Non-Return-to-Zero (NRZ) and Pulse-Width-Modulation (PWM). The UFS interface shall utilize the PWM signaling scheme in the LS-MODE as defined by the M-PHY specification for the State Machine Type I [MIPI M-PHY].

The UFS interface shall implement the following LCC categories: MISC, PWM-MODE and HS-MODE.

8.4 HS Burst

A UFS device shall support the HS-GEAR1 and the HS-GEAR2. Support for HS-GEAR3 is optional. The supported gears are indicated in the Capability Attribute Table 8-1 and Table 8-2.

SUBLINKS in a LINK may communicate with different HS-GEAR or PWM-GEAR.

8.4.1 HS Prepare Length Control

The TX_HS_PREPARE_LENGTH M-PHY configuration attribute defines the time to move from STALL to HS-BURST. At reset, M-TX sets TX_HS_PREPARE_LENGTH = 15.

8.4.2 HS Sync Length Control

The TX_HS_SYNC_LENGTH M-PHY configuration attribute defines the number of synchronization symbols before a HS Burst. In the UFS interface the synchronization sequence shall be generated by the M-TX. Support for protocol controlled synchronization is optional. M-TX starts at reset with TX_HS_SYNC_LENGTH = 15, in COARSE type.

8.5 PWM Burst

A UFS device shall support the PWM-G1 (default, mandated by [M-PHY]), PWM-G2, PWM-G3 and PWM-G4 GEARS. The PWM-G5, PWM-G6 and PWM-G7 are optional. The supported PWM-GEARS are indicated in the Capability Attributes Table 8-1 and Table 8-2.

NOTE Even if the physical layer supports PWM-G0, this gear can not be used because it is not supported by [MIPI-UniPro].

The PWM-G1 shall be the active one by default after power up or reset.

SUBLINKS in a LINK may communicate with different PWM-GEAR or HS-GEAR.

8.5.1 LS Prepare Length Control

The TX_LS_PREPARE_LENGTH M-PHY configuration attribute defines the time to move from SLEEP to PWM-BURST. At reset, M-TX sets TX_LS_PREPARE_LENGTH = 10.

8.6 UFS PHY Attributes

The MIPI M-PHY includes several configurable attributes. There is range of values defined for the attributes but it is left for the application to fix the actual required values inside the range. Following is the list of such attributes. The UFS application specific requirement for the values can be found from Table 8-1 and Table 8-2.

8.6 UFS PHY Attributes (cont'd)

Table 8-1 — UFS PHY M-TX Capability Attributes

Attribute Name	Attribute ID	Range	UFS Value	Notes
TX_HSMODE_Capability	0x01	False = 0, True = 1	1	
TX_HSGEAR_Capability	0x02	HS_G1_ONLY = 1, HS_G1_TO_G2 = 2, HS_G1_TO_G3 = 3	≥ 2	1
TX_PWMG0_Capability	0x03	0=No, 1=Yes	0, 1	2
TX_PWMGEAR_Capability	0x04	PWM_G1_ONLY = 1, PWM_G1_TO_G2 = 2, PWM_G1_TO_G3 = 3, PWM_G1_TO_G4 = 4, PWM_G1_TO_G5 = 5, PWM_G1_TO_G6 = 6, PWM_G1_TO_G7 = 7	≥ 4	3
TX_Amplitude_Capability	0x05	SA=1, LA=2, Both=3	3	4
TX_ExternalSYNC_Capability	0x06	0=False 1=True	0, 1	5
TX_HS_Unterminated_LINE_Drive_Capability	0x07	0=No, 1=Yes	1	6
TX_LS_Terminated_LINE_Drive_Capability	0x08	0=No, 1=Yes	1	6
TX_Min_SLEEP_NoConfig_Time_Capability	0x09	1 to 15	1 to 15	7
TX_Min_STALL_NoConfig_Time_Capability	0x0A	1 to 255	1 to 255	7
TX_Min_SAVE_Config_Time_Capability	0x0B	1 to 250	1 to 250	8
TX_REF_CLOCK_SHARED_Capability	0x0C	0=No, 1=Yes	1	
TX_PHY_MajorMinor_Release_Capability	0x0D	B[7:4] : Major version number	0 to 9	9
		B[3:0] : Minor version number	0 to 9	9
TX_PHY_Editorial_Release_Capability	0x0E	B[7:0] = 1 to 99	1 to 99	9
TX_Hibern8Time_Capability	0x0F	1 to 128	1	10
TX_Advanced_Granularity_Capability	0x10	B[2:1]: Step size B[0]: Supports fine granularity steps	B[2:1] = 0 to 3 B[0] = 0,1	11
TX_Advanced_Hibern8Time_Capability	0x11	1 to 128	1 to 128	
TX_HS_Equalizer_Setting_Capability	0x12	B[1:0]	B[1:0] = 0 to 3	

NOTE 1 All HS gears from HS-GEAR1 to TX_HSGEAR_Capability shall be supported.
NOTE 2 Even if TX_PWMG0_Capability = 1, PWM-G0 cannot be used because it is not supported by [MIPI-UniPro].
NOTE 3 All PWM gears from PWM-GEAR1 to TX_PWMGEAR_Capability shall be supported.
NOTE 4 UFS device shall support both drive levels.
NOTE 5 Support for external SYNC pattern is optional and its definition is device specific.
NOTE 6 A UFS device shall support un-terminated HS burst and terminated LS burst.
NOTE 7 Time defined in SI.
NOTE 8 Minimum reconfiguration time (in 40 ns steps).
NOTE 9 See clause 2, Normative Reference.
NOTE 10 Time defined in 0.1msec steps.
NOTE 11 B[2:1] step size: b00 = 4 µs, b01 = 8 µs, b10 = 16 µs, b11 = 32 µs. B[0]: No =0 (100 µs), Yes = 1

994 **8.6 UFS PHY Attributes (cont'd)**

995 **Table 8-2 — UFS PHY M-RX Capability Attributes**

Attribute Name	Attribute ID	Range	UFS Value	Notes
RX_HSMODE_Capability	0x81	No=0, Yes=1	1	
RX_HSGEAR_Capability	0x82	HS_G1_ONLY = 1, HS_G1_TO_G2 = 2, HS_G1_TO_G3 = 3	≥ 2	1
RX_PWMG0_Capability	0x83	0=No, 1=Yes	0, 1	2
RX_PWMGEAR_Capability	0x84	PWM_G1_ONLY = 1, PWM_G1_TO_G2 = 2, PWM_G1_TO_G3 = 3, PWM_G1_TO_G4 = 4, PWM_G1_TO_G5 = 5, PWM_G1_TO_G6 = 6, PWM_G1_TO_G7 = 7	≥ 4	3
RX_HS_Unterminated_Capability	0x85	0=No, 1=Yes	1	4
RX_LS_Terminated_Capability	0x86	0=No, 1=Yes	1	4
RX_Min_SLEEP_NoConfig_Time_Capability	0x87	1 to 15	1 to 15	
RX_Min_STALL_NoConfig_Time_Capability	0x88	1 to 255	1 to 250	
RX_Min_SAVE_Config_Time_Capability	0x89	1 to 250	1 to 250	5
RX_REF_CLOCK_SHARED_Capability	0x8A	0=No, 1=Yes	1	
RX_HS_G1_SYNC_LENGTH_Capability	0x8B	B[7:6] : SYNC_range FINE = 0, COARSE = 1 B[5:0] : SYNC_length 1 to 15 for FINE, 0 to 15 for COARSE	B[7:6] : 1,0 B[5:0] : 1 to 15 for FINE 0 to 15 for COARSE	
RX_HS_G1_PREPARE_LENGTH_Capability	0x8C	0 to 15	0 to 15	
RX_LS_PREPARE_LENGTH_Capability	0x8D	0 to 15	0 to 15	
RX_PWM_Burst_Closure_Length_Capability	0x8E	0 to 31	0 to 31	
RX_Min_ActivateTime_Capability	0x8F	1 to 9	1 to 9	6
RX_PHY_MajorMinor_Release_Capability	0x90	B[7:4] : Major version number	0 to 9	7
		B[3:0] : Minor version number	0 to 9	7
RX_PHY_Editorial_Release_Capability	0x91	1 to 99	1 to 99	7
RX_Hibern8Time_Capability	0x92	1 to 128	1	6
RX_PWM_G6_G7_SYNC_LENGTH_Capability	0x93	B[7:6] : SYNC_range FINE = 0, COARSE = 1 B[5:0] : SYNC_length 0 to 15	B[7:6] =1,0 B[5:0] ≤ 15	
RX_HS_G2_SYNC_LENGTH_Capability	0x94	B[7:6] : SYNC_range FINE = 0, COARSE = 1 B[5:0] : SYNC_length 1 to 15 for FINE, 0 to 15 for COARSE	B[7:6] : 1,0 B[5:0] : 1 to 15 for FINE 0 to 15 for COARSE	
RX_HS_G3_SYNC_LENGTH_Capability	0x95	B[7:6] : SYNC_range FINE = 0,COARSE = 1 B[5:0] : SYNC_length 1 to 15 for FINE, 0 to 15 for COARSE	B[7:6] : 1,0 B[5:0] : 1 to 15 for FINE 0 to 15 for COARSE	
RX_HS_G2_PREPARE_LENGTH_Capability	0x96	B[3:0] = 0 to 15	≤ 15	
RX_HS_G3_PREPARE_LENGTH_Capability	0x97	B[3:0] = 0 to 15	≤ 15	
RX_Advanced_Granularity_Capability	0x98	B[2:1]: Step size B[0]: Supports fine granularity steps	B[2:1] = 0 to 3 B[0] =0,1	
RX_Advanced_Hibern8Time_Capability	0x99	1 to 128	1 to 128	
RX_Advanced_Min_ActivateTime_Capability	0x9A	B[3:0] = 1 to 14	B[3:0] = 1 to 14	

8.6 UFS PHY Attributes (cont'd)

Table 8-2 — UFS PHY M-RX Capability Attributes (cont'd)

NOTE 1	All HS gears from HS-GEAR1 to RX_HSGEAR_Capability shall be supported.
NOTE 2	Even if RX_PWMG0_Capability = 1, PWM-G0 cannot be used because it is not supported by [MIPI-UniPro].
NOTE 3	All PWM gears from PWM-GEAR1 to RX_PWMGEAR_Capability shall be supported.
NOTE 4	A UFS device shall support un-terminated HS burst and terminated LS burst.
NOTE 5	Minimum reconfiguration time in 40ns steps.
NOTE 6	Time defined in 0.1msec steps.
NOTE 7	See clause 2, Normative Reference.

8.7 Electrical characteristics

8.7.1 Transmitter Characteristics

As defined in the M-PHY specification.

8.7.2 Receiver Characteristics

As defined in the M-PHY specification.

9 UFS UIC LAYER: MIPI UNIPRO

9.1 Overview

UFS builds on the MIPI Unified Protocol (UniPro) as its Interconnect (Service Delivery Subsystem) to provide basic transfer capabilities to the UFS Transport Protocol (UTP) Layer. On the data plane UTP and UniPro communicate via the Service Primitives of the UniPro Transport Layer CPorts (T_CO_SAPs). Control plane interaction (e.g., discovery, enumeration and configuration of the Link) between higher layer protocol functions of UFS and UniPro are accomplished using the Device Management Entity Service Primitives as defined by the UniPro specification.

9.2 Architectural Model

UniPro is internally composed of several sub-layers which are all well defined by the MIPI UniPro specification [MIPI-UniPro]. In the context of UFS the entire UniPro protocol stack shall be viewed as a black box model (see Figure 9-1) to the greatest extent possible. The following sections therefore only:

- Specify number and type of the required interfaces between UFS and UniPro
- Specify the mapping between UFS and UniPro addressing scheme
- Select optional features and definable attributes of the UniPro specification

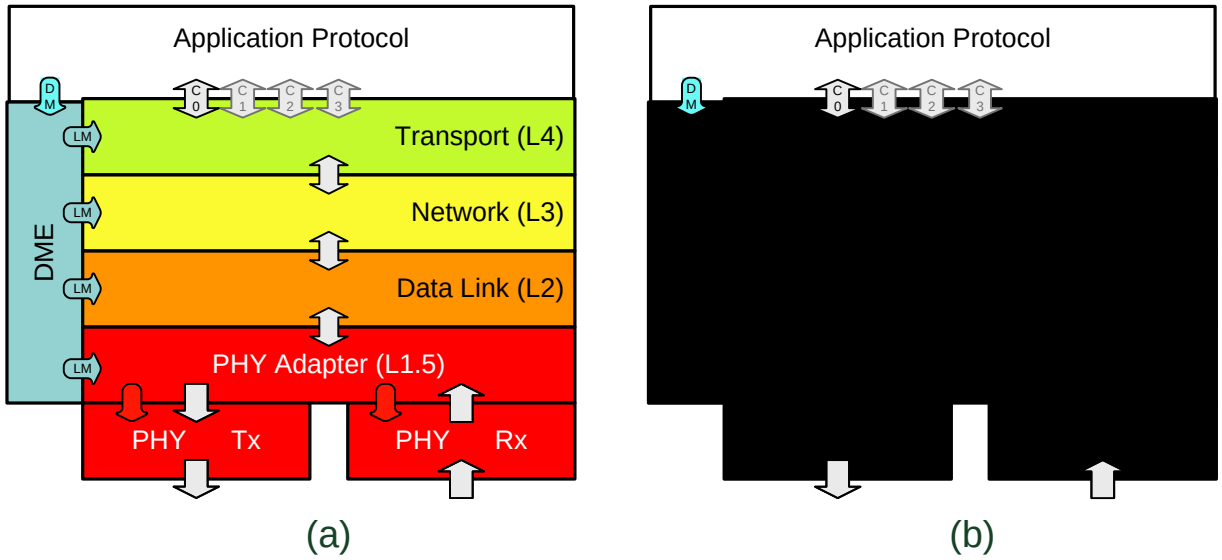


Figure 9-1 — UniPro internal layering view (a) and UniPro Black Box view (b)

9.3 UniPro/UFS Transport Protocol Interface (Data Plane)

UniPro provides CPorts as conceptual interfaces to applications or protocol layers on top of UniPro. CPorts can be viewed as instantiations of the T_CO_SAPs as specified in 8.8 of the UniPro specification. The physical implementation of a T_CO_SAP was deliberately not defined in MIPI as implementers should be free to choose, e.g., a SW implementations of higher UniPro layers, HW implementations based on buffering per CPort or DMA channels per CPort, etc.

A Service Access Point (SAP) provides Service Primitives (SP) which can be used by specifications of applications or protocols as UFS on top of UniPro to define their interactions. For more information on the concept of SAP/SP in protocol specifications please refer to Annex C of the UniPro specification.

The T_CO_SAP provides the following core data transfer service primitives (see UniPro specification, 8.8.1):

- T_CO_DATA.req(MessageFragment, EOM)
  - Issued by service user of UniPro to send a message (Fragment)

    NOTE Whenever a UFS layer requests the UIC layer to transfer data that UFS layer shall ensure that the last fragment of said data will be transmitted with the EOM flag set. One way to ensure such a behavior is for the UFS layer to invoke this UIC data transfer service primitive only once per atomic protocol data unit (e.g., once per UFS Transport Layer 'UPIU') with the EOM flag set to 'true' always.
- T_CO_DATA.cnf_L(L4CPortResultCode)
  - Issued by UniPro to report the result of a Message (Fragment) transfer request
- T_CO_DATA.ind(MessageFragment, EOM, SOM, MsgStatus)
  - Issued by UniPro to deliver a received Message (Fragment) towards the service user
  - EOM informs the service user that this is the last Message Fragment (EndOfMessage)
  - SOM informs the service user that this is the first Message Fragment (StartOfMessage)
- T_CO_DATA.rsp_L( )
  - Issued by a service user of UniPro to report readiness to receive the next Message (Fragment)

9.3.1 Flow control

UFS will not make use of the End-to-End Flow Control feature of UniPro for data communication as the UFS Transport Layer already avoids any overflow by a strict client-server communication model, tagged command queues and Device side throttling of Data transfers. Therefore UFS will not use the T_CO_FLOWCONTROL service primitive of UniPro and hence does not require its implementation.

9.3.2 Object sizes

A UniPro Message can be of any size and its content is not interpreted in any way by UniPro. Messages can be delivered from/to UniPro as multiple Message Fragments.

A Message Fragment is a portion of a Message that can be passed to, or received by, a CPort. Received Fragments are not generally identical to transmitted Fragments. Message Fragments may or may not carry an End-of-Message (EoM) flag.

A Message Fragment shall have maximum of T_MTU bytes to avoid further splitting in lower layers.

9.4 UniPro/UFS Control Interface (Control Plane)

UniPro provides access to its Device Management Entity (DME) via a Service Access Point (DME SAP) with the following services exposed to UFS allowing control of properties and the behavior of UniPro:

DME Configuration Primitives

- DME_GET / DME_SET
  - Provide read/write access to all UniPro and M-PHY attributes of the local UniPort
- DME_PEER_GET (optional) / DME_PEER_SET (optional)
  - Provide read/write access to all UniPro and M-PHY attributes of the peer UniPort

NOTE The order in which attributes are set is in some cases relevant for UniPro's correct operation. Therefore higher UFS layers shall preserve the ordering of DME Configuration Primitives invocations by UFS applications. If internally generated by UFS itself, DME Configuration Primitives shall be issued correctly ordered as defined by the UniPro specification.

DME Control Primitives

- DME_POWERON (optional) / DME_POWEROFF (optional)
  - Allow to power up or power down all UniPro layers (L1.5 through L4)
- DME_ENABLE
  - Allow enabling of the entire local UniPro stack (UniPro L1.5 -L4)
- DME_RESET
  - Allows to reset the entire local UniPro stack (UniPro L1.5-L4)
- DME_ENDPOINTRESET
  - Allows sending an end-point reset request command to a link end point.
- DME_LINKSTARTUP
  - Allows locally to startup the Link and informs about remote link startup invocation
- DME_HIBERNATE_ENTER / DME_HIBERNATE_EXIT
  - Allow to put the entire Link into HIBERNATE power mode and to wake the Link up
    - Affects the local and the peer UniPort (UniPro L1.5-L4 and M-PHY)

  NOTE After exit from Hibernate all UniPro Transport Layer attributes (including L4 T_PeerDeviceID, L4 T_PeerCPortID, L4 T_ConnectionState, etc.) will be reset to their reset values. All required attributes must be restored properly on both ends before communication can resume.
- DME_POWERMODE
  - Allows to change the power mode of one or both directions of the M-PHY Link
- DME_TEST_MODE (optional)
  - Allows to set the peer UniPro Device on the Link in a specific test mode
- DME_LINKLOST
  - Indication of the UniPro stack towards higher layers that the Link has been lost
- DME_ERROR
  - Indication of the UniPro stack towards higher layers that an error condition has been encountered in one of the UniPro Layers

9.5 UniPro/UFS Transport Protocol Address Mapping

UniPro has fundamentally two levels of addressing to control the exchange of information between remote UniPro entities

Network Layer (L3): Device ID, lowest level of addressability

- Provided for future UniPro networks of devices. During connection establishment the side creating a connection uses this value to select the physical entity on the remote end of the connection. It shall be considered static for the lifetime of this connection.

Transport Layer (L4): CPort ID, highest level of end-to-end addressability

- During connection establishment the side creating a connection uses this value to select the logical entity inside the targeted UniPro device on the remote end of the connection. It shall be considered static for the lifetime of this connection.

UFS adopts the addressing notation of the SCSI Architecture Model [SAM] based on Nexus definition.

The Nexus (I_T_L_Q) is composed of:

- Initiator Port Identifier (I)
- Target Port Identifier (T)
- Logical Unit Number (L)
- Command Identifier (Q).

An I_T_L_Q Nexus uniquely defines a specific command slot (Q) inside a specific Logical Unit (L) connected to a specific Device Target Port (T) accessed through a specific Host Initiator Port (I).

UFS Interconnect Layer addresses (Device ID and CPort ID) are only related to the I_T part of the Nexus.

This standard only requires and uses a single UniPro CPort on the device side and on the host side.

Mapping Rules

- UFS Initiator Port Identifier (I) and UFS Target Port Identifier (T) shall be 16 bit wide each and
  - UFS Initiator/Target Port Identifier shall contain the UniPro Network Layer Device ID of the entity (host or device) containing said UFS Port
    - The UniPro Network Layer Device ID reset value shall be 0 for the Host
    - The UniPro Network Layer Device ID reset value shall be 1 for the Device
  - UFS Initiator/Target Port Identifier contains the UniPro Transport Layer CPort ID which said UFS Port uses to communicate to the remote entity
    - The UniPro Transport Layer CPort ID reset value shall be 0 for the Host
    - The UniPro Transport Layer CPort ID reset value shall be 0 for the Device
- UFS Initiator Port Identifier shall contain the Initiator ID (IID).

Table 9-1 defines the Initiator Port Identifier (I) and Target Port Identifier (T) for UFS.

9.5 UniPro/UFS Transport Protocol Address Mapping (cont'd)

Table 9-1 — UFS Initiator and Target Port Identifiers

UFS Port	UFS Port IDs															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Initiator Port Identifier (I)	Device ID = 000 0000b							CPort ID = 0 0000b					IID			
Target Port Identifier (T)	Device ID = 000 0001b							CPort ID = 0 0000b					0000b			

The single UniPro connection between the UTP layer of a UFS host and the UTP layer of a UFS device can be uniquely identified by the UFS I_T Nexus above.

NOTE The UFS I_T Nexus elements (Device IDs and CPort IDs) can be modified by the Host after reset using the DME Service Primitives:

- The "I" element may be modified by the Host using the DME_SET primitive
- The "T" element may be modified by the Host using DME_PEER_SET primitive

All attributes of the CPort on the Host side (including, e.g., "T_ConnectionState") can be checked and modified by the Host using the DME_GET and DME_SET primitives after reset.

All attributes of the CPort on the Device side (including, e.g., the "T_ConnectionState") can be checked and modified by the Host using the DME_PEER_GET and DME_PEER_SET primitives after reset.

9.6 Options and Tunable Parameters of UniPro

MIPI UniPro has been designed as a versatile protocol specification and as such has several options and parameters which an application like UFS should specify for its specialized UniPro usage scenario. Annex E of the UniPro specification details all of the possible choices.

The remaining sections of this chapter define the specific requirements towards these options and parameters for this version of UFS standard. They apply to UniPro implementations for the UFS host side as well as to UniPro implementations for the UFS device side if not explicitly stated otherwise below.

9.6.1 UniPro PHY Adapter

For MIPI M-PHY related attribute values and implementation options as defined by UFS refer to 8.6, UFS PHY Attributes.

9.6.2 UniPro Data Link Layer

- Shall implement the Data Link Layer Traffic Class "Best Effort" (TC 0)
- Data Link Layer Traffic Class 1 (TC1: 'Low Latency') is not required
- TX preemption capability is not required
- Shall provide at least DL_MTU bytes of Data Link Layer RX and TX buffering
- Shall support transmission and reception of maximum sized L2 frames (DL_MTU)

9.6.3 UniPro Network Layer

- Shall support transmission and reception of maximum sized L3 packets (N_MTU)

9.6.4 UniPro Transport Layer

- UFS Hosts and UFS Devices shall implement at least 1 CPort

  NOTE This standard only requires and uses a single CPort on either side of the Link.

- UFS does not mandate any CPort arbitration scheme beyond the UniPro default if more than one CPort is implemented
- Shall support the UniPro Test Feature
- UFS does not require the UniPro End-to-End Flow Control mechanism
  - UFS will not use ‘Controlled Segment Dropping’ (CSD)
    - Hence CSD shall be disabled
- UFS will not use "CPort Safety Valve" (CSV). Hence, CSV shall be disabled.
- Shall support transmission and reception of maximum sized L4 segments (T_MTU) .

9.6.5 UniPro Device Management Entity Transport Layer

DME service primitives provide the means to

- retrieve or set attributes,
- control the reset and run mode of the entire UniPro protocol stack.

UFS Hosts and UFS Devices shall implement the following DME service primitives:

- DME_GET, DME_SET,
- DME_ENABLE,
- DME_RESET, DME_ENDPOINTRESET,
- DME_LINKSTARTUP, DME_LINKLOST,
- DME_HIBERNATE_ENTER, DME_HIBERNATE_EXIT,
- DME_POWERMODE,
- DME_ERROR.

UFS Hosts

- shall implement DME_PEER_GET primitive and DME_PEER_SET primitive, which are optional in [MIPI-UniPro].

UFS Devices

- shall not use DME_SET primitive to modify the local PA_PWRMode attribute,
- shall use DME_RESET only in the following cases: at power-on or hardware reset, or after a DME_LINKLOST.ind,
- shall not use the following primitives
  - DME_PEER_GET.req, DME_PEER_SET.req,
  - DME_POWERON.req, DME_POWEROFF.req,
  - DME_ENDPOINTRESET.req,
  - DME_HIBERNATE_ENTER.req, DME_HIBERNATE_EXIT.req,
  - DME_POWERMODE.req, DME_TEST_MODE.req.

9.6.6 UniPro Attributes

To optimize the UFS Boot procedure the UFS UIC implementation shall use the default reset values for all UniPro Attributes as defined by the MIPI UniPro specification. As an exception to this, the reset values of Network Layer Attributes and specific Attributes for *CPort 0* shall reflect the settings which have been defined in the sections above and therefore shall contain the values as depicted in Table 9-2.

Table -2 — UniPro Attribute

UniPro Attribute Name	UFS Host Reset Value	UFS Device Reset Value
N_DeviceID	0	1
N_DeviceID_valid	TRUE	TRUE
T_PeerDeviceID	1	0
T_PeerCPortID	0	0
T_CPortFlags	6 (E2E_FC off, CSD off, CSV off)	6 (E2E_FC off, CSD off, CSV off)
T_ConnectionState	1 (CONNECTED)	1 (CONNECTED)
T_TrafficClass	0	0
PA_MaxDataLanes	2	2
PA_AvailTxDataLanes	1, 2	1, 2
PA_AvailRxDataLanes	1, 2	1, 2
NOTE A UFS device will support either: one TX lane and one RX lane or two TX lanes and two RX lanes		

.

10 UFS TRANSPORT PROTOCOL (UTP) LAYER

10.1 Overview

The SCSI Architecture Model [SAM] is used as the general architectural model for UTP, and the SAM Task Management functions for task management. A task is generally a SCSI command or service request. While the model uses the SCSI command set as the command set, it is not necessary to use SCSI commands exclusively.

The SAM architecture is a client-server model or more commonly a request-response architecture. Clients are called Initiator devices and servers are called Target devices. Initiator devices and Target devices are mapped into UFS physical network devices. An Initiator device issues commands or service requests to a Target device that will perform the service requested. A Target device is a UFS device. A UFS device will contain one or more Logical Units. A Logical Unit is an independent processing entity within the device.

A client request is directed to a single Logical Unit within a device. A Logical Unit will receive and process the client command or request. Each Logical Unit has an address within the Target device called a Logical Unit Number (LUN).

Communication between the Initiator device and Target device is divided into a series of messages. These messages are formatted into UFS Protocol Information Units (UPIU) as defined within this standard. There are a number of different UPIU types defined. All UPIU structures contain a common header area at the beginning of the data structure (lowest address). The remaining fields of the structure vary according to the type of UPIU.

A Task is a command or sequence of actions that perform a requested service. A Logical Unit contains a task queue that will support the processing of one or more Tasks. The Task queue is managed by the Logical Unit. A unique Task Tag is generated by the Initiator device when building the Task. This Task Tag is used by the Target device and the Initiator device to distinguish between multiple Tasks. All transactions and sequences associated with a particular Task will contain that Task Tag in the transaction associated data structures.

Command structures consist of Command Descriptor Blocks (CDB) that contain a command opcode and related parameters, flags and attributes. The description of the CDB content and structure are defined in detail in [SAM], [SBC] and [SPC] INCITS T10 draft standards.

A command transaction consists of a Command, an optional Data Phase, and a Status Phase. These transactions are represented in the form of UPIU structures. The Command Phase delivers the command information and supporting parameters from the Initiator device to the Target device. If a Data Phase is required, the direction of data flow is relative to the Initiator device. A data WRITE travels from Initiator device to Target device. A data READ travels from Target device to Initiator device. At the completion of the command, the Target device will deliver a response to the Initiator device during the Status Phase. The response will contain the status and a UFS response status indicating successful completion or failure of the command. If an error is indicated the response will contain additional detailed UFS error information.

10.2 UTP and UniPro Specific Overview

UTP will deliver commands, data and responses as standard message packets (T_SDU) over the UniPro network.

The UFS transactions will be grouped into data structures called UFS Protocol Information Units (UPIU).

There are UPIU's defined for UFS SCSI commands, responses, data in and data out, task management, utility functions, vendor functions, transaction synchronization and control. The list is extensible for future additions.

For enumeration and configuration, UFS supports a system of Descriptors, Attributes and Flags that define and control the specifics of the device, including operating characteristics, interfaces, number of logical units, operating speeds, power profiles, etc. The system is a hierarchical tree of related elements. It is open to be expanded.

10.2.1 Phases

The SCSI based Command protocol requires that the UPIU packets follow the transitions required to execute a command. Briefly, a command execution requires the sending of a COMMAND UPIU, zero or more DATA IN UPIU or DATA OUT UPIU packets and terminates with a RESPONSE UPIU that contains the status.

10.2.2 Data Pacing

A device may have limited memory resources for buffering or limited processing throughput. During a Command that requires a large Data Out transaction the Target device can pace the Data Out phase by sending a READY TO TRANSFER UPIU when it is ready for the next DATA OUT UPIU. In addition, the READY TO TRANSFER UPIU contains an embedded transfer context that is used to initiate a DMA transfer on a per packet basis at the host.

During the Data In phase, no READY TO TRANSFER UPIU is required as the host has the ability to specify the size of the Data In transfer and thereby is able to allocate in advance the appropriate memory resources for the incoming data. A device issued DATA IN UPIU packet also contains an embedded DMA context that can be used to initiate a DMA transfer on a per packet basis.

10.2.3 UniPro

In keeping with the requirements of the UniPro Protocol the UFS Initiator device and Target device will divide its transactions into UniPro messages that will contain UPIU's. UniPro messages can handle T_SDU messages of theoretically unlimited size. UFS will impose a practicable limit on the maximum T_SDU message size. The limit is 65600 bytes which includes UPIU header, optional extended headers area and data segment. The minimum message size is determined by the basic header format, which is 32 bytes. There is a possibility that in the future this value will increase to allow a larger data segment area.

10.3 UFS Transport Protocol Transactions Overview

UFS transactions consist of packets called UFS Protocol Information Units (UPIU) that travel between devices on the UniPro bus. A transaction begins between an Initiator device and a Target device in the form of a Request-Response operation. The Initiator device starts the sequence of transactions by sending a request to a Target device and logical unit. The Target device will then respond with a series of transactions that eventually end in a response transaction.

All UFS UPIU's consist of a single basic header segment, transaction specific fields, possibly one or more extended header segments and zero or more data segments.

A basic header segment has a fixed length of 12 bytes. The minimum UPIU size is 32 bytes which includes a basic header segment and transaction specific fields.

The maximum UPIU size is defined as being 65600 bytes.

The UPIU format is flexible enough to be easily extended to support future transactions and larger data segments and will allow the application of this protocol to network protocols other than UniPro.

10.4 Service Delivery Subsystem

The Service Delivery Subsystem is an I/O system that transmits service requests and responses between Initiator devices and the Target device connected via a physical or logical bus. The UFS UTP attempts to define a protocol that is independent of the Service Delivery Subsystem. This will allow for the easy porting of UTP to different Service Delivery Subsystems.

Currently, UFS is using the MIPI UniPro bus and the MIPI M-PHY® as the Service Delivery Subsystem. For convenience and to aid in better understanding, portions of this standard directly reference UniPro and M-PHY®. Regardless of these references, the UTP protocol is independent of the Service Delivery Subsystem and should be able to port to other I/O systems.

UPIU structures will be handed off to MIPI UniPro as UniPro Service Data Units (T_SDU). Currently, the UniPro T_SDU requires no additional headers or trailer wrapped around the UPIU structure. This means that the T_SDU size will be exactly the UPIU size. The minimum size T_SDU will be 32 bytes. The maximum T_SDU size will be 65600 bytes.

10.5 UPIU Transactions

Every UPIU data structure contains a Transaction Code. This code defines the content and implied function or use of the UPIU data structure. Table 10-1 lists currently defined transaction codes.

Table 10-1 — UPIU Transaction Codes

Initiator To Target	Transaction Code	Target to Initiator	Transaction Code
NOP OUT	00 0000b	NOP IN	10 0000b
COMMAND	00 0001b	RESPONSE	10 0001b
DATA OUT	00 0010b	DATA IN	10 0010b
TASK MANAGEMENT REQUEST	00 0100b	TASK MANAGEMENT RESPONSE	10 0100b
Reserved	01 0001b	READY TO TRANSFER	11 0001b
QUERY REQUEST	01 0110b	QUERY RESPONSE	11 0110b
Reserved	01 1111b	REJECT UPIU	11 1111b
Reserved	Others	Reserved	Others
NOTE 1 Bit 5 of the Transaction Code indicates the direction of flow and the originator of the UPIU: when equal '0' the originator is the Initiator device, when equal '1' the originator is the Target device.			

10.5 UPIU Transactions (cont'd)

Table 10-2 — UPIU Transaction Code Definitions

UPIU Data Structure	Description
NOP Out	The NOP Out transaction acts as a ping from an initiator device to a target device. It can be used to check for a connection path to a device.
NOP In	The NOP In transaction is a target response to an initiator device when responding to a NOP Out request.
Command	The Command transaction originates in the Initiator device and is sent to a logical unit within a Target device. A COMMAND UPIU will contain a Command Descriptor Block as the command and the command parameters. This represents the COMMAND phase of the command.
Response	The Response transaction originates in the Target device and is sent back to the Initiator device. A RESPONSE UPIU will contain a command specific operation status and other response information. This represents the STATUS phase of the command.
Data Out	The Data Out transaction originates in the Initiator device and is used to send data from the Initiator device to the Target device. This represents the DATA OUT phase of a command.
Data In	The Data In transaction originates in the Target device and is used to send data from the Target to the Initiator device. This represents the DATA IN phase of a command.
Task Management Request	This transaction type carries SCSI Architecture Model (SAM) task management function requests originating at the Initiator device and terminating at the Target device. The standard functions are defined by [SAM].
Task Management Response	This transaction type carries SCSI Architecture Model (SAM) task management function responses originating in the Target device and terminating at the Initiator device.
Ready To Transfer	The Target device will send a Ready To Transfer transaction when it is ready to receive the next DATA OUT UPIU and has sufficient buffer space to receive the data. The Target device can send multiple Ready To Transfer UPIU if it has buffer space to receive multiple DATA OUT UPIU packets. The READY TO TRANSFER UPIU contains a DMA context and can be used to setup and trigger a DMA action within a host controller.
Query Request	This transaction originates in the Initiator device and is used to request descriptor data from the Target device. This transaction is defined outside of the Command and Task Management functions and is defined exclusively by UFS.
Query Response	This transaction originates in the Target device and provides requested descriptor information to the Initiator device in response of the Query Request transaction. This transaction is defined outside of the Command and Task Management functions and is defined exclusively by UFS.
Reject	The Reject transaction originates in the Target device and is sent back to the Initiator device. A REJECT UPIU is generated when Target device is not able to interpret and/or execute a UPIU received from the Initiator device due to wrong values in some of its fields.

UFS devices are able to process only either a NOP OUT or a QUERY REQUEST at any point of time.

10.6 General UFS Protocol Information Unit Format

Table 10-3 represents the general structure of a UPIU. All UPIU's will contain a fixed size and location basic header and additional fields as required to support the transaction type.

Table 10-3 — General format of the UFS Protocol Information Unit

General UPIU Format			
0 Transaction Type	1 Flags	2 LUN	3 Task Tag
4 Reserved / Command Set Type	5 Query Function / Task Manag. Function	6 Response	7 Status
8 Total EHS Length	9 Device Information	10 (MSB) / 11 (LSB) Data Segment Length	
12	13	14	15
16	Transaction Specific Fields		19
20			23
24			27
28			31
k	k+1	k+2	k+3
Extra Header Segment (EHS) 1			
…			
j	j+1	j+2	j+3
Extra Header Segment (EHS) N			
Header E2ECRC (omit if HD=0)			
Data Segment			
Data E2ECRC (omit if DD=0)			
NOTE 1 Extra Header Segments are not used in this standard, therefore the Total EHS Length value shall be set to zero.			

10.6.1 Overview

UPIU total size will vary depending upon the UPIU transaction type but all UPIU sizes will be an integer multiple of 32-bits, meaning they will be addressed on a 4-byte boundary. If the aggregation of data and header segments does not end on a 32-bit boundary then additional padding will be added to round up the UPIU to the next 32-bit, 4-byte address boundary.

The UPIU size can be fixed or variable depending upon the Transaction Type field and extension flags. Some Transaction Types will have different lengths for the same code others will always be a fixed size. In addition, any UPIU can be extended if necessary to include extra header and data segments. The general format allows for extension and has flags and size fields defined within the structure to indicate to the processing entity where the extension areas are located within the structure and their size (not including padding) and in some cases the type of extension data.

10.6.2 Basic Header Format

This is the format of the basic header contained within every UPIU structure. This data packet will be sent between Initiator devices and Target devices and will be part of a larger function specific UPIU. There is enough information in this header to allow the Initiator device or the Target device to track the destination and the source, the function request, if additional data and parameters are required and whether they are included in this UPIU or will follow in subsequent UPIU's.

The smallest sized UPIU is currently defined to have 32 bytes. The 32 bytes area will contain the basic header plus additional fields. This means that the smallest datum sent over the Service Delivery Subsystem will be 32 bytes.

Table 10-4 — Basic Header Format

Basic UPIU Header Format				
Transaction Type		Flags	LUN	Task Tag
Initiator ID	Command Set Type	Query Function, Task Manag. Function	Response	Status
Total EHS Length		Device Information	Data Segment Length	

The basic header formats are defined as follows:

a) Transaction Type

The Transaction Type indicates the type of request or response contained within the data structure. The Transaction Type contains an opcode as defined in 10.4.1.

Table 10-5 — Transaction Type Format

Transaction Type Bits							
7	6	5	4	3	2	1	0
HD	DD	Transaction Code					

b) HD

The HD bit when set to '1' specifies that an end-to-end CRC of all Header Segments is included within the UPIU. The CRC fields include all fields within the header area. The CRC is placed at the 32-bit word location following the header.

End-to-end CRC is not supported in this version of the standard, therefore HD shall be '0'.

10.6.2 Basic Header Format (cont'd)

c) DD

The DD bit when set to '1' specifies that an end-to-end CRC of the Data Segment is included with the UPIU. The 32-bit CRC is calculated over all the fields within the Data Segment. The 32-bit CRC word is placed at the end of the Data Segment. This will be the last word location of the UPIU.

End-to-end CRC is not supported in this version of the standard, therefore DD shall be '0'.

d) **Transaction Code**

The Transaction Code indicates the operation that is represented within the data fields of the UPIU and the number and location of the defined fields within the UPIU.

e) Flags

The content of the Flags field vary with the Transaction Type opcode [(1)].

Table 10-6 — UPIU Flags

UPIU Type	Operational Flags				Rsvd	CP[(2)]	Task Attribute	
	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
NOP Out	-	-	-	-	-	-	-	-
NOP In	-	-	-	-	-	-	-	-
Command	-	R	W	-	-	CP[(2)]	ATTR	
Response	-	O	U	D	-	-	-	-
Data Out	-	-	-	-	-	-	-	-
Data In	-	-	-	-	-	-	-	-
Ready to Transfer	-	-	-	-	-	-	-	-
Reject	-	-	-	-	-	-	-	-
Query Request	-	-	-	-	-	-	-	-
Query Response	-	-	-	-	-	-	-	-
Task Management Request	-	-	-	-	-	-	-	-
Task Management Response	-	-	-	-	-	-	-	-

NOTE 1 "-" denotes reserved values.

NOTE 2 CP = Command Priority

Table 10-7 — Task Attribute definition

Task Attribute	Bit 1	Bit 0
Simple	0	0
Ordered	0	1
Head of Queue	1	0
ACA (Not Used)	1	1

10.6.2 Basic Header Format (cont'd)

f) Response

If a response is required from a Target device, this field indicates whether the requested function succeeded or failed. This field is reserved in UPIU transactions from Initiator device to Target device.

Table 10-8 — UTP Response Values

Opcode	Response Description
00h	Target Success
01h	Target Failure
02h-7Fh	Reserved
80h-FFh	Vendor Specific

g) Status

This field contains the SCSI status (as defined in [SAM]) if the transaction is a RESPONSE UPIU for a COMMAND UPIU with Command Set Type = 00h (SCSI Command). Otherwise it contains an opcode specific status or it is reserved.

h) Reserved

All fields marked as reserved shall contain a value of zero.

i) LUN

This field contains the Logical Unit Number to which a request is targeted. Target devices will contain at a minimum one logical unit numbered unit 0. This field is generated by the Initiator device and maintained by the Target device and Initiator device for all UPIU transactions relating to a single request or task.

j) Task Tag

The Task Tag is generated by the Initiator device when creating a task request. This field will be maintained by the Initiator device and Target device for all UPIU transactions relating to a single task. The Initiator device will contain a register or variable that represents the Task Tag value. The Initiator device will generate unique Task Tag by incrementing the internal variable when creating a new task request. When a task request is made up of or generates a series of UPIU transactions, all UPIU will contain the same value in the Task Tag field.

In particular, the same Task Tag value shall be maintained for the UPIU grouped in each row of Table 10-9.

Table 10-9 — UPIU associated to a single task

Initiator UPIU	Target UPIU
NOP Out	NOP In
Command, Data Out	Ready to Transfer, Response
Command	Data In, Response
Task Management Request	Task Management Response
Query Request	Query Response

10.6.2 Basic Header Format (cont'd)

k) Initiator ID (IID)

The Initiator ID field is 4 bits wide, encoded in bits [7:4] of byte 4. This field indicates the identity of the Initiator device who created the task request.

The Initiator ID shall be set to zero if there is only one Initiator device.

UFS devices shall support all sixteen Initiator ID values. The Initiator ID shall be encoded in this field by the Host when creating a request. This field is maintained by the Initiator device and Target device for all UPIU transactions relating to the same task.

All requests from the same Initiator device have the same IID value. Details about Initiator ID assignment are available in UFS HCI standard specification.

l) Command Set Type

Command set type field is 4 bits wide, encoded in bits [3:0] of byte 4. This field indicates the command set type the Command and RESPONSE UPIU is associated with. This field is defined for the COMMAND UPIU and the RESPONSE UPIU. This field is reserved in all other UPIU's. This field shall be used to indicate the type of command that is in the CDB field. The currently supported command types are listed in Table 10-10.

Table 10-10 — Command Set Type

Value	Description
0h	SCSI Command Set (SPC, SBC)
1h	UFS Specific Command Set
2h … 7h	Reserved
8h … Fh	Vendor Specific Set
NOTE JESD220C does not define any UFS specific command, therefore the value 1h is reserved for future use.	

m) Query Function, Task Manag. Function

This field is used in QUERY REQUEST and QUERY RESPONSE UPIU's to define the Query function, and in TASK MANAGEMENT REQUEST UPIU to define the task management function.

n) Device Information

This field provides device level information required by specific UFS functionality in all RESPONSE UPIU.

o) Total Extra Header Segment Length

This field represents the size in 32-bit units (DWORDS) of all Extra Header Segments contained within the UPIU. This field is used if additional header segments are needed. The length of each Extra Header Segment shall be a multiple of four bytes. The value in this field is the number of total number of bytes in all EHS divided by four.

$$Total\ EHS\ Length\ value = \text{INTEGER}\left(\frac{Total\ Extra\ Header\ Segment\ Bytes + 3}{4}\right)$$

The maximum size of all EHS fields combined is 1024 bytes. A value of zero in this field indicates that there are no EHS contained within the UPIU. Extra Header Segments are not used in this standard, therefore the value of this field shall be set zero.

10.6.2 Basic Header Format (cont'd)

p) Data Segment Length

The Data Segment Length field contains the number of valid bytes within the Data Segment of the UPIU. When the number of bytes within the Data Segment is not a multiple of four then the last 32-bit field will be padded with zeros to terminate on the next nearest 32-bit boundary. The number of 32-bit units (DWORDS) that make up the Data Segment is calculated as follows:

$$Data\ Segment\ DWORDS = INTEGER\left(\frac{Data\ Segment\ Length + 3}{4}\right)$$

Since the Data Segment Length field size is two bytes, the data segment can contain a maximum of 65535 valid bytes. A value of zero in this field indicates that there is no Data Segment within the UPIU.

q) Transaction Specific Fields

Additional fields as required by certain Transaction Codes are located within this area. For UTP, this area starts at byte address 12 within the UPIU and terminates on a 32 byte boundary at byte address 31. Since all UPIU contain a 12 byte Basic Header this leaves 20 bytes remaining for this area.

r) Extra Header Segments

The Extra Header Segments exist if the Total EHS Length field contains a non-zero value. For UTP, this area will start at byte address 32 within the UPIU. The UPIU may contain zero or more EHS. The length of each Extra Header Segment shall be a multiple of four bytes. This version of standard does not use EHS.

s) Data Segment

The Data Segment field starts on the next 32-bit (DWORD) boundary after the EHS area within the UPIU. For UTP, there are no EHS areas used meaning that the Data Segment will begin at byte address 32 (byte address 36 if E2ECRC is enabled) within the UPIU. The Data Segment will be a multiple of 32-bits, thereby making the UPIU packet size a multiple of 4 bytes. The Data Segment Length field can contain a value that is not a multiple of 4 bytes but the Data Segment area will be padded with zeros to fill to the next nearest 32-bit (DWORD) boundary. The Data Segment Length field indicates the number of valid bytes within the Data Segment.

10.7 UFS Protocol Information Units

This section provides the details of each UFS Protocol Information Unit.

10.7.1 COMMAND UPIU

The COMMAND UPIU contains the basic UPIU header plus additional information needed to specify a command. The Initiator device will generate this UPIU and send it to a Target device to request a SCSI command service to be performed by the Target.

Table 10-11 — COMMAND UPIU

COMMAND UPIU				
0 xx00 0001b		1 Flags	2 LUN	3 Task Tag
4 IID	Command Set Type	5 Reserved	6 Reserved	7 Reserved
8 Total EHS Length (00h)		9 Reserved	10 (MSB) Data Segment Length (0000h)	11 (LSB)
12 (MSB)		13	14	15 (LSB)
Expected Data Transfer Length				
16 CDB[0]		17 CDB[1]	18 CDB[2]	19 CDB[3]
20 CDB[4]		21 CDB[5]	22 CDB[6]	23 CDB[7]
24 CDB[8]		25 CDB[9]	26 CDB[10]	27 CDB[11]
28 CDB[12]		29 CDB[13]	30 CDB[14]	31 CDB[15]
Header E2ECRC (omit if HD=0)				

10.7.1.1 Basic Header

The first 12 bytes of the COMMAND UPIU contain the Basic Header as described in 10.6.2, Basic Header Format. Specific details are as follows:

a) Transaction Type

A type code value of xx00 0001b indicates a COMMAND UPIU.

b) Flags

Table 10-12 describes the flags used in COMMAND UPIU.

Table 10-12 — Flags definition for COMMAND UPIU

<table>
<tr><th>Flag</th><th>Description</th></tr>
<tr><td>Flags.R</td><td>A value of ‘1’ in the .R flag indicates that the command requires a data transfer (incoming data) from Target device to Initiator device. If .R is set to ‘1’ then .W shall be set to ‘0’. If .R and .W are set to ‘0’ then no data transfer is required for this command and the Expected Data Transfer Length field is ignored.</td></tr>
<tr><td>Flags.W</td><td>A value of ‘1’ in the .W flag indicates that the command requires a data transfer (outgoing data) from Initiator device to Target device. If .W is set to ‘1’ then .R shall be set to ‘0’. If .W and .R are set to ‘0’ then no data transfer is required for this command and the Expected Data Transfer Length field is ignored.</td></tr>
<tr><td>Flags.ATTR</td><td>The .ATTR field contains the task attribute value as defined by [SAM].
ATTR Definition
<table>
<tr><th>Task Attribute</th><th>Bit 1</th><th>Bit 0</th></tr>
<tr><td>Simple</td><td>0</td><td>0</td></tr>
<tr><td>Ordered</td><td>0</td><td>1</td></tr>
<tr><td>Head of Queue</td><td>1</td><td>0</td></tr>
<tr><td>ACA (Not Used)</td><td>1</td><td>1</td></tr>
</table>
The relative order of execution between Head of Queue commands is left for implementation.</td></tr>
<tr><td>Flags.CP</td><td>The .CP field indicates the Command Priority; see [SAM] for details. In UFS, the .CP field supports only two values whereas [SAM] allows a larger range.
This 1-bit field specifies the relative scheduling importance of a command having a Simple task attribute in relation to other commands having Simple task attributes already in the task set. If the command has a task attribute other than Simple then this field has no meaning
A task manager may use command priority to determine an ordering to process commands with the Simple task attribute within the task set.
A value of “1” indicates high priority. A value of “0” indicates no priority.</td></tr>
<tr><td colspan="2">NOTE The bit assignment of the Flags field is shown in Table 10-6.</td></tr>
</table>

10.7.1.1 Basic header (cont'd)

c) Data Segment Length

The Data Segment Length field shall contain zero as there is no Data Segment in this UPIU.

d) Expected Data Transfer Length

The Expected Data Transfer Length field contains a value that represents the number of bytes to be transferred that are required to complete the SCSI command request as indicated in the CDB (e.g., TRANSFER LENGTH, ALLOCATION LENGTH, PARAMETER LIST LENGTH, etc.). Data may be transferred from the Initiator device to the Target device or from the Target device to the Initiator device. This field is valid only if one of the Flags.W or Flags.R bits are set to '1'.

For a data transfer from the Initiator device to the Target device, the .W flag shall be set to '1' and the .R flag shall be set to '0'. The value in the Expected Data Transfer Length field represents the number of bytes that the Initiator device expects to send to the Target device.

For a data transfer from the Target device to the Initiator device, the .R flag shall be set to '1' and the .W flag shall be set to '0'. The value in the Expected Data Transfer Length field represents the number of bytes that the Initiator device expects to receive from the Target device.

When the COMMAND UPIU encodes a SCSI WRITE or SCSI READ command (specifically WRITE (6), READ (6), WRITE (10), READ (10), WRITE (16), or READ (16)), the value of this field shall be the product of the Logical Block Size (bLogicalBlockSize) and the TRANSFER LENGTH field of the CDB.

This model requires that the Initiator device will allocate sufficient buffer space to receive the full size of the data requested by a command that requires a Data In operation. That size measured in bytes shall be the value in Expected Data Transfer Length field. This requirement is important in order to realize the full throughput of the Data In phase without the use of additional handshaking UPIU's.

The Initiator device may request a Data Out size larger than the size of the receive buffer in the Target device. In this case, the Target device will pace the DATA OUT UPIU's by sending READY TO TRANSFER UPIU's as needed. The Initiator device will not send a DATA OUT UPIU before it receives a READY TO TRANSFER UPIU.

e) CDB

The CDB fields contain the Command Descriptor Block. This area is an array of 16 bytes that will contain a standard Command Descriptor Block as defined by one of the supported UFS Command Set Types. For SCSI commands, specifications such as [SPC] can be referenced. Up to a 16 byte CDB can be utilized. The CDB size is implicitly indicated by the group bits of the operation code field in CDB[0] for SCSI, which is the SCSI command operation code. If the CDB size is lower than 16 bytes the unused COMMAND UPIU bytes are defined as reserved. For other commands, the CDB size is dependent upon the command opcode.

f) Initiator ID (IID)

The Initiator ID field (bits [7:4] of byte 4) indicates the identity of the Initiator device who created the task request. See Initiator ID description in 10.6.2, Basic Header Format, for details.

g) Command Set Type

The Command Set Type field will specify an enumerated value that indicates which particular command set is used to define the command bytes in the CDB fields. See Command Set Type description for details.

10.7.2 RESPONSE UPIU

The RESPONSE UPIU contains the basic UPIU header plus additional information indicating the command and device level status resulting from the successful or failed execution of a command. The Target will generate this UPIU and send it to the Initiator device after it has completed the requested task.

Before terminating a command which requires Data-Out data transfer and before sending the RESPONSE UPIU, the Target device shall wait until it receives all DATA OUT UPIUs related to any outstanding READY TO TRANSFER UPIUs. Also, the Target device should stop sending READY TO TRANSFER UPIUs for the command which requires Data-Out data transfer and to be terminated.

Table 10-13 — RESPONSE UPIU

RESPONSE UPIU			
0 xx10 0001b	1 Flags	2 LUN	3 Task Tag
4 IID / Command Set Type	5 Reserved	6 Response	7 Status
8 Total EHS Length (00h)	9 Device Information	10 (MSB) / 11 (LSB) Data Segment Length	
12 (MSB) / 13 / 14 / 15 (LSB) Residual Transfer Count			
16 / 17 / 18 / 19 Reserved			
20 / 21 / 22 / 23 Reserved			
24 / 25 / 26 / 27 Reserved			
28 / 29 / 30 / 31 Reserved			
Header E2ECRC (omit if HD=0)			
k (MSB) / k+1 (LSB) Sense Data Length		k+2 Sense Data[0]	k+3 Sense Data[1]
…	…	…	…
k+16 Sense Data[14]	k+17 Sense Data[15]	k+18 Sense Data[16]	k+19 Sense Data[17]
Data E2ECRC (omit if DD=0)			
NOTE 1 k = 32 if HD = 0.			

10.7.2.1 Basic Header

The first 12 bytes of the RESPONSE UPIU contain the Basic Header as described in 10.6.2, Basic Header Format. Specific details are as follows:

a) Transaction Type

A type code value of xx10 0001b indicates a RESPONSE UPIU.

b) Flags

Table 10-14 describes the flags used in RESPONSE UPIU.

Table 10-14 — Flags definition for RESPONSE UPIU

Flag	Description
Flags.O	The Flags.O flag will be set to '1' to indicate that a data overflow occurred during the task execution: the Target device has more data bytes to transfer than the Initiator device requested. The Residual Transfer Count field will indicate the number of available bytes not transferred from the Target device to the Initiator device or vice versa. The Residual Transfer Count will be set to the value difference of the total number of bytes available to be transferred and the Expected Data Transfer Length value received in the COMMAND UPIU. See below for further explanation.
Flags.U	The Flags.U flag will be set to '1' to indicate that a data underflow occurred during the task execution: the Target device has less data bytes to transfer than the Initiator device requested. The Residual Transfer Count field will indicate the number of bytes that were not transferred from the Target device to the Initiator device or vice versa. The Residual Transfer Count will be set to the value difference of the Expected Data Transfer Length value received in the COMMAND UPIU and the actual number of bytes transferred. See below for further explanation.
Flags.D	The .D flag will be set to '1' to indicate that a UTP Data Out Mismatch error occurred during the task execution: the data buffer offset and/or the data transfer count parameter in the Data Out UPIU doesn't match the corresponding parameters in the RTT request. See 10.7.13 for further explanation.
NOTE The bit assignment of the Flags field is shown in Table 10-6.	

c) Initiator ID (IID)

The Initiator ID field (bits [7:4] of byte 4) indicates the identity of the Initiator device who created the task request. See Initiator ID description in section 10.6.2, Basic Header Format, for details.

d) Command Set Type

The Command Set Type field will specify an enumerated value that indicates which particular command set is used to define the command bytes in the CDB fields. See 10.6.2, Basic Header Format, for details.

e) Response

The Response field will contain the UFS response that indicates the UFS defined overall success or failure of the series of Command, Data and RESPONSE UPIU's that make up the execution of a task. See 10.6.2, Basic Header Format, for details.

10.7.2.1 Basic Header (cont'd)

f) Status

The Status field contains the command set specific status for a specific command issued by the Initiator device. The Status field is command set specific. The Command Set Type field will indicate with which command set the status is associated. Specific command sets may or may not define detailed extended status indicated as Sense Data. If the command requires extended status, that information will be stored in the Sense Data field.

1) SCSI Command Set Status

When the Command Set Type field indicates SCSI Command Set the Status field will contain the standard SPC defined SCSI status value. Possible values are listed in the table below. See the [SPC] or [SAM] for detailed definition of the status conditions.

A GOOD status indicates successful SCSI completion and therefore no Sense Data will be returned.

A status of CHECK CONDITION requires that the Data Segment contain Sense Data for the failed command.

Other status values may or may not return Sense Data. In this case a non-zero value in the Data Segment Length field indicates that this UPIU contains Sense Data in the Data Segment area.

'**M**' indicates mandatory implementation of this field and the value specified if fixed. '**O**' indicates that the support of this field is optional; if it is not supported then a value of zero should be inserted in the field otherwise the value will be indicated as described. n/a indicates "not applicable" to UFS.

Table 10-15 — SCSI Status Values

Opcode	Response Description	Use
00h	GOOD	M
02h	CHECK CONDITION	M
04h	CONDITION MET	n/a
08h	BUSY	M
18h	RESERVATION CONFLICT	O
28h	TASK SET FULL	M
30h	ACA ACTIVE	n/a
40h	TASK ABORTED	n/a

GOOD - This status indicates that the device has completed the command without error.

CHECK CONDITION - This status indicates that the device has completed the command with error or other actions are required to process the result. Valid Sense Data for the last command processed will be returned within the response UPIU when this status occurs.

CONDITION MET - Not used for UFS.

BUSY - This status indicates that the logical unit is busy. When the logical unit is unable to accept a command this status will be returned. Issuing the command at a later time is the standard recovery action.

RESERVATION CONFLICT - This status is returned when execution of the command will result in a conflict of an existing reservation. UFS may support reserving areas of the device depending upon the device type and capabilities.

TASK SET FULL - This status is returned when the logical unit cannot process the command due to a lack of resources such as task queue being full or memory needed for command execution is temporarily unavailable.

ACA ACTIVE - This status is returned when an ACA condition exists. See [SAM] for further definition.

TASK ABORTED - This status shall be returned when a command is aborted by a command or task management function on another I_T nexus and the Control mode page TAS bit is set to one. Since in UFS TAS bit is zero TASK ABORTED status codes will never occur.

10.7.2.1 Basic Header (cont'd)

g) Device Information

The Device Information field provides information at device level not necessarily related with the logical unit executing the command.

In general, the information is about events that have an evolution much slower than the regular commands, and for which the host response latency is not critical. The use of this field avoids the execution of a continuous polling on some UFS attributes.

Only bit 0 of the Device Information field is defined, all the others are reserved and shall be set to zero.

Bit	Name	Description
bit 0	EVENT_ALERT	Exception Event Alert 0b: All exception sources not active 1b: At least one exception source is active
Others	Reserved	

The exception sources include: background operations, dynamic capacity, system data pool, etc. See 13.4.11, Exception Events Mechanism, for details.

h) Data Segment Length

The Data Segment Length field will contain the number of valid bytes in the Data Segment.

In the RESPONSE UPIU the Data Segment will contain the Sense Data bytes and the Sense Data Length field.

When this field contains zero it indicates that there is no Data Segment area in the UPIU and therefore no Sense Data is returned.

This version of the standard, when the Command Set Type field indicates SCSI Command Set, the number of Sense Data bytes is 18, therefore this field will contain a value of 20 (18 bytes of Sense Data + 2 bytes for Sense Data Length = 20 bytes).

As stated previously, the Data Segment field size is located on a 32-bit (DWORD) boundary. The Data Segment Length field indicates the number of "valid" bytes in the Data Segment area and therefore its value may not be an integer multiple of four.

i) Residual Transfer Count

This field is valid only if one of the Flags.U or Flags.O fields are set to '1', otherwise this field will contain zero.

When the Flags.O field is set to '1' then this field indicates the number of bytes that were not transferred from/to the Initiator device because the Expected Data Transfer Length field contained a value that was lower than the Target device expected to transfer. In other words, the Target device has more bytes to receive/send to complete the request but the Initiator device is not expecting more than the amount indicated in the Expected Data Transfer Length. For example, the Initiator device may intentionally request less bytes than it knows the Target device has available to transfer, because it only needs the first N bytes.

10.7.2.1 Basic Header (cont'd)

i) Residual Transfer Count (cont'd)

When the Flags.U field is set to '1' then this field indicates the number of bytes that were not transferred from/to the Initiator because the Expected Data Transfer Length field contained a value that was higher than the available data bytes. In other words, the Target device has less bytes to receive/send than the Initiator is requesting to transfer. For example, the Initiator device may intentionally request more bytes than the Target device has to transfer when it does not know how many bytes the Target device actually has and it asks for the max or more than possible.

Table 10-16 — Flags and Residual Count Relationship

Flags.O	Flags.U	Residual Transfer Count	Description
0	0	0	Expected Data Length bytes transferred
1	0	N	Target device expected to send N more bytes to Initiator device
0	1	N	Initiator device expected to receive N more bytes from Target device
1	1	X	Illegal condition

j) Sense Data Fields

The Sense Data fields will contain additional information on error condition.

For SCSI command they will provide a copy of first 18 sense data bytes as defined for the fixed format sense data, which corresponds to Response Code value of 70h. See the following subsection for further details.

A successfully executed command will not normally need to return Sense Data, therefore in this case the Data Segment may be empty and the Data Segment Length may have a zero value.

The Sense Data fields will be padded with zeros to place the data on the next nearest 32-bit boundary if the length of valid Sense Data fields plus two is not a multiple of 32-bit.

k) SCSI Sense Data Fields

The Sense Data fields will contain standard 18 byte SPC defined sense data when using format for a Response Code value of 70h. See [SPC] for further information.

Sense Data consists of three levels of error codes, each in increasing detail. The purpose is to provide the application client a means to determine the cause of an error or exceptional condition at various levels of detail. The Sense Key provides a general category of what error or exceptional condition occurred and has caused the current command from successfully completing. Further and finer error detail is provided in the Additional Sense Code field (ASC). The Additional Sense Code Qualifier (ASCQ) field refines the error information even further. It is required to implement the Sense Key value when indicating an error or exceptional condition. It is not required to implement the ASC or ASCQ values not described in this document; a value of zero can be placed in these fields if the implementation does not require more refined error detail.

All SCSI commands that terminate in error or exceptional condition will automatically return Sense Data in the RESPONSE UPIU, relieving the host from issuing a subsequent REQUEST SENSE command to retrieve the additional sense error information.

10.7.2.1 Basic Header (cont'd)

l) Sense Data Length

The Sense Data Length field indicates the number of valid Sense Data bytes that follow. The Sense Data Length plus two may be less than the number of bytes contained in the Data Segment area, if padding bytes have been added to reach 32-bit boundary.

A successfully executed command will not normally need to return Sense Data, therefore in this case the Data Segment area may be empty and the Data Segment Length may have a zero value.

A command that terminated in error or an exception may or may not return Sense Data. If the Sense Data Length indicates a value of zero, then that error condition did not return Sense Data. A zero value in the Data Segment Length also indicates that no Sense Data was returned. Otherwise, the Sense Data Length will contain a value that indicates the number of additional bytes of Sense Data information.

1) SCSI Sense Data Length

The Sense Data Length field shall indicate a value of 18 when using the SCSI Command Set.

m) Sense Data Format

Table 10-17 describes the sense data structure that gives detailed error information about the previously executed SCSI command. Eighteen bytes are returned and the Additional Sense Length field is set to a value of ten.

'**M**' indicates mandatory implementation of this field and the value specified if fixed. '**O**' indicates that the support of this field is optional; if it is not supported then a value of zero should be inserted in the field otherwise the value will be indicated as described.

10.7.2.1 Basic Header (cont'd)

Table 10-17 — SCSI fixed format sense data

Byte	Bits	Name	Description	Use
0	7:7	VALID	A VALID bit set to one indicates that the INFORMATION field contains valid data. Default value = 0b	O
	6:0	RESPONSE CODE	Value of 70h for fixed format sense data response	M
1	7:0	Obsolete	Not used Default value = 00h	M
2	7:7	FILEMARK	File mark found This bit is reserved for UFS. Default value = 0b	M
	6:6	EOM	End of media detected This bit is reserved for UFS. Default value = 0b	M
	5:5	ILI	Incorrect length detected This bit is reserved for UFS. Default value = 0b	M
	4:4	Reserved	Default value = 0b	M
	3:0	SENSE KEY	SENSE KEY code is the general SCSI error code for previous command (see Table 10-18)	M
3:6	7:0	INFORMATION	Sense Information	O
7	7:0	ADDITIONAL SENSE LENGTH	Length in bytes of additional sense information Value = 10 (0Ah) indicating 10 additional bytes (bytes 8 through 17)	M
8:11	7:0	COMMAND-SPECIFIC INFORMATION	Command Specific Information This field is reserved for UFS. Default value = 00h	M
12	7:0	ASC	ADDITIONAL SENSE CODE is an additional, more specific error code (see SCSI specs)	M
13	7:0	ASCQ	ADDITIONAL SENSE CODE QUALIFIER qualifies the Additional Sense Code (see SCSI specs)	M
14	7:0	FRUC	FIELD REPLACEABLE UNIT CODE Default value = 00h	M
15	7:7	SKSV	SKSV bit indicates if the SENSE KEY SPECIFIC field contains valid information. This bit is reserved for UFS. Default value = 0b	M
	6:0	SENSE KEY SPECIFIC	Sense key specific information This field is reserved for UFS. Default value = 00 00 00h	M
16:17	7:0	SENSE KEY SPECIFIC		

10.7.2.1 Basic Header (cont'd)

The SENSE KEY is used for normal error handling during operation.

The ADDITIONAL SENSE CODE (ASC) and the ADDITIONAL SENSE CODE QUALIFIER (ASCQ) are mainly used for detailed diagnostic and logging (post-mortem) information. If the device server does not have further information related to the error or exception condition, these fields shall be set to zero. Generally, except for a certain few, they're not mandatory and they may be set to zero, which means no additional information provided. See [SPC] for a list of additional sense codes and additional sense code qualifiers.

n) Sense Key

The Sense Key value provides a means to categorize errors and exceptional conditions. The Sense Key indicates a particular type of error. The Additional Sense Code and Additional Sense Code Qualifier can be used to further detail and describe the condition that the Sense Key indicates. Sense Keys are specific to the action performed by a particular command.

10.7.2.1 Basic Header (cont'd)

Table 10-18 — Sense Key

Sense Key	
Value	**Description**
00h	NO SENSE – Indicates that there is no specific sense key information to be reported. This would be the result of a successfully executed command.
01h	RECOVERED ERROR – Indicates that the last command completed successfully after error recovery actions were performed by the device server. Further details may be determined by examining the additional sense bytes (ASC and ASCQ fields).
02h	NOT READY – Indicates that the logical unit addressed cannot be accessed at this time.
03h	MEDIUM ERROR – Indicates that the last command was unsuccessful due to a non-recoverable error condition due to a flaw in the media or failed error recovery.
04h	HARDWARE ERROR – Indicates that the the device server detected a non-recoverable hardware error.
05h	ILLEGAL REQUEST – Indicates that there was an illegal parameter value in a command descriptor block or within additional parameter data supplied with some commands. If the device server detects an invalid parameter in the command descriptor block then it shall terminate the command without altering the media.
06h	UNIT ATTENTION – Indicates that the unit has been reset or unexpectedly powered-on or that removable media has changed.
07h	DATA PROTECT – Indicates that a command that reads or writes the medium was attempted on a block that is protected from this operation. The read or write operation shall not be performed.
08h	BLANK CHECK - Indicates that blank or unformatted media was encountered while reading or writing.
09h	VENDOR SPECIFIC – This Sense Key is available for reporting vendor specific error or exceptional conditions.
0Ah	COPY ABORTED – Not applicable for UFS device. Reserved
0Bh	ABORTED COMMAND – Indicates that the device server aborted the execution of the command. The application client may be able to recover by retrying the command.
0Ch	Reserved
0Dh	VOLUME OVERFLOW - Indicates that a buffered peripheral device has reached the end-of-partition and data may remain in the buffer that has not been written to the medium.
0Eh	MISCOMPARE – Indicates that the source data did not match the data read from the media.
0Fh	RESERVED
NOTE 1	See [SAM] for further details.

10.7.3 DATA OUT UPIU

The DATA OUT UPIU contains the basic UPIU header plus additional information needed to manage the data out transfer. The data transfer flows from Initiator device to Target device (write). The DATA OUT UPIU will usually contain a data segment. It is possible to have a null DATA OUT UPIU: the Data Segment is empty and Data Segment Length value is zero.

The DATA OUT UPIU is sent in response to READY TO TRANSFER UPIU generated by a Target device, according to the rules detailed in 10.7.13, Data out transfer rules.

Table 10-19 — DATA OUT UPIU

DATA OUT UPIU			
0 xx00 0010b	1 Flags	2 LUN	3 Task Tag
4 IID / Reserved	5 Reserved	6 Reserved	7 Reserved
8 Total EHS Length (00h)	9 Reserved	10 (MSB) / 11 (LSB) Data Segment Length	
12 (MSB) / 13 / 14 / 15 (LSB) Data Buffer Offset			
16 (MSB) / 17 / 18 / 19 (LSB) Data Transfer Count			
20 / 21 / 22 / 23 Reserved			
24 / 25 / 26 / 27 Reserved			
28 / 29 / 30 / 31 Reserved			
Header E2ECRC (omit if HD=0)			
k Data[0]	k+1 Data[1]	k+2 Data[2]	k+3 Data[3]
…	…	…	…
k+ Length-4 Data[Length - 4]	k+ Length-3 Data[Length - 3]	k+ Length-2 Data[Length - 2]	k+ Length-1 Data[Length - 1]
Data E2ECRC (omit if DD=0)			
NOTE 1 k = 32 if HD = 0			

10.7.3.1 Basic Header

The first 12 bytes of the DATA OUT UPIU contain the Basic Header as described in 10.6.2, Basic Header Format. Specific details are as follows:

a) Transaction Type

A type code value of 02h indicates a DATA OUT UPIU.

b) Initiator ID (IID)

The Initiator ID field (bits [7:4] of byte 4) indicates the identity of the Initiator device who created the task request. See Initiator ID description in section 10.6.2, Basic Header Format, for details.

c) Data Segment Length

The Data Segment Length shall indicate the number of valid bytes within the Data Segment area, and it shall not include the number of padding bytes (if present).

d) Data Buffer Offset

The Data Buffer Offset field contains the offset of this UPIU data payload within the complete data transfer area. The sum of the Data Buffer Offset and the Data Segment Length shall not exceed the Expected Data Transfer Length that was indicated in the COMMAND UPIU.

This field permits out of order sequencing of the DATA OUT UPIU packets. Therefore the order of the DATA OUT UPIU packets do not have to be sequential.

NOTE Out of order sequencing will only occur if a UFS device supports it (bDataOrdering = 01h) and if this feature is enabled (bOutOfOrderDataEn = 01h).

When the DATA OUT UPIU is a part of a SCSI WRITE transaction [i.e., a transaction which started with a WRITE (6), a WRITE (10), or a WRITE (16) command], the value of this field shall be equal to an integer multiple of the Logical Block Size (bLogicalBlockSize).

e) Data Transfer Count

This field indicates the number of bytes that the Initiator device is transferring to the Target device in this UPIU. This value is the number of bytes that are contained within the Data Segment of the UPIU. The maximum number of bytes that can be transferred within a single DATA OUT UPIU packet is 65535 bytes. Therefore, multiple DATA OUT UPIU packets will need to be issued by the Initiator device if the Expected Data Transfer Length of the original command requires more than 65535 bytes to be transferred.

When the DATA OUT UPIU is a part of a SCSI WRITE transaction [i.e. a transaction which started with a WRITE (6), a WRITE (10), or a WRITE (16) command], the value of this field shall be equal to an integer multiple of the Logical Block Size (bLogicalBlockSize).

This field and the Data Segment Length field of the UPIU shall contain the same value. This field is intended to be used along with the Data Buffer Offset field as part of a DMA context.

10.7.3.1 Basic Header (cont'd)

f) Data Segment

This is the Data Segment area that contains the data payload.

The maximum data payload size that can be transferred within a single DATA OUT UPIU packet is 65535 bytes.

The Data Segment area always starts on a 32-bit (DWORD) boundary. The Data Segment area shall be entirely filled with data payload to a 32-bit (DWORD) boundary unless the UPIU is the one that transmits the last data portion. In this case, if necessary, the Data Segment area shall be padded out to the next nearest 32-bit boundary.

When the DATA OUT UPIU is a part of a SCSI WRITE transaction [i.e., a transaction which started with a WRITE (6), a WRITE (10), or a WRITE (16) command], the Data Segment area shall contain an integer number of logical blocks.

NOTE For out of order DATA OUT UPIUs, the last data portion may not be transmitted by the final UPIU.

g) Data out transfer example

Figure 10-1 shows an example of data transfer from the Initiator device to the Target device. In particular, the command processing requires the transfer of 577-byte data. The data transfer is done out of order: at the beginning the middle portion of the data, then the last portion and finally the first portion.

NOTE The second DATA OUT UPIU delivers the last portion of the data. The Data Segment in this UPIU has 65 bytes of valid data and three pad bytes. The Data Segment in the other UPIUs is fully filled (no pad bytes).

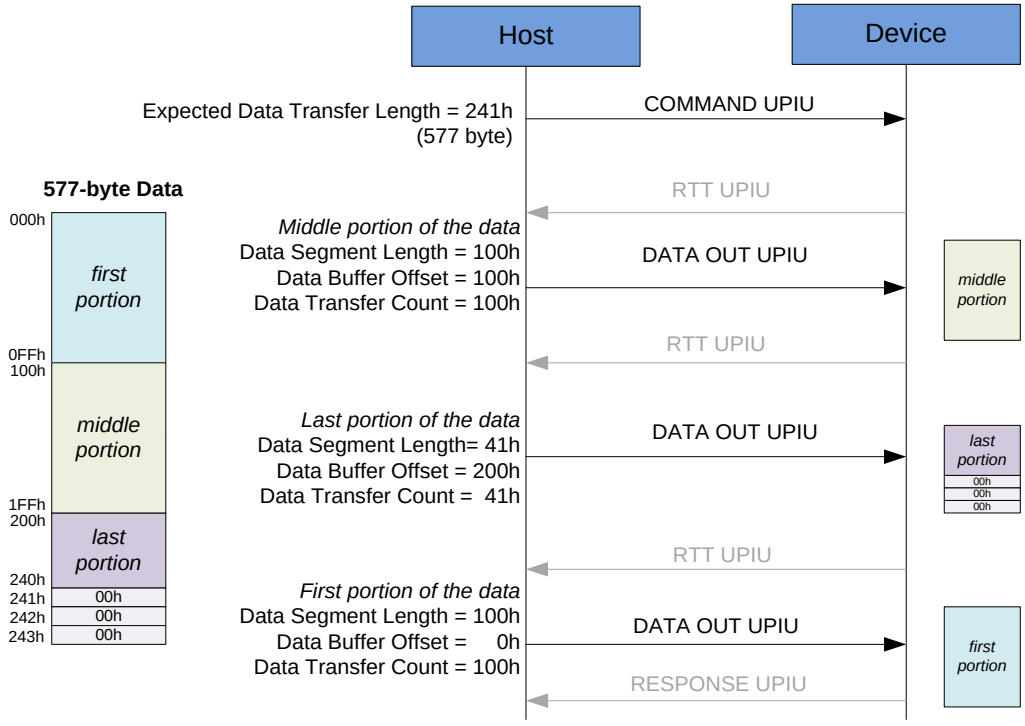


Figure 10-1 — Data out transfer example

10.7.4 DATA IN UPIU

The DATA IN UPIU contains the basic UPIU header plus additional information needed to manage the data in transfer. Data in flows from Target device to Initiator device (READ). The DATA IN UPIU will usually contain a data segment. It is possible to have a null DATA IN UPIU: the Data Segment is empty and Data Segment Length is 0.

Table 10-20 —DATA IN UPIU

DATA IN UPIU			
0 xx10 0010b	1 Flags	2 LUN	3 Task Tag
4 IID / Reserved	5 Reserved	6 Reserved	7 Reserved
8 Total EHS Length (00h)	9 Reserved	10 (MSB) / 11 (LSB) Data Segment Length	
12 (MSB) / 13 / 14 / 15 (LSB) Data Buffer Offset			
16 (MSB) / 17 / 18 / 19 (LSB) Data Transfer Count			
20 / 21 / 22 / 23 Reserved			
24 / 25 / 26 / 27 Reserved			
28 / 29 / 30 / 31 Reserved			
Header E2ECRC (omit if HD=0)			
k Data[0]	k+1 Data[1]	k+2 Data[2]	k+3 Data[3]
…	…	…	…
k+ Length-4 Data[Length -4]	k+ Length-3 Data[Length -3]	k+ Length-2 Data[Length -2]	k+ Length-1 Data[Length -1]
Data E2ECRC (omit if DD=0)			
NOTE 1 k = 32 if HD = 0.			

10.7.4 DATA IN UPIU (cont'd)

10.7.4.1 Basic Header

The first 12 bytes of the DATA IN UPIU contain the Basic Header as described in 10.6.2, Basic Header Format. Specific details are as follows:

a) Transaction Type

A type code value of xx10 0010b indicates a DATA IN UPIU.

b) Initiator ID (IID)

The Initiator ID field (bits [7:4] of byte 4) indicates the identity of the Initiator device who created the task request. See Initiator ID description in 10.6.2, Basic Header Format, for details.

c) Data Segment Length

The Data Segment Length shall indicate the number of valid bytes within the Data Segment area, and it shall not include the number of padding bytes (if present).

d) Data Buffer Offset

The Data Buffer Offset field contains the offset of this UPIU data payload within the complete data transfer area. The sum of the Data Buffer Offset and the Data Segment Length shall not exceed the Expected Data Transfer Length that was indicated in the COMMAND UPIU.

This field permits out of order sequencing of the DATA IN UPIU packets. Therefore the order of the SCSI In UPIU packets do not have to be sequential.

NOTE Out of order sequencing will only occur if a UFS device supports it (bDataOrdering = 01h) and if this feature is enabled (bOutOfOrderDataEn = 01h).

When the DATA IN UPIU is a part of a SCSI READ transaction [i.e. a transaction which started with a READ (6), a READ (10), or a READ (16) command], the value of this field shall be equal to an integer multiple of the Logical Block Size (bLogicalBlockSize).

e) Data Transfer Count

This field indicates the number of bytes that the Target device has placed in the UPIU Data Segment, for transfer back to the Initiator device. This value is the number of valid bytes that are contained within the Data Segment of this UPIU. The maximum number of bytes that can be transferred within a single DATA IN UPIU packet is 65535 bytes.

When the DATA IN UPIU is a part of a SCSI READ transaction [i.e. a transaction which started with a READ (6), a READ (10), or a READ (16) command], the value of this field shall be equal to an integer multiple of the Logical Block Size (bLogicalBlockSize).

This field and the Data Segment Length field of the UPIU shall contain the same value.

10.7.4.1 Basic Header (cont'd)

f) Data Segment

This is the Data Segment area that contains the data payload.

The maximum data payload size that can be transferred within a single DATA IN UPIU packet is 65535 bytes.

The Data Segment area always starts on a 32-bit (DWORD) boundary. The Data Segment area shall be entirely filled with data payload to a 32-bit (DWORD) boundary unless the UPIU is the one that transmits the last data portion. In this case, if necessary, the Data Segment area shall be padded out to the next nearest 32-bit boundary.

When the DATA IN UPIU is a part of a SCSI READ transaction [i.e., a transaction which started with a READ (6), a READ (10), or a READ (16) command], the Data Segment area shall contain an integer number of logical blocks.

NOTE For out of order DATA IN UPIUs, the final data portion may not be transmitted by the last UPIU.

g) Data in transfer example

Figure 10-2 shows an example of data transfer from the Target device to the Initiator device. In particular, during the command processing 577-byte data is sent to the Initiator device. The data transfer is done in sequence: at the beginning the first portion, then the middle portion and finally the last portion.

NOTE The last DATA IN UPIU delivers the last portion of the data. The Data Segment in this UPIU has 65 bytes of valid data and three pad bytes. The Data Segment in the other UPIUs is fully filled (no pad bytes).

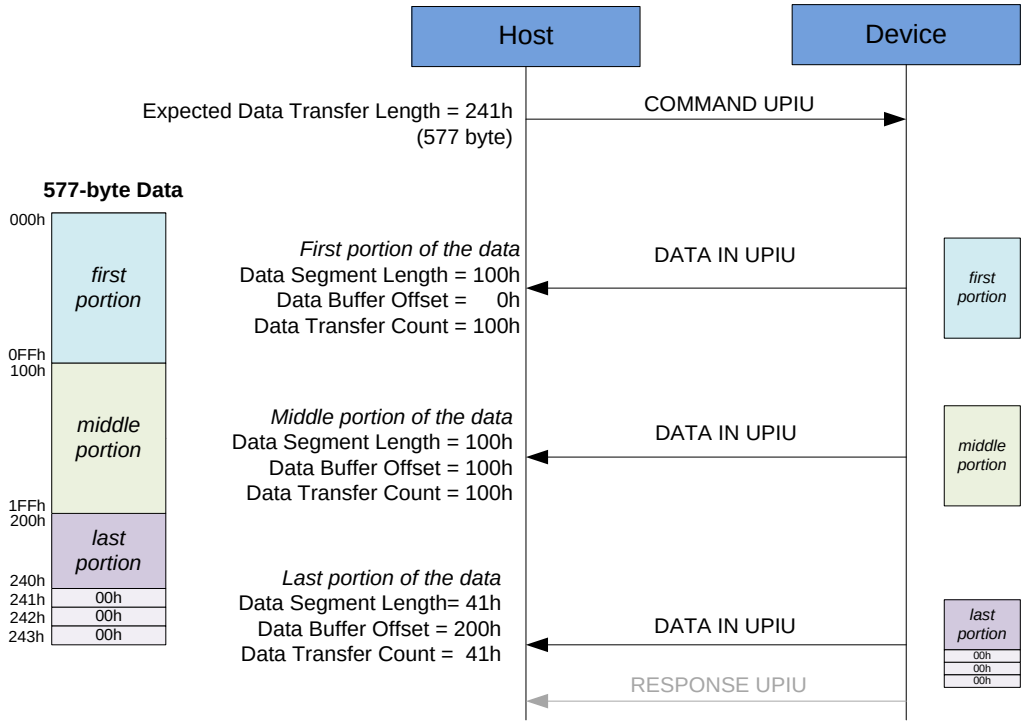


Figure 10-2 — Data in transfer example

10.7.5 READY TO TRANSFER UPIU

The READY TO TRANSFER UPIU is issued by the Target device when it is ready to receive data blocks when processing a SCSI command that requires a data out transfer (e.g., a write command). The Target device may request the data in sequence or out of order by setting the appropriate fields within the UPIU.

The Initiator device may respond to one or more READY TO TRANSFER UPIU packets with one or more DATA OUT UPIU packets, enough to satisfy the Expected Data Transfer Length that was indicated within the associated COMMAND UPIU. The maximum number of bytes that can be requested with a single READY TO TRANSFER UPIU shall not be greater than the value indicated by bMaxDataOutSize attribute.

See 10.7.13, Data out transfer rules, for further details about Initiator device to Target device data transfer.

Table 10-21 — READY TO TRANSFER UPIU

Ready To Transfer UPIU			
0 xx11 0001b	1 Flags	2 LUN	3 Task Tag
4 IID / Reserved	5 Reserved	6 Reserved	7 Reserved
8 Total EHS Length (00h)	9 Reserved	10 (MSB) / 11 (LSB) Data Segment Length (0000h)	
12 (MSB) / 13 / 14 / 15 (LSB) Data Buffer Offset			
16 (MSB) / 17 / 18 / 19 (LSB) Data Transfer Count			
20 / 21 / 22 / 23 Reserved			
24 / 25 / 26 / 27 Reserved			
28 / 29 / 30 / 31 Reserved			
Header E2ECRC (omit if HD=0)			

10.7.5 READY TO TRANSFER UPIU (cont'd)

10.7.5.1 Basic Header

The first 12 bytes of the READY TO TRANSFER UPIU contain the Basic Header as described in 10.6.2, Basic Header Format. Specific details are as follows:

a) Transaction Type

A type code value of xx11 0001b indicates a READY TO TRANSFER UPIU.

b) Initiator ID (IID)

The Initiator ID field (bits [7:4] of byte 4) indicates the identity of the Initiator device who created the task request. See Initiator ID description in 10.6.2, Basic Header Format, for details.

c) Data Segment Length

The Data Segment Length field shall contain zero as there is no Data Segment in this UPIU.

d) Data Buffer Offset

The Data Buffer Offset field indicates to the Initiator device the location of the beginning of the segment of data to send. The Target device may request the Initiator device to transfer the data in a number of UPIUs, not necessarily in sequential order. The sum of the Data Buffer Offset and the Data Transfer Count should not exceed the Expected Data Transfer Length that was indicated in the COMMAND UPIU.

The Data Buffer Offset shall be an integer multiple of four.

When the RTT UPIU is a part of a SCSI WRITE transaction [i.e. a transaction which started with a WRITE (6), a WRITE (10), or a WRITE (16) command], the value of this field shall be equal to an integer multiple of the Logical Block Size (bLogicalBlockSize).

e) Data Transfer Count

This field indicates the number of bytes the Target device is requesting.

The Data Transfer Count field shall be always an integer multiple of four bytes except for the READY TO TRANSFER UPIU which requests the final portion of data in the transfer.

When the RTT UPIU is a part of a SCSI WRITE transaction [i.e., a transaction which started with a WRITE (6), a WRITE (10), or a WRITE (16) command], the value of this field shall be equal to an integer multiple of the Logical Block Size (bLogicalBlockSize).

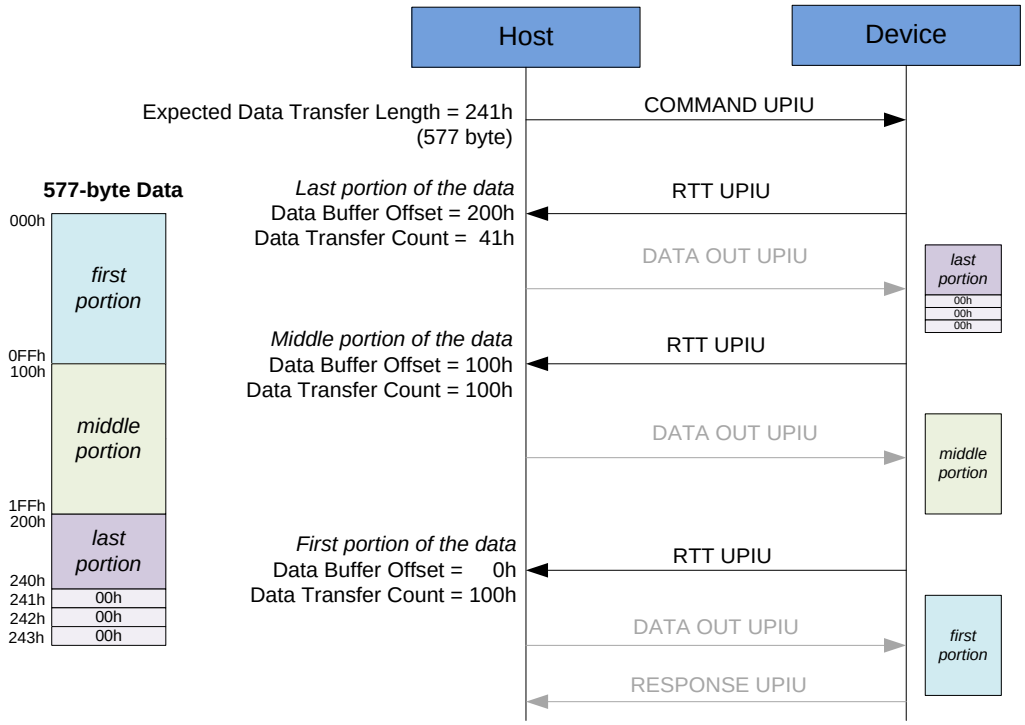
The maximum number of bytes that can be requested in a single READY TO TRANSFER UPIU shall not be greater than the value indicated by bMaxDataOutSize attribute.

f) READY TO TRANSFER UPIU sequence example

Figure 10-3 shows an example of READY TO TRANSFER UPIU sequence. In particular, during the command processing the Target device requests the Initiator device to send a total amount of 577-byte data. The data transfer is done out of order (reverse): at the beginning the last portion, then the middle portion and finally the first portion.

NOTE The first READY TO TRANSFER UPIU requests to send the last portion of the data.

1859 **10.7.5.1 Basic Header (cont'd)**



1860

1861 **Figure 10-3 — READY TO TRANSFER UPIU sequence example**

1862

10.7.6 TASK MANAGEMENT REQUEST UPIU

The TASK MANAGEMENT REQUEST UPIU is used by the Initiator device to manage the execution of one or more tasks within the Target device. The Task Management Request function closely follows the SCSI Architecture Model [SAM].

Task Management functions in UFS device requires the LU task manager to have the capability to process at least one Task Management Request. If more than one Task Management Request is accepted by a task manager in the device, the task manager may execute the requests in any order. The task manager may reject a Task Management Request with Task Management Function Failed service response when the number of outstanding Task Management Requests submitted by the Host exceeds the capability of the Task Manager.

Table 10-22 — Task Management Request UPIU

TASK MANAGEMENT REQUEST UPIU			
0 xx00 0100b	1 Flags	2 LUN	3 Task Tag
4 IID / Reserved	5 Task Manag. Function	6 Reserved	7 Reserved
8 Total EHS Length (00h)	9 Reserved	10 (MSB) / 11 (LSB) Data Segment Length (0000h)	
12 (MSB) / 13 / 14 / 15 (LSB) Input Parameter 1			
16 (MSB) / 17 / 18 / 19 (LSB) Input Parameter 2			
20 (MSB) / 21 / 22 / 23 (LSB) Input Parameter 3			
24 / 25 / 26 / 27 Reserved			
28 / 29 / 30 / 31 Reserved			
Header E2ECRC (omit if HD=0)			

10.7.6 TASK MANAGEMENT REQUEST UPIU (cont'd)

10.7.6.1 Basic Header

The first 12 bytes of the TASK MANAGEMENT REQUEST UPIU contain the Basic Header as described in 10.6.2 Basic Header Format. Specific details are as follows:

a) Transaction Type

A type code value of xx00 0100b indicates a TASK MANAGEMENT REQUEST UPIU.

b) Initiator ID (IID)

The Initiator ID field (bits [7:4] of byte 4) indicates the identity of the Initiator device who created the task request. See Initiator ID description in 10.6.2, Basic Header Format, for details.

c) Task Management Function

Table 10-23 defines UFS Task Management Functions based on [SAM].

Table 10-23 — Task Management Function values

Function	Value	Description
Abort Task	01h	Abort specific task in queue in a specific LU. Identify by LUN and Task Tag
Abort Task Set	02h	Abort the task queue list in a specific LU. Identify by LUN.
Clear Task Set	04h	Clear the task queue list in specific LU. Identify by LUN. Equivalent to Abort Task Set.
Logical Unit Reset	08h	Reset the designated LU. Identify by LUN
Query Task	80h	Query a specific task in a queue list in a specific LU. Identify by LUN and Task Tag. If the specific task is present in the queue, Function Succeeded is returned in the response. If the specific task is not present in the queue, Function Complete is returned in the response.
Query Task Set	81h	Query a specific LU to see if there is any Task in queue. Identify by LUN. If there is one of more tasks present in the queue, Function Succeeded is returned in the response. If no task is present in the queue, Function Complete is returned in the response.

d) Task Management Input Parameters

Table 10-24 — Task Management Input Parameters

Field	Description
Input Parameter 1	LSB: LUN of the logical unit operated on by the task management function. Other bytes: Reserved
Input Parameter 2	LSB: Task Tag of the task/command operated by the task management function. Other bytes: Reserved
Input Parameter 3	Bits [3:0]: IID of the task/command operated by the task management function. Other bits: Reserved

Input Parameter 1 and LUN field in the basic header should be set to the same value.

Input Parameter 3 and IID field in the basic header shall indicate the same value.

10.7.7 TASK MANAGEMENT RESPONSE UPIU

The TASK MANAGEMENT RESPONSE UPIU is sent by the Target device in response to a Task Management Request from the Initiator device. The Task Management Response function closely follows the SCSI Architecture Model [SAM].

If the Target device is processing a task which requires Data-Out data transfer, and it receives a task management request to abort that command, then Target device should stop sending READY TO TRANSFER UPIUs for the command requested to abort. Target device shall wait until it receives all DATA OUT UPIUs related to any outstanding READY TO TRANSFER UPIUs before sending the TASK MANAGEMENT RESPONSE UPIU.

Task management functions that may cause a task abort are: Abort Task, Abort Task Set, Clear Task Set and Logical Unit Reset.

Table 10-25 — Task Management Response UPIU

TASK MANAGEMENT RESPONSE UPIU			
0 xx10 0100b	1 Flags	2 LUN	3 Task Tag
4 IID | Reserved	5 Reserved	6 Response	7 Reserved
8 Total EHS Length (00h)	9 Reserved	10 (MSB) | 11 (LSB) Data Segment Length (0000h)	
12 (MSB)	13	14	15 (LSB)
Output Parameter 1			
16 (MSB)	17	18	19 (LSB)
Output Parameter 2			
20	21	22	23
Reserved			
24	25	26	27
Reserved			
28	29	30	31
Reserved			
Header E2ECRC (omit if HD=0)			

10.7.7 TASK MANAGEMENT RESPONSE UPIU (cont'd)

10.7.7.1 Basic Header

The first 12 bytes of the TASK MANAGEMENT RESPONSE UPIU contain the Basic Header as described in 10.6.2, Basic Header Format. Specific details are as follows:

a) Transaction Type

A type code value of xx10 0100b indicates a TASK MANAGEMENT RESPONSE UPIU.

b) Initiator ID (IID)

The Initiator ID field (bits [7:4] of byte 4) indicates the identity of the Initiator device who created the task request. See Initiator ID description in 10.6.2, Basic Header Format, for details.

c) Response

The Response field contains the UFS response that indicates the success or failure of the Task Management Request. See 10.6.2 for details.

d) Task Management Output Parameters

Table 10-26 — Task Management Output Parameters

Field	Description
Output Parameter 1	LSB: Task Management Service Response (see Table 10-27) Other bytes: Reserved
Output Parameter 2	Reserved

e) Task Management Service Response

Table 10-27 — Task Management Service Response

Service Response	Value
Task Management Function Complete	00h
Task Management Function Not Supported	04h
Task Management Function Failed (1)	05h
Task Management Function Succeeded	08h
Incorrect Logical Unit Number	09h
NOTE 1 This response shall be returned whenever the UFS device is not able to process the request due to TMR processing capacity exceed.	

10.7.8 QUERY REQUEST UPIU

The QUERY REQUEST UPIU is used to transfer data between the Initiator device and Target device that is outside domain of standard user data transfers for command read and write.

The QUERY REQUEST UPIU can be used to read and write parametric data to or from the Target device. It can be used to get information for configuration or enumeration, to set or clear bus or overall device conditions, to set or reset global flag values, parameters or attributes, to set or get power or bus or network information or to get or set descriptors, to get serial numbers or GUID's (globally unique identifiers), etc.

The Target device will send a QUERY RESPONSE UPIU in response to a QUERY REQUEST UPIU. After sending a QUERY REQUEST UPIU the Initiator device shall not send a new QUERY REQUEST UPIU until it receives the QUERY RESPONSE UPIU for the pending request. If the Target device receives a QUERY REQUEST UPIU while it is still processing a previous QUERY REQUEST UPIU, it shall ignore the latest request.

The QUERY REQUEST UPIU follows the general UPIU format with a field defined for query function. The Transaction Specific Fields are defined specifically for each type of operation.

The Data Segment Area is optional depending upon the query function value. The Data Segment Length field will be set to zero if there is no data segment in the packet.

Table 10-28 — QUERY REQUEST UPIU

QUERY REQUEST UPIU			
0 xx01 0110b	1 Flags	2 Reserved	3 Task Tag
4 Reserved	5 Query Function	6 Reserved	7 Reserved
8 Total EHS Length (00h)	9 Reserved	10 (MSB) / 11 (LSB) Data Segment Length	
12	13	14	15
Transaction Specific Fields			
16	17	18	19
Transaction Specific Fields			
20	21	22	23
Transaction Specific Fields			
24	25	26	27
Transaction Specific Fields			
28	29	30	31
Reserved			
Header E2ECRC (omit if HD=0)			
k Data[0]	k+1 Data[1]	k+2 Data[2]	k+3 Data[3]
…	…	…	…
k+ Length-4 Data[Length - 4]	k+ Length-3 Data[Length - 3]	k+ Length-2 Data[Length - 2]	k+ Length-1 Data[Length - 1]
Data E2ECRC (omit if DD=0)			

10.7.8 QUERY REQUEST UPIU (cont'd)

10.7.8.1 Overview

Queries are used to read and write data structures between the host and the device. This data is outside the scope of normal device reads or writes; data that would be considered system data, configuration data, production information, descriptors, special parameters and flags and other.

For UFS the query function will generally be used to read or write Descriptors, Attributes and Flags. There are also a range of Vendor Specific operations that can be used to transfer vendor specific data between host and device.

All these items reside within the device memory and used by the device to control or define its operation.

The following is a short overview of the most common data structures that are transferred using the Query Request function. Please see the related section for more detail on these data structures:

a) Descriptors

A Descriptor is a block or page of parameters that describe something about a Device. For example, there are Device Descriptors, Configuration Descriptors, Unit Descriptors, etc.

b) Attributes

An Attribute is a single parameter that represents a specific range of numeric values that can be set or read. This value could be a byte or word or floating point number. For example, baud rate or block size would be an attribute. Attribute size can be from 1-bit to 32-bit. Attributes of the same type can be organized in arrays, each element of them identified by an index.

c) Flags

A Flag is a single Boolean value that represents a TRUE or FALSE, '0' or '1', ON or OFF type of value. A Flag can be cleared or reset, set, toggled or read. Flags are useful to enable or disable certain functions or modes or states within the device.

10.7.8 QUERY REQUEST UPIU (cont'd)

10.7.8.2 Query Function

The Query Function field holds the requested query type describing the query function to perform. Common query functions are listed in Table 10-29. Currently, there are two general query functions defined: Read Request and Write Request. Additional Transaction Specific Fields will be used to specify further information needed for the transaction. These fields can describe the specific operation to perform, the target data or information to access, the amount of data to transfer and additional parameters and data.

Table 10-29 — Query Function field values

QUERY FUNCTION	
00h	Reserved
01h	STANDARD READ REQUEST
02h-3Fh	Reserved
40-7Fh	Vendor Specific Read Functions
80h	Reserved
81h	STANDARD WRITE REQUEST
82h-BFh	Reserved
C0h-FFh	Vendor Specific Write Functions

a) Standard Read Request

The Standard Read Request function type is used to read requested information from a Target device. The Target device will return the requested information to the Initiator device within a QUERY RESPONSE UPIU packet.

b) Standard Write Request

The Standard Write Request function type is used to write information and data to a Target device. The information and data to write to the Target device will be included within the Data Segment field of the QUERY REQUEST UPIU packet.

10.7.8 QUERY REQUEST UPIU (cont'd)

10.7.8.3 Transaction Specific Fields

The transaction specific fields are defined specifically for each type of operation. For the STANDARD READ REQUEST and STANDARD WRITE REQUEST, the operation specific fields are defined as in Table 10-30.

Table 10-30 — Transaction specific fields

Transaction Specific Fields for Standard Read/Write Request			
12 OPCODE	13 OSF[0]	14 OSF[1]	15 OSF[2]
16 OSF[3]	17 OSF[4]	18 (MSB) OSF[5]	19 (LSB)
20 (MSB)	21	22	23 (LSB)
OSF[6]			
24 (MSB)	25	26	27 (LSB)
OSF[7]			

a) OPCODE

The opcode indicates the operation to perform. Possible opcode values are listed in Table 10-31.

Table 10-31 — Query Function opcode values

OPCODE	Operation	QUERY FUNCTION
00h	NOP	Any value
01h	READ DESCRIPTOR	STANDARD READ REQUEST
02h	WRITE DESCRIPTOR	STANDARD WRITE REQUEST
03h	READ ATTRIBUTE	STANDARD READ REQUEST
04h	WRITE ATTRIBUTE	STANDARD WRITE REQUEST
05h	READ FLAG	STANDARD READ REQUEST
06h	SET FLAG	STANDARD WRITE REQUEST
07h	CLEAR FLAG	STANDARD WRITE REQUEST
08h	TOGGLE FLAG	STANDARD WRITE REQUEST
09h-EFh	Reserved	Reserved
F0h-FFh	Vendor Specific	Vendor Specific

b) OSF

The OSF field is an Opcode Specific Field. The OSF fields will be defined for each specific OPCODE.

10.7.8 QUERY REQUEST UPIU (cont'd)

10.7.8.4 Read Descriptor Opcode

The READ DESCRIPTOR OPCODE is used to retrieve a UFS Descriptor from the Target device. A descriptor can be a fixed or variable length. There are up to 256 possible descriptor types. The OSF fields are used to select a particular descriptor and to read a number of descriptor bytes. The OSF fields are defined in Table 10-32.

Table 10-32 — Read descriptor

Transaction Specific Fields for READ DESCRIPTOR OPCODE			
12 01h	13 DESCRIPTOR IDN	14 INDEX	15 SELECTOR
16 Reserved	17 Reserved	18 (MSB) LENGTH	19 (LSB)
20	21	22	23
Reserved			
24	25	26	27
Reserved			

a) DESCRIPTOR IDN

The Descriptor IDN field contains a value that indicates the particular type of descriptor to retrieve. For example, it could indicate a Device Descriptor or Unit Descriptor or String Descriptor. Some descriptor types are unique and can be fully identified by the Descriptor Type value. Other descriptors can exist in multiple forms, such as String Descriptors, and they are furthered identified with subsequent fields.

b) INDEX

The Index value is used to further identify a particular descriptor. For example, there may be multiple String Descriptors defined. In the case of multiple descriptors the INDEX field is used to select a particular one. Multiple descriptors are indexed starting from 0 through 255. The actual index value for a particular descriptor will be provided by other means, usually contained within a field of some other related descriptor.

c) SELECTOR

The SELECTOR field may be needed to further identify a particular descriptor.

d) LENGTH

The LENGTH field is used to indicate the number of bytes to read of the descriptor. These bytes will be returned in a QUERY RESPONSE UPIU packet. This is the requested length to read, which may be less than, or equal to, or greater than the number of bytes within the actual descriptor. If less than, or equal to the actual descriptor size, the number of bytes specified will be returned. If the LENGTH is greater than the descriptor size, the response will provide the exact descriptor size in the LENGTH field of the QUERY RESPONSE UPIU .

e) Data Segment

The Data Segment area is empty.

10.7.8 QUERY REQUEST UPIU (cont'd)

10.7.8.5 Write Descriptor Opcode

The WRITE DESCRIPTOR OPCODE is used to write a UFS Descriptor and it is sent from the host to the device. A descriptor can be a fixed or variable length. There are up to 256 possible descriptor types. The OSF fields are used to select a particular descriptor. The OSF fields are defined as listed in Table 10-33.

Table 10-33 — Write Descriptor

Transaction Specific Fields for WRITE DESCRIPTOR OPCODE			
12 02h	13 DESCRIPTOR IDN	14 INDEX	15 SELECTOR
16 Reserved	17 Reserved	18 (MSB) LENGTH	19 (LSB)
20 Reserved	21	22	23
24 Reserved	25	26	27

a) DESCRIPTOR IDN

The Descriptor IDN field contains a value that identifies a particular of descriptor to write. For example, it could indicate a Device Descriptor or Unit Descriptor or String Descriptor. Some descriptor types are unique and can be fully identified by the Descriptor IDN value. Other descriptors can exist in multiple forms, such as STRING DESCRIPTORS, and they are furthered identified with subsequent fields.

b) INDEX

The Index value is used to further identify a particular descriptor. For example, there may be multiple String Descriptors defined. In the case of multiple descriptors the INDEX field is used to select a particular one. Multiple descriptors are indexed starting from 0 through 255. The actual index value for a particular descriptor will be provided by other means, usually contained within a field of some other related descriptor.

c) SELECTOR

The SELECTOR field may be needed to further identify a particular descriptor.

d) LENGTH

The LENGTH field is used to indicate the number of descriptor bytes to write. Only the entire descriptor may be written; there is no partial write or update possible. These bytes will be contained within the DATA SEGMENT area of the QUERY REQUEST UPIU packet. The DATA SEGMENT LENGTH field of the UPIU shall also be set to this same value. If LENGTH is not equal to the descriptor size the operation will fail: the descriptor is not updated and the Query Response field of the QUERY RESPONSE UPIU is set FAILURE.

e) Data Segment

The Data Segment area contains the data to be written.

10.7.8 QUERY REQUEST UPIU (cont'd)

10.7.8.6 Read Attribute Opcode

The READ ATTRIBUTE OPCODE is used to retrieve a UFS Attribute from the Target device. Attribute size can be from 1-bit to 32-bit. There are up to 256 possible Attributes, identified by an identification number, IDN, which ranges from 0 to 255. The OSF fields for this opcode are listed in Table 10-34.

Table 10-34 — Read Attribute

Transaction Specific Fields for READ ATTRIBUTE OPCODE			
12 03h	13 ATTRIBUTE IDN	14 INDEX	15 SELECTOR
16 Reserved	17 Reserved	18 Reserved	19 Reserved
20	21	22	23
Reserved			
24	25	26	27
Reserved			

a) ATTRIBUTE IDN

The ATTRIBUTE IDN contains a value that identifies a particular Attribute to retrieve from the Target device.

b) INDEX

For attributes that are organized in array, the index value is used to identify the particular element. For example, the LUN is used as index to select the particular element of attributes that have logical unit specific values.

The range for the index is defined for each attribute, and it can be from 0 through 255. Index shall be set to zero for attributes composed by single element.

c) SELECTOR

The SELECTOR field may be needed to further identify a particular element of an attribute. Selector field shall be set to zero for attributes that do not require it.

d) Data Segment

The Data Segment area is empty.

10.7.8 QUERY REQUEST UPIU (cont'd)

10.7.8.7 Write Attribute Opcode

The **WRITE ATTRIBUTE OPCODE** is used to write a UFS Attribute to the Target device. Attribute size can be from 1-bit to 32-bit. There are up to 256 possible Attributes, identified by an identification number, IDN, which ranges from 0 to 255. The OSF fields for this opcode are listed in Table 10-35

Table 10-35 — Write Attribute

Transaction Specific Fields for WRITE ATTRIBUTE OPCODE			
12 04h	13 ATTRIBUTE IDN	14 INDEX	15 SELECTOR
16 Reserved	17 Reserved	18 Reserved	19 Reserved
20 (MSB) VALUE [31:24]	21 VALUE [23:16]	22 VALUE [15:8]	23 (LSB) VALUE [7:0]
24	25	26	27
Reserved			

a) ATTRIBUTE IDN

The ATTRIBUTE IDN contains a value that identifies a particular Attribute to write in the Target device.

b) INDEX

For attributes that are organized in array, the index value is used to identify the particular element. For example, the LUN is used as index to select the particular element of attributes that have logical unit specific values.

The range for the index is defined for each attribute, and it can be from 0 through 255. Index shall be set to zero for attributes composed by single element.

c) SELECTOR

The SELECTOR field may be needed to further identify a particular element of an attribute. Selector field shall be set to zero for attributes that do not require it.

d) VALUE [31:0]

The 32-bit VALUE field contains the data value of the Attribute. The VALUE is a right justified, big Endian value. Unused upper bits shall be set to zero.

e) Data Segment

The Data Segment area is empty.

10.7.8 QUERY REQUEST UPIU (cont'd)

10.7.8.8 Read Flag Opcode

The READ FLAG OPCODE is used to retrieve a UFS Flag value from the Target device. A Flag is a fixed size single byte value that represents a Boolean value. There can be defined up to 256 possible Flag values. A Flag is identified by its FLAG IDN, an identification number that ranges in value from 0 to 255. The OSF fields for this opcode are listed in Table 10-36.

The FLAG data, either one (01h) or zero (00h), is returned within the Transaction Specific Fields area of a QUERY RESPONSE UPIU packet.

Table 10-36 — Read Flag

Transaction Specific Fields for READ FLAG OPCODE			
12 05h	13 FLAG IDN	14 INDEX	15 SELECTOR
16 Reserved	17 Reserved	18 Reserved	19 Reserved
20	21	22	23
Reserved			
24	25	26	27
Reserved			

a) FLAG IDN

The FLAG IDN field contains a value that identifies a particular Flag to retrieve from the Target device.

b) INDEX

The index field may be needed to identify a particular element of a flag. Index field is not used in this version of the standard and its value shall be zero.

c) SELECTOR

The selector field may be needed to further identify a particular element of a flag. Selector field is not used in this version of the standard and its value shall be zero.

d) Operation

The Boolean value of the addressed flag is returned in a QUERY RESPONSE UPIU.

e) Data Segment

The Data Segment area is empty.

10.7.8 QUERY REQUEST UPIU (cont'd)

10.7.8.9 Set Flag

Table 10-37 — Set Flag

Transaction Specific Fields for SET FLAG OPCODE			
12 06h	13 FLAG IDN	14 INDEX	15 SELECTOR
16 Reserved	17 Reserved	18 Reserved	19 Reserved
20	21	22	23
Reserved			
24	25	26	27
Reserved			

a) FLAG IDN

The FLAG IDN field contains a value that identifies a particular Flag to set in the Target device.

b) INDEX

The index field may be needed to identify a particular element of a flag. Index field is not used in this version of the standard and its value shall be zero.

c) SELECTOR

The selector field may be needed to further identify a particular element of a flag. Selector field is not used in this version of the standard and its value shall be zero.

d) Operation

The Boolean value of the addressed flag is set to TRUE or one.

e) Data Segment

The Data Segment area is empty.

10.7.8 QUERY REQUEST UPIU (cont'd)

10.7.8.10 Clear Flag

Table 10-38 — Clear Flag

Transaction Specific Fields for CLEAR FLAG OPCODE			
12	13	14	15
07h	FLAG IDN	INDEX	SELECTOR
16	17	18	19
Reserved	Reserved	Reserved	Reserved
20	21	22	23
Reserved			
24	25	26	27
Reserved			

a) FLAG IDN

The FLAG IDN field contains a value that identifies a particular Flag to clear in Target device.

b) INDEX

The index field may be needed to identify a particular element of a flag. Index field is not used in this version of the standard and its value shall be zero.

c) SELECTOR

The selector field may be needed to further identify a particular element of a flag. Selector field is not used in this version of the standard and its value shall be zero.

d) Operation

The Boolean value of the addressed flag is cleared to FALSE or zero.

e) Data Segment

The Data Segment area is empty.

10.7.8 QUERY REQUEST UPIU (cont'd)

10.7.8.11 Toggle Flag

Table 10-39 — Toggle Flag

Transaction Specific Fields for TOGGLE FLAG OPCODE			
12 08h	13 FLAG IDN	14 INDEX	15 SELECTOR
16 Reserved	17 Reserved	18 Reserved	19 Reserved
20	21	22	23
Reserved			
24	25	26	27
Reserved			

a) FLAG IDN

The FLAG IDN field contains a value that identifies a particular Flag to toggle in the Target device.

b) INDEX

The index field may be needed to identify a particular element of a flag. Index field is not used in this version of the standard and its value shall be zero.

c) SELECTOR

The selector field may be needed to further identify a particular element of a flag. Selector field is not used in this version of the standard and its value shall be zero.

d) Operation

The Boolean value of the addressed flag is set to the negated current value.

e) Data Segment

The Data Segment area is empty.

10.7.8 QUERY REQUEST UPIU (cont'd)

10.7.8.12 NOP

Table 10-40 defines NOP OPCODE for QUERY REQUEST UPIU.

Table 10-40 — NOP

Transaction Specific Fields for NOP FLAG OPCODE			
12 00h	13 Reserved	14 Reserved	15 Reserved
16 Reserved	17 Reserved	18 Reserved	19 Reserved
20	21	22	23
Reserved			
24	25	26	27
Reserved			

a) Data Segment

The Data Segment area is empty.

10.7.9 QUERY RESPONSE UPIU

The QUERY RESPONSE UPIU is used to transfer data between the Target device and Initiator device in response to a QUERY REQUEST UPIU.

The QUERY RESPONSE UPIU is used to return parametric data to the requesting Initiator device in case of read descriptor/attribute/flag query request, or to provide response to write descriptor/attribute query request or set/clear/toggle flag query request.

Table 10-41 — QUERY RESPONSE

QUERY RESPONSE UPIU			
0 xx11 0110b	1 Flags	2 Reserved	3 Task Tag
4 Reserved	5 Query Function	6 Query Response	7 Reserved
8 Total EHS Length (00h)	9 Device Information	10 (MSB) 11 (LSB) Data Segment Length	
12 13 14 15 Transaction Specific Fields			
16 17 18 19 Transaction Specific Fields			
20 21 22 23 Transaction Specific Fields			
24 25 26 27 Transaction Specific Fields			
28 29 30 31 Reserved			
Header E2ECRC (omit if HD=0)			
k Data[0]	k+1 Data[1]	k+2 Data[2]	k+3 Data[3]
…	…	…	…
k+ Length-4 Data[Length - 4]	k+ Length-3 Data[Length - 3]	k+ Length-2 Data[Length - 2]	k+ Length-1 Data[Length - 1]
Data E2ECRC (omit if DD=0)			

The QUERY RESPONSE UPIU follows the general UPIU format a field defined for query function.

The transaction specific fields are defined specifically for each type of operation.

The Data Segment Area is optional depending upon the Query Function value. The Data Segment Length field will be set to zero if there is no data segment in the packet.

10.7.9 QUERY RESPONSE UPIU (cont'd)

10.7.9.1 Overview

The Query Response function will respond to the query function that was sent from the Initiator device in the QUERY REQUEST UPIU. The Query Response may or may not return data depending upon the function. If data needs to be returned it will be returned in the Data Segment of the UPIU or one of the Transaction Specific Fields.

10.7.9.2 Query Function

The Query Function field will contain the original query function value that was received in the corresponding QUERY REQUEST UPIU.

10.7.9.3 Query Response

The Query Response field indicates the completion code of the action taken in response to the QUERY REQUEST UPIU. Possible values are listed in Table 10-42.

NOTE In case of unsuccessful operation, the Target device may either set Query Response field to FFh, or optionally provide more detailed information about the failure using one of the other values.

Table 10-42 — Query Response Code

Value	Description
00h	Success
01h-F5h	Reserved
F6h	Parameter not readable
F7h	Parameter not writeable
F8h	Parameter already written (1)
F9h	Invalid LENGTH
FAh	Invalid value (2)
FBh	Invalid SELECTOR
FCh	Invalid INDEX
FDh	Invalid IDN
FEh	Invalid OPCODE
FFh	General failure

NOTE 1 This value applies to parameters with "Write once" or "Power on reset" write access property.

NOTE 2 This value applies to the following operations: write descriptor, write attribute, set flag, clear flag.

10.7.9 QUERY RESPONSE UPIU (cont'd)

10.7.9.4 Device Information

See Device Information field definition in RESPONSE UPIU, 10.7.2.1 Basic Header.

10.7.9.5 Transaction Specific Fields

The transaction specific fields are defined specifically for each type of operation. For the STANDARD READ REQUEST and STANDARD WRITE REQUEST, the operation specific fields are defined as in Table 10-43.

Table 10-43 — Transaction Specific Fields

Transaction Specific Fields for Standard Read/Write Request			
12 OPCODE	13 OSF[0]	14 OSF[1]	15 OSF[2]
16 OSF[3]	17 OSF[4]	18 (MSB) OSF[5]	19 (LSB)
20 (MSB)	21	22 OSF[6]	23 (LSB)
24 (MSB)	25	26 OSF[7]	27 (LSB)

a) OPCODE

The opcode indicates the operation to perform. Possible opcode values are listed in Table 10-31.

If in a QUERY REQUEST UPIU, the Query Function field is set to STANDARD READ REQUEST and the OPCODE field is set to WRITE DESCRIPTOR, WRITE ATTRIBUTE, SET FLAG, CLEAR FLAG, or TOGGLE FLAG; then the query request shall fail and the Query Response field shall be set to either "Invalid OPCODE" or "General failure".

If in a QUERY REQUEST UPIU, the Query Function field is set to STANDARD WRITE REQUEST and the OPCODE field is set to READ DESCRIPTOR, READ ATTRIBUTE, or READ FLAG; then the query request shall fail and the Query Response field shall be set to either "Invalid OPCODE" or "General failure".

b) OSF

The OSF field is an Opcode Specific Field. The OSF fields will be defined for each specific OPCODE.

10.7.9 QUERY RESPONSE UPIU (cont'd)

10.7.9.6 Read Descriptor Opcode

The READ DESCRIPTOR OPCODE is used to retrieve a UFS DESCRIPTOR from the Target device. A descriptor can be a fixed or variable length. There are up to 256 possible descriptor types. The OSF fields are used to select a particular descriptor and to read a number of descriptor bytes. The OSF fields are defined in Table 10-44.

The READ DESCRIPTOR OPCODE is returned in response to a QUERY REQUEST UPIU containing the same value in the OPCODE field.

Table 10-44 — Read Descriptor

Transaction Specific Fields for READ DESCRIPTOR OPCODE			
12 01h	13 DESCRIPTOR IDN	14 INDEX	15 SELECTOR
16 Reserved	17 Reserved	18 (MSB) LENGTH	19 (LSB)
20 Reserved	21	22	23
24 Reserved	25	26	27

a) DESCRIPTOR IDN

The DESCRIPTOR IDN field contains the same DESCRIPTOR IDN value sent from the corresponding QUERY REQUEST UPIU.

b) INDEX

The Index field value returned is the same INDEX value of the corresponding QUERY REQUEST UPIU.

c) SELECTOR

The SELECTOR field returned is the same SELECTOR value of the corresponding QUERY REQUEST UPIU.

d) LENGTH

The LENGTH field is used to indicate the number of bytes returned in response to the corresponding QUERY REQUEST UPIU. This value could be less than the requested size, if the size of the data item is smaller than the size requested in the corresponding QUERY REQUEST UPIU.

e) Data Segment

The Data Segment area contains the descriptor data.

10.7.9 QUERY RESPONSE UPIU (cont'd)

10.7.9.7 Write Descriptor Opcode

The WRITE DESCRIPTOR OPCODE is used to respond to a write descriptor query request. A descriptor can be a fixed or variable length. There are up to 256 possible descriptor types. The OSF fields are used to select a particular descriptor. The OSF fields are defined in Table 10-45.

Table 10-45 — Write Descriptor

Transaction Specific Fields for WRITE DESCRIPTOR OPCODE			
12 02h	13 DESCRIPTOR IDN	14 INDEX	15 SELECTOR
16 Reserved	17 Reserved	18 (MSB) LENGTH	19 (LSB)
20 Reserved	21	22	23
24 Reserved	25	26	27

a) DESCRIPTOR IDN

The Descriptor IDN field contains the same DESCRIPTOR IDN value sent from the corresponding QUERY REQUEST UPIU.

b) INDEX

The Index field value returned is the same INDEX value of the corresponding QUERY REQUEST UPIU.

c) SELECTOR

The SELECTOR field returned is the same SELECTOR value of the corresponding QUERY REQUEST UPIU.

d) LENGTH

The LENGTH field is used to indicate the number of descriptor bytes written. Only the entire descriptor may be written; there is no partial write or update possible.

e) Data Segment

The Data Segment area is empty.

10.7.9 QUERY RESPONSE UPIU (cont'd)

10.7.9.8 Read Attribute Opcode

The READ ATTRIBUTE OPCODE is used to retrieve a UFS attribute from the Target device. Attribute size can be from 1-bit to 32-bit. There are up to 256 possible attributes, identified by an identification number, IDN, which ranges from 0 to 255.

The response to a READ ATTRIBUTE request will be returned in a QUERY RESPONSE UPIU. A success or failure code for the entire operation will be contained within the RESPONSE field.

The attribute data will be returned within the transaction specific fields. The Transaction Specific fields are formatted as indicated in Table 10-46. The first two 32-bit words of those fields will echo the first two 32-bit words of the transaction specific fields of the QUERY REQUEST UPIU. The third word will contain the Attribute data.

Table 10-46 — Read Attribute Response Data Format

Transaction Specific Fields for READ ATTRIBUTE OPCODE			
12 03h	13 ATTRIBUTE IDN	14 INDEX	15 SELECTOR
16 Reserved	17 Reserved	18 Reserved	19 Reserved
20 (MSB) VALUE [31:24]	21 VALUE [23:16]	22 VALUE [15:8]	23 (LSB) VALUE [7:0]
24	25	26	27
Reserved			

a) ATTRIBUTE IDN

The ATTRIBUTE IDN field contains the same ATTRIBUTE IDN value sent from the corresponding QUERY REQUEST UPIU.

b) INDEX

The Index field value returned is the same INDEX value of the corresponding QUERY REQUEST UPIU.

c) SELECTOR

The SELECTOR field returned is the same SELECTOR value of the corresponding QUERY REQUEST UPIU.

d) VALUE [31:0]

The 32-bit VALUE field contains the data value of the ATTRIBUTE. The VALUE is a right justified, big Endian value. Unused upper bits shall be set to zero.

e) Data Segment

The Data Segment area is empty.

10.7.9 QUERY RESPONSE UPIU (cont'd)

10.7.9.9 Write Attribute Opcode

The WRITE ATTRIBUTE OPCODE is used to respond to a write attribute query request. Attribute size can be from 1-bit to 32-bit. There are up to 256 possible attributes, identified by an identification number, IDN, which ranges from 0 to 255. The OSF fields for this opcode are listed in Table 10-47

Table 10-47 — Write Attribute

Transaction Specific Fields for WRITE ATTRIBUTE OPCODE			
12 04h	13 ATTRIBUTE IDN	14 INDEX	15 SELECTOR
16 Reserved	17 Reserved	18 Reserved	19 Reserved
20 (MSB) VALUE [31:24]	21 VALUE [23:16]	22 VALUE [15:8]	23 (LSB) VALUE [7:0]
24	25	26	27
Reserved			

a) ATTRIBUTE IDN

The ATTRIBUTE IDN field contains the same ATTRIBUTE IDN value sent from the corresponding QUERY REQUEST UPIU.

b) INDEX

The Index field value returned is the same INDEX value of the corresponding QUERY REQUEST UPIU.

c) SELECTOR

The SELECTOR field returned is the same SELECTOR value of the corresponding QUERY REQUEST UPIU.

d) VALUE [31:0]

This field contains the same data provided in write attribute query request.

e) Data Segment

The Data Segment area is empty.

10.7.9 QUERY RESPONSE UPIU (cont'd)

10.7.9.10 Read Flag Opcode

The READ FLAG OPCODE is used to retrieve a UFS FLAG value from the Target device. A FLAG is a fixed size single byte value that represents a Boolean value. There can be defined up to 256 possible FLAG values. A FLAG is identified by its FLAG IDN, an identification number that ranges in value from 0 to 255. The FLAG data, either '1' or '0', is returned within the Transaction Specific Fields area of a QUERY RESPONSE UPIU packet.

The response to a READ ATTRIBUTE request will be returned in a QUERY RESPONSE UPIU. A success or failure code for the entire operation will be contained within the RESPONSE field.

The attribute data will be returned within the transaction specific fields. The Transaction Specific fields are formatted as indicated in Table 10-48. The first two 32-bit words of those fields will echo the first two 32-bit words of the transaction specific fields of the QUERY REQUEST UPIU. The third word will contain the FLAG data, a '0' or '1' value.

Table 10-48 — Read Flag Response Data Format

Transaction Specific Fields for READ FLAG OPCODE			
12 05h	13 FLAG IDN	14 INDEX	15 SELECTOR
16 Reserved	17 Reserved	18 Reserved	19 Reserved
20 Reserved	21 Reserved	22 Reserved	23 FLAG VALUE
24	25	26	27
Reserved			

a) FLAG IDN

The FLAG IDN field contains the same FLAG IDN value sent from the corresponding QUERY REQUEST UPIU

b) INDEX

The Index field value returned is the same INDEX value of the corresponding QUERY REQUEST UPIU.

c) SELECTOR

The SELECTOR field returned is the same SELECTOR value of the corresponding QUERY REQUEST UPIU

d) FLAG VALUE

The FLAG VALUE field contains the FLAG data: 00h or 01h.

e) Data Segment

The Data Segment area is empty.

10.7.9 QUERY RESPONSE UPIU (cont'd)

10.7.9.11 Set Flag

Table 10-49 — Set Flag

Transaction Specific Fields for SET FLAG OPCODE			
12 06h	13 FLAG IDN	14 INDEX	15 SELECTOR
16 Reserved	17 Reserved	18 Reserved	19 Reserved
20 Reserved	21 Reserved	22 Reserved	23 FLAG VALUE
24	25	26	27
Reserved			

a) FLAG IDN

The FLAG IDN field contains the same FLAG IDN value sent from the corresponding QUERY REQUEST UPIU

b) INDEX

The INDEX field value returned is the same INDEX value of the corresponding QUERY REQUEST UPIU.

c) SELECTOR

The SELECTOR field returned is the same SELECTOR value of the corresponding QUERY REQUEST UPIU.

d) FLAG VALUE

The FLAG VALUE field contains the FLAG data: 00h or 01h.

This field is valid only if the Query Response field indicates that the operation has been successfully completed ("Success").

e) Data Segment

The Data Segment area is empty.

10.7.9 QUERY RESPONSE UPIU (cont'd)

10.7.9.12 Clear Flag

Table 10-50 — Clear Flag

Transaction Specific Fields for CLEAR FLAG OPCODE			
12 07h	13 FLAG IDN	14 INDEX	15 SELECTOR
16 Reserved	17 Reserved	18 Reserved	19 Reserved
20 Reserved	21 Reserved	22 Reserved	23 FLAG VALUE
24	25	26	27
Reserved			

a) FLAG IDN

The FLAG IDN field contains the same FLAG IDN value sent from the corresponding QUERY REQUEST UPIU

b) INDEX

The Index field value returned is the same INDEX value of the corresponding QUERY REQUEST UPIU.

c) SELECTOR

The SELECTOR field returned is the same SELECTOR value of the corresponding QUERY REQUEST UPIU.

d) FLAG VALUE

The FLAG VALUE field contains the FLAG data: 00h or 01h.

This field is valid only if the Query Response field indicates that the operation has been successfully completed ("Success").

e) Data Segment

The Data Segment area is empty.

10.7.9 QUERY RESPONSE UPIU (cont'd)

10.7.9.13 Toggle Flag

Table 10-51 — Toggle Flag

Transaction Specific Fields for TOGGLE FLAG OPCODE			
12 08h	13 FLAG IDN	14 INDEX	15 SELECTOR
16 Reserved	17 Reserved	18 Reserved	19 Reserved
20 Reserved	21 Reserved	22 Reserved	23 FLAG VALUE
24 Reserved	25	26	27

a) FLAG IDN

The FLAG IDN field contains the same FLAG IDN value sent from the corresponding QUERY REQUEST UPIU.

b) INDEX

The Index field value returned is the same INDEX value of the corresponding QUERY REQUEST UPIU.

c) SELECTOR

The SELECTOR field returned is the same SELECTOR value of the corresponding QUERY REQUEST UPIU

d) FLAG VALUE

The FLAG VALUE field contains the FLAG data: 00h or 01h.

This field is valid only if the Query Response field indicates that the operation has been successfully completed ("Success").

e) Data Segment

The Data Segment area is empty.

10.7.9 QUERY RESPONSE UPIU (cont'd)

10.7.9.14 NOP

Table 10-52 defines NOP OPCODE for QUERY RESPONSE UPIU.

Table 10-52 — NOP

Transaction Specific Fields for NOP FLAG OPCODE			
12 00h	13 Reserved	14 Reserved	15 Reserved
16 Reserved	17 Reserved	18 Reserved	19 Reserved
20	21	22	23
Reserved			
24	25	26	27
Reserved			

a) Data Segment

The Data Segment area is empty.

10.7.10 REJECT UPIU

All UPIU packets include the basic header segment and some transaction specific fields. In addition to them, UPIU packets may have: Data Segment, Extra Header Segment, Header E2ECRC, Data E2ECRC.

The purpose of the REJECT UPIU is to simplify the software development and the system debug.

Table 10-53 — Reject UPIU

Reject UPIU			
0 xx11 1111b	1 Flags	2 LUN	3 Task Tag
4 IID | Reserved	5 Reserved	6 Response (01h)	7 Reserved
8 Total EHS Length (00h)	9 Device Information (00h)	10 (MSB) | 11 (LSB) Data Segment Length (0000h)	
12 Basic Header Status	13 Reserved	14 E2E Status	15 Reserved
16 Reserved	17	18	19
20 Reserved	21	22	23
24 Reserved	25	26	27
28 Reserved	29	30	31
Header E2ECRC (omit if HD=0)			

The device shall send a REJECT UPIU if it receives a UPIU with an invalid Transaction Type.

The Transaction Type is defined in 10.6.2a, Basic Header Format, and it is composed by the following fields: HD bit, DD bit and the Transaction Code.

Since this version of the standard does not support end-to-end CRC for header and data segments, a Transaction Type value is valid if;

- HD bit and DD bit are set to zero.
- the Transaction Code identifies one of the defined UPIU transactions from the Initiator device to Target device, see Table 10.1, UPIU Transaction Codes, (reserved values excluded).

The device shall not respond with a REJECT UPIU in the following cases:

- Incorrect LUN field or Command Set Type field in a COMMAND UPIU: the device shall send a RESPONSE UPIU. In particular, in case of an incorrect Command Set Type field value, the Data Segment Area of the RESPONSE UPIU shall be empty (Data Segment Length shall be equal to zero).
- Incorrect LUN field or Task Management Function field in TASK MANAGEMENT REQUEST UPIU: the device shall send a TASK MANAGEMENT RESPONSE UPIU.
- Incorrect Query Function field in QUERY REQUEST UPIU: the device shall send a QUERY RESPONSE UPIU

10.7.10.1 Basic Header

The first 12 bytes of the Reject UPIU contain the Basic Header as described in 10.6.2, Basic Header Format. Specific details are as follows:

a) Transaction Type

A type code value of xx11 1111b indicates a Reject UPIU.

b) Flags

The Flags field value shall be equal to zero.

c) LUN

The LUN shall be equal to the LUN value of the rejected UPIU.

d) Task Tag

The Task Tag shall be equal to the Task Tag value of the rejected UPIU.

e) Initiator ID (IID)

The IID shall be equal to the IID value of the rejected UPIU.

f) Response

The Response field shall be set to 01h (Target Failure) indicating that the Target device was not able to execute the requested operation.

g) Data Segment Length

The Data Segment Length field shall contain zero as there is no Data Segment in this UPIU.

h) Basic Header Status

The Basic Header Status field provides information about error detected in the UPIU received by the Initiator device.

Table 10-54 defines the possible values for the Basic Header Status field.

Table 10-54 — Basic Header Status Description

Value	Name
00h	Reserved
01h	Invalid Transaction Type
02h to FFh	Reserved

i) E2E Status

The E2E Status field provides the result of the end-to-end CRC of the rejected UPIU for both Header and Data. E2E Status is reserved if end-to-end CRC is not supported.

Table 10-55 — E2E Status Definition

Bit	Description
Bit 0	0: Header E2ECRC validated or not supported 1: Header E2ECRC error
Bit 1	0: Data E2ECRC validated or not supported 1: Data E2ECRC error
Others	Reserved

10.7.11 NOP OUT UPIU

The Initiator device may use NOP OUT UPIU to check the connection to a device. The Target device will respond to a NOP OUT UPIU sending a NOP IN UPIU back to the Initiator device.

Table 10-56 — NOP OUT UPIU

NOP OUT UPIU			
0 xx00 0000b	1 Flags	2 Reserved	3 Task Tag
4 Reserved	5 Reserved	6 Reserved	7 Reserved
8 Total EHS Length (00h)	9 Reserved	10 (MSB) / 11 (LSB) Data Segment Length (0000h)	
12 / 13 / 14 / 15 Reserved			
16 / 17 / 18 / 19 Reserved			
20 / 21 / 22 / 23 Reserved			
24 / 25 / 26 / 27 Reserved			
28 / 29 / 30 / 31 Reserved			
32 (MSB) / 33 / 34 / 35 (LSB) Header E2ECRC (omit if HD=0)			

10.7.11.1 Basic Header

The first 12 bytes of the NOP OUT UPIU contain the Basic Header as described in 10.6.2, Basic Header Format. Specific details are as follows:

a) Task Tag

Task Tag normally is related to I_T_L_Q nexus addressing of SCSI while here it is used in a pure UTP (device level) context.

b) Transaction Type

A type code value of xx00 00000b indicates a NOP OUT UPIU.

c) Flags

The Flags field value shall be equal to zero.

d) Data Segment Length

The Data Segment Length field shall contain zero as there is no Data Segment in this UPIU.

10.7.12 NOP IN UPIU

NOP IN UPIU is the response from the Target device to a NOP OUT UPIU sent by the Initiator device.

Table 10-57 — NOP IN UPIU

NOP IN UPIU			
0 xx10 0000b	1 Flags	2 Reserved	3 Task Tag
4 Reserved	5 Reserved	6 Response (00h)	7 Reserved
8 Total EHS Length (00h)	9 Device Information (00h)	10 (MSB) / 11 (LSB) Data Segment Length (0000h)	
12	13	14	15
Reserved			
16	17	18	19
Reserved			
20	21	22	23
Reserved			
24	25	26	27
Reserved			
28	29	30	31
Reserved			
32 (MSB)	33	34	35 (LSB)
Header E2ECRC (omit if HD=0)			

10.7.12.1 Basic Header

The first 12 bytes of the NOP IN UPIU contain the Basic Header as described in 10.6.2, Basic Header Format. Specific details are as follows:

a) Transaction Type

A type code value of xx10 00000b indicates a NOP IN UPIU.

b) Flags

The Flags field value shall be equal to zero.

c) Task Tag

The Task Tag shall be equal to the Task Tag value of the corresponding NOP OUT UPIU.

d) Response

The Response field shall be set to 00h (Target Success) indicating that the Target device was able to respond to the NOP OUT UPIU.

e) Data Segment Length

The Data Segment Length field shall contain zero as there is no Data Segment in this UPIU.

10.7.13 Data out transfer rules

Data out transfer rules related with RTT have been defined for the following reasons:

- To have the same implementation of UFS data transfer mechanisms for data out transactions across various host and device vendors.
- To limit the number of outstanding RTTs sent by the device based on host capability.

RTT requests related to several commands and from any logical units may be sent in any order.

Examples of commands with data out transfer are: MODE SELECT (10), WRITE (6), WRITE (10), WRITE (16), FORMAT UNIT, SECURITY PROTOCOL OUT, etc.

The following rules are applicable for device in generating the RTT and the host in handling the RTT:

- Rule 1 - The host shall send only one DATA OUT UPIU for each RTT received from the device.
  - Figure 10-4 shows an example of UTP traffic related to a write command processing: the host sends one and only one DATA OUT UPIU for each READY TO TRANSFER UPIU sent by the device.

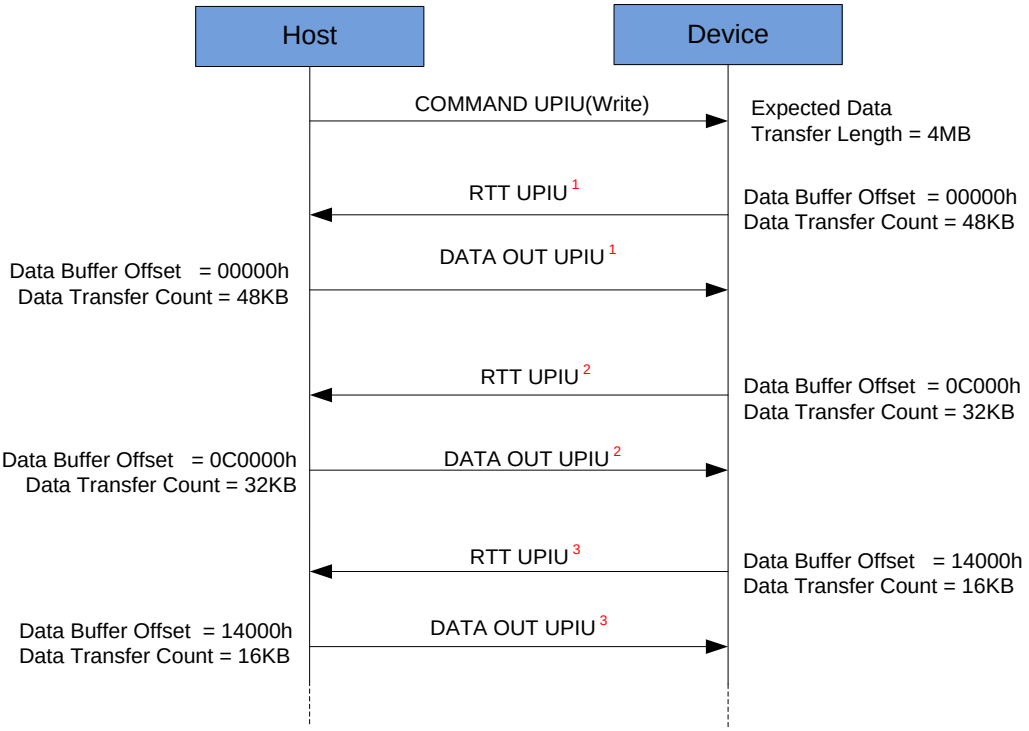


Figure 10-4 — Example for data out transfer rule 1

  - If the Data Buffer Offset field value or the Data Transfer Count field value in the DATA OUT UPIU does not match the corresponding parameters in the RTT request, the device shall terminate the command by sending RESPONSE UPIU with UTP Data Out Mismatch Error flag (Flags.D) set and Status = CHECK CONDITION with SENSE KEY = ABORTED COMMAND. In this case, the device shall wait until it receives all DATA OUT UPIUs related to any outstanding READY TO TRANSFER UPIUs before sending the Response UPIU with UTP Data Out Mismatch Error flag (Flags.D) set and Status = CHECK CONDITION with SENSE KEY = ABORTED COMMAND. Figure 10-5 describes a scenario, where the Data Transfer Count value does not match.

10.7.13 Data out transfer rules (cont'd)

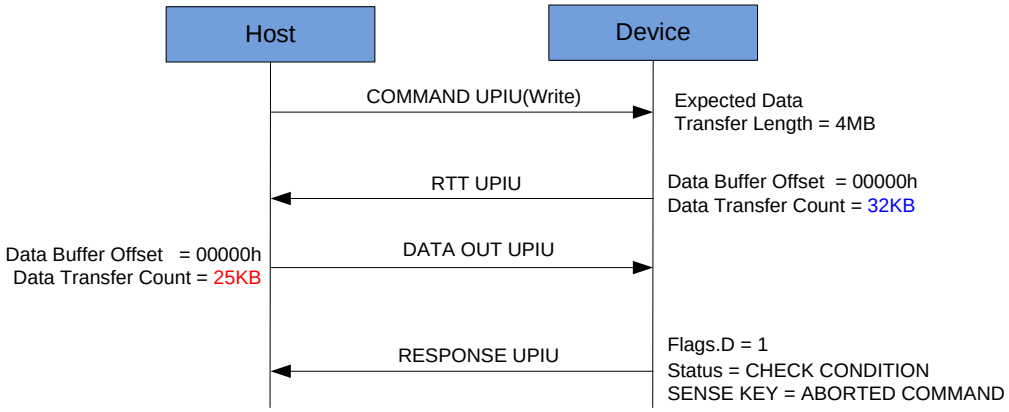


Figure 10-5 — Example for Data Transfer Count mismatch

- Rule 2 – Device shall not have outstanding RTTs more than specified by host
  - bDeviceRTTCap read-only parameter in Device Descriptor defines the maximum number of outstanding RTTs which can be supported by device. bMaxNumOfRTT read-write attribute limits the number of outstanding RTTs generated by the device. bMaxNumOfRTT is equal to two after device manufacturing, and it may be changed by the host according to its capability to increase performance. In particular, bMaxNumOfRTT value shall not be set to a value greater than bDeviceRTTCap value, and it may be set only when the command queues of all logical units are empty. If the host attempts to write a value higher than what indicated by bDeviceRTTCap, the value shall not be changed and the QUERY RESPONSE UPIU shall have Query Response field set to "Invalid VALUE". If the host attempts to write bMaxNumOfRTT when there is at least one logical unit with command queue not empty, the operation shall fail, and Query Response field in the QUERY RESPONSE UPIU shall be set to FFh ("General failure").
  - Figure 10-6 shows an example of UTP traffic related to a write command processing assuming bMaxNumOfRTT = 2. The Target device sends RTT UPIU[1] and RTT UPIU[2], and the Initiator device starts to provide data related to first request. There are two outstanding RTTs (RTT UPIU[1] and RTT UPIU[2]) therefore the Target device cannot send additional RTT UPIUs. The Target device sends the third data request (RTT UPIU[3]) only after receiving DATA OUT UPIU[1].

10.7.13 Data out transfer rules (cont'd)

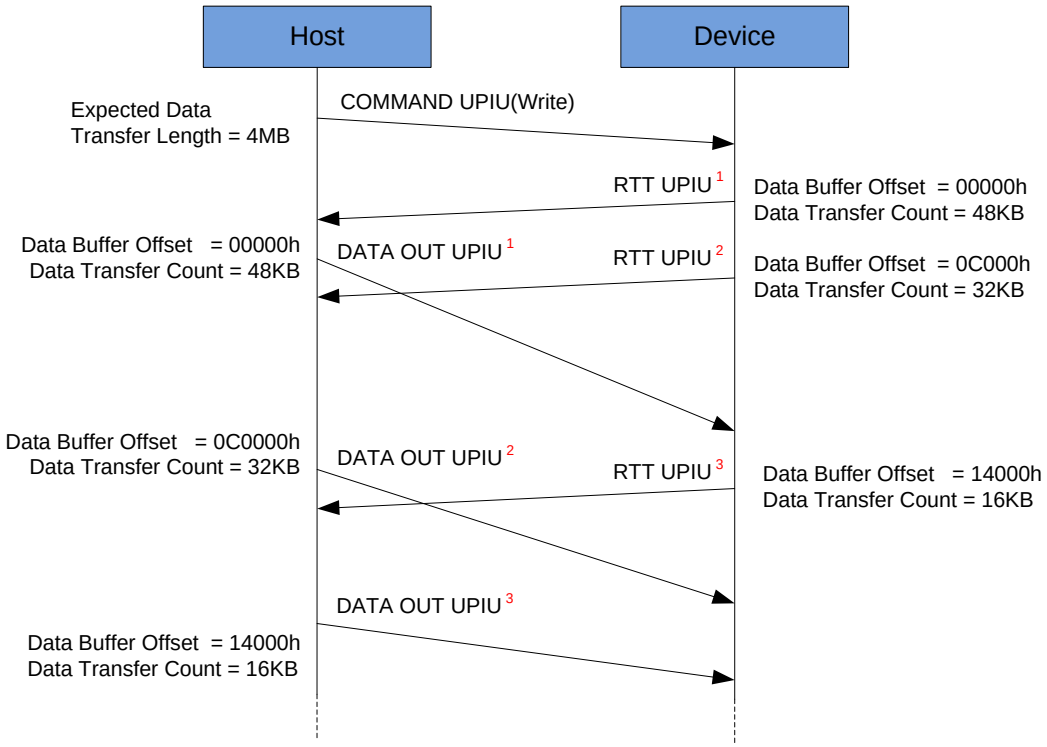


Figure 10-6 — Example for data out transfer rule 2

- Rule 3 – Across all logical units, DATA OUT UPIUs shall be sent in the same order of RTT UPIUs: data for RTT^{N} shall be transferred before transfering data for RTT^{N+1}.
  - For example, if the host first receives RTT $UPIU^{1}$ and then RTT $UPIU^{2}$ from the device, it shall send data related to the first request (Data1) prior to data related to the second request (Data2).

10.7.13 Data out transfer rules (cont'd)

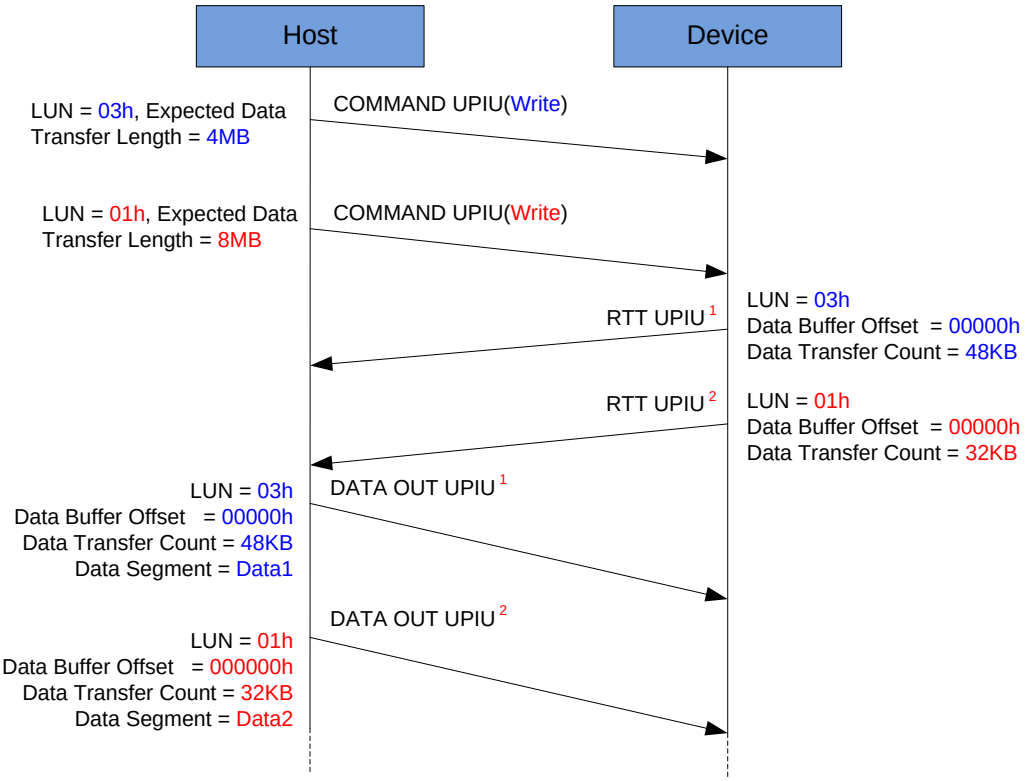


Figure 10-7 — Example for data out transfer rule 3

- It is recommended for device to determine the sequence of RTTs based on logical unit priority, command priority, internal optimization, etc.

10.7.13.1 Implementation

The following parameters are defined to implement data out transfer rules.

Table 10-58 — Parameters related to data out transfer rules

bMaxNumOfRTT	Defines the current maximum number of outstanding RTTs that are allowed. This value can be set by the host.
bDeviceRTTCap	Defines the maximum number of outstanding RTTs supported by device
UTP Data Out Mismatch Error Flag(D)	The D flag shall be set to '1' to indicate that a Data Out mismatch error occurred during the command transaction

10.8 Logical Units

This section gives more details on the definition of logical unit in the UFS Standard.

10.8.1 UFS SCSI Domain

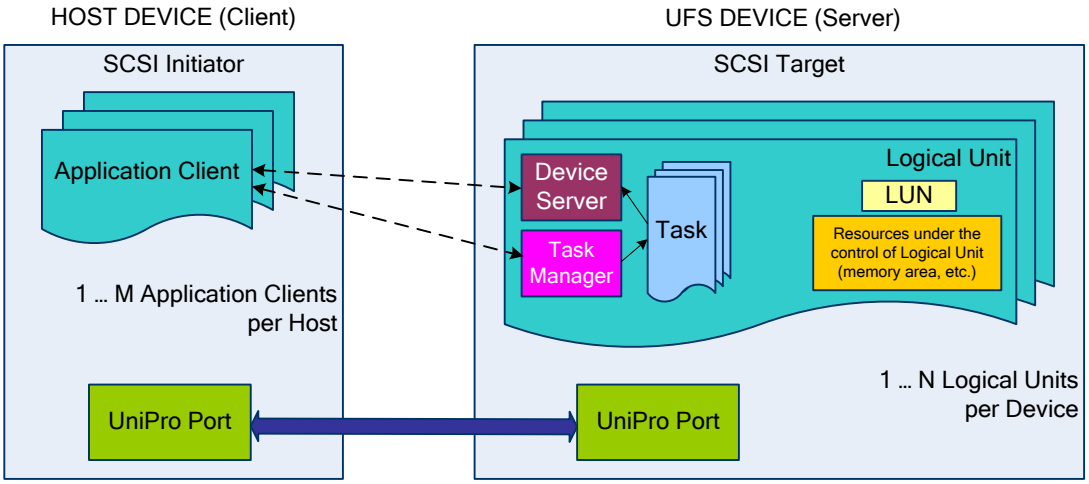


Figure 10-8 — UFS SCSI domain

10.8.2 UFS Logical Unit Definition

A logical unit (LU) is an externally addressable, independent, processing entity that processes SCSI tasks (commands) and performs task management functions.

- Each logical unit is independent of other logical units in a device
- UFS shall support the amount of logical units specified by bMaxNumberLU, in addition to the well known logical units defined in 10.8.5.
- Logical units may be used to store boot code, application code and mass storage data applications

Commands addressed to logical unit ‘i’ are handled by logical unit ‘i’ exclusively, not visible, handled or processed by logical unit ‘j’

A logical unit contains the following:

- DEVICE SERVER: A conceptual object within a logical unit that processes SCSI commands.
- TASK MANAGER: A conceptual object within a logical unit that controls the sequencing of commands and performs task management functions.
- TASK SET: A conceptual group of 1 or more commands (a list, queue, etc.)

10.8.3 Well Known Logical Unit Definition

Well known logical units, as defined by SCSI, support very specific types of commands, usually only four or five commands such as REPORT LUNS command to allow an application client to issue requests to receive specific information usually relating to the entire device.

In this standard, additional well known logical units are defined for specific UFS functions, including Boot and RPMB. Each well known logical unit has a well known logical unit number (W-LUN).

10.8.4 Logical Unit Addressing

The 8-bit LUN field in UPIU is used to provide either LUN or W-LUN. In particular, the most significant bit of this field (WLUN_ID) shall be set according to the logical unit type as follows:

- WLUN_ID = 0b for logical unit,
- WLUN_ID = 1b for well known logical unit.

The remaining 7 bits of the LUN field (UNIT_NUMBER_ID) shall be set to either the LUN value or the W-LUN value, depending on the logical unit type.

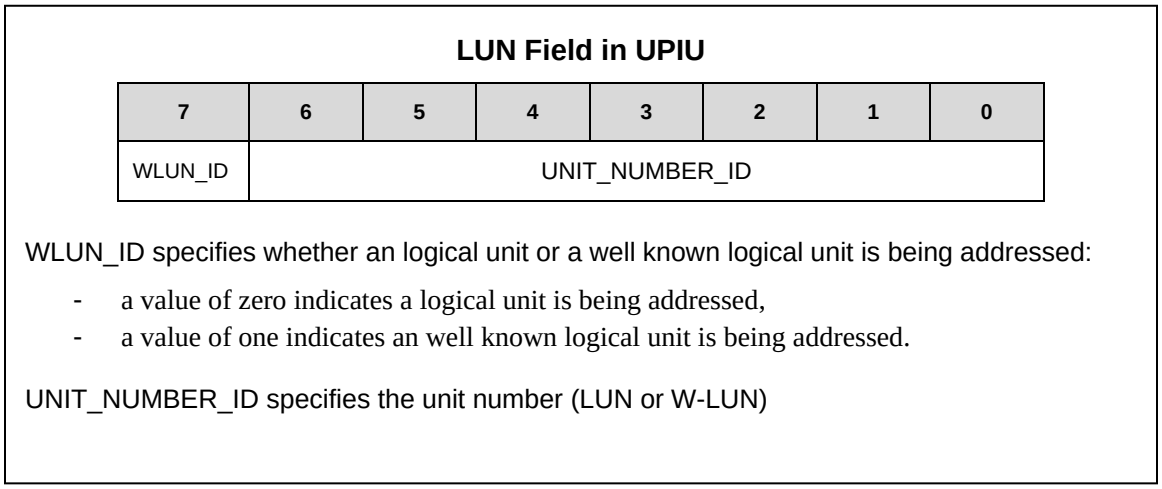


Figure 10-9 — Logical Unit Addressing

Therefore, the encoding of the LUN field in UPIU supports up to 128 LUN's and up to 128 W-LUN's (0 <= UNIT_NUMBER_ID <= 127) .

10.8.5 Well Known Logical Unit Defined in UFS

The following well known logical units are defined in this standard for SCSI and UFS specific functions: REPORT LUNS, UFS Device, Boot, RPMB.

The REPORT LUNS well known logical unit is defined in [SPC] and provides the logical unit inventory. The UFS Device well known logical unit provides UFS device level interaction (i.e., Power mode control, Wipe Device). The Boot well known logical unit is a virtual reference to the actual logical unit containing boot code, as designated by the host. The Boot well known logical unit is read at the system startup to access the boot code. The RPMB well known logical unit supports the RPMB function with its own independent processes and memory space as dictated by the RPMB security definition.

Each well known logical unit shall only process the commands listed in Table 10-59. If one of the four well known logical units receives a command that is not listed in Table 10-59, then the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID COMMAND OPERATION CODE.

Table 10-59 — Well known logical unit commands

Well known logical unit	W-LUN	LUN Field in UPIU	Command name
REPORT LUNS	01h	81h	INQUIRY, REQUEST SENSE, TEST UNIT READY, REPORT LUNS
UFS Device	50h	D0h	INQUIRY, REQUEST SENSE, TEST UNIT READY, START STOP UNIT, FORMAT UNIT
Boot	30h	B0h	INQUIRY, REQUEST SENSE, TEST UNIT READY, READ (6), READ (10), READ (16)
RPMB	44h	C4h	INQUIRY,REQUEST SENSE, TEST UNIT READY, SECURITY IN, SECURITY OUT

NOTE If bBootEnable field in the Device Descriptor is set to zero or if the Boot well known logical is not mapped to an enabled logical unit (see bLUEnable, bBootLunID and bBootLunEn), then the Boot well known logical unit shall terminate TEST UNIT READY, READ (6), READ (10), and READ (16) commands with CHECK CONDITION status.

10.8.6 Translation of 8-bit UFS LUN to 64-bit SCSI LUN Address

The SCSI Architecture Model describes a 64-bit LUN addressing scheme. The value of C1h in the first 8 bits of the 64-bit address indicates a well known LUN address in the SAM SCSI format. Examples of translation of the 8-bit LUN field value in UPIU to 64-bit SCSI address are shown in Table 10-60.

Table 10-60 — Examples of logical unit representation format

Logical unit			
Logical Unit Name	**LUN field in UPIU**	**LUN**	**SAM LUN**
Logical Unit 1	01h	01h	00 01 00 00 00 00 00 00h
Logical Unit 6	06h	06h	00 06 00 00 00 00 00 00h
Well known logical unit			
Logical Unit Name	**LUN field in UPIU**	**W-LUN**	**SAM LUN**
Boot	B0h	30h	C1 30 00 00 00 00 00 00h
RPMB	C4h	44h	C1 44 00 00 00 00 00 00h

10.8.7 SCSI Write Command

The execution of a SCSI write command is composed by the following subsequent phases: command, data, status. In the command phase a write command is sent to the device using COMMAND UPIU.

During the subsequent phase data is delivered to the UFS device using DATA OUT UPIU: the UFS device paces the data delivery by sending a READY TO TRANSFER UPIU when it is ready for the next DATA OUT UPIU. (The UFS device may send new READY TO TRANSFER UPIU's before it receives data for previous request.)

The write command terminates with a RESPONSE UPIU that contains the status.

Figure 10-10 shows an example of UPIU sequence for SCSI write command.

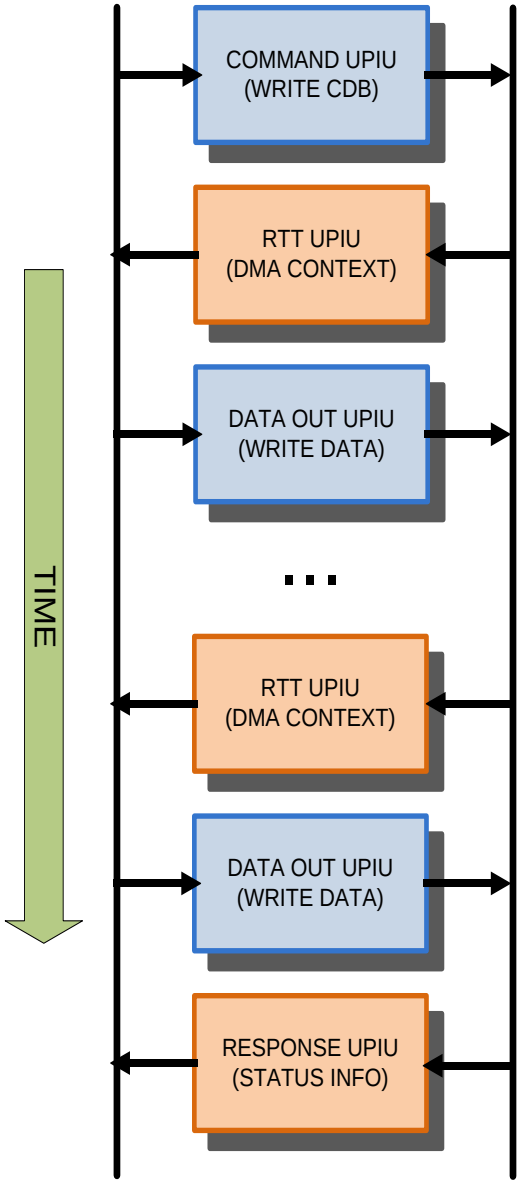


Figure 10-10 — SCSI Write

10.8.8 SCSI Read Command

The execution of a SCSI read command is composed by the following subsequent phases: command, data, status. In the command phase a read command is sent to the device using COMMAND UPIU. During the subsequent phase the UFS device delivers data to the host using DATA IN UPIU. The read command terminates with a RESPONSE UPIU that contains the status.

Figure 10-11 shows an example of UPIU sequence for SCSI read command.

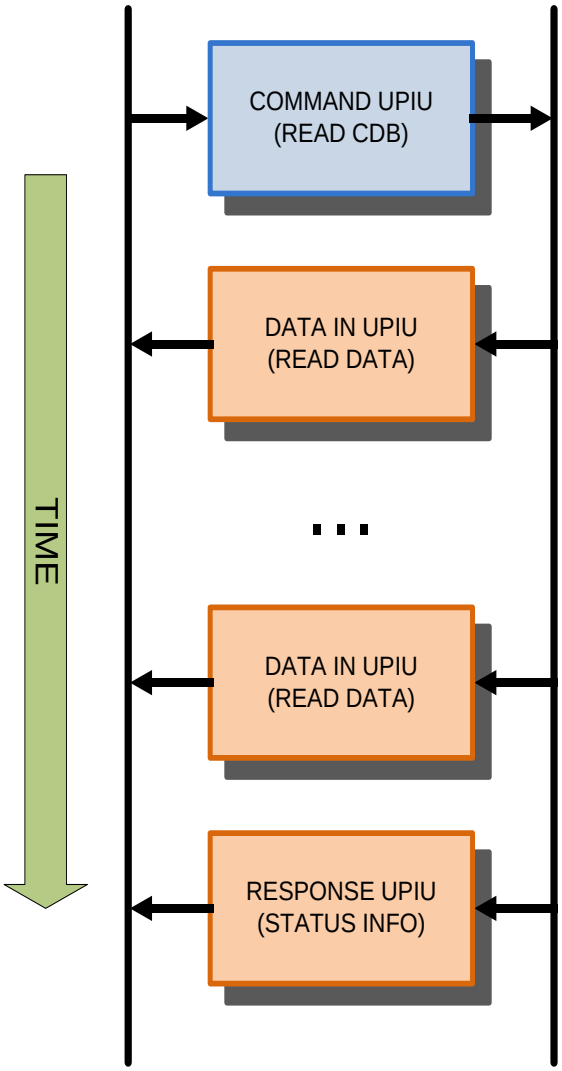


Figure 10-11 — SCSI Read

10.9 Application Layer and Device Manager Transport Protocol Services

10.9.1 UFS Initiator Port and Target Port Attributes

Table 10-61 — UFS Initiator Port and Target Port Attributes

Attribute	Value
Maximum CDB Length	16 bytes
Command Identifier Size	16 bits
Task Attributes Supported	Simple, Head of Queue, Ordered, ACA (not supported)
Maximum Data-In Buffer Size	FFFFh
Maximum Data-Out Buffer Size	FFFFh
Maximum CRN	Not Applicable
Command Priority Supported	Yes (field width = 1 bit)
Maximum Sense Data Length	FFFFh
Status Qualifier Supported	No
Additional Response Information Supported	Yes
Bidirectional Commands Supported	No
Task Management Functions Supported	Abort Task, Abort Task Set, Clear Task Set, Logical Unit Reset, Query Task, Query Task Set

10.9.2 Execute Command procedure call transport protocol services

All SCSI transport protocol standards shall define the SCSI transport protocol specific requirements for implementing the Send SCSI Command request, the SCSI Command Received indication, the Send Command Complete response, and the Command Complete Received confirmation SCSI transport protocol services.

All SCSI initiator devices shall implement the Send SCSI Command request and the Command Complete Received confirmation SCSI transport protocol services as defined in the applicable SCSI transport protocol standards. All SCSI target devices shall implement the SCSI Command Received indication and the Send Command Complete response SCSI transport protocol services as defined in the applicable SCSI transport protocol standards.

Initiator device	Target device
Send SCSI Command request	SCSI Command Received indication
Command Complete Received confirmation	Send Command Complete response

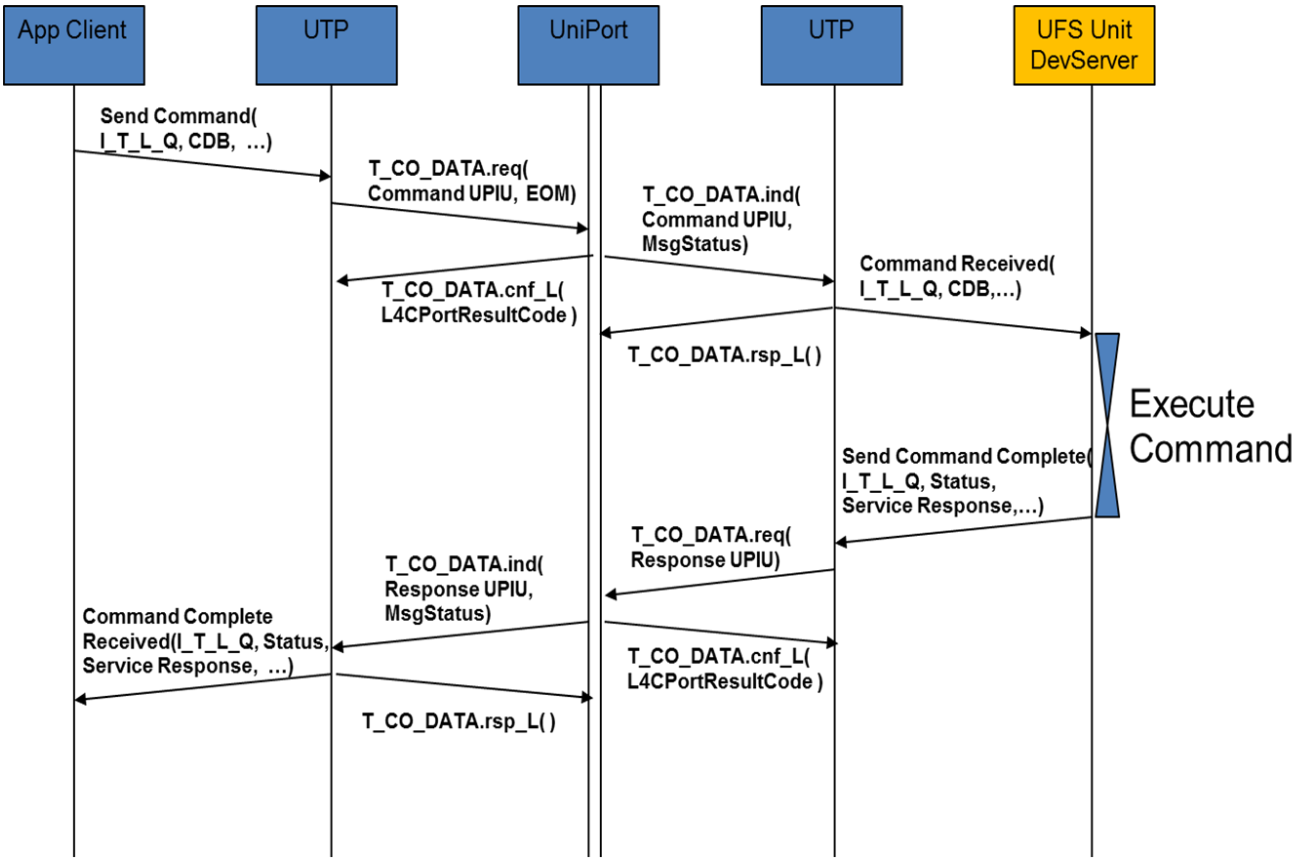


Figure 10-12 — Command without Data Phase

10.9.3 SCSI Command transport protocol service

An application client uses the Send Command transport protocol service request to request a UFS Initiator port transmit a COMMAND UPIU via UniPort

- Send SCSI Command (IN (I_T_L_Q Nexus, CDB, Task Attribute, [Data-in Buffer Size], [Data-out Buffer], [Data-out Buffer Size], [CRN], [Command Priority], [First Burst Enabled]))

Table 10-62 — Send SCSI Command transport protocol service

Argument	Implementation
I_T_L_Q Nexus	I specifies the initiator port to send the COMMAND UPIU T specifies the target port to which the COMMAND UPIU is to be sent L specifies the LUN field in the COMMAND UPIU Q specifies the Task Tag field in the COMMAND UPIU
CDB	Specifies the CDB field in the COMMAND UPIU
Task Attribute	Specifies the Flags.ATTR of the Flag field in the COMMAND UPIU
[Data-in Buffer Size]	Expected Data Transfer Length
[Data-out Buffer]	Internal to UFS Initiator port
[Data-out Buffer Size]	Expected Data Transfer Length
[CRN]	Reserved
[Command Priority]	Specifies the Flags.CP of the Flag field in the COMMAND UPIU. The mapping of 16 levels in SCSI Command Priority to 2 levels in Flags.CP field is left for implementation.
[First Burst Enabled]	An argument specifying that a SCSI transport protocol specific number of bytes from the Data-Out Buffer shall be delivered to the logical unit without waiting for the device server to invoke the Receive Data-Out SCSI transport protocol service. A non-zero value in the Data Segment Length field indicates First Burst Enabled.

10.9.4 SCSI Command Received transport protocol

A UFS target port uses the SCSI Command Received transport protocol service indication to notify a task manager that it has received a COMMAND UPIU.

- SCSI Command Received (IN (I_T_L_Q Nexus, CDB, Task Attribute, [CRN], [Command Priority], [First Burst Enabled))

Table 10-63 — SCSI Command Received transport protocol

Argument	Implementation
I_T_L_Q Nexus	I indicates the initiator port from which COMMAND UPIU was received T indicates the target port which receives the COMMAND UPIU L indicates the LUN field in the COMMAND UPIU Q indicates the Task Tag field in the COMMAND UPIU
CDB	Specifies the CDB field in the COMMAND UPIU
Task Attribute	Specifies the Flags.ATTR of the Flag field in the COMMAND UPIU
[CRN]	Reserved
[Command Priority]	Specifies the Flags.CP of the Flag field in the COMMAND UPIU. The mapping of 16 levels in SCSI Command Priority to 2 levels in Flags.CP field is left for implementation.
[First Burst Enabled]	An argument specifying that a SCSI transport protocol specific number of bytes from the Data-Out Buffer shall be delivered to the logical unit without waiting for the device server to invoke the Receive Data-Out SCSI transport protocol service. A non-zero value in the Data Segment Length field indicates First Burst Enabled.

10.9.5 Send Command Complete transport protocol service

A device server uses the Send Command Complete transport protocol service response to request that a UFS target port transmit a RESPONSE UPIU.

- Send Command Complete (IN (I_T_L_Q Nexus, [Sense Data], [Sense Data Length], Status, [Status Qualifier], Service Response))

A device server shall only call Send Command Complete () after receiving SCSI Command Received ().

A device server shall not call Send Command Complete () for a given I_T_L_Q nexus until:

a) all its outstanding Receive Data-Out () calls for that I_T_L_Q nexus have been responded to with

   - Data-Out Received (); and

b) all its outstanding Send Data-In () calls for that I_T_L_Q nexus have been responded to with Data-In Delivered ().

Table 10-64 — Send Command Complete transport protocol service

Argument	Implementation
I_T_L_Q Nexus	I specifies the initiator port to send the RESPONSE UPIU T specifies the target port to which the RESPONSE UPIU is to be sent L specifies the LUN field in the RESPONSE UPIU Q specifies the Task Tag field in the RESPONSE UPIU
[Sense Data]	Specifies the Sense Data field in the RESPONSE UPIU
[Send Data Length]	Specifies the Sense Data Length field in the RESPONSE UPIU
Status	Specifies the Status Length field in the RESPONSE UPIU
[Status Qualifier]	Ignored
Service Response	Specifies the Response field in the RESPONSE UPIU

10.9.6 Command Complete Received transport protocol service

A UFS initiator port uses the Command Complete Received transport protocol service confirmation to notify an application client that it has received a response for its COMMAND UPIU.

- Command Complete Received (IN (I_T_L_Q Nexus, [Data-in Buffer], [Sense Data], [Sense Data Length], Status, [Status Qualifier], Service Response))

Table 10-65 — Command Complete Received transport protocol service

Argument	Implementation
I_T_L_Q Nexus	I specifies the initiator port to send the RESPONSE UPIU T specifies the target port to which the RESPONSE UPIU is to be sent L specifies the LUN field in the RESPONSE UPIU Q specifies the Task Tag field in the RESPONSE UPIU
[Data-in Buffer]	A buffer containing command specific information returned by the logical unit on command completion
[Sense Data]	Specifies the Sense Data field in the RESPONSE UPIU
[Send Data Length]	Specifies the Sense Data Length field in the RESPONSE UPIU
Status	Specifies the Status Length field in the RESPONSE UPIU
[Status Qualifier]	Ignored
Service Response	Specifies the Response field in the RESPONSE UPIU

10.9.7 Data transfer SCSI transport protocol services

The data transfer services provide mechanisms for moving data to and from the SCSI initiator port while processing commands. All SCSI transport protocol standards shall define the protocols required to implement these services.

The application client's Data-In Buffer and/or Data-Out Buffer each appears to the device server as a single, logically contiguous block of memory large enough to hold all the data required by the command.

10.9.7.1 Send Data-In transport protocol service

A device server uses the Send Data-In transport protocol service request to request that a UFS target port sends data.

- Send Data-In (IN (I_T_L_Q Nexus, Device Server Buffer, Application Client Buffer Offset, Request Byte Count))

A device server shall only call Send Data-In () during a read operation.

A device server shall not call Send Data-In () for a given I_T_L_Q nexus after it has called Send Command Complete () for that I_T_L_Q nexus (e.g., a RESPONSE UPIU with for that I_T_L_Q nexus) or called Task Management Function Executed for a task management function that terminates that task (e.g., an ABORT TASK).

Table 10-66 — Send Data-In transport protocol service

Argument	Implementation
I_T_L_Q Nexus	I_T_L_Q of the corresponding COMMAND UPIU
Device Server Buffer	Internal to device server
Application Client Buffer Offset	Offset in bytes from the beginning of the application client's buffer to the first byte of transferred data.
Request Byte Count	Specifies the length of the read data specified by the command

10.9.7.2 Data-In Delivered transport protocol service

This confirmation notifies the device server that the specified data was successfully delivered to the application client buffer, or that a UniPro delivery subsystem error occurred while attempting to deliver the data.

- Data-In Delivered (IN (I_T_L_Q Nexus, Delivery Result))

Table 10-67 — Data-In Delivered transport protocol service

Argument	Implementation
I_T_L_Q Nexus	I_T_L_Q of the corresponding COMMAND UPIU
Delivery Result	DELIVERY SUCCESSFUL: The data was delivered successfully. DELIVERY FAILURE: A UTP service delivery error occurred while attempting to deliver the data.

10.9.7.2 Data-In Delivered transport protocol service (cont'd)

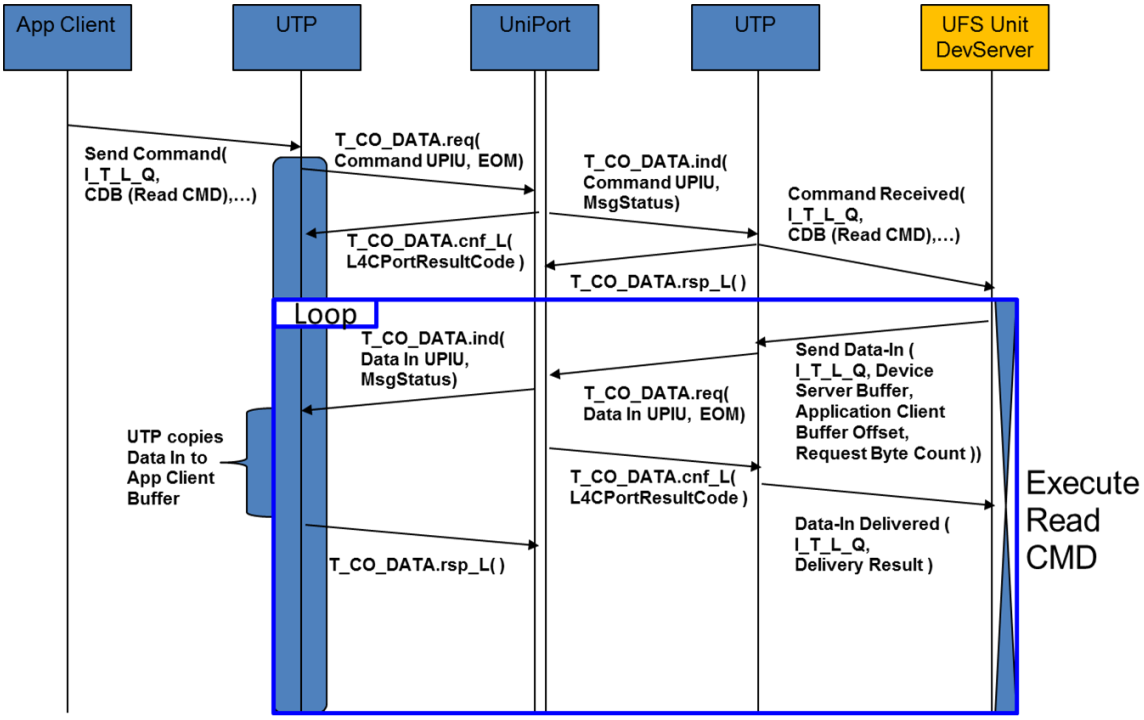


Figure 10-13 — Command + Read Data Phase 1/2

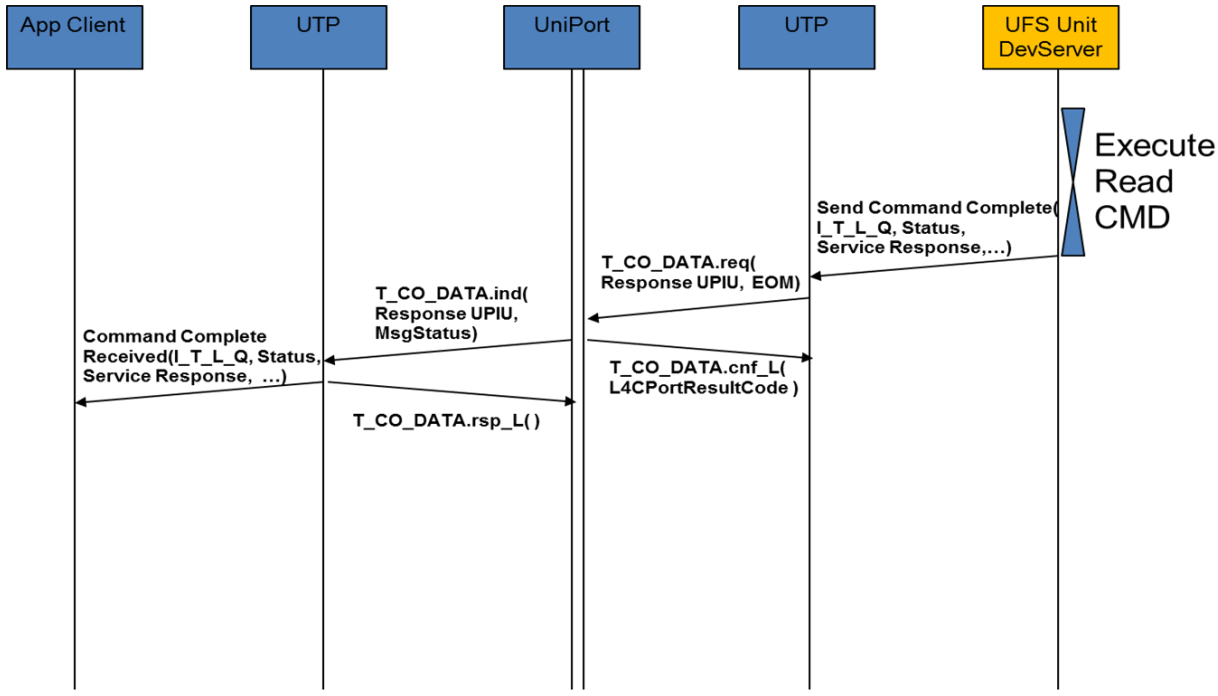


Figure 10-14 — Command + Read Data Phase 2/2

10.9.7.3 Receive Data-Out transport protocol service

A device server uses the Receive Data-Out transport protocol service request to request that a UFS target port receives data.

- Receive Data-Out (IN (I_T_L_Q Nexus, Application Client Buffer Offset, Request Byte Count, Device Server Buffer))

A device server shall only call Receive Data-Out () during a write operation.

A device server shall not call Receive Data-Out () for a given I_T_L_Q nexus after a Send Command Complete () has been called for that I_T_L_Q nexus or after a Task Management Function Executed () has been called for a task management function that terminates that command (e.g., an ABORT TASK).

Table 10-68 — Receive Data-Out transport protocol service

Argument	Implementation
I_T_L_Q Nexus	I_T_L_Q of the corresponding COMMAND UPIU
Application Client Buffer Offset	Offset in bytes from the beginning of the application client's buffer to the first byte of transferred data
Device Server Buffer	Internal to device server
Request Byte Count	Number of bytes to be moved by this request

10.9.7.4 Data-Out Received transport protocol service

A UFS target port uses the Data-Out Received transport protocol service indication to notify a device server that it has received data.

- Data-out Received (IN (I_T_L_Q Nexus, Delivery Result))

Table 10-69 — Data-Out Received transport protocol service

Argument	Implementation
I_T_L_Q Nexus	I_T_L_Q of the corresponding COMMAND UPIU
Delivery Result	DELIVERY SUCCESSFUL: The data was delivered successfully. DELIVERY FAILURE: A UTP service delivery error occurred while attempting to deliver the data.

10.9.7.4 Data-Out Received transport protocol service (cont'd)

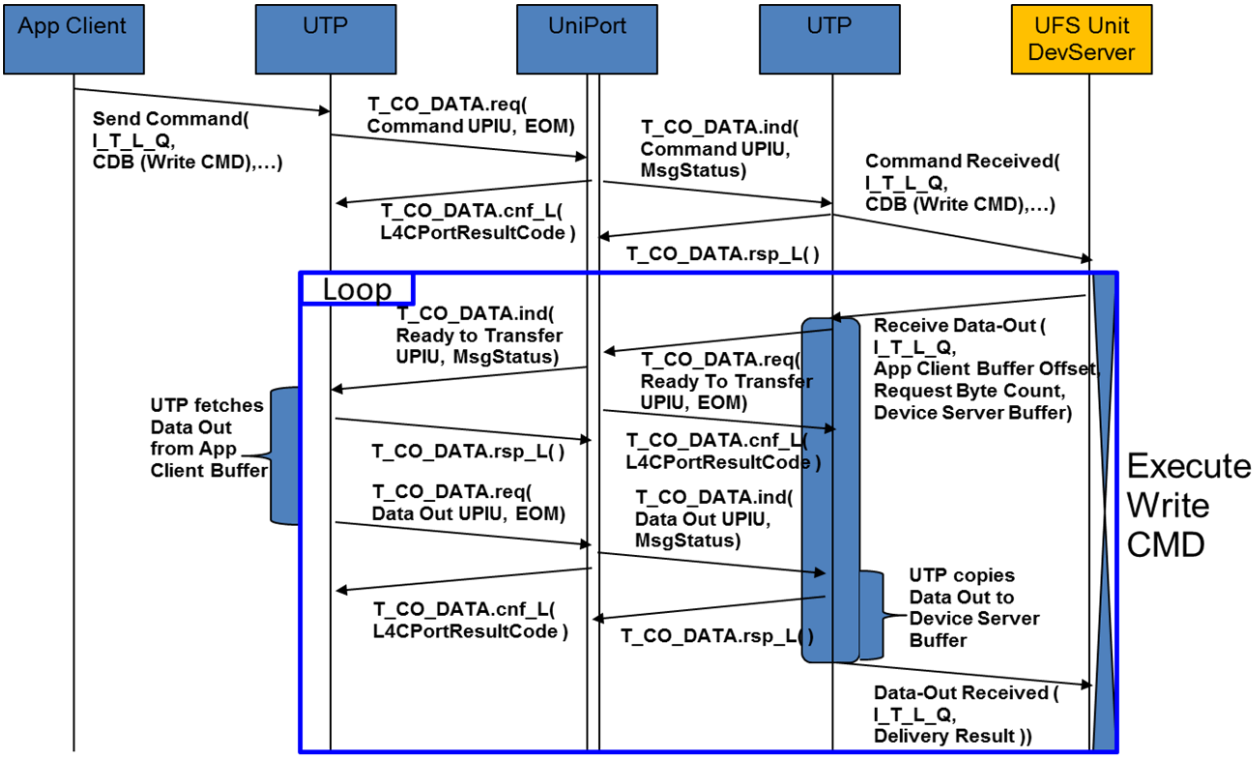


Figure 10-15 — Command + Write Data Phase ½

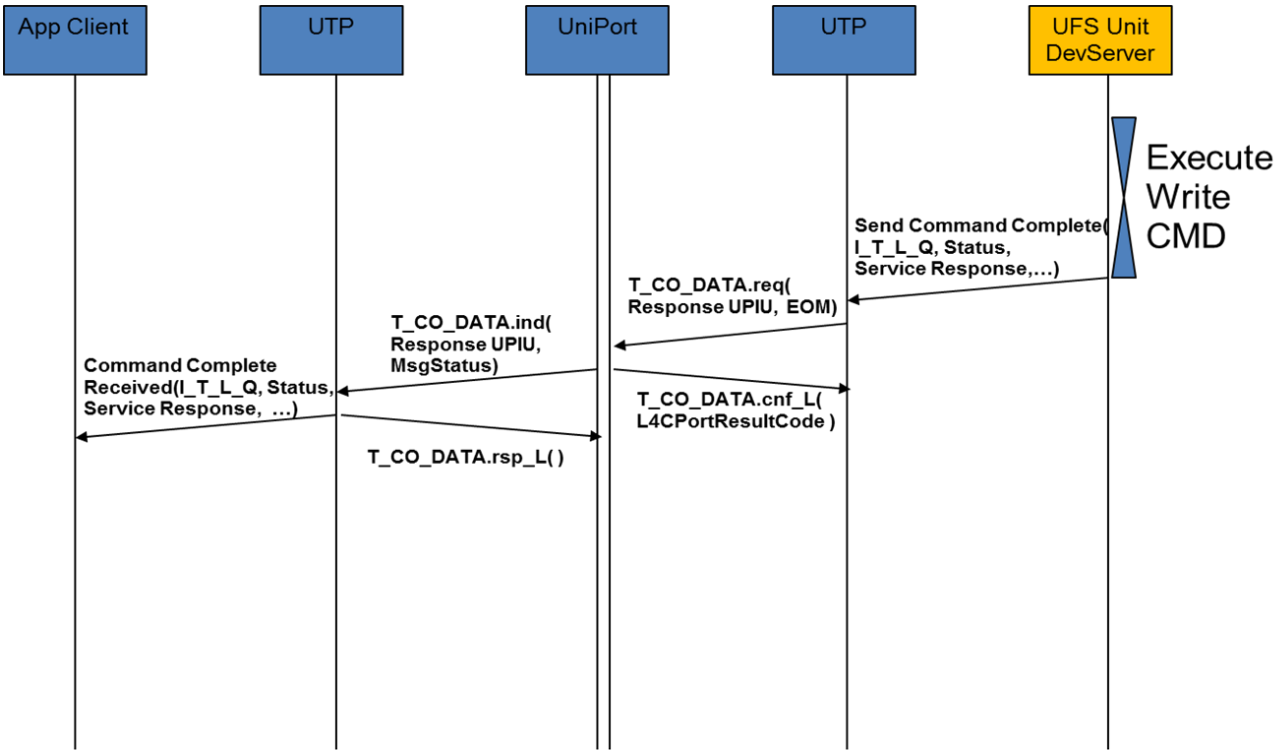


Figure 10-16 — Command + Write Data Phase 2/2

10.9.8 Task Management Function procedure calls

An application client requests the processing of a task management function by invoking the SCSI transport protocol services:

- Service Response = Function name (IN (Nexus), OUT ([Additional Response Information])

Table 10-70 — Task Management Function procedure calls

Task Management Function (Function name)	Nexus argument
Abort Task	I_T_L_Q Nexus
Abort Task Set	I_T_L Nexus
Clear Task Set	I_T_L Nexus
Logical Unit Reset	I_T_L Nexus
Query Task	I_T_L_Q Nexus
Query Task Set	I_T_L Nexus

One of the following SCSI transport protocol specific service responses shall be returned:

Table 10-71 — SCSI transport protocol service responses

FUNCTION COMPLETE	A task manager response indicating that the requested function is complete. Unless another response is required, the task manager shall return this response upon completion of a task management request supported by the logical unit or SCSI target device to which the request was directed.
FUNCTION SUCCEEDED	A task manager response indicating that the requested function is supported and completed successfully. This task manager response shall only be used by functions that require notification of success (e.g., QUERY TASK, QUERY TASK SET)
FUNCTION REJECTED	A task manager response indicating that the requested function is not supported by the logical unit or SCSI target device to which the function was directed.
INCORRECT LOGICAL UNIT NUMBER	A task router response indicating that the function requested processing for an incorrect logical unit number.
SERVICE DELIVERY OR TARGET FAILURE	The request was terminated due to a service delivery failure SCSI target device malfunction. The task manager may or may not have successfully performed the specified function.

10.9.8.1 ABORT TASK

This function shall be supported by all logical units.

The task manager shall abort the specified command, if it exists. Previously established conditions, including mode parameters, and reservations shall not be changed by the ABORT TASK function.

A response of FUNCTION COMPLETE shall indicate that the command was aborted or was not in the task set. In either case, the SCSI target device shall guarantee that no further requests or responses are sent from the command.

If the processing of the task that is requested to be aborted requires Data-Out data transfer, then Target device shall wait until it receives all DATA OUT UPIUs related to any outstanding READY TO TRANSFER UPIUs before sending the task management response.

All SCSI transport protocol standards shall support the ABORT TASK task management function.

Service Response = ABORT TASK (IN (I_T_L_Q Nexus))

10.9.8.2 ABORT TASK SET

This function shall be supported by all logical units.

The task manager shall abort all commands in the task set that were received on the specified I_T_L nexus. Commands received on other I_T nexuses or in other task sets shall not be aborted. This task management function performed is equivalent to a series of ABORT TASK requests.

All pending status and sense data for the commands that were aborted shall be cleared. Other previously established conditions, including mode parameters, and reservations shall not be changed by the ABORT TASK SET function.

If the processing of one or more tasks in the task set require Data-Out data transfer, then Target device shall wait until it receives all DATA OUT UPIUs related to any outstanding READY TO TRANSFER UPIUs before sending the task management response.

All SCSI transport protocol standards shall support the ABORT TASK SET task management function.

Service Response = ABORT TASK SET (IN (I_T_L Nexus))

10.9.8.3 CLEAR TASK SET

This function shall be supported by all logical units.

The task manager shall abort all commands in the task set.

If the processing of one or more tasks in the task set require Data-Out data transfer, then Target device shall wait until it receives all DATA OUT UPIUs related to any outstanding READY TO TRANSFER UPIUs before sending the task management response.

All pending status and sense data for the task set shall be cleared. Other previously established conditions, including mode parameters, and reservations shall not be changed by the CLEAR TASK SET function.

All SCSI transport protocol standards shall support the CLEAR TASK SET task management function.

Service Response = CLEAR TASK SET (IN (I_T_L Nexus))

10.9.8.4 LOGICAL UNIT RESET

This function shall be supported by all logical units.

Before returning a FUNCTION COMPLETE response, the logical unit shall perform the logical unit reset functions.

If the processing of one or more tasks in the logical unit requires Data-Out data transfer, then Target device shall wait until it receives all DATA OUT UPIUs related to any outstanding READY TO TRANSFER UPIUs before sending the task management response.

All SCSI transport protocol standards shall support the LOGICAL UNIT RESET task management function.

Service Response = LOGICAL UNIT RESET (IN (I_T_L Nexus))

10.9.8.5 QUERY TASK

UFS transport protocols shall support QUERY TASK.

The task manager in the specified logical unit shall:

1) If the specified command is present in the task set, then return a service response set to FUNCTION SUCCEEDED; or

2) If the specified command is not present in the task set, then return a service response set to FUNCTION COMPLETE.

Service Response = QUERY TASK (IN (I_T_L_Q Nexus))

10.9.8.6 QUERY TASK SET

UFS transport protocols shall support QUERY TASK SET.

The task manager in the specified logical unit shall:

1) If there is any command present in the task set specified I_T_L nexus, then return a service response set to FUNCTION SUCCEEDED; or

2) If there is no command present in the task set specified I_T nexus, then return a service response set to FUNCTION COMPLETE.

Service Response = QUERY TASK SET (IN (I_T_L Nexus))

10.9.8.7 Task Management SCSI Transport Protocol Services

UFS standard shall define the SCSI transport protocol specific requirements for implementing the Send Task Management Request request, the Task Management Request Received indication, the Task Management Function Executed response, and the Received Task Management Function Executed confirmation SCSI transport protocol services.

A SCSI transport protocol standard may specify different implementation requirements for the Send Task Management Request request SCSI transport protocol service for different values of the Function Identifier argument.

All SCSI initiator devices shall implement the Send Task Management Request request and the Received Task Management Function Executed confirmation SCSI transport protocol services as defined in the applicable SCSI transport protocol standards.

10.9.8.7 Task Management SCSI Transport Protocol Services (cont'd)

All SCSI target devices shall implement the Task Management Request Received indication and the Task Management Function Executed response SCSI transport protocol services as defined in the applicable SCSI transport protocol standards.

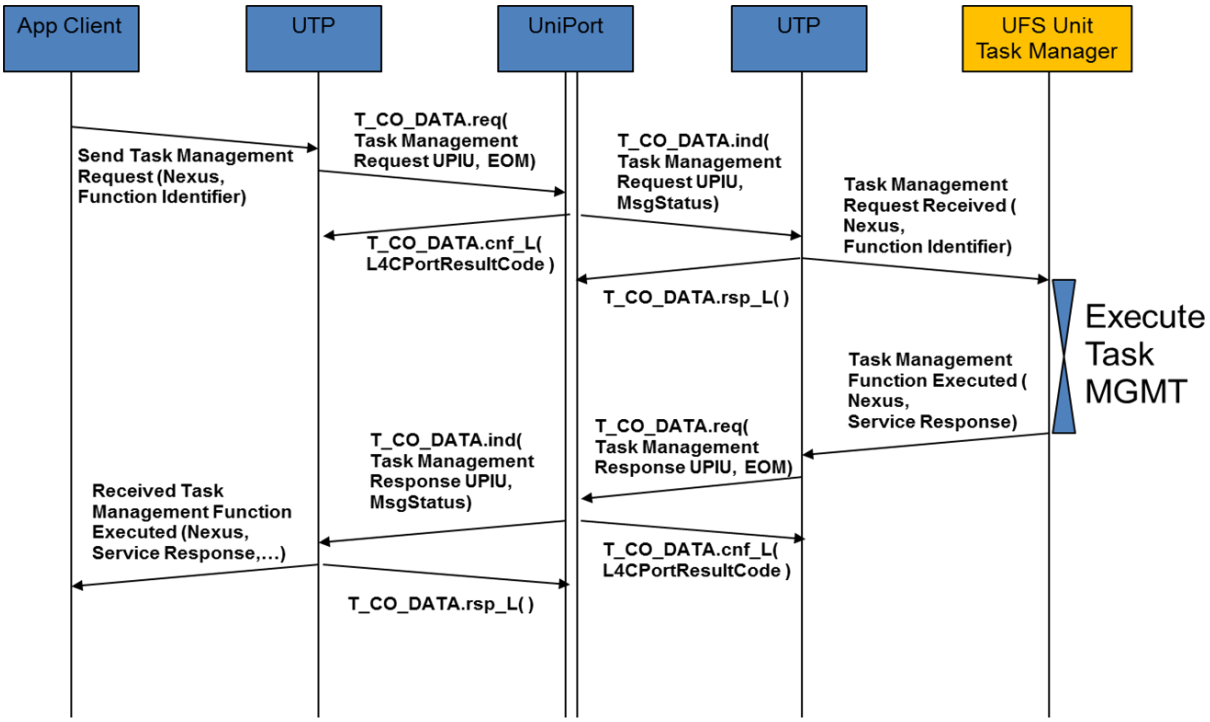


Figure 10-17 — Task Management Function

10.9.8.8 Send Task Management Request SCSI transport protocol service request

An application client uses the Send Task Management Request SCSI transport protocol service request to request that a SCSI initiator port send a task management function.

Send Task Management Request SCSI transport protocol service request:

Send Task Management Request (IN (Nexus, Function Identifier))

Table 10-72 — Send Task Management Request SCSI transport protocol service request

Argument	Implementation
Nexus	I_T nexus, I_T_L nexus, or I_T_L_Q nexus
Function Identifier:	Argument encoding the task management function to be performed.

10.9.8.9 Task Management Request Received SCSI transport protocol service indication

A task router uses the Task Management Request Received SCSI transport protocol service indication to notify a task manager that it has received a task management function.

Task Management Request Received SCSI transport protocol service indication:

Task Management Request Received (IN (Nexus, Function Identifier))

Table 10-73 — Task Management Request Received SCSI transport protocol service indication

Argument	Implementation
Nexus	I_T nexus, I_T_L nexus, or I_T_L_Q nexus
Function Identifier:	Argument encoding the task management function to be performed.

10.9.8.10 Task Management Function Executed SCSI transport protocol service response

A task manager uses the Task Management Function Executed SCSI transport protocol service response to request that a SCSI target port transmit task management function executed information.

Task Management Function Executed SCSI transport protocol service response:

Task Management Function Executed (IN (Nexus, Service Response, [Additional Response Information]))

Table 10-74 — Task Management Function Executed SCSI transport protocol service response

Argument	Implementation
Nexus	I_T nexus, I_T_L nexus, or I_T_L_Q nexus
Service Response	FUNCTION COMPLETE
	FUNCTION SUCCEEDED:
	FUNCTION REJECTED
	INCORRECT LOGICAL UNIT NUMBER
	SERVICE DELIVERY OR TARGET FAILURE
Additional Response Information	The Additional Response Information output argument for the task management procedure call

10.9.8.11 Received Task Management Function Executed SCSI transport protocol service confirmation

A SCSI initiator port uses the Received Task Management Function Executed SCSI transport protocol service confirmation to notify an application client that it has received task management function executed information.

Received Task Management Function Executed SCSI transport protocol service confirmation:

Received Task Management Function Executed (IN (Nexus, Service Response, [Additional Response Information]))

Table 10-75 — Received Task Management Function Executed SCSI transport protocol service confirmation

Argument	Implementation
Nexus	I_T nexus, I_T_L nexus, or I_T_L_Q nexus
Service Response	FUNCTION COMPLETE
	FUNCTION SUCCEEDED:
	FUNCTION REJECTED
	INCORRECT LOGICAL UNIT NUMBER
	SERVICE DELIVERY OR TARGET FAILURE
Additional Response Information	The Additional Response Information output argument for the task management procedure call

10.9.9 Query Function transport protocol services

UFS defines Query Function to get/set UFS-specific device-level registers and parameters (not part of SCSI definition).

10.9.9.1 Send Query Request UFS transport protocol service

An application client uses the Send Query Request UFS transport protocol service request to request that a UFS initiator port send a Query Request function.

Send Query Request UFS transport protocol service request:

Send Query Request(I_T Nexus, Operation: Read|Write, Type: UFS|Vendor, Identifier: Descriptor|Attribute|Flag,Length: n, [Index], [Selector], [WriteData])

Table 10-76 — Send Query Request UFS transport protocol service

Argument	Implementation
Nexus	I_T nexus
Operation	Argument encoding the operation to be performed
Type	Indicates UFS defined or vendor-specific operation
Identifier	Identifier for descriptor type, attribute or flag
Length	Number of bytes to read or write
Index	Index reference for the descriptor, attribute or flag
Selector	Reserved
WriteData	Data to be written in a write query request

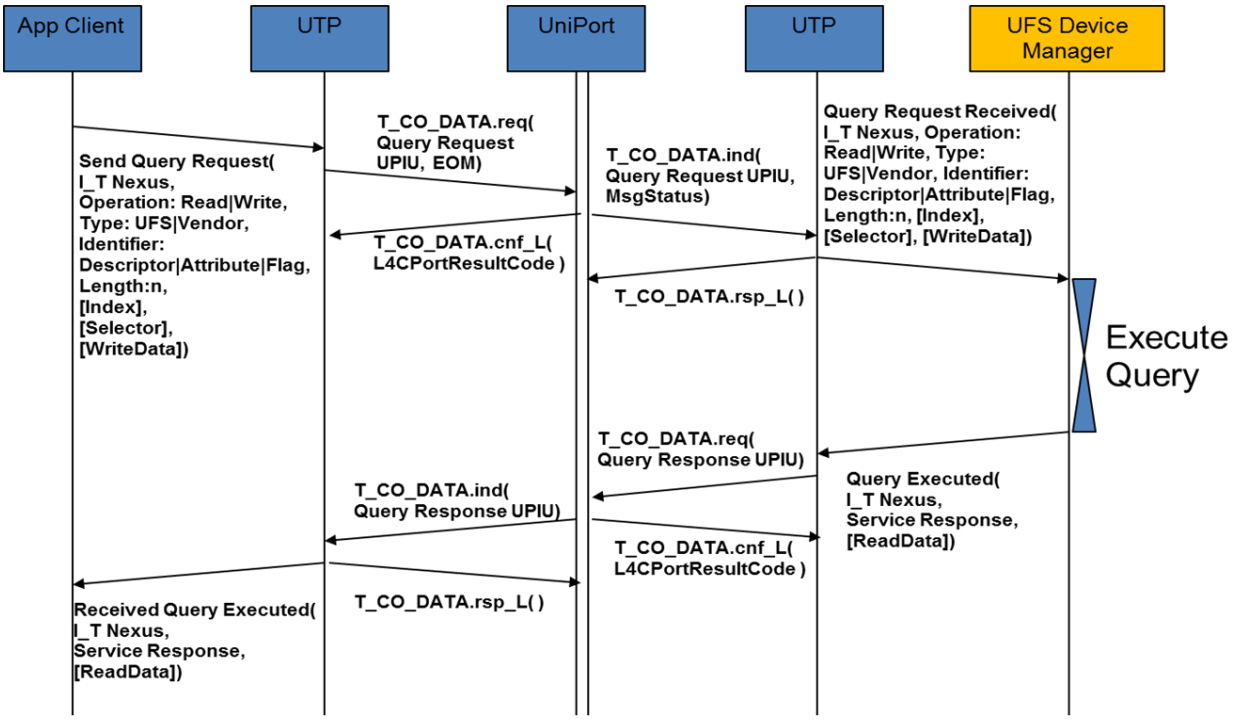


Figure 10-18 — UFS Query Function

10.9.9.2 Query Request Received UFS transport protocol service indication

A UFS target port uses the Query Request Received UFS transport protocol service indication to notify a UFS device manager that it has received a query function.

Query Request Received UFS transport protocol service indication:

Query Request Received(I_T Nexus, Operation: Read|Write, Type: UFS|Vendor, Identifier:Descriptor|Attribute|Flag,Length:n, [Index], [Selector], [WriteData])

Table 10-77 — Query Request Received UFS transport protocol service indication

Argument	Implementation
Nexus	I_T nexus
Operation	Argument encoding the operation to be performed
Type	Indicates UFS defined or vendor-specific operation
Identifier	Identifier for descriptor type, attribute or flag
Length	Number of bytes to read or write
Index	Index reference for the descriptor, attribute or flag
Selector	Reserved
WriteData	Data to be written in a write query request

10.9.9.3 Query Function Executed UFS transport protocol service response

A device manager uses the Query Function Executed UFS transport protocol service response to request that a UFS target port transmit query function executed information.

Query Function Executed UFS transport protocol service response:

Query Executed(I_T Nexus, Service Response,[ReadData])

Table 10-78 — Query Function Executed UFS transport protocol service response

Argument	Implementation
Nexus	I_T nexus
Service Response	FUNCTION SUCCEEDED
	FUNCTION FAILED
ReadData	Data to be returned in a read query request

10.9.9.4 Received Query Function Executed UFS transport protocol service confirmation

A UFS initiator port uses the Received Query Function Executed UFS transport protocol service confirmation to notify an application client that it has received task management function executed information.

Received Query Function Executed UFS transport protocol service confirmation:

Received Query Executed(I_T Nexus, Service Response,[ReadData])

Table 10-79 — Received Query Function Executed UFS transport protocol service confirmation

Argument	Implementation
Nexus	I_T nexus
Service Response	FUNCTION SUCCEEDED
	FUNCTION FAILED
ReadData	Data to be returned in a read query request

11 UFS APPLICATION (UAP) LAYER – SCSI COMMANDS

11.1 Universal Flash Storage Command Layer (UCL) Introduction

This chapter defines the mandatory commands set supported by the UFS device.

Commands may belong to the UFS Native command set or to the UFS SCSI command set.

This version of the standard does not define UFS native commands.These command set may be defined in the future to support specific flash storage or UFS native basic needs.

The UFS SCSI command set (USC) consists of a selection of commands from SCSI Primary Commands – 4 [SPC], and SCSI Block Commands – 3 [SBC]. Both command types share similar command descriptor block (CDB) format.

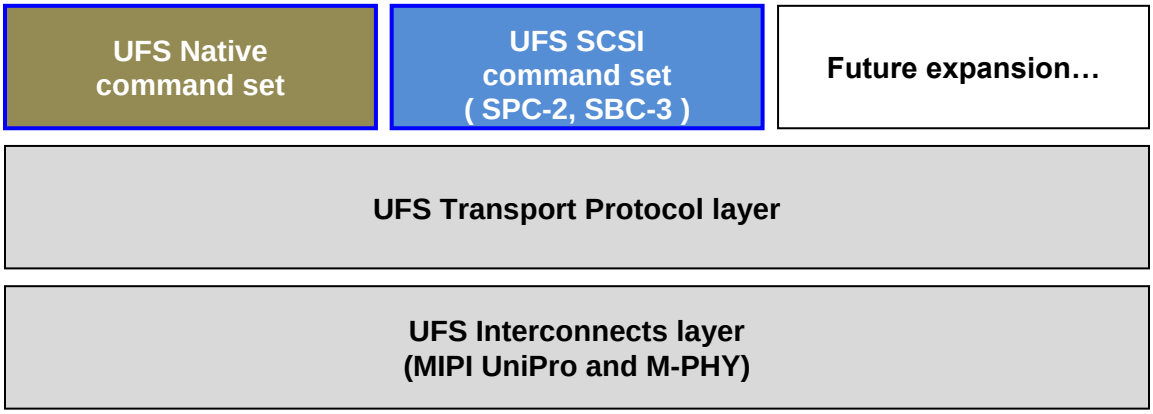


Figure 11-1 — UFS Command Layer

11.1.1 The Command Descriptor Block (CDB)

SCSI commands are communicated by sending the SCSI Command Descriptor Block (CDB) to the device. There are only fixed length CDB format for UFS, unlike SCSI which has additional Variable Length CDB format.

All UFS CDBs shall have an OPERATION CODE field as their first byte and these values shall be defined by each SCSI and UFS. Detail SCSI CDB usages and structure are defined in SPC-4, 4.3, The Command Descriptor Block (CDB).

The General Common CDB fields are defined in SPC-4, 4.3.5, Common CDB fields.

Operation code

The first byte of a SCSI and USC CDB shall contain an operation code identifying the operation being requested by the CDB.

The OPERATION CODE of the CDB contains a GROUP CODE field and COMMAND CODE field. The GROUP CODE field provides eight groups of command codes and the COMMAND CODE provides thirty-two command codes in each group, see [SPC] for further details.

11.2 Universal Flash Storage Native Commands (UNC)

The UFS Native commands are not defined in this version of the standard, they may be defined in future versions if needed.

11.3 Universal Flash Storage SCSI Commands

The Basic Universal Flash Storage (UFS) SCSI commands are compatible with SCSI Primary Commands - 4 [SPC] and SCSI Block Commands - 3 [SBC].

If enabled (bLUEnable = 01h), each logical unit shall support the commands defined as mandatory in Table 11-1.

Table 11-1 — UFS SCSI Command Set

Command name	Opcode	Command Support
FORMAT UNIT	04h	M
INQUIRY	12h	M
MODE SELECT (10)	55h	M
MODE SENSE (10)	5Ah	M
PRE-FETCH (10)	34h	M
PRE-FETCH (16)	90h	O
READ (6)	08h	M
READ (10)	28h	M
READ (16)	88h	O
READ BUFFER	3Ch	O
READ CAPACITY (10)	25h	M
READ CAPACITY (16)	9Eh	M
REPORT LUNS	A0h	M
REQUEST SENSE	03h	M
SECURITY PROTOCOL IN	A2h	M
SECURITY PROTOCOL OUT	B5h	M
SEND DIAGNOSTIC	1Dh	M
START STOP UNIT	1Bh	M
SYNCHRONIZE CACHE (10)	35h	M
SYNCHRONIZE CACHE (16)	91h	O
TEST UNIT READY	00h	M
UNMAP	42H	M
VERIFY (10)	2Fh	M
WRITE (6)	0Ah	M
WRITE (10)	2Ah	M
WRITE (16)	8Ah	O
WRITE BUFFER	3Bh	M
M: mandatory, O: optional, R: RPMB		
NOTE 1 SECURITY PROTOCOL IN command and SECURITY PROTOCOL OUT command are supported by the RPMB well known logical unit.		

11.3.1 General information about SCSI commands in UFS

The remaining part of this section describes the SCSI commands used in UFS devices. A dedicated paragraph for each command provides: CDB table, brief command description, relevant command fields, details about mandatory and optional features, and some other fundamental information.

Fields that are not supported by UFS should be set to zero, and are documented using the notation "= 00h" (e.g., RDPROTECT = 000b). The device may ignore values in fields that are not supported by UFS.

NOTE The values enclosed in parenthesis are defined in SCSI standards and are not UFS specific (e.g., OPERATION CODE (12h)).

In the following some information that apply to all SCSI commands used in UFS.

CONTROL

The CONTROL byte is present in several CDB and it is defined in [SAM]. The CONTROL byte is not used in this standard: the CONTROL byte should be set to zero and it shall be ignored by UFS device. No vendor specific interpretation and Normal ACA are assumed.

Auto contingent allegiance (ACA)

Establishing an ACA condition the application client may request that the device server alter command processing when a command terminates with a CHECK CONDITION status.

UFS device does not support ACA.

11.3.2 INQUIRY Command

The INQUIRY command (see Table 11-2) is a request for information regarding the logical units and UFS target device be sent to the application client. Refer to [SPC] for more details regarding the INQUIRY command.

Table 11-2 — INQUIRY command

Byte \ Bit	7	6	5	4	3	2	1	0
0	OPERATION CODE (12h)							
1	Reserved						Obsolete	EVPD
2	PAGE CODE							
3	(MSB)	ALLOCATION LENGTH*						
4								(LSB)
5	CONTROL = 00h							
* Allocation Length = Number of response bytes to return								

11.3.2.1 VITAL PRODUCT DATA

When EVPD = 1, the device server shall return the vital product data specified by the PAGE CODE field as defined in [SPC]. Support for all vital product data except Mode Page Policy VPD is optional for UFS. Mode Page Policy VPD shall be supported by UFS device to provide information about Mode pages which are applicable to Device level or logical unit level. See [SPC] for data format definition of Mode Page Policy VPD page.

11.3.2.2 STANDARD INQUIRY DATA

When EVPD =0 and Page Code = 0, the Standard INQUIRY DATA is responded to INQUIRY command. The standard INQUIRY data format is shown on Table 11-3, INQUIRY data shall contain at least 36 bytes. Table 11-4 defines the INQUIRY response data for UFS.

The INQUIRY command requests that information regarding the logical unit and SCSI target device be sent to the Application Client.

The INQUIRY command may be used by an Application Client after a hard reset or power on condition to determine information about the device for system configuration. If a INQUIRY command is received with a pending UNIT ATTENTION condition (i.e., before the device server reports CHECK CONDITION status), the device server shall perform the INQUIRY command.

INQUIRY information is returned in standard INQUIRY RESPONSE data structure (described below)

- Client requests number of bytes to return
- First 36 bytes are defined for UFS as standard
- Requesting zero byte is valid
- Requesting 36 bytes will result in the device returning the complete record
- Requesting more bytes than defined will result in truncation to max number device has defined

The Device Server should process the INQUIRY command even when an error occurs that prohibits normal command completion

- When in UNIT ATTENTION
- During other conditions that may affect medium access

The Command CDB shall be sent in a single COMMAND UPIU

Table 11-3 — Standard INQUIRY data format

Bit Byte	7	6	5	4	3	2	1	0
0	PERIPHERAL QUALIFIER			PERIPHERAL DEVICE TYPE				
1	RMB	Reserved						
2	VERSION							
3	Obsolete	Obsolete	NORMACA	HISUP	RESPONSE DATA FORMAT			
4	ADDITIONAL LENGTH (n-4)							
5	SCCS	ACC	TPGS		3PC	Reserved		PROTECT
6	Obsolete	ENCSERV	VS	MULTIP	Obsolete	Obsolete	Obsolete	ADDR16
7	Obsolete	Obsolete	WBUS16	SYNC	Obsolete	Obsolete	CMDQUE	VS
8	(MSB)	VENDOR IDENTIFICATION						
15								(LSB)
16	(MSB)	PRODUCT IDENTIFICATION						
31								(LSB)
32	(MSB)	PRODUCT REVISION LEVEL						
35								(LSB)

11.3.2.3 Inquiry Command Data Response

- Data returned from an INQUIRY command will be transferred to the Application Client in a single DATA IN UPIU
- The Device Server will transfer up to 36 bytes of Response Data in the Data Segment area of a DATA IN UPIU
  - Return 36 bytes if Allocation Length in CDB >= 36
  - Return Allocation Length bytes if Allocation Length in CDB < 36
  - An Allocation Length of zero specifies that no data shall be transferred. This condition shall not be considered as an error, and DATA IN UPIU shall not be generated.
- Data will be returned in the indicated Response Data Format described below
- No DATA IN UPIU will be transferred if an error occurs

11.3.2.4 Inquiry Response Data

Table 11-4 —Standard INQUIRY Response Data

Byte	Bit	Value	Description
0	7:5	000b	PERIPHERAL QUALIFIER: 0
0	4:0	00h/1Eh	PERIPHERAL DEVICE TYPE: 00h: Direct Access Device for logical unit (non well known) 1Eh: Well known logical unit
1	7	0b	RMB: Medium not removable
1	6:0	0000000b	RESERVED
2	7:0	06h	VERSION: Conformance to [SPC]
3	7:4	0000b	N/A
3	3:0	0010b	RESPONSE DATA FORMAT: Type 2
4	7:0	31d	ADDITIONAL LENGTH: 31 bytes
5	7:0	00h	N/A
6	7:0	00h	N/A
7	7:2	000000b	N/A
7	1:1	1b	CMDQUE: Support command management (SAM)
7	0:0	0b	N/A
8:15	7:0	ASCII	VENDOR IDENTIFICATION: Left justified (e.g., “Micron “)
16:31	7:0	ASCII	PRODUCT IDENTIFICATION: Left justified (e.g., “UFS MSD M33-X ”)
32:35	7:0	ASCII	PRODUCT REVISION LEVEL: Left justified (e.g., “1.23”)
NOTE 1 The fields marked with N/A are not applicable for UFS, and their values shall be zero.			

The 4-byte PRODUCT REVISION LEVEL in the Inquiry Response Data shall identify the firmware version of the UFS device and shall be uniquely encoded for any firmware modification implemented by the UFS device vendor.

11.3.2.5 Inquiry Command Status Response

- STATUS response will be sent in a single RESPONSE UPIU
- If the requested data is successfully transferred, the INQUIRY command will terminate with a STATUS response of GOOD
- If the unit is not ready to accept a new command (e.g., still processing previous command) a STATUS response of BUSY will be returned
- When the INQUIRY command fails a STATUS response of CHECK CONDITION will be returned along with an appropriate SENSE KEY, such as
  - ILLEGAL REQUEST (range or CDB errors)
  - HARDWARE ERROR (hardware failure)
- Will not fail due to a pending UNIT ATTENTION condition

11.3.3 MODE SELECT (10) Command

MODE SELECT command provides a means for the application client to specify medium, logical unit, or peripheral device parameters to the device server.

- Parameters are managed by means of parameter pages called mode pages
  - UFS devices shall support the following mode pages
    - CONTROL, CACHING, READ-WRITE ERROR RECOVERY
  - UFS devices may support vendor specific mode pages
  - See 11.4 for further details.
- Writes parameters to one or more mode pages in a list
  - The Application Client can specify a single, multiple or all supported pages in a single command
- Complementary command to the MODE SENSE command

The Command CDB shall be sent in a single COMMAND UPIU

Table 11-5 — MODE SELECT (10) command

Byte \ Bit	7	6	5	4	3	2	1	0
0	OPERATION CODE (55h)							
1	Reserved			PF = 1b	Reserved			SP
2								
3								
4				Reserved				
5								
6								
7	PARAMETER LIST LENGTH							
8								
9	CONTROL = 00h							

3046 **11.3.3.1 Mode Select Command Parameters**

3047 **Table 11-6 — Mode Select Command Parameters**

Byte	Bit	Description
1	4:4	**PF**: PAGE FORMAT. A page format bit set to zero specifies that all parameters after the block descriptors are vendor specific. A PF bit set to one specifies that the MODE SELECT parameters following the header and block descriptor(s) are structured as pages of related parameters as defined in the SCSI standard.
1	0:0	**SP**: SAVE PAGES. A save pages (SP) bit set to zero specifies that the device server shall perform the specified MODE SELECT operation, and shall not save any mode pages. If the logical unit implements no distinction between current and saved mode pages and the SP bit is set to zero, the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST. An SP bit set to one specifies that the device server shall perform the specified MODE SELECT operation, and shall save to a nonvolatile vendor specific location all the saveable mode pages including any sent in the Data-Out Buffer. Mode pages that are saved are specified by the parameter saveable (PS) bit that is returned in the first byte of each mode page when read via the MODE SENSE command. If the PS bit is set to one in the MODE SENSE data, then the mode page shall be saveable when issuing a MODE SELECT command with the SP bit set to one. If the logical unit does not implement saved pages and the SP bit is set to one, the command shall terminate with a CHECK CONDITION status and a sense key set to ILLEGAL REQUEST.
7:8	7:0	**PARAMETER LIST LENGTH**: Specifies length in bytes of the mode parameter list that the Application Client will transfer to the Device Server. A PARAMETER LIST LENGTH of zero specifies that no data shall be transferred, this shall not be considered an error and in this case the device shall not send RTT UPIU.

3048 **11.3.3.2 Mode Select Command Data Transfer**

3049 The Device Server requests to transfer the mode parameter list from the Application Client data-out
3050 buffer by issuing one or more READY TO TRANSFER UPIU's (RTT).

3051 The mode parameter list is delivered in one or more segments sending DATA OUT UPIU packets, as
3052 indicated in the RTT requests,.

3053 Zero or an incomplete number of segments may be requested, if an error occurs before the entire data
3054 transfer is complete.

3055 Mode parameters are changed as specified in the received mode parameter list if the command completes
3056 succefully.

3057 See 11.4.1.2, Mode parameter list format, for details about the mode parameter list.

3058

11.3.3.3 Mode Select Command Status Response

- STATUS response will be sent in a single RESPONSE UPIU
- If the requested data is successfully transferred and written, the MODE SELECT command will terminate with a STATUS response of GOOD
- If the unit is not ready to accept a new command (e.g., still processing previous command) a STATUS response of BUSY will be returned
- Failure can occur for numerous reasons. When the MODE SELECT command fails a STATUS response of CHECK CONDITION will be returned along with an appropriate SENSE KEY, such as
    - ILLEGAL REQUEST (CDB or parameter errors)
    - MEDIUM ERROR (medium failure, ECC, etc.)
    - HARDWARE ERROR (hardware failure)
    - UNIT ATTENTION (reset, power-on, etc.)

11.3.4 MODE SENSE (10) Command

The MODE SENSE command provides a means for a Device Server to report parameters to an Application Client

- Parameters are managed by means of parameter pages called Mode Pages
    - UFS devices shall support the following mode pages
        - CONTROL, READ-WRITE ERROR RECOVERY, CACHING
    - UFS devices may support vendor specific mode pages
    - See 11.4 for further details
- Reads parameter pages in a list
    - The Application Client may request any one or all of the supported pages from the Device Server
    - If all pages requested, they will be returned in ascending page order
- Mode Sense returns DEVICE-SPECIFIC PARAMETER in header
    - See paragraph 11.4.1.3, Mode Parameter Header for details.
- Complementary command to the MODE SELECT command

The Command CDB shall be sent in a single COMMAND UPIU

11.3.4 MODE SENSE (10) Command (cont'd)

Table 11-7 — MODE SENSE (10) command

Byte \ Bit	7	6	5	4	3	2	1	0
0	OPERATION CODE (5Ah)							
1	Reserved =000b			LLBAA =0b	DBD = 1b	Reserved = 000b		
2	PC		PAGE CODE					
3	SUBPAGE CODE							
4	Reserved							
5								
6								
7	(MSB)		ALLOCATION LENGTH					
8								(LSB)
9	CONTROL = 00h							

11.3.4.1 Mode Sense Command Parameters

Table 11-8 — Mode Sense Command Parameters

Byte	Bit	Description
2	7:6	**PC**: The page control (PC) field specifies the type of mode parameter values to be returned in the mode pages. See Table 11-9 for its definition.
2	5:0	**PAGE CODE**: Specifies which mode page to return. The Page and Subpage code specify the page to return.
3	7:0	**SUBPAGE CODE**: Specifies which subpage mode page to return
7:8	7:0	**ALLOCATION LENGTH**: Specifies the maximum number of bytes the Device Server will transfer to the Application Client

11.3.4.2 Page Control Function

Table 11-9 — Page Control Function

Page Control (PC)	Description	Use
00b	Current Values	Return the current operational mode page parameter values.
01b	Changeable Values	Return a bit mask denoting values that are changeable. Changeable bits in the mode parameters will have a value of '1', non-changeable will have a value of '0'.
10b	Default Values	Return the default values of the mode parameters.
11b	Saved Values	Return the saved values of the mode parameters.

11.3.4.3 Mode Sense Command Data Transfer

The Device Server will transfer up to Allocation Length number of data bytes of Mode Parameter Data to the Application Client.

- Less than Allocation Length will be transferred if Device Server contains less bytes

The Device Server will transfer the data as indicated by the Mode Parameter Layout

- Header (8 bytes)
- Block Descriptor (0 bytes for UFS)
- Page Data (N bytes, dependent up page requested)

Data will be transferred from the Device Server to the Application Client via a series of DATA IN UPIU's

- The data transferred from the Device Server will be contained within the Data Segment of the DATA IN UPIU

Zero or an incomplete number of DATA IN UPIU's will be transferred if an error occurs before the entire data transfer is complete.

11.3.4.4 Mode Sense Command Status Response

- STATUS response will be sent in a single RESPONSE UPIU
- If the requested data is successfully transferred and written, the MODE SENSE command will terminate with a STATUS response of GOOD
- If the unit is not ready to accept a new command (e.g., still processing previous command) a STATUS response of BUSY will be returned
- Failure occurs when a requesting a page or subpage that is not supported. A STATUS response of CHECK CONDITION will be returned with a SENSE KEY indicating ILLEGAL REQUEST
- Failure can occur for numerous reasons. When the MODE SENSE command fails a STATUS response of CHECK CONDITION will be returned along with an appropriate SENSE KEY, such as
    - ILLEGAL REQUEST (CDB errors)
    - HARDWARE ERROR (hardware failure)
    - UNIT ATTENTION (reset, power-on, etc.)

11.3.5 READ (6) Command

The READ (6) command (see Table 11-10) requests that the Device Server read from the medium the specified number of logical block(s) and transfer them to the Application Client.

The Command CDB shall be sent in a single COMMAND UPIU.

Table 11-10 — READ (6) command

Bit / Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (08h)							
1	Reserved			(MSB)				
2	LOGICAL BLOCK ADDRESS							
3								(LSB)
4	TRANSFER LENGTH							
5	CONTROL = 00h							

11.3.5.1 Read (6) Command Parameters

- LOGICAL BLOCK ADDRESS: Address of first block
- TRANSFER LENGTH: Number of contiguous logical blocks of data that shall be read and transferred. A transfer length of zero specifies that 256 logical blocks shall be read. Any other value specifies the number of logical blocks that shall be read.

11.3.5.2 Read (6) Command Data Transfer

- The Device Server will read the specified logical block(s) from the medium and transfer them to the Application Client in a series of DATA IN UPIU's
- The data segment of each DATA IN UPIU shall contain an integer number of logical blocks
- Zero or an incomplete number of DATA IN UPIU's could be transferred if a read error occurs before the entire data transfer is complete

11.3.5.3 Read (6) Command Status Response

- Status response will be sent in a single RESPONSE UPIU
- If all requested data is successfully read and transferred, the READ command will terminate with a STATUS response of GOOD
- If the unit is not ready to accept a new command (e.g., still processing previous command) a STATUS response of BUSY will be returned
- Failure can occur for numerous reasons. When the READ command fails a STATUS response of CHECK CONDITION will be returned along with an appropriate SENSE KEY, such as
  - ILLEGAL REQUEST (range or CDB errors)
  - MEDIUM ERROR (medium failure, ECC, etc.)
  - HARDWARE ERROR (hardware failure)
  - UNIT ATTENTION (reset, power-on, etc.)
  - etc.

11.3.6 READ (10) Command

The READ (10) command (see Table 11-11) requests that the Device Server read from the medium the specified number of logical block(s) and transfer them to the Application Client.

The Command CDB shall be sent in a single COMMAND UPIU

Table 11-11 — READ (10) command

Byte \ Bit	7	6	5	4	3	2	1	0
0	OPERATION CODE (28h)							
1	RDPROTECT = 000b			DPO	FUA	Reserved	FUA_NV = 0b	Obsolete
2	(MSB)							
3				LOGICAL BLOCK ADDRESS				
4								
5								(LSB)
6	Reserved			GROUP NUMBER				
7	(MSB)							
8				TRANSFER LENGTH				(LSB)
9	CONTROL = 00h							

- The RDPROTECT field is set to zero for UFS.

11.3.6.1 Read (10) Command Parameters

- DPO = Disable Page Out
  "0" = specifies that the retention priority shall be determined by the RETENTION PRIORITY fields in the Caching mode page
  "1" = specifies that the device server shall assign the logical blocks accessed by this command the lowest retention priority for being fetched into or retained by the cache. A DPO bit set to one overrides any retention priority specified in the Caching mode page.
- FUA: Force Unit Access
  '0' = The Device Server may read the logical blocks from the cache and/or the medium.
  '1' = The Device Server shall read the logical blocks from the medium. If a cache contains a more recent version of a logical block, then the device server shall write the logical block to the medium before reading it.
- FUA_NV is defined per SBC. Since non-volatile cache support is not currently defined in this standard, the FUA_NV parameter value in the CDB is ignored by the UFS Device Server.
- LOGICAL BLOCK ADDRESS: Address of first block
- TRANSFER LENGTH: Number of contiguous logical blocks of data that shall be read and transferred. A transfer length of zero specifies that no logical blocks will be read. This condition shall not be considered an error.
- GROUP NUMBER: Notifies the Target device that the data linked to a ContextID:

GROUP NUMBER Value	Function
00000b	Default, no Context ID is associated with the read operation.
00001b to 01111b (0XXXXb)	Context ID. (XXXX from 0001b to 1111b - Context ID value)
10000b to 11111b	Reserved

  In case the GROUP NUMBER is set to a reserved value, the operation shall fail and a status response of CHECK CONDITION will be returned along with the sense key set to ILLEGAL REQUEST.

11.3.6.2 Read (10) Command Data Transfer

- The Device Server will read the specified logical block(s) from the medium and transfer them to the Application Client in a series of DATA IN UPIU's
- The data segment of each DATA IN UPIU shall contain an integer number of logical blocks
- Zero or an incomplete number of DATA IN UPIU's could be transferred if a read error occurs before the entire data transfer is complete.

11.3.6.3 Read (10) Command Status Response

- Status response will be sent in a single RESPONSE UPIU
- If all requested data is successfully read and transferred, the READ command will terminate with a STATUS response of GOOD
- If the unit is not ready to accept a new command (e.g., still processing previous command) a STATUS response of BUSY will be returned
- Failure can occur for numerous reasons. When the READ command fails a STATUS response of CHECK CONDITION will be returned along with an appropriate SENSE KEY, such as
  - ILLEGAL REQUEST (range or CDB errors)
  - MEDIUM ERROR (medium failure, ECC, etc.)
  - HARDWARE ERROR (hardware failure)
  - UNIT ATTENTION (reset, power-on, etc.)
  - etc.

11.3.7 READ (16) Command

The READ (16) command (see Table 11-12) requests that the Device Server read from the medium the specified number of logical block(s) and transfer them to the Application Client

The Command CDB shall be sent in a single COMMAND UPIU

Table 11-12 — READ (16) command

Bit / Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (88h)							
1	RDPROTECT = 000b			DPO	FUA	Reserved	FUA_NV = 0b	Reserved
2	(MSB)							
….	LOGICAL BLOCK ADDRESS							
9								(LSB)
10	(MSB)							
….	TRANSFER LENGTH							
13								(LSB)
14	Reserved	Reserved	GROUP NUMBER					
15	CONTROL = 00h							

- The RDPROTECT field is set to zero for UFS.

11.3.7.1 Read (16) Command Parameters

- DPO = Disable Page Out
  "0" = specifies that the retention priority shall be determined by the RETENTION PRIORITY fields in the Caching mode page
  "1" = specifies that the device server shall assign the logical blocks accessed by this command the lowest retention priority for being fetched into or retained by the cache. A DPO bit set to one overrides any retention priority specified in the Caching mode page.
- **FUA**: Force Unit Access
  '0' = The Device Server may read the logical blocks from the cache and/or the medium. '1' = The Device Server shall read the logical blocks from the medium. If a cache contains a more recent version of a logical block, then the device server shall write the logical block to the medium before reading it.
- FUA_NV is defined per SBC. Since non-volatile cache support is not currently defined in this standard, the FUA_NV parameter value in the CDB is ignored by the UFS Device Server.
- **LOGICAL BLOCK ADDRESS**: Address of first block
- **TRANSFER LENGTH**: Number of contiguous logical blocks of data that shall be read and transferred. A transfer length of zero specifies that no logical blocks will be read. This condition shall not be considered an error.
- GROUP NUMBER: See Read (10) Command.

11.3.7.2 Read (16) Command Data Transfer

- The Device Server will read the specified logical block(s) from the medium and transfer them to the Application Client in a series of DATA IN UPIU's
- The data segment of each DATA IN UPIU shall contain an integer number of logical blocks
- Zero or an incomplete number of DATA IN UPIU's could be transferred if a read error occurs before the entire data transfer is complete

11.3.7.3 Read (16) Command Status Response

- Status response will be sent in a single RESPONSE UPIU
- If all requested data is successfully read and transferred, the READ command will terminate with a STATUS response of GOOD
- If the unit is not ready to accept a new command (e.g., still processing previous command) a STATUS response of BUSY will be returned
- Failure can occur for numerous reasons. When the READ command fails a STATUS response of CHECK CONDITION will be returned along with an appropriate SENSE KEY, such as
  - ILLEGAL REQUEST (range or CDB errors)
  - MEDIUM ERROR (medium failure, ECC, etc.)
  - HARDWARE ERROR (hardware failure)
  - UNIT ATTENTION (reset, power-on, etc.)
  - etc.

11.3.8 READ CAPACITY (10) Command

The READ CAPACITY (10) command provides a means for the application client to discover the logical unit capacity. Refer to [SBC] for more details regarding the READ CAPACITY (10) command.

- The READ CAPACITY (10) command requests that the device server transfer 8 bytes of parameter data describing the capacity and medium format of the direct-access block device

The Command CDB shall be sent in a single COMMAND UPIU

Table 11-13 — READ CAPACITY (10) command

Byte \ Bit	7	6	5	4	3	2	1	0
0	OPERATION CODE (25h)							
1	Reserved							Obsolete = 0b
2	(MSB)							
5	LOGICAL BLOCK ADDRESS							(LSB)
6	Reserved							
7	Reserved							
8	Reserved							PMI = 0b
9	CONTROL = 00h							

- PMI = 0 for UFS
- Logical Block Address = 0 for UFS

11.3.8.1 Read Capacity (10) Data Response

- Data returned from a READ CAPACITY (10) command will be transferred to the Application Client in a single DATA IN UPIU
- The Device Server will transfer 8 bytes of Capacity Data in the Data Segment area of a DATA IN UPIU
- Data will be returned in the indicated Read Capacity Parameter format described below
- No DATA IN UPIU will be transferred if an error occurs

11.3.8.2 Read Capacity (10) Parameter Data

Table 11-14 — Read Capacity (10) Parameter Data

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)							
1				RETURNED LOGICAL BLOCK ADDRESS				
2								
3								(LSB)
4	(MSB)							
5				LOGICAL BLOCK LENGTH IN BYTES				
6								
7								(LSB)

- RETURNED LOGICAL BLOCK ADDRESS: last addressable block on medium under control of logical unit (LU)
  - If the number of logical blocks exceeds the maximum value that is able to be specified in the RETURNED LOGICAL BLOCK ADDRESS field, then the device server shall set the RETURNED LOGICAL BLOCK ADDRESS field to FFFF FFFFh. The application client should then issue a READ CAPACITY (16) command to request that the device server transfer the READ CAPACITY (16) parameter data to the data-in buffer.
- LOGICAL BLOCK LENGTH IN BYTES: size of block in bytes
  - For UFS minimum size shall be 4096 bytes

11.3.8.3 Read Capacity (10) Status Response

- STATUS response will be sent in a single RESPONSE UPIU
- If the requested data is successfully transferred, the READ CAPACITY command will terminate with a STATUS response of GOOD
- If the unit is not ready to accept a new command (e.g., still processing previous command) a STATUS response of BUSY will be returned
- Failure can occur for a number of reasons. When the READ CAPACITY command fails a STATUS response of CHECK CONDITION will be returned along with an appropriate SENSE KEY, such as
  - ILLEGAL REQUEST (range or CDB errors)
  - HARDWARE ERROR (hardware failure)
  - UNIT ATTENTION (reset, power-on, etc.)
  - etc.

11.3.9 READ CAPACITY (16) Command

The READ CAPACITY (16) command provides a means for the application client to discover the logical unit capacity. Refer to [SBC] for more details regarding the READ CAPACITY (16) command.

- The READ CAPACITY (16) command requests that the device server transfer 32 bytes of parameter data describing the capacity and medium format of the direct-access block device

The Command CDB shall be sent in a single COMMAND UPIU.

Table 11-15 — READ CAPACITY (16) command

Bit / Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (9Eh)							
1	Reserved			SERVICE ACTION (10h)				
2	(MSB)							
….	LOGICAL BLOCK ADDRESS							
9								(LSB)
10	(MSB)							
….	ALLOCATION LENGTH							
13								(LSB)
14	Reserved							PMI = 0b
15	CONTROL = 00h							

- PMI = 0 for UFS
- Logical Block Address = 0 for UFS
- Allocation Length = the maximum number of bytes that the application client has allocated for the returned parameter data.

11.3.9.1 Read Capacity (16) Data Response

- Data returned from a READ CAPACITY (16) command will be transferred to the Application Client in a single DATA IN UPIU
- The Device Server will transfer 32 bytes of Capacity Data in the Data Segment area of a DATA IN UPIU
- Data will be returned in the indicated Read Capacity Parameter format described below
- No DATA IN UPIU will be transferred if an error occurs

11.3.9.2 Read Capacity (16) Parameter Data

Table 11-16 — Read Capacity (16) Parameter Data

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)							
….		RETURNED LOGICAL BLOCK ADDRESS						
7								(LSB)
8	(MSB)							
….		LOGICAL BLOCK LENGTH IN BYTES						
11								(LSB)
12	Reserved					P_TYPE		PROT_EN
13	P_I_EXPONENT					LOGICAL BLOCKS PER PHYSICAL BLOCK EXPONENT		
14	TPE	TPRZ	(MSB)	LOWEST ALIGNED LOGICAL BLOCK ADDRESS				
15								(LSB)
16								
….		Reserved						
31								

- RETURNED LOGICAL BLOCK ADDRESS: last addressable block on medium under control of logical unit (LU)
- LOGICAL BLOCK LENGTH IN BYTES: size of block in bytes
  - For UFS minimum size shall be 4096 bytes
- P_TYPE, PROT_EN are set to ‘0’ for UFS
- The P_I_EXPONENT field can be ignored for PROT_EN = ‘0’
- Logical Blocks per Physical Block Exponent (vendor-specific information)
  - 0: one or more physical blocks per logical block
  - n>0: 2^n logical blocks per physical block
- TPE is set to ‘0’ if bProvisioningType parameter in UFS Configuration Descriptor is set to 00h. TPE is set to ‘1’ if bProvisioningType is set to 02h or 03h.
- TPRZ value is set by bProvisioningType parameter in UFS Configuration Descriptor. If the thin provisioning read zeros (TPRZ) bit is set to one, then, for an unmapped LBA specified by a read operation, the device server shall send user data with all bits set to zero to the data in buffer. If the TPRZ bit is set to zero, then, for an unmapped LBA specified by a read operation, the device server shall send user data with all bits set to any value to the data in buffer. Lowest Aligned Logical Block Address indicates the LBA of the first logical block that is located at the beginning of a physical block (vendor-specific information)

11.3.9.3 Read Capacity (16) Status Response

- STATUS response will be sent in a single RESPONSE UPIU
- If the requested data is successfully transferred, the READ CAPACITY command will terminate with a STATUS response of GOOD
- If the unit is not ready to accept a new command (e.g., still processing previous command) a STATUS response of BUSY will be returned
- Failure can occur for a number of reasons. When the READ CAPACITY command fails a STATUS response of CHECK CONDITION will be returned along with an appropriate SENSE KEY, such as
    - ILLEGAL REQUEST (range or CDB errors)
    - HARDWARE ERROR (hardware failure)
    - UNIT ATTENTION (reset, power-on, etc.)
    - etc.

11.3.10 START STOP UNIT Command

The START STOP UNIT command requests that the device server change the power condition of the logical unit or load or eject the medium.

- Enable or disable the direct-access block device for medium access operations by controlling power

The Command CDB shall be sent in a single COMMAND UPIU.

Table 11-17 — START STOP UNIT command

Bit / Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (1Bh)							
1	Reserved							IMMED
2	Reserved							
3	Reserved				POWER CONDITION MODIFIER = 0h			
4	POWER CONDITIONS				Reserved	NO_FLUSH	LOEJ = 0b	START
5	CONTROL = 00h							

IMMED = 0 : Return STATUS (RESPONSE UPIU) after operation is completed

IMMED = 1 : Return STATUS after CDB is decoded

11.3.10.1 START STOP UNIT Parameters

Power Condition Modifier shall be set to ‘0’ (reserved) in UFS spec.

Use of the other parameters is defined in the Power Mode section.

11.3.10.2 Start Stop Unit Data Response

- The START STOP UNIT command does not have a data phase
- No DATA IN or DATA OUT UPIU’s are transferred

11.3.10.3 Start Stop Unit Status Response

- STATUS response will be sent in a single RESPONSE UPIU
- If IMMED = 1 in the CDB, the STATUS response will be sent to the Application Client before the device operations have been completed
  - Usually used for shutting down
- If the requested operation is successful, the START STOP UNIT command will terminate with a STATUS response of GOOD
- If the unit is not ready to accept a new command (e.g., still processing previous command) a STATUS response of BUSY will be returned
- Failure can occur for few reasons. When the START STOP UNIT command fails a STATUS response of CHECK CONDITION will be returned along with an appropriate SENSE KEY, such as
  - ILLEGAL REQUEST (range or CDB errors)
  - HARDWARE ERROR (hardware failure)
  - UNIT ATTENTION

11.3.11 TEST UNIT READY Command

The TEST UNIT READY command provides a means to check if the logical unit is ready. This is not a request for a self-test.

- If the logical unit is able to accept an appropriate medium-access command without returning CHECK CONDITION status, this command shall return a GOOD status.
- If the logical unit is unable to become operational or is in a state such that an Application Client action (e.g., START UNIT command) is required to make the logical unit ready, the command shall be terminated with CHECK CONDITION status, with the sense key set to NOT READY (02h).

The Command CDB shall be sent in a single COMMAND UPIU

Table 11-18 — TEST UNIT READY command

Bit / Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (00h)							
1	Reserved							
2	Reserved							
3	Reserved							
4	Reserved							
5	CONTROL = 00h							

11.3.11.1 Test Unit Ready Data Response

- The TEST UNIT READY command does not have a data response
- No DATA IN or DATA OUT UPIU's are transferred

11.3.11.2 Test Unit Ready Status Response

- STATUS response will be sent in a single RESPONSE UPIU.
- If the command succeeds without error, the TEST UNIT READY command will terminate with a STATUS response of GOOD
- If the unit is not ready to accept a new command (e.g., still processing previous command) a STATUS response of BUSY will be returned
- Failure can occur for numerous reasons. When the TEST UNIT READY command fails a STATUS response of CHECK CONDITION will be returned along with an appropriate SENSE KEY, such as
    - ILLEGAL REQUEST (CDB errors)
    - HARDWARE ERROR (hardware failure)
    - UNIT ATTENTION (reset, power-on, etc.)
    - etc.

11.3.12 REPORT LUNS Command

The REPORT LUNS command requests that the peripheral device logical unit inventory be sent to the Application Client.

- The logical unit inventory is a list that shall include the logical unit numbers of all logical units accessible to a UFS Application Client
- If a REPORT LUNS command is received with a pending unit attention condition (i.e., before the device server reports CHECK CONDITION status), the device server shall perform the REPORT LUNS command

The Command CDB shall be sent in a single COMMAND UPIU

Table 11-19 — REPORT LUNS command

Bit / Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (A0h)							
1	Reserved							
2	SELECT REPORT							
3	(MSB)			Reserved				
5								(LSB)
6	(MSB)			ALLOCATION LENGTH				
9								(LSB)
10	Reserved							
11	CONTROL = 00h							

11.3.12.1 Report LUNS Command Parameters

Table 11-20 — Report LUNS Command Parameters

Byte	Bit	Description
2	7:0	**SELECT REPORT**: Specifies the type of logical unit addresses that shall be reported. The report types available are listed in the table below. UFS will support all report types.
6:9	7:0	**ALLOCATION LENGTH**: Specifies the maximum number of bytes of buffer space that the Application Client has allocated for data reception.

11.3.12.2 Report LUNS Command Select Report Field Values

Table 11-21 — SELECT REPORT field

CODE	DESCRIPTION
00h	The list shall contain all accessible logical units using the single level LUN structure using the peripheral device addressing method, if any. If there are no logical units, the LUN LIST LENGTH field shall be zero.
01h	The list shall contain all accessible well known logical units, if any using the well known logical unit extended addressing format. If there are no well known logical units, the LUN LIST LENGTH field shall be zero.
02h	The list shall contain all accessible logical units using their respective addressing format.
03h-FFh	Reserved

NOTE The well known logical units are not included in the list when the SELECT REPORT field is set to zero.

11.3.12.3 Report LUNS Data Response

- Data returned from a REPORT LUNS command will be transferred to the Application Client in a one or more DATA IN UPIU's
- Most likely one DATA IN UPIU
- The Device Server will transfer less than or equal to Allocation Length data bytes to the Application Client.
- Less if Device Server has less total data than requested
- Data will be returned in the indicated Parameter Data Format described below
- Each reportable logical unit will produce 8 bytes of data in a format described below

11.3.12.4 Report LUNS Parameter Data Format

Table 11-22 — Report LUNS Parameter Data Format

Byte	Description
0:3	**LUN LIST LENGTH**: Length = N – 7. This field shall contain the length in bytes of the LUN list that is available to be transferred. The LUN list length is the number of logical unit numbers in the logical unit inventory multiplied by eight.
4:7	**RESERVED**: 00000000h
8:15	**FIRST LUN RECORD**: 8 byte record that contains the first LUN
…	… next LUN record …
N-7:N	**LAST LUN RECORD**: 8 byte record that contains the last LUN

- Total List Length = LUN LIST LENGTH + 8 (i.e., 8*number of LUN's + 8)
- Each LUN record is 8 bytes in length
- UFS uses two formats:
  - The single level LUN structure using peripheral device addressing method
  - The well known logical unit extended addressing format

11.3.12.5 Report LUNS LUN Addressing Formats

Table 11-23 — Single level LUN structure using peripheral device addressing method

Peripheral Device Addressing Method								
Byte	7	6	5	4	3	2	1	0
0	00b		000000b					
1	LUN							
2	NULL (0000h)							
3								
4	NULL (0000h)							
5								
6	NULL (0000h)							
7								

- Format used for standard Logical Unit addressing
- LUN = Logical Unit Number
  - For UFS: 00h <= LUN <= 7Fh

NOTE The expected value is the SCSI LUN and not the LUN field in UPIU.

11.3.12.5 Report LUNS LUN Addressing Formats (cont'd)

Table 11-24 — Well Known Logical Unit Extended Addressing Format

Well Known Logical Unit Extended Addressing Format								
Byte	7	6	5	4	3	2	1	0
0	11b		00b		0001b			
1	W-LUN							
2	NULL (0000h)							
3								
4	NULL (0000h)							
5								
6	NULL (0000h)							
7								

- Format used for well known logical unit addressing
- W-LUN = Well Known Logical Unit Number
  - For UFS: 00h <= W-LUN <= 7Fh

    NOTE The expected value is the SCSI LUN and not the LUN field in UPIU.

11.3.12.6 Report LUNS Status Response

- Status response will be sent in a single RESPONSE UPIU
- If all requested data is successfully read and transferred, the REPORT LUNS command will terminate with a STATUS response of GOOD
- The REPORT LUNS command will succeed when a pending UNIT ATTENTION condition exists
- Failure can occur for very few reasons, mainly for illegal values in the CDB. When the REPORT LUNS command fails a STATUS response of CHECK CONDITION will be returned along with an appropriate SENSE KEY, such as
  - ILLEGAL REQUEST (CDB errors)

11.3.12.7 UFS LUN Format

Table 11-25 — Format of LUN field in UPIU

7	6	5	4	3	2	1	0
WLUN_ID	UNIT_NUMBER_ID						

The UFS 8-bit LUN field in UPIU supports two types of LUN addressing:

- If WLUN_ID bit = ‘0’ then the UNIT_NUMBER_ID field addresses a standard logical unit (LUN)
- If WLUN_ID bit = ‘1’ then the UNIT_NUMBER_ID field addresses a well known logical unit (W-LUN)

Up to 128 LUN’s and up to 128 W-LUN’s

- 0 <= UNIT_NUMBER_ID <= 127
- 0 <= UNIT_NUMBER_ID <= 127

The following table defines the logical unit number for UFS well known logical units (WLUN_ID bit set to ‘1’)

Table 11-26 — Well known logical unit numbers

Well known logical unit	WLUN_ID	UNIT_NUMBER_ID	LUN Field in UPIU
REPORT LUNS	1b	01h	81h
UFS Device	1b	50h	D0h
RPMB	1b	44h	C4h
BOOT	1b	30h	B0h

11.3.13 VERIFY (10) Command

The VERIFY command requests that the UFS device verify that the specified logical block(s) and range on the medium can be accessed.

- Logical units that contain cache shall write referenced cached logical blocks to the medium for the logical unit before verification

The Command CDB shall be sent in a single COMMAND UPIU.

Table 11-27 — VERIFY (10) command

Byte \ Bit	7	6	5	4	3	2	1	0
0	OPERATION CODE (2Fh)							
1	VRPROTECT = 000b			DPO = 0b	Reserved		BYTCHK = 0b	Obsolete = 0b
2	(MSB)							
3				LOGICAL BLOCK ADDRESS				
4								
5								(LSB)
6	Reserved			GROUP NUMBER = 00000b				
7	(MSB)							
8				VERIFICATION LENGTH				(LSB)
9	CONTROL = 00h							

UFS device is required to support only the value zero for the byte check (BYTCHK) bit. Therefore the BYTCHK bit should be set to zero, and the device shall perform a medium verification with no data comparison and not transfer any data from the data-out buffer for any mapped LBA specified by the command.

11.3.13.1 Verify Command Parameters

Table 11-28 — Verify Command Parameters

Byte	Bit	Description
2:5	7:0	**LOGICAL BLOCK ADDRESS**: Address of first block
7:8	7:0	**VERIFICATION LENGTH**: Number of contiguous logical blocks of data that shall be verified, starting with the logical block specified by the LOGICAL BLOCK ADDRESS field. A transfer length of zero specifies that no logical blocks will be verified. This condition shall not be considered an error.

11.3.13.2 Verify Command Data Transfer

The VERIFY command does not have a data response

- No DATA IN or DATA OUT UPIU's are transferred

11.3.13.3 Verify Command Status Response

- STATUS response will be sent in a single RESPONSE UPIU
- If all requested data is successfully verified against the medium, the VERIFY command will terminate with a STATUS response of GOOD
- If the unit is not ready to accept a new command (e.g., still processing previous command) a STATUS response of BUSY will be returned
- Other failures can occur for numerous reasons. When the WRITE command fails a STATUS response of CHECK CONDITION will be returned along with an appropriate SENSE KEY, such as:
    - ILLEGAL REQUEST (range or CDB errors)
    - MEDIUM ERROR (medium failure, ECC, etc.)
    - HARDWARE ERROR (hardware failure)
    - UNIT ATTENTION (reset, power-on, etc.)
    - etc.

11.3.14 WRITE (6) Command

The WRITE (6) UFS command requests that the Device Server transfer the specified number of logical blocks(s) from the Application Client and write them to the medium.

The Command CDB shall be sent in a single COMMAND UPIU.

Table 11-29 — WRITE (6) command

Byte \ Bit	7	6	5	4	3	2	1	0
0	OPERATION CODE (0Ah)							
1	Reserved			(MSB)				
2	LOGICAL BLOCK ADDRESS							
3								(LSB)
4	TRANSFER LENGTH							
5	CONTROL = 00h							

11.3.14.1 Write (6) Command Parameters

- LOGICAL BLOCK ADDRESS: Address of first block
- TRANSFER LENGTH: Number of contiguous logical blocks of data that shall be transferred and written. A transfer length of zero specifies that 256 logical blocks shall be written. Any other value specifies the number of logical blocks that shall be written.

11.3.14.2 Write (6) Command Data Transfer

The Device Server requests to transfer the specified logical block(s) from the Application Client data-out buffer by issuing one or more READY TO TRANSFER UPIU's (RTT).

The data is delivered in one or more segments sending DATA OUT UPIU packets, as indicated in the RTT requests. The data contained in DATA OUT UPIU is written.

The number of bytes requested and the Data Buffer Offset field in each RTT shall both be integer multiples of the Logical Block Size (bLogicalBlockSize).

The data segment of each DATA OUT UPIU shall contain an integer number of logical blocks.

Zero or an incomplete number of segments may be requested, if an error occurs before the entire data transfer is complete.

11.3.14.3 Write (6) Command Status Response

- STATUS response will be sent in a single RESPONSE UPIU
- If all requested data is successfully transferred and written, the WRITE command will terminate with a STATUS response of GOOD
- If the logical blocks are transferred directly to a cache then the Device Server may complete the command with a GOOD status prior to writing the logical blocks to the medium
- If the unit is not ready to accept a new command (e.g., still processing previous command) a STATUS response of BUSY will be returned
- Failure can occur for numerous reasons. When the WRITE command fails a STATUS response of CHECK CONDITION will be returned along with an appropriate SENSE KEY, such as
    - ILLEGAL REQUEST (range or CDB errors)
    - MEDIUM ERROR (medium failure, ECC, etc.)
    - HARDWARE ERROR (hardware failure)
    - UNIT ATTENTION (reset, power-on, etc.)
    - DATA PROTECT (permanent, power-on, secure write protect, etc.)
    - etc.

11.3.15 WRITE (10) Command

The WRITE (10) UFS command requests that the Device Server transfer the specified number of logical blocks(s) from the Application Client and write them to the medium.

The Command CDB shall be sent in a single COMMAND UPIU.

Table 11-30 — WRITE (10) command

Byte \ Bit	7	6	5	4	3	2	1	0
0	OPERATION CODE (2Ah)							
1	WRPROTECT = 000b			DPO	FUA	Reserved	FUA_NV = 0b	Obsolete
2	(MSB)							
3				LOGICAL BLOCK ADDRESS				
4								
5								(LSB)
6	Reserved				GROUP NUMBER			
7	(MSB)							
8				TRANSFER LENGTH				(LSB)
9	CONTROL = 00h							

- The WRPROTECT field is set to zero for UFS.

11.3.15.1 Write (10) Command Parameters

- DPO: Disable Page Out
  "0" = specifies that the retention priority shall be determined by the RETENTION PRIORITY fields in the Caching mode page
  "1" = specifies that the device server shall assign the logical blocks accessed by this command the lowest retention priority for being fetched into or retained by the cache. A DPO bit set to one overrides any retention priority specified in the Caching mode page.
- **FUA**: Force Unit Access
  - '0' = The Device Server shall write the logical blocks to the cache and/or the medium.
  - '1' = The Device Server shall write the logical blocks to the medium, and shall not complete the command with GOOD status until all the logical blocks have been written on the medium without error.
- FUA_NV is defined per SBC. Since non-volatile cache support is not currently defined in this standard, the FUA_NV parameter value in the CDB is ignored by the UFS Device Server.
- **LOGICAL BLOCK ADDRESS**: Address of first block
- **TRANSFER LENGTH**: Number of contiguous logical blocks of data that shall be transferred and written. A transfer length of zero specifies that no logical blocks will be written. This condition shall not be considered an error.
- GROUP NUMBER: Notifies the Target device that the data has System Data characteristics or linked to a ContextID:

GROUP NUMBER Value	Function
00000b	Default, no Context ID or System Data characteristics is associated with the write operation.
00001b to 01111b (0XXXXb)	Context ID. (XXXX from 0001b to 1111b - Context ID value)
10000b	Data has System Data characteristics
10001b to 11111b	Reserved

In case the GROUP NUMBER is set to a reserved value, then the operation shall fail and a status response of CHECK CONDITION will be returned along the sense key set to ILLEGAL REQUEST.

11.3.15.2 Write(10) Command Data Transfer

The Device Server requests to transfer the specified logical block(s) from the Application Client data-out buffer by issuing a series of READY TO TRANSFER UPIU's (RTT).

The data is delivered in one or more segments sending DATA OUT UPIU packets, as indicated in the RTT requests. The data contained in DATA OUT UPIU is written.

The number of bytes requested and the Data Buffer Offset field in each RTT shall both be integer multiples of the Logical Block Size (bLogicalBlockSize).

The data segment of each DATA OUT UPIU shall contain an integer number of logical blocks.

Zero or an incomplete number of segments may be requested, if an error occurs before the entire data transfer is complete.

11.3.15.3 Write (10) Command Status Response

- STATUS response will be sent in a single RESPONSE UPIU.
- If all requested data is successfully transferred and written, the WRITE command will terminate with a STATUS response of GOOD.
- If the logical blocks are transferred directly to a cache then the Device Server may complete the command with a GOOD status prior to writing the logical blocks to the medium.
- If the unit is not ready to accept a new command (e.g., still processing previous command) a STATUS response of BUSY will be returned.
- Failure can occur for numerous reasons. When the WRITE command fails a STATUS response of CHECK CONDITION will be returned along with an appropriate SENSE KEY, such as
    - ILLEGAL REQUEST (range or CDB errors)
    - MEDIUM ERROR (medium failure, ECC, etc.)
    - HARDWARE ERROR (hardware failure)
    - UNIT ATTENTION (reset, power-on, etc.)
    - DATA PROTECT (permanent, power-on, secure write protect, etc.)
    - etc.

11.3.16 WRITE (16) Command

The WRITE (16) UFS command requests that the Device Server transfer the specified number of logical blocks(s) from the Application Client and write them to the medium.

The Command CDB shall be sent in a single COMMAND UPIU.

Table 11-31 — WRITE (16) command

Byte \ Bit	7	6	5	4	3	2	1	0
0	OPERATION CODE (8Ah)							
1	WRPROTECT = 000b			DPO	FUA	Reserved	FUA_NV = 0b	Reserved
2	(MSB)							
….	LOGICAL BLOCK ADDRESS							
9								(LSB)
10	(MSB)							
….	TRANSFER LENGTH							
13								(LSB)
14	Reserved[1]	Reserved		GROUP NUMBER				
15	CONTROL = 00h							

NOTE 1 Bit 7 of byte 14 shall be ignored.

- The WDPROTECT field is set to zero for UFS.

11.3.16.1 Write (16) Command Parameters

- DPO: Disable Page Out
  "0" = specifies that the retention priority shall be determined by the RETENTION PRIORITY fields in the Caching mode page
  "1" = specifies that the device server shall assign the logical blocks accessed by this command the lowest retention priority for being fetched into or retained by the cache. A DPO bit set to one overrides any retention priority specified in the Caching mode page.
- **FUA**: Force Unit Access

  '0' = The Device Server shall write the logical blocks to the cache and/or the medium.

  '1' = The Device Server shall write the logical blocks to the medium, and shall not complete the command with GOOD status until all the logical blocks have been written on the medium without error.
- FUA_NV is defined per SBC. Since non-volatile cache support is not currently defined in this standard, the FUA_NV parameter value in the CDB is ignored by the UFS Device Server.
- **LOGICAL BLOCK ADDRESS**: Address of first block
- **TRANSFER LENGTH**: Number of contiguous logical blocks of data that shall be transferred and written. A transfer length of zero specifies that no logical blocks will be written. This condition shall not be considered an error.
- GROUP NUMBER: See Write (10) Command.

11.3.16.2 Write (16) Command Data Transfer

The Device Server requests to transfer the specified logical block(s) from the Application Client data-out buffer by issuing a series of READY TO TRANSFER UPIU's (RTT).

The data is delivered in one or more segments sending DATA OUT UPIU packets, as indicated in the RTT requests. The data contained in DATA OUT UPIU is written.

The number of bytes requested and the Data Buffer Offset field in each RTT shall both be integer multiples of the Logical Block Size (bLogicalBlockSize).

The data segment of each DATA OUT UPIU shall contain an integer number of logical blocks.

Zero or an incomplete number of segments may be requested, if an error occurs before the entire data transfer is complete.

11.3.16.3 Write (16) Command Status Response

- STATUS response will be sent in a single RESPONSE UPIU
- If all requested data is successfully transferred and written, the WRITE command will terminate with a STATUS response of GOOD.
- If the logical blocks are transferred directly to a cache then the Device Server may complete the command with a GOOD status prior to writing the logical blocks to the medium.
- If the unit is not ready to accept a new command (e.g., still processing previous command) a STATUS response of BUSY will be returned.
- Failure can occur for numerous reasons. When the WRITE command fails a STATUS response of CHECK CONDITION will be returned along with an appropriate SENSE KEY, such as
  - ILLEGAL REQUEST (range or CDB errors)
  - MEDIUM ERROR (medium failure, ECC, etc.)
  - HARDWARE ERROR (hardware failure)
  - UNIT ATTENTION (reset, power-on, etc.)
  - DATA PROTECT (permanent, power-on, secure write protect, etc.)
  - etc.

11.3.17 REQUEST SENSE Command

The REQUEST SENSE Command requests that the Device Server transfer parameter data containing sense data information to the Application Client.

- Sense Data describes error or exception condition and/or current operational status of device
  - i.e., get the device "status"
- UFS devices will return a fixed format data record of 18 bytes of sense data as described below.
- Three tiered error code for detailed status
  - Sense Key: main indicator
  - ASC: Additional Sense Code
  - ASCQ: Additional Sense Code Qualifier
- If a REQUEST SENSE command is received with a pending UNIT ATTENTION condition (i.e., before the device server reports CHECK CONDITION status), the device server shall perform the REQUEST SENSE command and clear the UNIT ATTENTION condition.

The Command CDB shall be sent in a single COMMAND UPIU

Table 11-32 — REQUEST SENSE command

Byte \ Bit	7	6	5	4	3	2	1	0
0	OPERATION CODE (03h)							
1	Reserved							DESC = 0b
2	(MSB)	Reserved						
3								(LSB)
4	ALLOCATION LENGTH							
5	CONTROL = 00h							

UFS devices are not required to support descriptor format sense data.

11.3.17.1 Request Sense Data Response

- Data returned from a REQUEST SENSE command will be transferred to the Application Client in a single DATA IN UPIU.
- The Device Server will transfer up to 18 bytes of Response Data in the Data Segment area of a DATA IN UPIU.
  - Return 18 bytes if Allocation Length in CDB >= 18.
  - Return Allocation Length bytes if Allocation Length in CDB < 18.
  - An Allocation Length of zero specifies that no data shall be transferred. This condition shall not be considered as an error, and DATA IN UPIU shall not be generated.
- Data will be returned in the indicated Sense Data Format described in 11.3.17.2.

11.3.17.2 Sense Data

See Table 10-17.

11.3.17.3 Sense Key

See Table 10-18.

11.3.17.4 Request Sense Status Response

- STATUS response will be sent in a single RESPONSE UPIU.
- If the requested data is successfully transferred, the REQUEST SENSE command will terminate with a STATUS response of GOOD.
- If the unit is not ready to accept a new command (e.g., still processing previous command) a STATUS response of BUSY will be returned.
- Failure is very rare. When the REQUEST SENSE command fails a STATUS response of CHECK CONDITION will be returned along with an appropriate SENSE KEY, such as
  - ILLEGAL REQUEST (range or CDB errors).
- Will not fail due to a pending UNIT ATTENTION condition.
- If the REQUEST SENSE command was received with a pending unit attention condition, the returned sense data will indicate the cause of the unit attention condition, and the unit attention condition within the device server will be cleared.
- If a REQUEST SENSE command is terminated with CHECK CONDITION status, then the device server shall not clear the pending unit attention condition.

11.3.18 FORMAT UNIT Command

The FORMAT UNIT command requests that the Device Server format the medium into Application Client accessible logical blocks as specified in the parameter lists. The Device Server may also certify the medium and create control structures for the management of the medium and defects. The degree that the medium is altered by the command is vendor specific.

A FORMAT UNIT command sent to the Device well known logical unit requests the device format all enabled logical units except the RPMB well known logical unit (see in 12.2.3.4, Wipe Device).

If the medium is write-protected, then the command shall be terminated with CHECK CONDITION status with the sense key set to DATA PROTECT.

Following a successful format operation all LBAs:

a) shall be mapped on a fully provisioned logical unit (bProvisioningType set to 00h)
b) shall be unmapped on a thin provisioned logical unit (bProvisioningType set to 02h or 03h)

For a LBA in a formatted logical unit specified by a read operation, the device server shall send user data with all bits set to zero to the data in buffer.

The Command CDB shall be sent in a single COMMAND UPIU.

Table 11-33 — FORMAT UNIT command

Bit / Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (04h)							
1	FMTPINFO = 00b		LONGLIST	FMTDATA = 0b	CMPLST	DEFECT LIST FORMAT= 000b		
2	Vendor Specific = 00b							
3	Obsolete							
4								
5	CONTROL = 00h							

11.3.18.1 Format Unit Command Parameters

Table 11-34 — Format Unit Command Parameters

Byte	Bit	Description
1	7:6	**FMTPINFO**: Specifies the FORMAT PROTECTION INFORMATION as detailed in [SBC].
1	5	**LONGLIST**: If set to '0' then the parameter list, if any, will use the SHORT parameter list header as defined in [SBC]. If set to '1' then the parameter list uses the LONG format.
1	4	**FMTDATA**: If set to '1' specifies that parameter list data shall be transferred from the data-out buffer.
1	3	**CMPLST**: Complete List. '0' indicates that the parameter list contains a partial growing list of defects. A '1' indicates the list is complete. See [SBC].
1	2:0	**DEFECT LIST FORMAT**: If the FMTDATA bit is set to one, then the DEFECT LIST FORMAT field specifies the format of the address descriptors in the defect list. See [SBC].
2	7:0	**VENDOR SPECIFIC**: Vendor specified field.

11.3.18.2 Format Unit Command Data Transfer

- If needed, the Device Server requests to transfer the FORMAT UNIT parameter list from the Application Client data-out buffer by issuing a series of READY TO TRANSFER UPIU's (RTT).
- The FORMAT UNIT parameter list is delivered in one or more segments sending DATA OUT UPIU packets, as indicated in the RTT requests.
- Zero or an incomplete number of segments may be requested if an error occurs before the entire data transfer is complete.

11.3.18.3 Format Unit Command Status Response

- STATUS response will be sent in a single RESPONSE UPIU.
- If the requested format of the medium is performed successfully then the command will terminate with a STATUS response of GOOD.
- If the unit is not ready to accept a new command (e.g., still processing previous command) a STATUS response of BUSY will be returned.
- Other failures can occur for numerous reasons. When the FORMAT UNIT command fails, a STATUS response of CHECK CONDITION will be returned along with an appropriate SENSE KEY, such as:
  - ILLEGAL REQUEST (range or CDB errors)
  - MEDIUM ERROR (medium failure, ECC, etc.)
  - HARDWARE ERROR (hardware failure)
  - UNIT ATTENTION (reset, power-on, etc.)
  - DATA PROTECT (permanent, power-on, secure write protect, etc.)
  - etc.

11.3.19 PRE-FETCH (10) Command

The PRE-FETCH (10) command shall require the device server to transfer the logical blocks for the mapped LBAs specified by the command from the medium to the volatile cache (if such cache exists) and for any unmapped LBAs specified by the command to update the volatile cache to prevent retrieval of stale data as defined in [SBC].

The Command CDB shall be sent in a single COMMAND UPIU.

Table 11-35 — PRE_FETCH command

Bit / Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (34h)							
1	Reserved						IMMED	Obsolete = 0b
2	(MSB)							
3				LOGICAL BLOCK ADDRESS				
4								
5								(LSB)
6	Reserved			GROUP NUMBER = 00000b				
7	(MSB)							
8				PREFETCH LENGTH				(LSB)
9	CONTROL= 00h							

3762 **11.3.19.1 PRE-FETCH (10) Command Parameters**

3763 **Table 11-36 — PRE-FETCH Command Parameters**

Byte	Bit	Description
1	1	**IMMED**: An immediate (IMMED) bit set to zero specifies that the device server shall not return status until the operation has been completed. An IMMED bit set to one specifies that the device server shall return status as soon as the CDB has been validated. If the IMMED bit is set to one, and the device server does not support the IMMED bit, then the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.
2:5	7:0	**LOGICAL BLOCK ADDRESS**: The LOGICAL BLOCK ADDRESS field specifies the LBA of the first logical block accessed by this command. If the specified LBA exceeds the capacity of the medium, then the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to LOGICAL BLOCK ADDRESS OUT OF RANGE.
6	4:0	**GROUP NUMBER:** The GROUP NUMBER field specifies the group into which attributes associated with the command should be collected. A GROUP NUMBER field set to zero specifies that any attributes associated with the command shall not be collected into any group. The use of GROUP NUMBER field for PRE-FETCH command is not defined in this standard therefore this field should be set to zero.
7:8	7:0	**PREFETCH LENGTH**: The PREFETCH LENGTH field specifies the number of contiguous logical blocks that shall be pre-fetched, starting with the logical block specified by the LOGICAL BLOCK ADDRESS field. A NUMBER OF LOGICAL BLOCKS field set to zero specifies that all logical blocks starting with the one specified in the LOGICAL BLOCK ADDRESS field to the last logical block on the medium shall be pre-fetched. If the LBA plus the prefetch length exceeds the capacity of the medium, then the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to LOGICAL BLOCK ADDRESS OUT OF RANGE. The device server is not required to transfer logical blocks that already are contained in the cache.
9	7:0	**CONTROL**: The CONTROL byte is not used in this standard: the CONTROL byte should be set to zero and it shall be ignored by UFS device.

3764

11.3.19.2 PRE-FETCH Command Data Transfer

The PRE-FETCH command does not have a data transfer phase

- No DATA IN or DATA OUT UPIU's are transferred

11.3.19.3 PRE-FETCH Command Status Response

- STATUS response will be sent in a single RESPONSE UPIU
- If the command is performed successfully then it will terminate with a STATUS response of GOOD
- If the unit is not ready to accept a new command (e.g., still processing previous command) a STATUS response of BUSY will be returned
- Other failures can occur for numerous reasons. When the PRE-FETCH command fails, a STATUS response of CHECK CONDITION will be returned along with an appropriate SENSE KEY, such as:
    - ILLEGAL REQUEST (range or CDB errors)
    - MEDIUM ERROR (medium failure, ECC, etc.)
    - HARDWARE ERROR (hardware failure)
    - UNIT ATTENTION (reset, power-on, etc.)
    - etc.

11.3.20 PRE-FETCH (16) Command

See PRE-FETCH (10) command for details.

Table 11-37 — PRE-FETCH (16) command

Bit / Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (90h)							
1	Reserved						IMMED	Reserved
2	(MSB)							
….	LOGICAL BLOCK ADDRESS							
9								(LSB)
10	(MSB)							
….	PREFETCH LENGTH							
13								(LSB)
14	Reserved			GROUP NUMBER = 00000b				
15	CONTROL= 00h							

11.3.21 SECURITY PROTOCOL IN Command

The SECURITY PROTOCOL IN command is used to retrieve security protocol information or the results of one or more SECURITY PROTOCOL OUT commands. See [SPC] for details.

Table 11-38 — SECURITY PROTOCOL IN command

Byte \ Bit	7	6	5	4	3	2	1	0
0	OPERATION CODE (A2h)							
1	SECURITY PROTOCOL							
2	SECURITY PROTOCOL SPECIFIC							
3								
4	INC_512	Reserved						
5	Reserved							
6	(MSB)	ALLOCATION LENGTH						
9								(LSB)
10	Reserved							
11	CONTROL = 00h							

11.3.21.1 SECURITY PROTOCOL IN Command Parameter

- The SECURITY PROTOCOL field specifies which security protocol is being used. UFS devices shall support the following value:
  - ECh: JEDEC UFS application

  Support of other SECURITY PROTOCOL values is device specific.
- A INC_512 bit set to one specifies that the ALLOCATION LENGTH is expressed in increments of 512 bytes.

11.3.21.2 SECURITY PROTOCOL IN Command Data Transfer

- The Device Server transfers security protocol data to the Application Client using one or more DATA IN UPIU's.

11.3.21.3 SECURITY PROTOCOL IN Command Status Response

- Status response shall be sent in a single RESPONSE UPIU
- If the command is successfully executed, it shall terminate with a STATUS response of GOOD
- Failure can occur for numerous reasons. When the command fails a STATUS response of CHECK CONDITION will be returned along with an appropriate SENSE KEY, such as
- ILLEGAL REQUEST (range or CDB errors)
- HARDWARE ERROR (hardware failure)
- UNIT ATTENTION (reset, power-on, etc.)
- etc.

11.3.22 SECURITY PROTOCOL OUT Command

The SECURITY PROTOCOL OUT command is used to send data to the logical unit. The data sent specifies one or more operations to be performed by the logical unit. The format and function of the operations depends on the contents of the SECURITY PROTOCOL field. See [SPC] for details.

Table 11-39 — SECURITY PROTOCOL OUT command

Byte \ Bit	7	6	5	4	3	2	1	0
0	OPERATION CODE (B5h)							
1	SECURITY PROTOCOL							
2	SECURITY PROTOCOL SPECIFIC							
3								
4	INC_512	Reserved						
5	Reserved							
6	(MSB)	TRANSFER LENGTH						
9								(LSB)
10	Reserved							
11	CONTROL = 00h							

11.3.22.1 SECURITY PROTOCOL OUT Command Parameter

- The SECURITY PROTOCOL field specifies which security protocol is being used. UFS devices shall support the following value:
  - ECh: JEDEC UFS application

  Support of other SECURITY PROTOCOL values is device specific.
- A INC_512 bit set to one specifies that the TRANSFER LENGTH is expressed in increments of 512 bytes.

11.3.22.2 SECURITY PROTOCOL OUT Command Data Transfer

The Device Server requests to transfer data from the Application Client data-out buffer by issuing a series of READY TO TRANSFER UPIU's (RTT).

The data is delivered in one or more segments sending DATA OUT UPIU packets, as indicated in the RTT requests.

Zero or an incomplete number of segments may be requested, if an error occurs before the entire data transfer is complete.

11.3.22.3 SECURITY PROTOCOL OUT Command Status Response

- Status response shall be sent in a single RESPONSE UPIU
- If the command is successfully executed, it shall terminate with a STATUS response of GOOD
- Failure can occur for numerous reasons. When the command fails a STATUS response of CHECK CONDITION will be returned along with an appropriate SENSE KEY, such as
- ILLEGAL REQUEST (range or CDB errors)
- HARDWARE ERROR (hardware failure)
- UNIT ATTENTION (reset, power-on, etc.)
- etc.

11.3.23 SEND DIAGNOSTIC Command

The SEND DIAGNOSTIC command requests the Device Server to perform diagnostic operations on the SCSI target device, on the logical unit or on both. Logical units shall implement, at a minimum, the default self-test feature (i.e., the SELFTEST bit equal to one and a parameter list length of zero).

The Command CDB shall be sent in a single COMMAND UPIU.

Table 11-40 — SEND DIAGNOSTIC command

Byte \ Bit	7	6	5	4	3	2	1	0
0	OPERATION CODE (1Dh)							
1	SELF-TEST CODE			PF	Reserved = 0b	SELFTEST	DEVOFFL	UNITOFFL
2	Reserved							
3	(MSB)	PARAMETER LIST LENGTH						
4								(LSB)
5	CONTROL = 00h							

11.3.23.1 Send Diagnostic Parameters

Table 11-41 — Send Diagnostic Parameters

Byte	Bit	Description
1	7:5	**SELF-TEST CODE**: Specifies the self-test code as defined in SPC4.
1	4	**PF**: Specifies the format of any parameter list sent by the Application Client
1	2	**SELFTEST**: Specifies the device server shall perform a self test.
1	1	**DEVOFFL**: If set to ‘0’, the device server will perform a self-test without exhibiting any effects on the logical unit. If set to ‘1’, the device server may perform a self-test that affects the logical unit.
1	0	**UNITOFFL**: If set to ‘0’, the device server will perform a self-test without exhibiting any effects on the user accessible medium of the logical unit. If set to ‘1’, the device server may perform operations that affect the user accessible medium.
1	5	**PARAMETER LIST LENGTH**: Specifies the length in bytes that shall be transferred from the Application Client to the device server. A parameter list length of zero specifies that no data shall be transferred.

11.3.23.2 Send Diagnostic Command Data Transfer

The SEND DIAGNOSTIC command will transfer out the number of bytes specified by the Parameter List Length. If that value is zero then no data out transfer will occur.

The Device Server will request the transfer the specified bytes from the Application Client by issuing a series READY TO TRANSFER UPIU (RTT) followed by a DATA OUT UPIU containing the number of bytes to transfer.

11.3.23.3 Send Diagnostic Command Status Response

- STATUS response will be sent in a single RESPONSE UPIU.
- If the requested diagnostics are performed successfully then the command will terminate with a STATUS response of GOOD.
- If the unit is not ready to accept a new command (e.g., still processing previous command) a STATUS response of BUSY will be returned.
- Other failures can occur for numerous reasons. When the SEND DIAGNOSTIC command fails, a STATUS response of CHECK CONDITION will be returned along with an appropriate SENSE KEY, such as
    - ILLEGAL REQUEST (range or CDB errors)
    - MEDIUM ERROR (medium failure, ECC, etc.)
    - HARDWARE ERROR (hardware failure)
    - UNIT ATTENTION (reset, power-on, etc.)
    - etc.

11.3.24 SYNCHRONIZE CACHE (10) Command

The SYNCHRONIZE CACHE (10) command requests that the device server ensure that the specified logical blocks have their most recent data values recorded on the medium.

Table 11-42 — SYNCHRONIZE CACHE (10) command

Byte \ Bit	7	6	5	4	3	2	1	0
0	OPERATION CODE (35h)							
1	Reserved					SYNC_NV	IMMED	Obsolete =0b
2	(MSB)							
3				LOGICAL BLOCK ADDRESS				
4								
5								(LSB)
6	Reserved			GROUP NUMBER = 00000b				
7	(MSB)							
8				NUMBER OF LOGICAL BLOCKS				(LSB)
9	CONTROL = 00h							

11.3.24.1 Synchronize Cache Command Parameters

Table 11-43 — Synchronize Cache Command Parameters

Byte	Bit	Description
1	2	**SYNC_NV**: SYNC_NV is defined per SBC. Since non-volatile cache support is not currently defined in this standard, the SYNC_NV parameter value in the CDB is ignored by the UFS Device Server.
1	1	**IMMED**: An immediate (IMMED) bit set to zero specifies that the device server shall not return status until the operation has been completed. An IMMED bit set to one specifies that the device server shall return status as soon as the CDB has been validated. If the IMMED bit is set to one, and the device server does not support the IMMED bit, then the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.
2:5	7:0	**LOGICAL BLOCK ADDRESS**: The LOGICAL BLOCK ADDRESS field specifies the LBA of the first logical block accessed by this command. If the specified LBA exceeds the capacity of the medium, then the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to LOGICAL BLOCK ADDRESS OUT OF RANGE.
6	4:0	**GROUP NUMBER:** The use of GROUP NUMBER field for SYNCHRONIZE CACHE command is not defined in this standard therefore this field should be set to zero.
7:8	7:0	**NUMBER OF LOGICAL BLOCKS**: The NUMBER OF LOGICAL BLOCKS field specifies the number of logical blocks that shall be synchronized, starting with the logical block specified by the LOGICAL BLOCK ADDRESS field. A NUMBER OF LOGICAL BLOCKS field set to zero specifies that all logical blocks starting with the one specified in the LOGICAL BLOCK ADDRESS field to the last logical block on the medium shall be synchronized. If the LBA plus the number of logical blocks exceeds the capacity of the medium, then the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to LOGICAL BLOCK ADDRESS OUT OF RANGE. A logical block within the range that is not in cache is not considered an error.
9	7:0	**CONTROL:** The CONTROL byte is not used in this standard: the CONTROL byte should be set to zero and it shall be ignored by UFS device.

11.3.24.2 Synchronize Cache Command Data Transfer

The SYNCHRONIZE CACHE command does not have a data transfer phase.

- No DATA IN or DATA OUT UPIU's are transferred.

11.3.24.3 Synchronize Cache Command Status Response

- STATUS response will be sent in a single RESPONSE UPIU.
- If the command is performed successfully then it will terminate with a STATUS response of GOOD.
- If the unit is not ready to accept a new command (e.g., still processing previous command) a STATUS response of BUSY will be returned.
- Other failures can occur for numerous reasons. When the SYNCHRONIZE CACHE command fails, a STATUS response of CHECK CONDITION will be returned along with an appropriate SENSE KEY, such as
  - ILLEGAL REQUEST (range or CDB errors)
  - MEDIUM ERROR (medium failure, ECC, etc.)
  - HARDWARE ERROR (hardware failure)
  - UNIT ATTENTION (reset, power-on, etc.)
  - etc.

11.3.25 SYNCHRONIZE CACHE (16) Command

See SYNCHRONIZE CACHE (10) command for details.

Table 11-44 — SYNCHRONIZE CACHE (16) Command Descriptor Block

Byte \ Bit	7	6	5	4	3	2	1	0
0	OPERATION CODE (91h)							
1	Reserved					SYNC_NV	IMMED	Reserved
2	(MSB)							
…	LOGICAL BLOCK ADDRESS							
9								(LSB)
10	(MSB)							
…	NUMBER OF LOGICAL BLOCKS							
13								(LSB)
14	Reserved				GROUP NUMBER = 00000b			
15	CONTROL = 00h							

11.3.26 UNMAP Command

The UNMAP command shall require the device server to cause one or more LBAs to be unmapped (de-allocated).

UFS defines that a logical unit shall be either Full Provisioning or Thin Provisioning as described in SCSI SBC. To use UNMAP command, SCSI SBC requires that a logical unit to be thin-provisioned and support logical block provisioning management. UNMAP command is not supported in a full-provisioned logical unit.

In UFS, a thin provisioned logical unit shall have sufficient physical memory resources to support the logical block address space when the device is configured by the user. Mapped State and De-Allocated State are mandatory in a UFS thin provisioned logical unit.

Table 11-45 — UNMAP command

Bit / Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (42h)							
1	Reserved							ANCHOR = 0b
2	(MSB)							
3				Reserved				
4								
5								(LSB)
6	Reserved				GROUP NUMBER = 00000b			
7	(MSB)							
8				PARAMETER LIST LENGTH				(LSB)
9	CONTROL = 00h							

- GROUP NUMBER = ‘0’
- ANCHOR = 0 for UFS. If ANCHOR = 1, the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.
- The PARAMETER LIST LENGTH field specifies the length in bytes of the UNMAP parameter data that are sent from the application client to the device server.

11.3.26.1 UNMAP parameter list

The UNMAP parameter list contains the data sent by an application client along with an UNMAP command. Included in the data are an UNMAP parameter list header and block descriptors for LBA extents to be processed by the device server for the UNMAP command. The LBAs specified in the block descriptors may contain overlapping extents, and may be in any order.

Table 11-46 — UNMAP parameter list

Byte \ Bit	7	6	5	4	3	2	1	0
0	(MSB)	UNMAP DATA LENGTH (n-1)						
1								(LSB)
2	(MSB)	UNMAP BLOCK DESCRIPTOR DATA LENGTH (n-7)						
3								(LSB)
4	(MSB)	Reserved						
….								
7								(LSB)
	UNMAP block descriptors							
8	(MSB)	UNMAP block descriptor (first)						
….								
23								(LSB)
	…							
n-15	(MSB)	UNMAP block descriptor (last)						
…								
n								(LSB)

- The UNMAP DATA LENGTH field specifies the length in bytes of the following data that is available to be transferred from the data-out buffer. The unmap data length does not include the number of bytes in the UNMAP DATA LENGTH field.
- The UNMAP BLOCK DESCRIPTOR DATA LENGTH field specifies the length in bytes of the UNMAP block descriptors that are available to be transferred from the data-out buffer. The unmap block descriptor data length should be a multiple of 16. If the unmap block descriptor data length is not a multiple of 16, then the last unmap block descriptor is incomplete and shall be ignored. If the UNMAP BLOCK DESCRIPTOR DATA LENGTH is set to zero, then no unmap block descriptors are included in the UNMAP parameter data. This condition shall not be considered an error.

11.3.26.2 UNMAP block descriptor

Table 11-47 — UNMAP block descriptor

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)							
….				UNMAP LOGICAL BLOCK ADDRESS				
7								(LSB)
8	(MSB)							
….				NUMBER OF LOGICAL BLOCKS				
11								(LSB)
12								
….				Reserved				
15								

- The UNMAP LOGICAL BLOCK ADDRESS field contains the first LBA of the UNMAP block descriptor to be unmapped.
- The NUMBER OF LOGICAL BLOCKS field contains the number of LBAs to be unmapped beginning with the LBA specified by the UNMAP LOGICAL BLOCK ADDRESS field.
- To minimize performance degradation, the entire LBA region to be unmapped should be aligned with the bOptimalWriteBlockSize value in the Unit Descriptor where possible (but not required).
- If the NUMBER OF LOGICAL BLOCKS is set to zero, then no LBAs shall be unmapped for this UNMAP block descriptor. This condition shall not be considered an error.
- If the LBA specified by the UNMAP LOGICAL BLOCK ADDRESS field plus the number of logical blocks exceeds the capacity of the medium, then the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to LOGICAL BLOCK ADDRESS OUT OF RANGE.
- If the UFS device does not support Block Limits VPD page then MAXIMUM UNMAP LBA COUNT value and MAXIMUM UNMAP BLOCK DESCRIPTOR COUNT value are defined as in the following:
  - MAXIMUM UNMAP LBA COUNT = LBA count reported in READ CAPACITY
  - MAXIMUM UNMAP BLOCK DESCRIPTOR COUNT = 1
- If Vital Product Data Page is supported by the device, the MAXIMUM UNMAP LBA COUNT and MAXIMUM UNMAP BLOCK DESCRIPTOR COUNT are set by the device manufacturer in the Block Limits VPD page. If the total number of logical blocks specified in the UNMAP block descriptor data exceeds the value indicated in the MAXIMUM UNMAP LBA COUNT field in the Block Limits VPD page or if the number of UNMAP block descriptors exceeds the value of the MAXIMUM UNMAP BLOCK DESCRIPTOR COUNT field in the Block Limits VPD page, then the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

11.3.26.3 UNMAP parameter list transfer

- The Device Server requests to transfer the UNMAP parameter list from the Application Client data-out buffer by issuing a series of READY TO TRANSFER UPIU's (RTT).
- The UNMAP parameter list is delivered in one or more segments sending DATA OUT UPIU packets, as indicated in the RTT requests.
- Zero or an incomplete number of segments may be requested, if an error occurs before the entire data transfer is complete.

11.3.27 READ BUFFER Command

The READ BUFFER command is used in conjunction with the WRITE BUFFER command for

- testing logical unit buffer memory
- testing the integrity of the service delivery subsystem
- Downloading microcode
- Retrieving error history and statistics

The READ BUFFER command transfers a specified number of data bytes from a specified offset within a specified buffer in the Device Server to a buffer in the Application Client.

The Command CDB shall be sent in a single COMMAND UPIU.

Table 11-48 — READ BUFFER command

Bit / Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (3Ch)							
1	Reserved			MODE				
2	BUFFER ID							
3	(MSB)							
4	BUFFER OFFSET							
5								(LSB)
6	(MSB)							
7	ALLOCATION LENGTH							
8								(LSB)
9	CONTROL = 00h							

11.3.27.1 Read Buffer Command Parameters

Table 11-49 — Read Buffer Command Parameters

Byte	Bit	Description
1	4:0	**MODE**: Specifies the function of this command. DATA MODE (02h) shall be used to transfer UFS specific data. See table below for more detail.
2	7:0	**BUFFER ID**: Specifies a buffer within the logical unit. Buffer 0 shall be supported. If more than one buffer is supported, then additional BUFFER ID codes shall be assigned contiguously, beginning with one.
3:5	7:0	**BUFFER OFFSET**: Specifies the byte offset within the specified buffer from which data shall be transferred.
6:8	7:0	**ALLOCATION LENGTH**: Specifies the maximum number of bytes of buffer space that the Application Client has allocated for data reception.

11.3.27.2 Read Buffer Command Mode Field Values

Table 11-50 — Read Buffer Command Mode Field Values

MODE	DESCRIPTION
00h	Not used in UFS
01h	Vendor Specific
02h	Data
03h-1Ch	Not used in UFS
1Dh-1Fh	Reserved

- The device shall support the MODE value of 02h, indicating Data Mode. The BUFFER ID field specifies a buffer from which data shall be transferred. The BUFFER OFFSET field contains the byte offset from which data shall be transferred.
- The definition and structure of the data being transferred in Data Mode is device specific.

11.3.27.3 Read Buffer Command Data Transfer

- The Device Server will read up to Allocation Length number of data bytes from the specified Buffer Offset within a buffer specified by the Buffer ID in the Device Server and transfer them to a buffer in the Application Client
    - Less than Allocation Length will be transferred if Device Server contains less bytes
- Data will be transferred from the Device Server to the Application Client via a series of DATA IN UPIU's
    - The data transferred from the Device Server will be contained within the Data Segment of the DATA IN UPIU
- Zero or an incomplete number of DATA IN UPIU's will be transferred if an error occurs before the entire data transfer is complete

11.3.27.4 Read Buffer Command Status Response

- Status response will be sent in a single RESPONSE UPIU
- If all requested data is successfully read and transferred, the READ BUFFER command will terminate with a STATUS response of GOOD
- If the unit is not ready to accept a new command (e.g., still processing previous command) a STATUS response of BUSY will be returned
- Failure can occur for numerous reasons. When the READ BUFFER command fails a STATUS response of CHECK CONDITION will be returned along with an appropriate SENSE KEY, such as:
    - ILLEGAL REQUEST (range or CDB errors)
    - MEDIUM ERROR (medium failure, ECC, etc.)
    - HARDWARE ERROR (hardware failure)
    - UNIT ATTENTION (reset, power-on, etc.)
    - etc.

11.3.28 WRITE BUFFER Command

The WRITE BUFFER command is used in conjunction with the READ BUFFER command for

- testing logical unit buffer memory
- testing the integrity of the service delivery subsystem
- Field Firmware Update
- Retrieving error history and statistics

The WRITE BUFFER command transfers a specified number of data bytes from a buffer in the Application Client to a specified buffer in the Device Server at a specified buffer offset

The Command CDB shall be sent in a single COMMAND UPIU

Table 11-51 — WRITE BUFFER command

Bit Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (3Bh)							
1	Reserved			MODE				
2	BUFFER ID							
3	(MSB)							
4	BUFFER OFFSET							
5								(LSB)
6	(MSB)							
7	PARAMETER LIST LENGTH							
8								(LSB)
9	CONTROL = 00h							

11.3.28.1 Write Buffer Command Parameters

Table 11-52 — Write Buffer Command Parameters

Byte	Bit	Description
1	4:0	**MODE**: Specifies the function of this command. See Table 11-53 for more detail.
2	7:0	**BUFFER ID**: Specifies a buffer within the logical unit. Buffer 0 shall be supported. If more than one buffer is supported, then additional BUFFER ID codes shall be assigned contiguously, beginning with one.
3:5	7:0	**BUFFER OFFSET**: Specifies the byte offset within the specified buffer from which data shall be transferred.
6:8	7:0	**PARAMETER LIST LENGTH**: Specifies the maximum number of bytes the Application Client buffer will transfer to the Device Server.

11.3.28.2 Write Buffer Command Mode Field Values

Table 11-53 — Write Buffer Command Mode Field Values

MODE	DESCRIPTION
00h	Not used in UFS
01h	Vendor Specific
02h	Data
03h	Not used in UFS
04h	Not used in UFS
05h	Not used in UFS
06h	Not used in UFS
07h	Not used in UFS
08h-0Dh	Not used in UFS
0Eh	Download microcode with offsets, save and defer active
0Fh	Not used in UFS
10h-1Ch	Not used in UFS
1Dh-1Fh	Reserved

- The device shall support the MODE value of 02h, indicating Data Mode. The BUFFER ID field specifies a buffer to which data shall be transferred. The BUFFER OFFSET field specifies the location to which the data is written.
- The definition and structure of the data being transferred in Data Mode is device specific.
- UFS device shall support a MODE value of 0Eh for microcode download as defined in section Field Firmware Update.

11.3.28.2.1 Field Firmware Update

UFS Field Firmware Update (FFU) is based on microcode download definition in [SPC].

[SPC] describes multiple operation modes for microcode download which are selected using the MODE field in the WRITE BUFFER command. UFS supports only the MODE field value 0Eh: "Download microcode with offsets, save, and defer active".

The deferred microcode shall be activated and no longer considered deferred when a power on or a hard reset occurs. Note that in UFS, START STOP UNIT command, FORMAT UNIT command or WRITE BUFFER command (MODE=0Fh) will not activate the microcode.

UFS FFU uses the following mechanism:

1) Host delivers the microcode using one or more WRITE BUFFER commands through any logical unit which supports the WRITE BUFFER command. The host specifies: MODE = 0Eh, BUFFER OFFSET, which should be aligned to 4 Kbyte, BUFFER ID = 00h, and the PARAMETER LIST LENGTH field indicating the number of bytes to be transferred.
   All WRITE BUFFER commands should be sent to the same logical unit with task attribute set to simple or ordered. In the sequence of WRITE BUFFER commands used to deliver the microcode, the BUFFER OFFSET values should be in increasing order and it should start from zero.
2) bFFUTimeout indicates the maximum time in which the device may handle the WRITE BUFFER command. Within this time access to the device is limited or not possible.
3) Following a successful delivery of the microcode, the host activates the new firmware using a hardware reset or a power cycle. The UFS device shall use new firmware upon hard reset or power up. Host should be aware that the first initialization flow after a successful delivery of the microcode may be longer than usual.
4) After device initialization, the host should read bDeviceFFUStatus attribute and verify that the new firmware was updated successfully.

Other modes of WRITE BUFFER command are not supported by the UFS device for FFU process.

11.3.28.3 Write Buffer Command Data Transfer

The Device Server requests to transfer the buffer data from the Application Client data-out buffer by issuing a series of READY TO TRANSFER UPIU's (RTT).

The buffer data is delivered in one or more segments sending DATA OUT UPIU packets, as indicated in the RTT requests.

Zero or an incomplete number of segments may be requested, if an error occurs before the entire data transfer is complete.

The received data is written to the location specified by the BUFFER OFFSET field within the buffer specified by the BUFFER ID field.

11.3.28.4 Write Buffer Command Status Response

- STATUS response will be sent in a single RESPONSE UPIU
- If the requested data is successfully transferred and written, the WRITE BUFFER command will terminate with a STATUS response of GOOD
- If the unit is not ready to accept a new command (e.g., still processing previous command) a STATUS response of BUSY will be returned
- Failure can occur for numerous reasons. When the WRITE BUFFER command fails a STATUS response of CHECK CONDITION will be returned along with an appropriate SENSE KEY, such as:
  - ILLEGAL REQUEST (range or CDB errors)
  - MEDIUM ERROR (medium failure, ECC, etc.)
  - HARDWARE ERROR (hardware failure)
  - UNIT ATTENTION (reset, power-on, etc.)
  - etc.

11.4 Mode Pages

This section describes the mode pages used with MODE SELECT command and MODE SENSE command. Subpages are identical to mode pages except that they include a SUBPAGE CODE field that further differentiates the mode page contents.

11.4.1 Mode Page Overview

11.4.1.1 Mode Page/Subpage Codes

- Mode pages and subpages are selected by Page Code field and the Subpage Code field
- UFS devices are not required to support subpages (subpage = 0)

Table 11-54 — Mode page code usage

Page Code	Subpage Code	Description	Page Format
00h	Vendor specific	Vendor specific page	Vendor specific format
01h to 1Fh (SCSI SPECIFIC)	00h	Device specific STANDARD page (subpage 0)	Page 0 format
	01h to DFh	Device specific SUBPAGE	Subpage format
	E0h to FEh	Vendor specific SUBPAGE	Subpage Format
	FFh	Return all SUBPAGES for the specified device specific mode page	Page 0 format for subpage 00h, subpage format for subpages 01h to FEh
20h to 3Eh (VENDOR SPECIFIC)	00h	Vendor specific STANDARD page (subpage 0)	Page 0 format
	01h to FEh	Vendor specific SUBPAGE	Subpage format
	FFh	Return all SUBPAGES for the specified vendor specific mode page	Page 0 format for subpage 00h, subpage format for subpages 01h to FEh
3Fh (Return ALL pages)	00h	Return all STANDARD pages (subpage 0)	Page 0 format
	01h to FEh	Reserved	NA
	FFh	Return all subpages for all mode pages	Page 0 format for subpage 00h, sub_page format for subpages 01h to FEh

11.4.1.2 Mode parameter list format

General format for reading or writing mode pages.

UFS will not implement Block Descriptor field

Table 11-55 — UFS Mode parameter list

Bit / Byte	7	6	5	4	3	2	1	0
	Mode Parameter Header							
	Block Descriptor(s)							
	Mode Page(s) or Subpages or Vendor specific pages							

11.4.1.3 Mode Parameter Header

The mode parameter header that is used by the MODE SELECT (10) command and the MODE SENSE (10) command is defined in the following table.

Table 11-56 — UFS Mode parameter header (10)

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)	MODE DATA LENGTH						
1								(LSB)
2	MEDIUM TYPE = 00h							
3	DEVICE SPECIFIC PARAMETER							
	WP	Reserved = 00b		DPOFUA	Reserved = 0000b			
4	Reserved = 00h							LONGLBA = 0b
5	Reserved = 00h							
6	(MSB)	BLOCK DESCRIPTOR LENGTH = 0000h						
7								(LSB)

When using the MODE SENSE command, the MODE DATA LENGTH field indicates the length in bytes of the following data that is available to be transferred. The mode data length does not include the number of bytes in the MODE DATA LENGTH field. When using the MODE SELECT command, this field is reserved.

11.4.1.4 Mode Parameter Header Detail

Table 11-57 — Mode Parameter Header Detail

Byte	Bit	Description
0:1	7:0	**MODE DATA LENGTH**: Indicates the length in bytes of data following this field that is available to transfer. This value does not include the size of this field (2 bytes). For MODE SENSE 10-byte CDB, this value will be calculated as 6 + page data bytes.
2	7:0	**MEDIUM TYPE**: Indicates the medium type of the device. For UFS this value shall be set to 00h, indicating Data Medium.
3	7:0	**DEVICE SPECIFIC PARAMETER**: Direct access device specific value. When used with the MODE SELECT command, the write protect (WP) bit is reserved. When used with the MODE SENSE command, a WP bit set to one indicates that the medium is write-protected, a WP bit set to zero indicates that the medium is not write-protected (1). When used with the MODE SELECT command, the DPOFUA bit is reserved. When used with the MODE SENSE command, a DPOFUA bit set to zero indicates that the device server does not support the DPO and FUA bits. When used with the MODE SENSE command, a DPOFUA bit set to one indicates that the device server supports the DPO and FUA bits
6:7	7:0	**BLOCK DESCRIPTOR LENGTH**: Length of block descriptor in parameter list. For UFS this value shall be 00h indicating that there is no block descriptor(s) used in the parameter list.

NOTE 1 The WP bit shall be set to one when the logical unit is write-protected by any method, including any one of the following:

- the software write protect (SWP) bit in the Control mode page is set to one
- the logical unit is configured as permanently write protected and fPermanentWPEn = 1
- the logical unit is configured as power on write protected and fPowerOnWPEn = 1
- the device is write-protected by vendor-specific electrical or mechanical mechanism.

4117 **11.4.1.5 Page_0 mode page format**

4118 **Table 11-58 — Page_0 mode page format**

Byte \ Bit	7	6	5	4	3	2	1	0
0	PS	SPF (0b)	PAGE CODE					
1	PAGE LENGTH (n – 1)							
2	Mode Parameters							
n								

4119

4120 **Table 11-59 — Page 0 Format parameters**

Byte	Bit	Description
0	7:7	**PS**: Indicates the page parameters can be saved. When using the MODE SENSE command, the PS bit set to one indicates that the mode page may be saved by the logical unit in a nonvolatile, vendor specific location. A PS bit set to zero indicates that the device server is not able to save the supported parameters. When using the MODE SELECT command, the PS bit is reserved.
0	6:6	**SPF**: Indicates SUBPAGE format. When set to zero indicates that the PAGE 0 format is being used. When set to one, indicates the SUBPAGE mode page format is being used.
0	5:0	**PAGE CODE**: Indicates the format and parameters for particular mode page.
1	7:0	**PAGE LENGTH**: Indicates the size in bytes of the following mode page parameters.
2:N	7:0	**MODE PARAMETERS**: The contents of the indicated mode page.

4121

4122 **11.4.1.6 Sub_page mode page format**

4123 **Table 11-60 — Sub_page mode page format**

Byte \ Bit	7	6	5	4	3	2	1	0
0	PS	SPF (1b)	PAGE CODE					
1	SUBPAGE CODE							
2	(MSB)	PAGE LENGTH (n – 3)						
3								(LSB)
4		Mode Parameters						
n								

4124

4125 **Table 11-61 — Subpage Format parameters**

Byte	Bit	Description
0	7:7	**PS**: Indicates the page parameters can be saved. When using the MODE SENSE command, the PS bit set to one indicates that the mode page may be saved by the logical unit in a nonvolatile, vendor specific location. A PS bit set to zero indicates that the device server is not able to save the supported parameters. When using the MODE SELECT command, the PS bit is reserved.
0	6:6	**SPF**: Indicates SUBPAGE format. When set to zero indicates that the PAGE 0 format is being used. When set to one, indicates the SUBPAGE mode page format is being used.
0	5:0	**PAGE CODE**: Page and Subpage code indicates the format and parameters for particular mode page.
1	7:0	**SUBPAGE CODE:** Page and subpage code indicates the format and parameters for particular mode page.
2:3	7:0	**PAGE LENGTH**: Indicates the size in bytes of the following mode page parameters.
4:N	7:0	**MODE PARAMETERS**: The contents of the indicated mode page.

4126

11.4.2 UFS Supported Pages

Table 11-62 shows the mode pages supported by UFS device. This standard does not define any additional subpages.

Table 11-62 — UFS Supported Pages

PAGE NAME	PAGE CODE	SUBPAGE CODE	DESCRIPTION
CONTROL	0Ah	00h	Return CONTROL mode page
READ-WRITE ERROR RECOVERY	01h	00h	Return READ-WRITE ERROR RECOVERY mode page
CACHING	08h	00h	Return CACHING mode page
ALL PAGES	3Fh	00h	Return all mode pages (not including subpages)
ALL SUBPAGES	3Fh	FFh	Return all mode pages and subpages

If the device has more than one logical unit, host should read Mode Page Policy VPD in order to know whether the logical unit maintains its own copy of the mode page and subpage or all logical units share the mode page and subpage.

11.4.2.1 Control Mode Page

The Control mode page provides controls over SCSI features that are applicable to all device types (e.g., task set management and error logging).

Table 11-63 defines the Control mode page default value (PC = 10b).

Table 11-63 — Control Mode Page default value

Byte \ Bit	7	6	5	4	3	2	1	0
0	PS	SPF (0)	PAGE CODE (0Ah)					
1	PAGE LENGTH (0Ah)							
2	TST = 000b			TMF_ONLY = 0b	DPICZ = 0b	D_SENSE = 0b	GLTSD = 0b	RLEC = 0b
3	QUEUE ALGORITHM MODIFIER = 0001b				NUAR = 0b	QERR = 00b		Obsolete = 0b
4	VS = 0b	RAC = 0b	UA_INTLCK_CTRL = 00b		SWP = 0b	Obsolete = 000b		
5	ATO = 0b	TAS = 0b	ATMPE = 0b	RWWP = 0b	Reserved = 0b	AUTOLOAD MODE = 000b		
6	Obsolete = 0000h							
7								
8	(MSB)	BUSY TIMEOUT PERIOD						
9								(LSB)
10	(MSB)	EXTENDED SELF-TEST COMPLETION TIME						
11								(LSB)

NOTE 1 Default values for PS bit, BUSY TIMEOUT PERIOD field and EXTENDED SELF-TEST COMPLETION TIME field are device specific.

The following Control mode page field shall be changeable: SWP. The following Control mode page fields are not changeable: TST and BUSY TIMEOUT PERIOD. Other fields may or may not be changeable, refer to the vendor datasheet for details.

4144 **11.4.2.1.1 Control Mode Page Parameters**

4145 **Table 11-64 — Control Mode Page Parameters**

Byte	Bit	Description
1	7:5	**TST**: Indicates Task Set Type. 000b indicates the logical unit maintains one task set for all I_T nexuses. Others: reserved.
4	3:3	**SWP**: A software write protect (SWP) bit set to one specifies that the logical unit shall inhibit writing to the medium after writing all cached or buffered write data, if any. When SWP is one, all commands requiring writes to the medium shall be terminated with CHECK CONDITION status, with the sense key set to DATA PROTECT
8:9	7:0	**BUSY TIMEOUT PERIOD**: The BUSY TIMEOUT PERIOD field specifies the maximum time, in 100 milliseconds increments, that the application client allows for the device server to return BUSY status for commands from the application client. A 0000h value in this field is undefined. An FFFFh value in this field is defined as an unlimited period.
NOTE 1 In addition to the software write protection, logical units may be configured as permanently write protected or power on write protected. A logical unit is writeable if all types of write protection are disabled. Logical units may be write protected setting SWP to one or using one of the methods described in 12.3, Device Data Protection.		

4146

11.4.2.2 Read-Write Error Recovery Mode Page

The Read-Write Error Recovery mode page specifies the error recovery parameters the device server shall use during any command that performs a read or write operation to the medium (e.g., READ command, WRITE command, or VERIFY command).

Table 11-65 defines the Read-Write Error Recovery mode page default value (PC = 10b).

Table 11-65 — Read-Write Error Recovery Mode Page default value

Bit Byte	7	6	5	4	3	2	1	0
0	PS	SPF (0b)	PAGE CODE (01h)					
1	PAGE LENGTH (0Ah)							
2	AWRE = 1b	ARRE = 0b	TB = 0b	RC = 0b	EER = 0b	PER = 0b	DTE = 0b	DCR = 0b
3	READ RETRY COUNT							
4	Obsolete = 00h							
5	Obsolete = 00h							
6	Obsolete = 00h							
7	TPERE = 0b	Reserved = 00000b					Restricted for MMC-6 = 00b	
8	WRITE RETRY COUNT							
9	Reserved = 00h							
10	(MSB)	RECOVERY TIME LIMIT						
11								(LSB)

NOTE 1 Default values for PS field, READ RETRY COUNT field, WRITE RETRY COUNT field and RECOVERY TIME LIMIT are device specific.

This standard does not define which Read-Write Error Recovery mode page fields are changeable, refer to vendor datasheet for details.

11.4.2.2.1 Read-Write Error Recovery Parameters

Table 11-66 — Read-Write Error Recovery Parameters

Byte	Bit	Description
3	7:0	**READ RETRY COUNT**: The READ RETRY COUNT field specifies the number of times that the device server shall attempt its recovery algorithm during read operations.
8	7:0	**WRITE RETRY COUNT**: The WRITE RETRY COUNT field specifies the number of times that the device server shall attempt its recovery algorithm during write operations.
10:11	7:0	**RECOVERY TIME LIMIT**: The RECOVERY TIME LIMIT field specifies in milliseconds the maximum time duration that the device server shall use for data error recovery procedures. When both a retry count and a recovery time limit are specified, the field that specifies the recovery action of least duration shall have priority.

11.4.2.3 Caching Mode Page

The Caching mode page defines the parameters that affect the use of the cache. A UFS device shall implement support for following parameters.

Table 11-67 defines the Caching mode page default value (PC = 10b).

Table 11-67 — Caching Mode Page default value

Byte \ Bit	7	6	5	4	3	2	1	0
0	PS	SPF (0b)	PAGE CODE (08h)					
1	PAGE LENGTH (12h)							
2	IC = 0b	ABPF = 0b	CAP = 0b	DISC = 0b	SIZE = 0b	WCE =1b	MF = 0b	RCD =0b
3	DEMAND READ RETENTION PRIORITY = 0000b				WRITE RETENTION PRIORITY = 0000b			
4	(MSB)	DISABLE PRE-FETCH TRANSFER LENGTH = 0000h						
5								(LSB)
6	(MSB)	MINIMUM PRE-FETCH = 0000h						
7								(LSB)
8	(MSB)	MAXIMUM PRE-FETCH = 0000h						
9								(LSB)
10	(MSB)	MAXIMUM PRE-FETCH CEILING = 0000h						
11								(LSB)
12	FSW = 0b	LBCSS = 0b	DRA = 0b	Vendor Specific = 00b		Reserved = 00b		NV_DIS = 0b
13	NUMBER OF CACHE SEGMENTS = 00h							
14	(MSB)	CACHE SEGMENT SIZE = 0000h						
15								(LSB)
16	Reserved = 00h							
17		Obsolete = 000000h						
18								
19								

The following Caching mode page fields shall be changeable: WCE and RCD. Other fields may or may not be changeable, refer to the vendor datasheet for details.

11.4.2.3.1 Caching Mode Page Parameters

Table 11-68 — Caching Mode Page Parameters

Byte	Bit	Description
2	2:2	**WCE**: WRITE BACK CACHE ENABLE. A writeback cache enable bit set to zero specifies that the device server shall complete a WRITE command with GOOD status only after writing all of the data to the medium without error. A WCE bit set to one specifies that the device server may complete a WRITE command with GOOD status after receiving the data without error and prior to having written the data to the medium.
2	0:0	**RCD**: READ CACHE DISABLE. A read cache disable bit set to zero specifies that the device server may return data requested by a READ command by accessing either the cache or medium. A RCD bit set to one specifies that the device server shall transfer all of the data requested by a READ command from the medium (i.e., data shall not be transferred from the cache).
NOTE 1 Fields that are not supported by UFS should be set to zero, and are documented assigning a value of zero to them (e.g., PS=0b). The device may ignore values in fields that are not supported by UFS.		

11.5 Vital product data parameters

11.5.1 Overview

The vital product data (VPD) pages are returned by an INQUIRY command with the EVPD bit set to one and contain vendor specific product information about a logical unit and SCSI target device.

A UFS device shall support the following VPD pages:

- Supported VPD Pages
- Mode Page Policy;

Support for other VPD pages is optional.

11.5.2 VPD page format

Table 11-69 shows the VPD page structure.

Table 11-69 — VPD page format

Bit / Byte	7	6	5	4	3	2	1	0
0	PERIPHERAL QUALIFIER			PERIPHERAL DEVICE TYPE				
1	PAGE CODE							
2	(MSB)	PAGE LENGTH (n-3)						
3								(LSB)
4	(MSB)	VPD parameters						
n								(LSB)

The PERIPHERAL QUALIFIER field and the PERIPHERAL DEVICE TYPE field are the same as defined for standard INQUIRY data (see 11.3.2.2).

The PAGE CODE field identifies the VPD page and contains the same value as in the PAGE CODE field in the INQUIRY CDB (see 11.3.2).

The PAGE LENGTH field indicates the length in bytes of the VPD parameters that follow this field.

See [SPC] for further details.

11.5.3 Supported VPD Pages VPD page

The Supported VDP Pages VPD page contains a list of the VPD page codes supported by the logical unit (see Table 11-70).

Table 11-70— Supported VPD Pages VPD page

Bit / Byte	7	6	5	4	3	2	1	0
0	PERIPHERAL QUALIFIER			PERIPHERAL DEVICE TYPE				
1	PAGE CODE (00h)							
2	(MSB)	PAGE LENGTH (n-3)						
3								(LSB)
4	Supported VPD page list							
n								

The supported VPD page list shall contain a list of all VPD page codes implemented by the logical unit in ascending order beginning with page code 00h.

11.5.4 Mode Page Policy VPD page

The Mode Page Policy VPD page (see Table 11-71) indicates which mode page policy is in effect for each mode page supported by the logical unit.

Table 11-71 — Mode Page Policy VPD page

Bit / Byte	7	6	5	4	3	2	1	0
0	PERIPHERAL QUALIFIER			PERIPHERAL DEVICE TYPE				
1	PAGE CODE (87h)							
2	(MSB)			PAGE LENGTH (n-3)				
3								(LSB)
	Mode page policy descriptor list							
4				Mode page policy descriptor [first]				
7								
	…							
n-3				Mode page policy descriptor [last]				
n								

Each mode page policy descriptor (see Table 11-72) contains information describing the mode page policy for one or more mode pages or subpages.

Table 11-72 — Mode page policy descriptor

Bit / Byte	7	6	5	4	3	2	1	0
0	Reserved		POLICY PAGE CODE					
1	POLICY SUBPAGE CODE							
2	MLUS	Reserved					MODE PAGE POLICY = 00b	
3	Reserved							

The POLICY PAGE CODE field and POLICY SUBPAGE CODE field indicate the mode page and subpage to which the descriptor applies. See [SPC] for further details.

11.5.4 Mode Page Policy VPD page (cont'd)

If more than one logical unit are configured in the device, a multiple logical units share (MLUS) bit set to one indicates the mode page and subpage identified by the POLICY PAGE CODE field and POLICY SUBPAGE CODE field is shared by more than one logical unit.

A MLUS bit set to zero indicates the logical unit maintains its own copy of the mode page and subpage identified by the POLICY PAGE CODE field and POLICY SUBPAGE CODE field.

Table 11-73 describes the mode page policies.

Table 11-73 — MODE PAGE POLICY field

Code	Description
00b	Shared
01b	Per target port
10b	Obsolete
11b	Per I_T nexus

NOTE This standard defines only one target port and one initator port.

MODE PAGE POLICY field shall be set to zero (Shared).

See [SPC] for further details about Mode Page Policy VPD page.

12 UFS SECURITY

This section summarizes UFS device security features and the implementation details. These features include: Secure mode operation, data and register protection, RPMB and reset.

12.1 UFS Security Feature Support Requirements

The security features defined in this standard are mandatory for all devices.

The following security features are defined: replay protected memory block (RPMB), secure mode and different types of logical unit write protection.

12.2 Secure Mode

12.2.1 Description

UFS devices will be used to store user's personal and/or corporate data information. The UFS device provides a way to remove the data permanently from the device when requested, ensuring that it cannot be retrieved using reverse engineering on the memory device.

The UFS device shall support a secure and insecure mode of operation. In the secure mode all operations that result in the removal or retiring of information on the device will purge this information in a secure manner, as outlined in 12.2.2.1, Secure Removal.

The secure mode is applied at the logical unit level, so different logical unit may have different secure modes.

12.2.2 Requirements

12.2.2.1 Secure Removal

The way in which data is removed securely from the device is dependent on the type of memory technology that is used to implement the UFS device. Three common methods that apply to most memory types implemented at the time of this spec are:

1) The device controller shall issue an erase operation to the addressed location.
2) The device controller shall overwrite the addressed locations with a single character and erase the device.
3) The device controller shall overwrite the addressed locations with a character, its complement, then a random character

UFS devices shall support at least one secure removal method.

12.2.2.2 Erase Operation

Erase is an operation that moves data from the mapped address space to the unmapped address space. The regions in the mapped address space where erase was applied will be set to the erased value of zero. This operation places no requirement on what the device is required to do with the data in the unmapped address space. After an erase is executed, software on the host should not be able to retrieve the erased data.

The minimum data range that an erase operates on is the smallest region that can be written.

12.2.2.3 Discard Operation

Discard is a non-secure variant of the erase functionality. The distinction between discard and erase is the device behavior where the device is not required to guarantee that host would not retrieve the original data from one or more LBA’s that were marked for discard when a read operation is directed to the LBA’s.

12.2.2.4 Purge Operation

The Purge operation operates on the unmapped address space. When the operation is executed it results in removing all the data from the unmapped address space. This is done in accordance with the bSecureRemovalType parameter value of the Device Descriptor. This mode allows the host system to protect against die level attacks.

12.2.3 Implementation

12.2.3.1 Erase

The erase functionality is implemented using the UNMAP command and it is enabled if the bProvisioningType parameter in the Unit Descriptor is set to 03h (TPRZ = 1).

The device behavior shall comply with the UNMAP definition in [SBC] when the TPRZ bit in the READ CAPACITY(16) parameter data is set to one.

As defined in [SBC],

- The UNMAP command causes a mapped LBA to transition from mapped state to deallocated state if an unmap operation completes without error.
- Since the TPRZ bit is set to one if the erase functionality is enabled, a READ command specifying a deallocated LBA shall return zero.
- The device server may maintain a deallocated LBA in deallocated state until a write operation specifying that LBA is completed without error.
- Or, the device server may transition a deallocated LBA from deallocated state to mapped state at any time (autonomous state transition). For UFS, if TPRZ bit is set to one and an autonomous transition to the mapped state occurs, the LBA shall be mapped to a physical block(s) containing data with all bits set to zero.

LBA's to be erased may be aligned to multiples of the dEraseBlockSize parameter value, where it is possible, to minimize performance impact. dEraseBlockSize is a parameter included in the Unit Descriptor.

12.2.3.2 Discard

The discard functionality is implemented using the UNMAP command and it is enabled if the bProvisioningType parameter in the Unit Descriptor is set to 02h (TPRZ = 0). The device behavior shall comply with the UNMAP definition in [SBC] when the TPRZ bit in the READ CAPACITY(16) parameter data is set to zero.

As defined in [SBC],

- The UNMAP command causes a mapped LBA to transition from mapped state to deallocated state if an unmap operation completes without error.
- Since the TPRZ bit is set to zero if the discard functionality is enabled, a READ command specifying a deallocated LBA may return any data..
- The device server may maintain a deallocated LBA in deallocated state until a write operation specifying that LBA is completed without error.
- Or, the device server may transition a deallocated LBA from deallocated state to mapped state at any time (autonomous state transition). For UFS, if TPRZ bit is set to zero and an autonomous transition to the mapped state occurs, the LBA shall be mapped to a physical block(s) containing any data including the original data before UNMAP operation.

LBA's to be discarded may align to multiples of the dEraseBlockSize where possible to minimize performance impact. dEraseBlockSize is a parameter included in the Unit Descriptor.

12.2.3.3 Purge operation

The purge operation is implemented via Query Functions with Attributes and Flags. In particular, the fPurgeEnable flag allows to enable or disable the execution of a purge operation, and the bPurgeStatus attribute provides information about the operation status.

- fPurgeEnable flag
  - Write only volatile flag, set to zero after power on or reset.
  - Purge operation is enabled when this flag is equal to one, otherwise it is disabled.
  - This flag can only be set when the command queue of all logical units are empty.
  - This flag is automatically cleared by the UFS device when the operation completes or an error condition occurs.
  - This flag can be cleared by the host to interrupt an ongoing purge operation.
- bPurgeStatus attribute
  - Read only attribute.
  - This attribute can be set to one of the following values:
    - 00h: Idle (purge operation disabled).
    - 01h: Purge operation in progress.
    - 02h: Purge operation stopped prematurely by the host.
    - 03h: Purge operation completed successfully.
    - 04h: Purge operation failed due to logical unit queue not empty
    - 05h: Purge operation general failure.

    Other values are reserved and shall not be set.
  - bPurgeStatus is set to 00h (Idle) after power on or reset.
  - When the host enables the purge operation setting fPurgeEnable flag to one, and if all logical unit command queue are empty, the bPurgeStatus will be set to 01h to indicate that the purge operation is in progress. The bPurgeStatus shall be set to 03h if the operation is completed successfully, or to 05h if a failure occurred.
  - If the host attempts to enable the purge operation when there is at least one logical unit with command queue not empty, the setting of fPurgeEnable flag shall fail, Query Response field in the QUERY RESPONSE UPIP shall be set to FFh (“General Failure”), the purge operation shall not start, and the bPurgeStatus shall be set to 04h.
  - If an ongoing purge operation is interrupted by the host setting the fPurgeEnable flag to zero, the bPurgeStatus shall be set to 02h.
  - When the bPurgeStatus is equal to the values 02h, 03h, 04h or 05h, the bPurgeStatus shall be automatically cleared to 00h (Idle) the first time that it is read. The bPurgeStatus values of 00h and 01h shall not be modified as a result of a read.

12.2.3.3 Purge operation (cont'd)

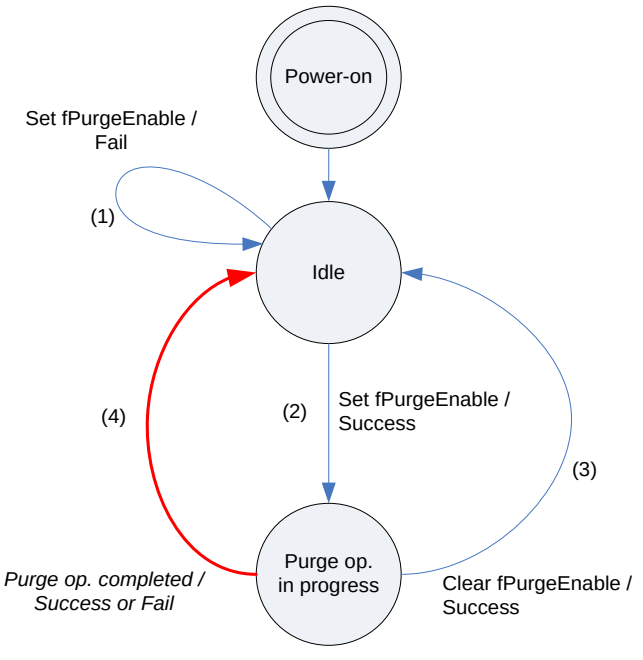
- If a purge operation is in progress (bPurgeStatus = 01h) commands sent to any logical units or to the RPMB well known logical unit will fail. The device shall return the sense key "NOT READY" to show that the command failed because a purge operation was in progress. Descriptors, attributes and flags may be read when a purge operation is in progress, while only fPurgeEnable flag may be written. A query request to write descriptors, attributes or flags (except fPurgeEnable) shall be terminated with Query Response field set to "General Failure".
- If the host needs to execute a command urgently when a purge operation is in progress, it may interrupt the purge operation. In particular, before issuing any command, the host sets fPurgeEnable flag to zero, waits until the device interrupts the operation, and then sets the bPurgeStatus attribute to 02h (purge operation stopped prematurely).
- If a power failure occurs the fPurgeEnable flag and bPurgeStatus attribute shall be reset to zero. In this case the device will not indicate that operation failed.

Figure 12-1 shows the Purge operation state machine. There are two states: "Idle" and "Purge Op. in progress".

After power on, the purge operation state is "Idle", and the purge operation is disabled.

To enable the execution of a purge operation, the host sets fPurgeEnable flag to one sending a QUERY REQUEST UPIU. If the setting is executed successfully, the state will transition to "Purge Op. in progress" and the purge operation will start (bPurgeStatus = 01h). If there is a least one logical unit with command queue not empty, the setting of fPurgeEnable flag shall fail, the purge operation shall not start, the state shall remain "Idle", and bPurgeStatus shall be set to 04h.

When the purge operation is completed, the state will transition automatically to "Idle", and bPurgeStatus shall be set to 03h if the operation is completed successfully, or 05h in case of failure.



State	bPurgeStatus
Idle	00h, 02h, 03h, 04h, 05h
Purge op. in progress	01h

Transition	Description
1	Set fPurgeEnable and LU command queues not empty
2	Set fPurgeEnable and LU command queues empty
3	Clear fPurgeEnable
4	Automatic transition when the purge operation completes successfully or with failure

NOTE 1 On each transition the input event (triggering the state transition) and the output of the transition itself are mentioned.

Figure 12-1 — Purge operation state machine

The host may interrupt an ongoing purge operation clearing the fPurgeEnable flag, when the operation has been interrupted the state will transition to "Idle" and bPurgeStatus will be set to 02h.

12.2.3.4 Wipe Device

The wipe device operation is fulfilled issuing the FORMAT UNIT command to all enabled logical units. If the logical unit is write protected using one of the methods described in 12.3, Device Data Protection, or if the SWP bit in Control Mode Page is one, then the FORMAT UNIT command shall fail and the content of the medium shall not be altered.

A FORMAT UNIT command sent to the Device well known logical unit requests the device format all enabled logical units except the RPMB well known logical unit. If any logical unit is write protected when the FORMAT UNIT command is issued to Device well known logical unit, the FORMAT UNIT command shall fail and the content of the medium shall not be altered.

The fields of the FORMAT UNIT command should be set as described in the following:

- The Format data (FMTDATA) bit shall be set to zero to specify that no parameter list will be provided.
- The DEFECT LIST FORMAT shall be set to 000b.
- The format protection information (FMTPINFO) shall be set to 00b.
- The vendor specific byte shall be set to 00h.

The UFS device shall ignore CMPLST and LONGLIST bits since FMTDATA is set to zero.

12.2.3.5 bProvisioningType Parameter

Logical units can be configured in secure mode using bProvisioningType parameter of the Unit Descriptor. This parameter allows to enable thin provisioning and define TPRZ bit value in the READ CAPACITY parameter data.

The secure mode is enabled if thin provisioning is enabled and TPRZ bit is equal to one. In this mode all operations shall be performed using the mode defined by bSecureRemovalType parameter in the Device Descriptor. Only one type of removal type can be defined for an entire device.

bProvisioningType parameter can be set to the following values:

- 00h: to disable thin provisioning
- 02h: to enable thin provisioning and set TPRZ to zero
- 03h: to enable thin provisioning and set TPRZ to one

TPRZ bit value of zero indicates the device is in normal mode.

As all other Unit Descriptor configurable parameters, the bProvisioningType value is set writing the Configuration Descriptor. See 14.1.4.3, Configuration Descriptor, for details.

12.2.3.6 bSecureRemovalType Parameter

The bSecureRemovalType parameter within the Device Descriptor defines how information is removed from the physical memory during a Purge operation. This parameter may be set during system integration writing the Configuration Descriptors. bSecureRemovalType values are defined as follows:

- Value of ‘03h’ will result in the information being removed using a vendor defined mechanism.
- Value of ’02h’ will result in all information being removed by overwriting the addressed locations with a character, its complement, then a random character.
- Value of ‘01h’ will result in all information being removed by overwriting the addressed locations with a single character followed by an erase.
- Value of ‘00h’ will result in all information being removed by an erase of the physical memory (default).
- Other values are reserved for future use and shall not be set.

Device manufacturers are only required to support the mechanism required by their memory array technology.

For additional information please refer to the http://www.killdisk.com/dod.htm or to the following documents for more details:

- DoD 5220.22-M (http://www.dtic.mil/whs/directives/corres/pdf/522022m.pdf) and
- NIST SP 800-88 (http://csrc.nist.gov/publications/nistpubs/800-88/NISTSP800-88_rev1.pdf)

12.3 Device Data Protection

12.3.1 Description and Requirements

UFS device data content can be protected at the logical unit level. The following protection modes shall be available:

- Permanent write protection (permanent: once enabled it cannot be reversed)
- Power on write protection (write protection can be cleared with a power cycle or a hardware reset event)
- Secure write protection (write protection can only be configured and enabled/disabled using secure authenticated methods)

These modes of write protections are not implemented in the RPMB well known logical unit.

There shall also be a method to read the protection mode that is currently enabled for a logical unit.

12.3.2 Implementation

The protection mode can be defined at logical unit level configuring the bLUWriteProtect parameter of the Unit Descriptor. The write protection modes are encoded as shown below:

00h: Logical unit not write protected (or secure write protected if one or more secure write protect entry is set)

01h: Logical unit power on write protected

02h: Logical unit permanently write protected

A power on write protected logical unit (bLUWriteProtect = 01h) can be written only if fPowerOnWPEn flag is equal to zero. The fPowerOnWPEn flag is set to zero after a power cycle or hardware reset event, once it is set to one it cannot be toggled or cleared by the host.

The fPermanentWPEn flag shall be set to one to enable the write protection of all permanently write protected logical unit (bLUWriteProtect = 02h); logical unit can be written if fPermanentWPEn flag is equal to 0b. The fPermanentWPEn flag is write once: it cannot be toggled or cleared once it is set. The fPermanentWPEn flag shall be zero after device manufacturing.

LBA areas within logical units may be write protected using the secure write protection mode.

Secure write protect areas are configured setting the Secure Write Protect Configuration Block, and they may only be created in logical units configured as “not write protected” (bLUWriteProtect=00h).

One logical unit may have up to four secure write protect areas. However the total number of secure write protect areas in a device shall not be more than bNumSecureWPArea.

A secure write protect area can be written only if write protection is disabled in the related Secure Write Protect Entry (WPF flag = 0b). See 12.4.3.1 for more details.

It is recommended to set fPowerOnWPEn flag, fPermanentWPEn flag or WPF flag when all command queues are empty, and wait device query response before enqueuing write command.

If an LBA is write protected, then otherwise valid commands that request unmap, format or alteration of the medium related to that LBA shall be rejected with CHECK CONDITION status with the sense key set to DATA PROTECT.

12.4 RPMB

12.4.1 Introduction

A signed access to a Replay Protected Memory Block is provided. This function provides means for the system to store data to the specific memory area in an authenticated and replay protected manner. This is provided by first programming authentication key information to the UFS device memory (shared secret).

As the system cannot be authenticated yet in this phase the authentication key programming have to take in a secure environment like in an OEM production. Further on the authentication key is utilized to sign the read and write accesses made to the replay protected memory area with a Message Authentication Code (MAC).

Usage of random number generation and count register are providing additional protection against replay of messages where messages could be recorded and played back later by an attacker.

12.4.2 RPMB Well Known Logical Unit Description

The RPMB is contained in a unique well known logical unit whose size is defined in the RPMB Unit Descriptor. RPMB well known logical unit size shall be a multiple of 128 Kbytes, therefore its minimum size is 128 Kbytes. The contents of the RPMB well known logical unit can only be read or written via successfully authenticated read and write accesses. The data may be overwritten by the host but can never be erased.

All accesses to the RPMB will reference the specific RPMB well known logical unit number (W-LUN).

12.4.3 Requirements

12.4.3.1 RPMB Resources

- Authentication Key
  - Type: Write once, not erasable or readable
  - Size: 32 bytes
  - Description: Authentication key register which is used to authenticate accesses when MAC is calculated.
- Write Counter
  - Type: Read only
  - Size: 4 bytes
  - Description: Counter value for the total amount of successful authenticated data write requests made by the host. The initial value of this register after production is 0000 0000h. The value will be incremented by one automatically by the UFS device along with each successful programming access. The value cannot be reset. After the counter has reached the maximum value of FFFF FFFFh, it will not be incremented anymore (overflow prevention).
- RPMB Data Area
  - Type: Readable and writable
  - Size: Multiples of 128 Kbytes defined in RPMB Unit Descriptor
    - 128 Kbytes minimum, 16 Mbytes maximum.
  - Description: Data which can only be read and written via successfully authenticated read/write access. This data may be overwritten by the host but can never be erased.
- Secure Write Protect Configuration Block
  - Type: Readable and writable
  - Size : 256 Bytes
  - Description: This block is used for configuring secure write protect areas in logical units. There is one Secure Write Protect Configuration Block for each logical unit. Each Secure Write Protect Configuration Block has up to four Secure Write Protect Entries. Each entry represents one secure write protect area. If an entry is not used, then the related fields shall contain a value of zero. The Secure Write Protect Configuration Block is structured as shown in Table 12-1.

12.4.3.1 RPMB Resources (cont'd)

Table 12-1 — Secure Write Protect Configuration Block

Bit / Byte (1)	7	6	5	4	3	2	1	0
0 (228)	LUN							
1 (229)	DATA LENGTH							
2 (230)	Reserved							
…								
15 (243)								
16 (244)	Secure Write Protect Entry 0							
…								
31 (259)								
32 (260)	Secure Write Protect Entry 1							
…								
47 (275)								
48 (276)	Secure Write Protect Entry 2							
…								
63 (291)								
64 (292)	Secure Write Protect Entry 3							
…								
79 (307)								
80 (308)	Reserved							
..								
255 (483)								

NOTE 1 Values in parenthesis indicates the byte number in the RPMB Message Data Frame.

12.4.3.1 RPMB Resources (cont'd)

a) LUN

The LUN field indicates the logical unit to which secure write protection shall apply. Valid values are from 0 to the number of LU specified by bMaxNumberLU.

b) DATA LENGTH

The DATA LENGTH field specifies the length in bytes of the Secure Write Protect Entries (16 for one entry, 32 for two entries, etc.). In a write request, the device shall ignore the bytes from DATA LENGTH + 16 to 255 and set these bytes of the Secure Write Protect Configuration Block to zero.

c) Secure Write Protect Entry 0 to Entry 3

The Secure Write Protect Configuration Block may contain only the Entry 0, the Entries 0 and 1, the Entries 0, 1 and 2, or all four Entries.

Table 12-2 defines the structure of Secure Write Protect Entry.

Table 12-2 — Secure Write Protect Entry

Bit / Byte (1)	7	6	5	4	3	2	1	0
0	Reserved					WPT		WPF
1	Reserved							
2	Reserved							
3	Reserved							
4	(MSB)							
…				LOGICAL BLOCK ADDRESS				
11								(LSB)
12	(MSB)							
…				NUMBER OF LOGICAL BLOCKS				
15								(LSB)

12.4.3.1 RPMB Resources (cont'd)

d) WPT (Write Protect Type)

The write protect type field (WPT) specifies how WPF bit may be modified.

Table 12-3 — Write Protect Type field

Code	Definition
00b	NV-type WPF bit is persistent through power cycle and hardware reset. WPFlag value may only be changed writing to Secure Write Protect Configuration Block.
01b	P-type WPF bit is automatically cleared to 0b after power cycle or hardware reset.
10b	NV-AWP-type WPF bit is automatically set to 1b after power cycle or hardware reset.
11b	Reserved

e) WPF (Write Protect Flag)

0b : Secure Write Protection is disabled.

1b : Secure Write Protection is enabled.

A WPF set to one specifies that the logical unit shall inhibit alteration of the medium for LBA within the range indicated by LOGICAL BLOCK ADDRESS field and NUMBER OF LOGICAL BLOCKS field. Commands requiring writes to the medium shall be terminated with CHECK CONDITION status, with the sense key set to DATA PROTECT, and the additional sense code set to WRITE PROTECTED.

Logical units that contain cache shall write all cached logical blocks to the medium (e.g., as they would do in response to a SYNCHRONIZE CACHE command with the LOGICAL BLOCK ADDRESS field and the NUMBER OF LOGICAL BLOCKS field set to the values indicated in the Secure Write Protect Entry) prior to enabling the write protection.

A WPF bit set to zero specifies that the logical unit may allow writing to the medium, depending on other write inhibit mechanisms implemented by the logical unit.

WPF shall be set to zero after device manufacturing.

f) LOGICAL BLOCK ADDRESS

This field specifies the LBA of the first logical block of the Secure Write Protect area.

g) NUMBER OF LOGICAL BLOCKS

This field specifies the number of contiguous logical blocks that belong to the Secure Write Protect area.

If the NUMBER OF LOGICAL BLOCKS field is set to zero, then the secure write protection shall apply to the entire logical unit. In that case, only Entry-0 needs to be configured to enable secure write protection for the entire logical unit.

12.4.3.2 Algorithm and Key for MAC Calculation

The message authentication code (MAC) is calculated using HMAC SHA-256 as defined in [HMAC-SHA]. The HMAC SHA-256 calculation takes as input a key and a message. The resulting MAC is 256 bits (32 bytes), which are embedded in the data frame as part of the request or response.

The key used for the MAC calculation is always the 256-bit Authentication Key stored in the UFS device. The message used as input to the MAC calculation is the concatenation of the fields in the RPMB packet.

12.4.3.3 RPMB Message Components

Each RPMB message includes specific components. These components are displayed in Table 12-4.

Table 12-4 — RPMB Message Components

Component Name	Length	Request	Response	Description
Request Message Type	2 bytes	Yes	Yes	This component indicates the request message type. See 12.4.3.4.
Response Message Type	2 bytes	No	Yes	This component indicates the response message type. See 12.4.3.5.
Authentication Key	32 bytes	Yes	No	This component is used only when programming the Authentication Key.
MAC	32 bytes	Yes	Yes	Message Authentication Code
Result	2 bytes	No	Yes	This component provides the operation result. See12.4.3.6.
Write Counter	4 bytes	Yes	Yes	Total amount of successful authenticated data write operations.
Address	2 bytes	Yes	Yes	Logical block address of data to be programmed to or read from the RPMB.
Nonce	16 bytes	Yes	Yes	Random number generated by the host for the requests and copied to response by the RPMB engine.
Data	256 bytes	Yes	Yes	Data to be written or read by signed access.
Block Count	2 bytes	Yes	Yes	Number of 256-byte logical blocks requested to be read or programmed.

12.4.3.4 Request Message Types

The following request message types are defined to support RPMB. These messages are sent from the host to the device.

- Authentication Key programming request
- Write Counter read request
- Authenticated data write request
- Authenticated data read request
- Result read request
- Secure Write Protect Configuration Block write request
- Secure Write Protect Configuration Block read request

Table 12-5 defines the Request Message Type codes for the various messages.

Table 12-5 — Request Message Types

Code	Request Message Types
0001h	Authentication Key programming request
0002h	Write Counter read request
0003h	Authenticated data write request
0004h	Authenticated data read request
0005h	Result read request
0006h	Secure Write Protect Configuration Block write request
0007h	Secure Write Protect Configuration Block read request
Others	Reserved

12.4.3.5 Response Message Types

The following response message types are defined to support RPMB. These messages are sent from the device to the host.

- Authentication Key programming response
- Write Counter read response
- Authenticated data write response
- Authenticated data read response
- Secure Write Protect Configuration Block write response
- Secure Write Protect Configuration Block read response

Table 12-6 defines the Response Message Type codes for the various messages.

Table 12-6 — Response Message Types

Code	Response Message Types
0100h	Authentication Key programming response
0200h	Write Counter read response
0300h	Authenticated data write response
0400h	Authenticated data read response
0500h	Reserved
0600h	Secure Write Protect Configuration Block write response
0700h	Secure Write Protect Configuration Block read response
Others	Reserved

12.4.3.6 RPMB Operation Result

- Result component of an RPMB message is composed of two bytes. The most significant byte is reserved and shall be set to zero.
- Bit 7 of Result field shall indicate if the write counter has expired (i.e., reached its maximum value) or not
  - Value of one will represent an expired write counter
  - Value of zero will represent a valid write counter
- The other bits shall indicate the operation status
  - Operation Okay (00h)
  - General Failure (01h)
  - Authentication Failure (02h)
    - MAC comparison not matching, MAC calculation failure
  - Counter Failure (03h)
    - Counters not matching in comparison, counter incrementing failure
  - Address Failure (04h)
    - Address out of range, wrong address alignment
  - Write Failure (05h)
    - Data, Counter or Result write failure
  - Read Failure (06h)
    - Data, Counter or Result write failure
  - Authentication key not yet programmed (07h)
    - This value is the only valid result until the Authentication Key has been programmed, after which it can never occur again
  - Secure Write Protect Configuration Block access failure (08h).
    - Secure Write Protect Configuration read or write failure.
  - Invalid Secure Write Protect Block Configuration parameter (09h).
    - Invalid LUN (or logical unit not enabled), DATA LENGTH, LOGICAL BLOCK ADDRESS, NUMBER OF LOGICAL BLOCKS, or overlapped areas.
  - Secure Write Protection not applicable (0Ah).
    - Logical unit configured with other write protection modes (permanent or power-on)

Table 12-7 — Result data structure

Bit[15:8]	Bit[7]	Bit[6:0]
Reserved	Write Counter Status	Operation Status

12.4.3.6 RPMB Operation Result (cont'd)

Table 12-8 — Result code definition

Code	Description
0000h (0080h)	Operation OK
0001h (0081h)	General failure
0002h (0082h)	Authentication failure • MAC comparison not matching, MAC calculation failure
0003h (0083h)	Counter failure • Counters not matching in comparison, counter incrementing failure
0004h (0084h)	Address failure • Address out of range, wrong address alignment
0005h (0085h)	Write failure • Data / Counter / Result write failure
0006h (0086h)	Read failure • Data / Counter / Result read failure
0007h	Authentication Key not yet programmed. • This value is the only valid Result value until the Authentication Key has been programmed. Once the key is programmed, this value will no longer be used.
0008h (0088h)	Secure Write Protect Configuration Block access failure • Secure Write Protect Configuration read or write failure
0009h (0089h)	Invalid Secure Write Protect Block Configuration parameter • Invalid LUN or logical unit not enabled, DATA LENGTH, LOGICAL BLOCK ADDRESS, NUMBER OF LOGICAL BLOCKS, or overlapped areas
000Ah (008Ah)	Secure Write Protection not applicable • Logical unit configured with other write protection modes (permanent or power-on)
NOTE The values in parenthesis are valid when Write Counter has expired.	

12.4.4 Implementation

12.4.4.1 RPMB Message

An RPMB Message may be composed of one or more RPMB Message Data Frames.

RPMB Message Data Frame size is 512 bytes and it is organized as shown in Table 12-9.

Table 12-9 — RPMB Message Data Frame

Byte \ Bit	7	6	5	4	3	2	1	0
0	(MSB)							
195				Stuff Bytes				(LSB)
196	(MSB)							
227				Key / MAC				(LSB)
228				Data [255]				
483				Data [0]				
484	(MSB)							
499				Nonce				(LSB)
500	(MSB)							
503				Write Counter				(LSB)
504	(MSB)							
505				Address				(LSB)
506	(MSB)							
507				Block Count				(LSB)
508	(MSB)							
509				Result				(LSB)
510	(MSB)							
511				Request Message Type / Response Message Type				(LSB)

12.4.4.2 MAC Calculation

The key used for the MAC calculation is always the 256 bit Authentication Key stored in the device. Input to the MAC calculation is the concatenation of the fields in the RPMB Message Data Frames from byte 228 to byte 511 (stuff bytes and the MAC are excluded).

If RPMB Message is composed by several RPMB Message Data Frames then the input message to MAC is the concatenation of bytes [228:511] of each data frame in the order in which the data frames are sent. The MAC is valid only in the last data frame.

12.4.4.3 RPMB Message Data Frame Delivery

The RPMB messages are delivered using SCSI security protocol commands:

- SECURITY PROTOCOL IN is used to send request messages to the device
- SECURITY PROTOCOL OUT is used to request to the device the sending of response messages.

12.4.5 SECURITY PROTOCOL IN/OUT Commands

SECURITY PROTOCOL IN command and SECURITY PROTOCOL OUT command defined in [SPC] are used to encapsulate and deliver data packets of any security protocol between host and device without interpreting, dis-assembling or re-assembly the data packets for delivery.

The SECURITY PROTOCOL IN command and SECURITY PROTOCOL OUT command contain a SECURITY PROTOCOL field. A unique security protocol ID is assigned by T10 for JEDEC UFS application.

- SECURITY PROTOCOL = ECh (JEDEC Universal Flash Storage)

This standard assigns a unique identifier to the SECURITY PROTOCOL SPECIFIC field of both the SECURITY PROTOCOL IN command and SECURITY PROTOCOL OUT command.

- RPMB Protocol ID = 0001h

4658 **12.4.5.1 CDB format of SECURITY PROTOCOL IN/OUT commands**

4659 **Table 12-10 — CDB format of SECURITY PROTOCOL IN/OUT commands**

Byte \ Bit	7	6	5	4	3	2	1	0
0	OPERATION CODE (1)							
1	SECURITY PROTOCOL							
2	SECURITY PROTOCOL SPECIFIC							
3								
4	INC_512 = 0b	Reserved						
5	Reserved							
6	(MSB) ALLOCATION / TRANSFER LENGTH (2)							
9								(LSB)
10	Reserved							
11	CONTROL = 00h							

NOTE 1 OPERATION CODE = A2h for SECURITY PROTOCOL IN command, OPERATION CODE = B5h for SECURITY PROTOCOL OUT command.

NOTE 2 ALLOCATION LENGTH for SECURITY PROTOCOL IN command, TRANSFER LENGTH for SECURITY PROTOCOL OUT command.

4660 The RPMB well known logical unit shall support the following SECURITY PROTOCOL field values:

4661 • 00h: Security protocol information

4662 • ECh: JEDEC Universal Flash Storage (security protocol ID assigned for JEDEC UFS application)

4663 Other values are invalid.

4664 SECURITY PROTOCOL IN/OUT commands shall consider the unique Security Protocol ID assigned to
4665 JEDEC UFS application as the only valid Security Protocol ID.

4666 INC_512 bit shall be set to zero to specify that the ALLOCATION LENGTH or the TRANSFER
4667 LENGTH field expresses the number of bytes to be transferred.

4668 If the ALLOCATION LENGTH field in a SECURITY PROTOCOL IN command is not equal to an
4669 integer multiple of 512, then the command shall be terminated with CHECK CONDITION status.

4670 If the TRANSFER LENGTH field in a SECURITY PROTOCOL OUT command is not equal to an
4671 integer multiple of 512, then the command shall be terminated with CHECK CONDITION status.

4672

12.4.5.2 Security Protocol Information Description

As required by [SPC], the SECURITY PROTOCOL value of 00h (security protocol information) shall be supported if the SECURITY PROTOCOL IN command is supported by the device. The security protocol information security protocol (i.e., the SECURITY PROTOCOL field set to 00h in a SECURITY PROTOCOL IN command) is used to transfer security protocol related information from the logical unit.

When the SECURITY PROTOCOL field is set to 00h in a SECURITY PROTOCOL IN command, the two bytes SECURITY PROTOCOL SPECIFIC field shall contain a numeric value as defined in Table 12-11.

Table 12-11 — Security Protocol specific field values

Security Protocol Specific Field Value		Description	Support
CDB Byte 2	CDB Byte 3		
00h	00h	Supported security protocol list	Mandatory
00h	01h	Certificate data	Mandatory
Other values		Reserved	

12.4.5.3 Supported security protocols list description

According to [SPC], if the SECURITY PROTOCOL field is set to 00h and the SECURITY PROTOCOL SPECIFIC field is set to 0000h in a SECURITY PROTOCOL IN command, the parameter data shall have the format shown in Table 12-12.

Table 12-12 — Supported security protocols SECURITY PROTOCOL IN parameter data

Byte \ Bit	7	6	5	4	3	2	1	0
0	Reserved							
5								
6	(MSB)	SUPPORTED SECURITY PROTOCOL LIST LENGTH (m-7)						
7								(LSB)
8	SUPPORTED SECURITY PROTOCOL [first] (00h)							
	. .							
m	SUPPORTED SECURITY PROTOCOL [last]							
m+1	Pad Bytes (optional)							
n								

Security protocol information (00h) and the JEDEC Universal Flash Storage (ECh) are the only valid security protocol ID's supported by the RPMB well known logical unit, therefore Table 12-12 shall be implemented as defined in Table 12-13.

Table 12-13 — UFS Supported security protocols SECURITY PROTOCOL IN parameter data

Byte \ Bit	7	6	5	4	3	2	1	0
0	Reserved							
5								
6	(MSB)	0002h (SUPPORTED SECURITY PROTOCOL LIST LENGTH)						
7								(LSB)
8	00h (Security protocol information)							
9	ECh (JEDEC Universal Flash Storage)							
10	Pad bytes (optional)							
n								

12.4.5.4 Certificate data description

If the SECURITY PROTOCOL field is set to 00h and the SECURITY PROTOCOL SPECIFIC field is set to 0001h in a SECURITY PROTOCOL IN command, the parameter data shall have the format shown in Table 12-14.

Table 12-14 — Certificate data SECURITY PROTOCOL IN parameter data

Byte \ Bit	7	6	5	4	3	2	1	0
0	Reserved							
1								
2	(MSB) CERTIFICATE LENGTH (m-3)							
3								(LSB)
4	CERTIFICATE							
m								
m+1	Pad bytes (optional)							
n								

The Device Server does not have a certificate to transfer, the CERTIFICATE LENGTH field shall be set to 0000h. therefore Table 12-14 shall be implemented as defined in Table 12-15.

Table 12-15 — UFS certificate data SECURITY PROTOCOL IN parameter data

Byte \ Bit	7	6	5	4	3	2	1	0
0	Reserved							
1								
2	(MSB) 0000h (CERTIFICATE LENGTH)							
3								(LSB)
4	Pad bytes (optional)							
n								

12.4.6 RPMB Operations

12.4.6.1 Request Type Message Delivery

- Only one RPMB operation can be executed at any time.
- Host sends request type message to RPMB well known logical unit to request the execution of an operation by the RPMB well known logical unit.
- To deliver a request type message, the host sends a SECURITY PROTOCOL OUT command in a COMMAND UPIU in the command phase of a SCSI transaction.
- For an authenticated data write request, the data to be written into the RPMB data area is included in the request message. The maximum data size in a single Authenticated Data Write request is equal to bRPMB_ReadWriteSize × 256 bytes; multiple Authenticated Data Write operations should be executed if the desired data size exceeds this value.
- For SECURITY PROTOCOL OUT command, the Flags.W in the COMMAND UPIU is set to one since data is transferred from the host to the device.
- Table 12-16 defines the Expected Data Transfer Length field value in the COMMAND UPIU for the various cases.

Table 12-16 — Expected Data Transfer Length value for Request Type Messages

RPMB Data Frame	Value
Authentication Key programming request, Result read request, Write Counter read request, Authenticated data read request, Secure Write Protect Configuration Block write request, Secure Write Protect Configuration Block read request	512
Authenticated data write request	512 × Block Count
NOTE Block Count is equal to the data size divided by 256.	

- The device indicates to the host that it is ready to receive the request type message sending READY TO TRANSFER UPIU. If the Expected Data Transfer Length is 512 byte, then Data Buffer Offset field shall be set to a value of zero and Data Transfer Count field shall be set to a value of 512.
- The number of bytes requested in a single READY TO TRANSFER UPIU shall not be greater than the value indicated by bMaxDataOutSize attribute.
- In response to each READY TO TRANSFER UPIU, the host delivers the requested portion of the message sending DATA OUT UPIU. See 10.7.13 for details about data transfer.
- To complete the operation, the device returns a RESPONSE UPIU with the status of the operation in the status phase.

12.4.6.2 Response Type Message Delivery

- Host requests the RPMB well known logical unit to send a response type message to read the result of a previous operation, to read the Write Counter, to read data from the RPMB data area, or to read the contents of a Secure Write Protect Configuration Block.
- To request the delivery of a response type message, the host sends a SECURITY PROTOCOL IN command in a COMMAND UPIU in the command phase of a SCSI transaction.
- For an authenticated data read the data from the RPMB data area is included in the response message.
- For SECURITY PROTOCOL IN command, the Flags.R in the COMMAND UPIU is set to one since data is transferred from the device to the host.
- Table 12-17 defines the Expected Data Transfer Length field value in the COMMAND UPIU for the various cases.

Table 12-17 — Expected Data Transfer Length value for Response Type Messages

RPMB Data Frame	Value
Response for Result read request – Authentication Key programming response – Authenticated data write response – Secure Write Protect Configuration Block write response Write Counter read response Secure Write Protect Configuration Block read response	512
Authenticated data read response	512 × Block Count
NOTE Block Count is equal to the data size divided by 256.	

- The device returns the result or data requested in the RPMB message. The RPMB message is delivered by sending one or more DATA IN UPIU in the data phase.
- The data size in DATA IN UPIU shall not exceed the value indicated by bMaxDataInSize attribute.
- To complete the operation, the device sends a RESPONSE UPIU with the status of the operation in the status phase.

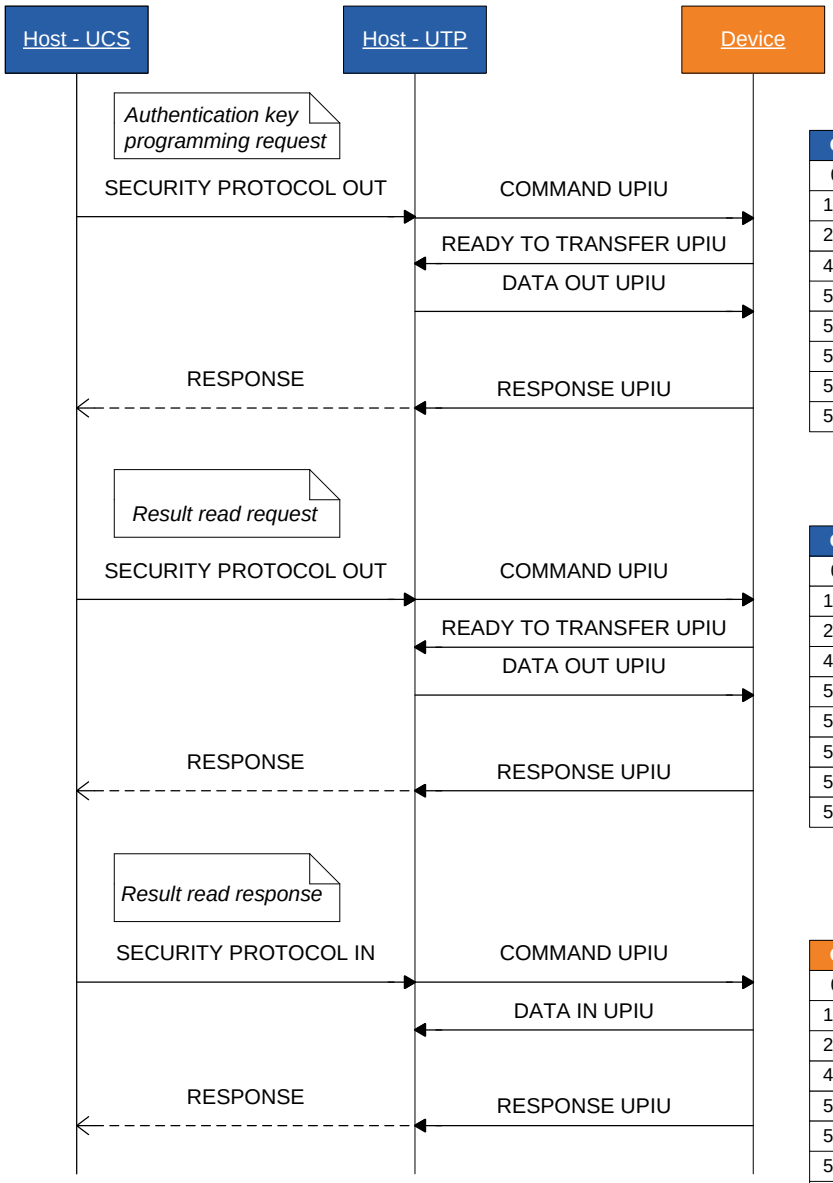
12.4.6.3 Authentication Key Programming

- The Authentication Key programming is initiated by a SECURITY PROTOCOL OUT command
- The RPMB data frame delivered from the host to the device includes the Request Message Type = 0001h and the Authentication Key
- The device returns GOOD status in status response when Authentication Key programming is completed
- Host initiates the Authentication Key programming verification process by issuing a SECURITY PROTOCOL OUT command with delivery of a RPMB data frame contains the Request Message Type = 0005h
- The device returns GOOD status in status response when the verification result is ready for retrieval
- Host retrieves the verification result by issuing a SECURITY PROTOCOL IN command
- Device returns the RPMB data frame containing the Response Message Type = 0100h and the Result code

  If programming of Authentication Key fails then returned result is 0005h (Write failure). If some other error occurs during Authentication Key programming then returned result is 0001h (General failure).

Access to RPMB data area is not possible before the Authentication Key is programmed. The state of the device can be checked by trying to write/read data to/from the RPMB data area: if the Authentication Key is not programmed then the Result field in the response message will be set to 0007h (Authentication Key not yet programmed).

12.4.6.3 Authentication Key Programming (cont'd)



Authentication Key Programming Request

Offset	Field Name	Value
0 :195	Stuff bytes	0...00h
196:227	MAC / Key	*Key*
228:483	Data	0...00h
484:499	Nonce	0...00h
500:503	Write counter	0...00h
504:505	Address	0000h
506:507	Block Count	0000h
508:509	Result	0000h
510:511	Req./Resp.	0001h

Result Read Request

Offset	Field Name	Value
0 :195	Stuff bytes	0...00h
196:227	MAC / Key	0...00h
228:483	Data	0...00h
484:499	Nonce	0...00h
500:503	Write counter	0...00h
504:505	Address	0000h
506:507	Block Count	0000h
508:509	Result	0000h
510:511	Req./Resp.	0005h

Result Read Response

Offset	Field Name	Value
0 :195	Stuff bytes	0...00h
196:227	MAC / Key	0...00h
228:483	Data	0...00h
484:499	Nonce	0...00h
500:503	Write counter	0000 0000h
504:505	Address	0000h
506:507	Block Count	0000h
508:509	Result	*Result code*
510:511	Req./Resp.	0100h

Figure 12-2 — Authentication Key Programming Flow

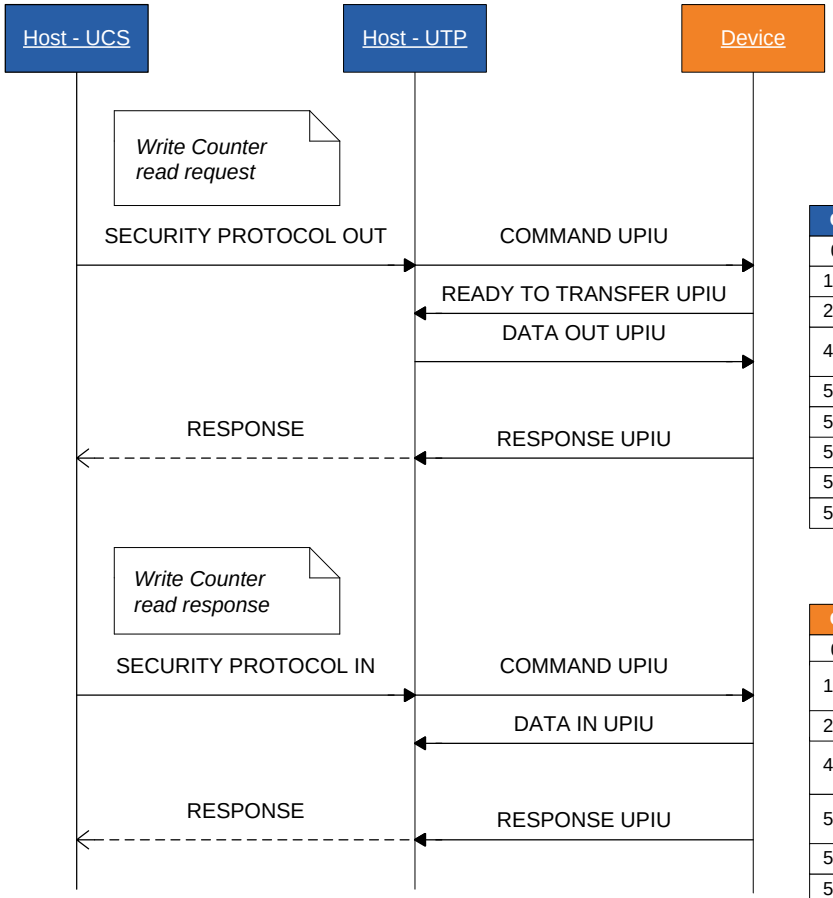
12.4.6.4 Read Counter Value

- The Read Counter Value sequence is initiated by a SECURITY PROTOCOL OUT command.
- The RPMB data frame delivered from the host to the device includes the Request Message Type = 0002h and the Nonce.
- When the host received a GOOD status in the status response from the device, it sends a SECURITY PROTOCOL IN command to the device to retrieve the write counter value.
- The device returns a RPMB data frame with Response Message Type = 0200h, a copy of the Nonce received in the request, the Write Counter value, the MAC and the Result.

If reading of the counter value fails then returned result is 0006h (Read failure).

If some other error occurs then Result is 0001h (General failure).

If counter has expired also bit 7 is set to 1 in returned results (Result values 0080h, 0086h and 0081h, respectively).



Write Counter Read Request

Offset	Field Name	Value
0 :195	Stuff bytes	0...00h
196:227	MAC / Key	0...00h
228:483	Data	0...00h
484:499	Nonce	*Nonce from the host*
500:503	Write counter	0...00h
504:505	Address	0000h
506:507	Block Count	0000h
508:509	Result	0000h
510:511	Req./Resp.	0002h

Write Counter Read Response

Offset	Field Name	Value
0 :195	Stuff bytes	0...00h
196:227	MAC / Key	MAC from the device
228:483	Data	0...00h
484:499	Nonce	*Copy of the Nonce*
500:503	Write counter	*Counter Value*
504:505	Address	0000h
506:507	Block Count	0000h
508:509	Result	*Result code*
510:511	Req./Resp.	0200h

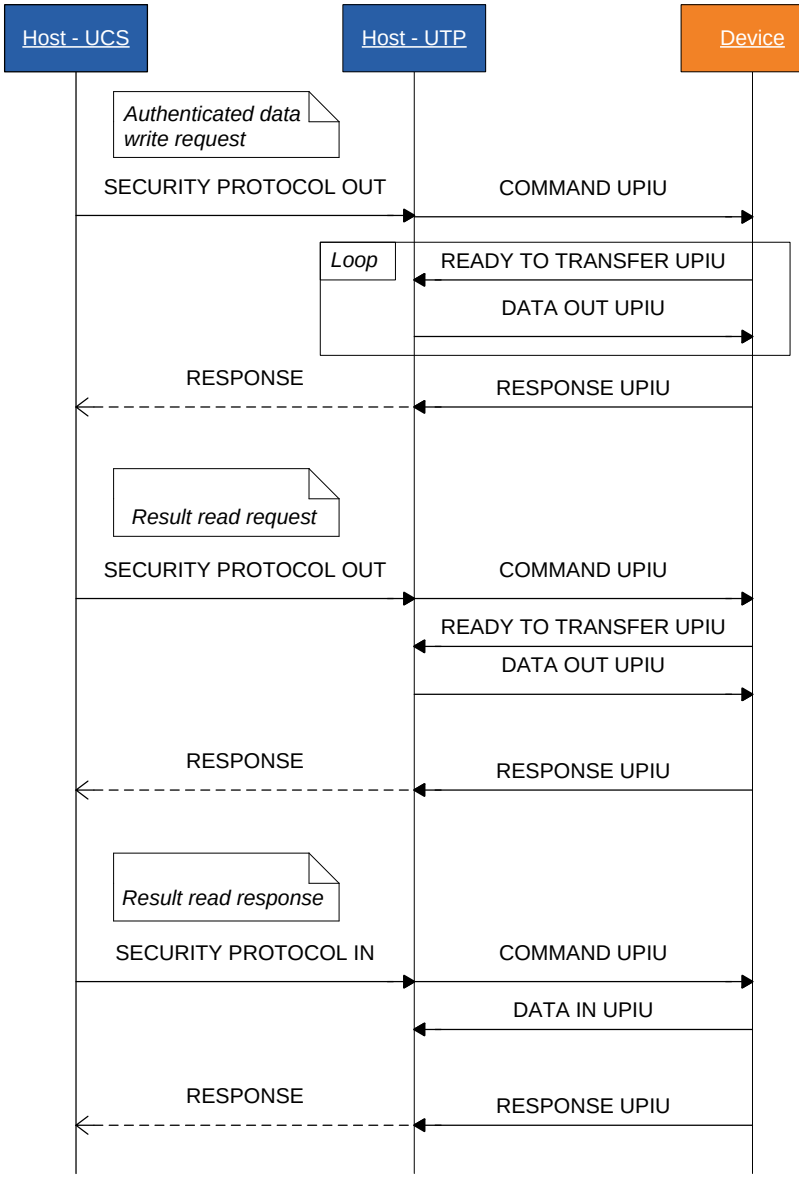
Figure 12-3 — Read Counter Value Flow

12.4.6.5 Authenticated Data Write

- The Authenticated Data Write sequence is initiated by a SECURITY PROTOCOL OUT command.
- The RPMB message delivered from the host to the device is composed of one or more RPMB message data frames, each of which includes: Request Message Type = 0003h, Block Count, Address, Write Counter, Data and MAC.
    - When the device receives the RPMB message, it first checks whether the write counter has expired. If the write counter is expired then the device sets the Result to 0085h (write failure, write counter expired). No data is written to the RPMB data area.
    - Next the address is checked. If the Address value is equal or greater than qLogicalBlockCount parameter value in the RPMB Unit Descriptor, then the Result is set to 0004h (address failure). No data is written to the RPMB data area.
    - If the Address value plus the Block Count value is greater than qLogicalBlockCount parameter value, then the Result is set to 0004h (address failure). No data is written to the RPMB data area.
    - If the Block Count indicates a value greater than bRPMB_ReadWriteSize, then the authenticated data write operation fails and the Result is set to 0001h (General failure).
    - If the write counter was not expired then the device calculates the MAC of request type, block count, write counter, address and data, and compares this with the MAC in the request. If the two MAC's are different, then the device sets the Result to 0002h (authentication failure). No data is written to the RPMB data area.
    - If the MAC in the request and the calculated MAC are equal then the device compares the write counter in the request with the write counter stored in the device. If the two counters are different then the device sets the Result to 03h (counter failure). No data is written to the RPMB data area.
    - If the MAC and write counter comparisons are successful then the write request is considered to be authenticated. The data is written to the address indicated in the request.
    - The write counter is incremented by one if the write operation is successfully executed.
    - If write fails then returned result is 0005h (write failure).
    - If some other error occurs during the write procedure then returned result is 0001h (General failure).
    - In an authenticated data write request with Block Count greater than one
        - the MAC is included only in the last RPMB message data frame. The MAC field is zero in all previous data frames. The device behavior is undefined if a MAC field is non-zero in any but the last RPMB message data frame.
        - In each data frame, the write counter indicates the current counter value, the address is the start address of the full access (not address of the individual logical block) and the block count is the total count of the blocks (not the block numbers).
- When the authenticated data write operation is completed, the device may return GOOD status in response to the SECURITY PROTOCOL OUT command regardless of whether the Authenticated data write was successful or not.
- The Result field in a response type RPMB data frame provides the operation result. To retrieve it, the host sends a SECURITY PROTOCOL OUT command delivering an RPMB data frame with the Request Message Type = 0005h.

12.4.6.5 Authenticated Data Write (cont'd)

- The device returns GOOD status when the result is ready for retrieval.
- The host requests the device to transfer the RPMB data frame sending a SECURITY PROTOCOL IN command.
- Device returns the RPMB data frame containing the Response Message Type = 0300h, the counter value (incremented if the write operation is successfully executed), the address received in the Authenticated data write request, the MAC and result of the authenticated data write operation.



Authenticated Data Write Request

Offset	Field Name	Value
0 :195	Stuff bytes	0...00h
196:227	MAC / Key	*MAC from the host*
228:483	Data	*Data*
484:499	Nonce	0...00h
500:503	Write counter	*Current Counter value*
504:505	Address	Address
506:507	Block Count	Number of 256B blocks
508:509	Result	0000h
510:511	Req./Resp.	0003h

Result Read Request

Offset	Field Name	Value
0 :195	Stuff bytes	0...00h
196:227	MAC / Key	0...00h
228:483	Data	0...00h
484:499	Nonce	0...00h
500:503	Write counter	0...00h
504:505	Address	0000h
506:507	Block Count	0000h
508:509	Result	0000h
510:511	Req./Resp.	0005h

Result Read Response

Offset	Field Name	Value
0 :195	Stuff bytes	0...00h
196:227	MAC / Key	*MAC from the device*
228:483	Data	0...00h
484:499	Nonce	0...00h
500:503	Write counter	*New Counter value* (1)
504:505	Address	Address
506:507	Block Count	0000h
508:509	Result	Result code
510:511	Req./Resp.	0300h

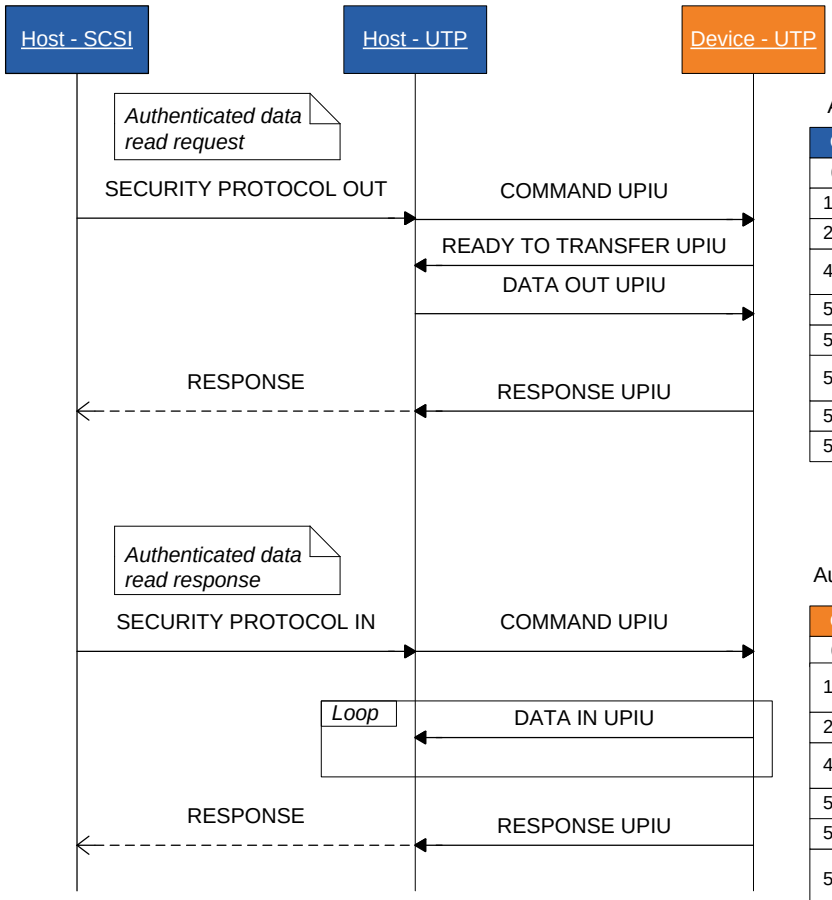
NOTE 1 The Write Counter is incremented only if the operation was successfully completed

Figure 12-4 — Authenticated Data Write Flow

12.4.6.6 Authenticated Data Read

- The Authenticated Data Read sequence is initiated by a SECURITY PROTOCOL OUT command.
- The RPMB data frame delivered from the host to the device includes the Request Message Type = 0004h, the nonce, the data address, and the block count.
    - When the device receives this request it first checks the address. If the Address value is equal or greater than qLogicalBlockCount parameter value in the RPMB Unit Descriptor, then Result is set to 0004h (address failure). The data read is not valid.
    - If the Address value plus the Block Count value is greater than qLogicalBlockCount parameter value, then the Result is set to 0004h (address failure). No data is read from the RPMB data area.
    - After successful data fetch the MAC is calculated from response type, nonce, address, data and result. If the MAC calculation fails then returned result is 0002h (Authentication failure).
- If the SECURITY PROTOCOL OUT command completes with GOOD status, the host sends a SECURITY PROTOCOL IN command to retrieve the data.
- The device returns a RPMB data frame with Response Message Type (0400h), the block count, a copy of the nonce received in the request, the address received in the Authenticated data read request, the data, the MAC and the result.
- In an authenticated data read response with Block Count greater than one,
    - the MAC is included only in the last RPMB message data frame, The MAC field is zero in all previous data frames.
    - In each data frame, the Nonce contains a copy of the received nonce, the address is the start address of the full access (not address of the individual logical block) and the block count is the total count of the blocks (not the sequence number of blocks).
- When the authenticated data read operation is completed, the device may return GOOD status in response to the SECURITY PROTOCOL IN command regardless of whether the Authenticated data read was successful or not.
- If data fetch from addressed location inside device fails then returned result is 0006h (Read failure). If some other error occurs during the read procedure then returned result is 0001h (General failure).

12.4.6.6 Authenticated Data Read (cont'd)



Authenticated Data Read Request

Offset	Field Name	Value
0 :195	Stuff bytes	0...00h
196:227	MAC / Key	0...00h
228:483	Data	0...00h
484:499	Nonce	*Nonce from the host*
500:503	Write counter	0...00h
504:505	Address	Address
506:507	Block Count	Number of 256B blocks
508:509	Result	0000h
510:511	Req./Resp.	0004h

Authenticated Data Read Response

Offset	Field Name	Value
0 :195	Stuff bytes	0...00h
196:227	MAC / Key	*MAC from the device*
228:483	Data	*Data*
484:499	Nonce	*Copy of the Nonce*
500:503	Write counter	0...00h
504:505	Address	Address
506:507	Block Count	Number of 256B blocks
508:509	Result	Result code
510:511	Req./Resp.	0400h

Figure 12-4 — Authenticated Data Read Flow

12.4.6.7 Authenticated Secure Write Protect Configuration Block Write

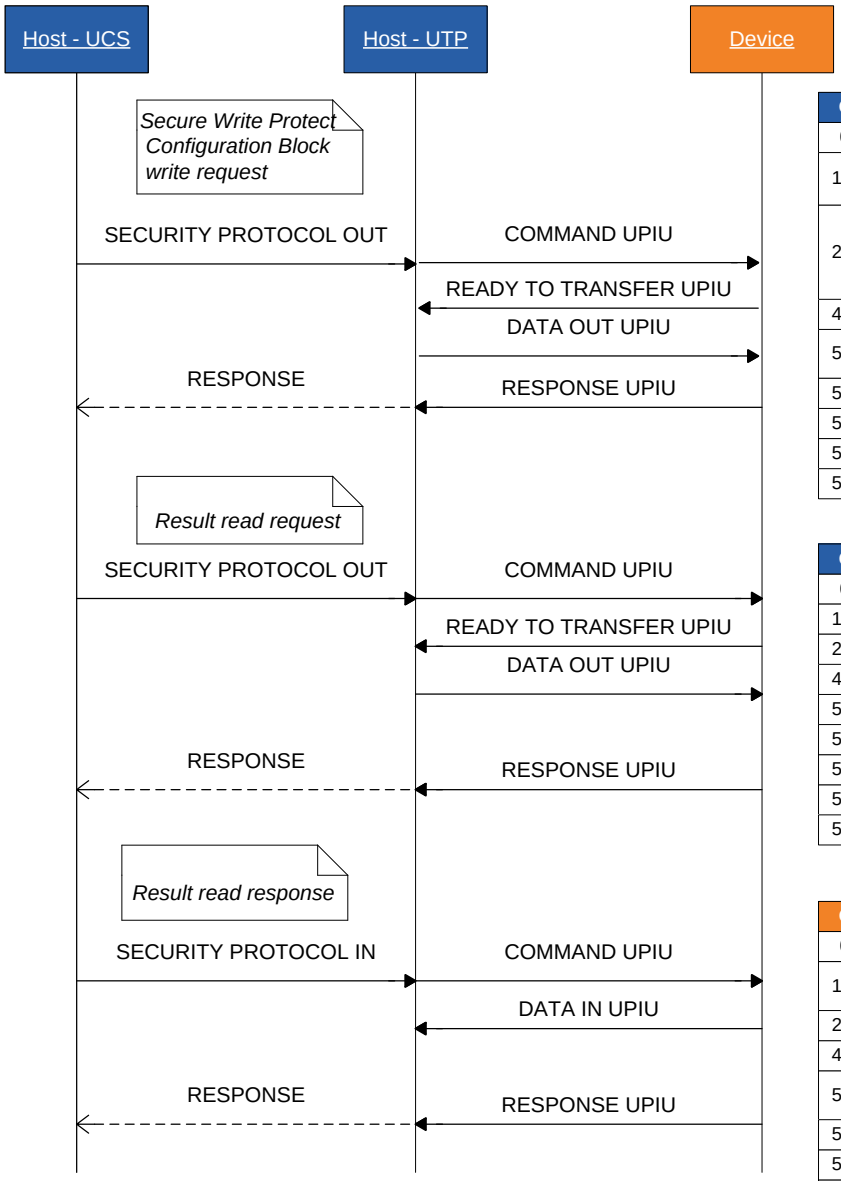
- The Authenticated Secure Write Protect Configuration Block write sequence is initiated by a SECURITY PROTOCOL OUT command.
- If the INC_512 bit and TRANSFER LENGTH field in the SECURITY PROTOCOL OUT command do not indicate 512 bytes, then the command shall be terminated with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.
- The SECURITY PROTOCOL OUT command delivers a single RPMB message data frame which contains the Secure Write Protect Configuration Block in the Data field. The Secure Write Protect Configuration Block is specific of the logical unit indicated by the LUN field (byte 228 of the data frame).
- The other fields of RPMB data frame are set as specified in the following: Request Message Type = 0006h, Result = 0000h, Block Count = 0001h, Address = 0000h, Write Counter = current counter value, Nonce = 00...0h, and MAC = see 12.4.4.2.
    - When the device receives the RPMB message data frame, it first checks whether the write counter has expired. If the write counter is expired then the device sets the result to 0085h (write failure, write counter expired). The Secure Write Protect Configuration Block is not updated.
    - If the write counter was not expired, then the device calculates the MAC of request type, block count, write counter, address and data, and compares this with the MAC in the request. If the two MAC's are different, then the device sets the result to 0002h (authentication failure). The Secure Write Protect Configuration Block is not updated.
    - If the MAC in the request and the calculated MAC are equal, then the device compares the write counter in the request with the write counter stored in the device. If the two counters are different then the device sets the result to 03h (counter failure). The Secure Write Protect Configuration Block is not updated.
    - If the MAC and write counter comparisons are successful then the write request is considered to be authenticated.
    - If the LUN field indicates a logical unit with bLUWriteProtect set to a value different from zero, then the device sets the result to 000Ah (Secure Write Protection not applicable). The Secure Write Protect Configuration Block is not updated.
    - The device sets the result to 0009h (Invalid Secure Write Protect Block Configuration parameter) and it does not update the Secure Write Protect Configuration Block if one or more of the following conditions occurs.
        - the LUN field is invalid: greater than the value specified by bMaxNumberLU or if the logical unit is not enabled (bLUEnable=00h).
        - the DATA LENGTH is set to a value different from the following ones: 16, 32, 48, 64.
        - the LOGICAL BLOCK ADDRESS in a Secure Write Protect entry exceeds the logical unit capacity,
        - the LOGICAL BLOCK ADDRESS plus the NUMBER OF LOGICAL BLOCKS in a Secure Write Protect entry exceeds the logical unit capacity,
        - two or more Secure Write Protect entries specify overlapping areas,
        - with this request, the number of Secure Write Protect areas set in the entire device is increased to a value greater than what indicated by bNumSecureWPArea.

12.4.6.7 Authenticated Secure Write Protect Configuration Block Write (cont'd)

- If some other error occurs during the write procedure then returned result is 0008h (Secure Write Protect Configuration Block access failure). The Secure Write Protect Configuration Block is not updated.
- If no error occurred, then the Secure Write Protect Configuration Block is updated overwriting the former configuration and the write counter is incremented by one.

- The device may return GOOD status in response to the SECURITY PROTOCOL OUT command regardless of whether the Authenticated Secure Write Protect Configuration Block Write was successful or not.
- The successfulness of the programming of the data shall be checked by the host by reading the result register of the RPMB. Host initiates the Authenticated Write Protected Area updating verification process by issuing a SECURITY PROTOCOL OUT command with delivery of a RPMB data frame contains the Request Message Type = 0005h.
- The device returns "Good" status in status response when the verification result is ready for retrieval.
- Host retrieves the verification result by issuing a SECURITY PROTOCOL IN command.
- Device returns the RPMB data frame containing the Response Message Type = 0600h, the incremented counter value, the MAC and result of the Authenticated Secure Write Protect Configuration Block write operation.

4930 **12.4.6.7 Authenticated Secure Write Protect Configuration Block Write (cont'd)**



Secure Write Protect Configuration Block write request

Offset	Field Name	Value
0 :195	Stuff bytes	0...00h
196:227	MAC / Key	*MAC from the host*
228:483	Data	*LUN, DATA LENGTH, Secure Write Protect Entries*
484:499	Nonce	0...00h
500:503	Write counter	*Current Counter value*
504:505	Address	0000h
506:507	Block Count	0001h
508:509	Result	0000h
510:511	Req./Resp.	0006h

Result Read Request

Offset	Field Name	Value
0 :195	Stuff bytes	0...00h
196:227	MAC / Key	0...00h
228:483	Data	0...00h
484:499	Nonce	0...00h
500:503	Write counter	0...00h
504:505	Address	0000h
506:507	Block Count	0000h
508:509	Result	0000h
510:511	Req./Resp.	0005h

Result Read Response

Offset	Field Name	Value
0 :195	Stuff bytes	0...00h
196:227	MAC / Key	*MAC from the device*
228:483	Data	0...00h
484:499	Nonce	0...00h
500:503	Write counter	*New Counter value* (1)
504:505	Address	0000h
506:507	Block Count	0000h
508:509	Result	*Result code*
510:511	Req./Resp.	0600h

NOTE 1 The Write Counter is incremented only if the operation was successfully completed.

4931

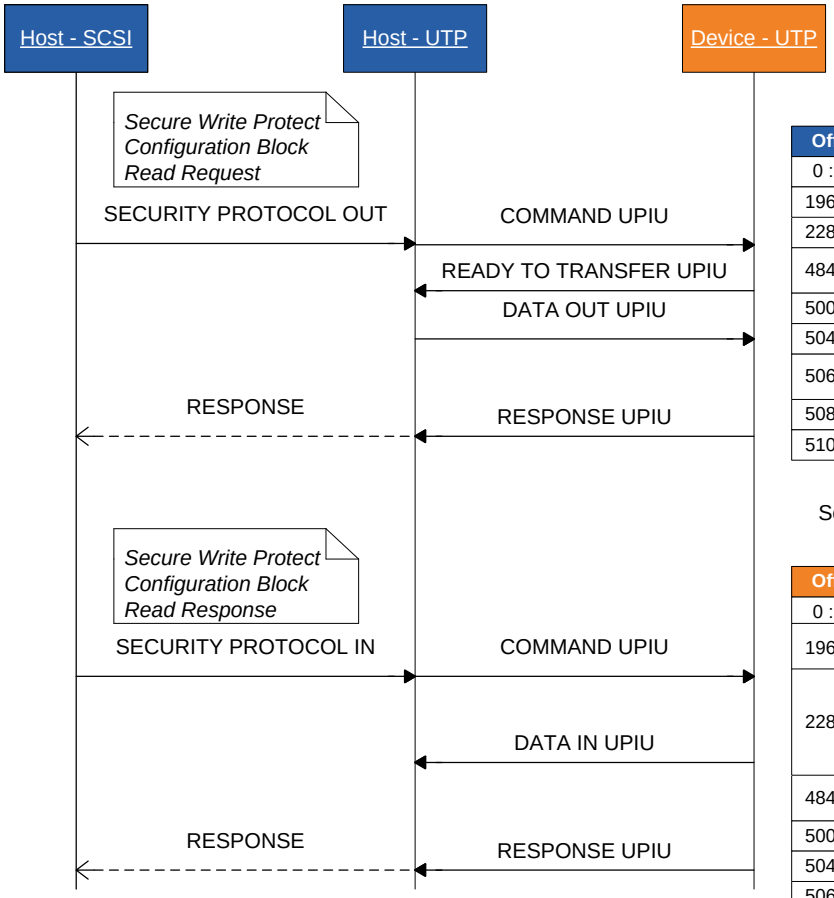
4932 **Figure 12-5 —Authenticated Secure Write Protect Configuration Block Write Flow**

4933

12.4.6.8 Authenticated Secure Write Protect Configuration Block Read

- The Authenticated Secure Write Protect Configuration Block Read sequence is initiated by a SECURITY PROTOCOL OUT command.
- If the INC_512 bit and TRANSFER LENGTH field in the SECURITY PROTOCOL OUT command do not indicate 512 bytes, then the command shall be terminated with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.
- The SECURITY PROTOCOL OUT command delivers a single RPMB message data frame which contains in the Data field a LUN value (byte 228 of the data frame). The Secure Write Protect Configuration Block that will be returned is specific of the logical unit indicated by the LUN field .
- The RPMB data frame delivered from the host to the device includes the Request Message Type = 0007h, Block Count=0001h, Address=0000h, the Data and Nonce. In this request, the LUN is the only relevant byte of the Data field, all other bytes shall be considered reserved and they shall be ignored.
    - If the LUN field is not valid, then the device sets the result to 0009h (Invalid Secure Write Protect Block Configuration parameter). The LUN field is invalid if it is greater than the value specified by bMaxNumberLU or if the logical unit is not enabled (bLUEnable=00h).
    - If the LUN field indicates a logical unit with bLUWriteProtect set to a value different from zero, then the device sets the result to 000Ah (Secure Write Protection not applicable).
    - After successful fetch of the Secure Write Protect Configuration Block, the MAC is calculated from response type, nonce, address, data and result. If the MAC calculation fails then returned result is 0002h (Authentication failure).
- If the SECURITY PROTOCOL OUT command completes with GOOD status, then the Secure Write Protect Configuration Block can be retrieved sending a SECURITY PROTOCOL IN command, in which INC_512 bit and ALLOCATION LENGTH field indicate 512 bytes.
- The device returns a RPMB data frame with Response Message Type = 0700h, the block count, a copy of the nonce received in the request, the contents of the Secure Write Protect Configuration Block in the Data field, the MAC and the result
- If data fetch from addressed location inside device fails or some other error occurs during the read procedure then returned result is 0008h (Secure Write Protect Configuration Block access failure).

12.4.6.8 Authenticated Secure Write Protect Configuration Block Read (cont'd)



Secure Write Protect Configuration Block Read Request

Offset	Field Name	Value
0 :195	Stuff bytes	0...00h
196:227	MAC / Key	0...00h
228:483	Data	*LUN, 0...00h*
484:499	Nonce	*Nonce from the host*
500:503	Write counter	0...00h
504:505	Address	0...00h
506:507	Block Count	0001h
508:509	Result	0000h
510:511	Req./Resp.	0007h

Secure Write Protect Configuration Block Read Response

Offset	Field Name	Value
0 :195	Stuff bytes	0...00h
196:227	MAC / Key	*MAC from the device*
228:483	Data	*LUN, DATA LENGTH, Secure Write Protect Entries*
484:499	Nonce	*Copy of the Nonce*
500:503	Write counter	0...00h
504:505	Address	0...00h
506:507	Block Count	0001h
508:509	Result	Result code
510:511	Req./Resp.	0700h

Figure 12-6 — Authenticated Secure Write Protect Configuration Block Read Flow

12.5 Malware Protection

The UFS device will also have the option to protect boot, bus configuration settings and other important device configuration settings so that once they are set they cannot be modified. The implementation of the protection of these parameters is defined within the spec where the parameter is defined.

12.6 Mechanical

Packaging and requirements for UFS embedded device should adhere to the following guidelines if possible

- Reset and data transfer pins should be located in the second (PoP) or third row (MCP) in from the side of the package to prevent access.

13 UFS FUNCTIONAL DESCRIPTIONS

13.1 UFS Boot

13.1.1 Introduction

Some computing systems can have the need to download the system boot loader from an external non-volatile source. This task can be accomplished through an internal boot ROM contained in the host SOC whose code when executed determines a minimal initialization of the system to start the boot code transfer.

Several features of the boot functionality can be configured in order to be adapted to different system requirements.

Moreover specific features to ensure boot data integrity and no corruption of boot code are defined.

13.1.2 Boot Configuration

During boot operation the UFS host controller retrieves the system boot code stored in the UFS device.

In this version of the standard, the boot mechanism is defined for a point-to-point topology (see Figure 13-1).

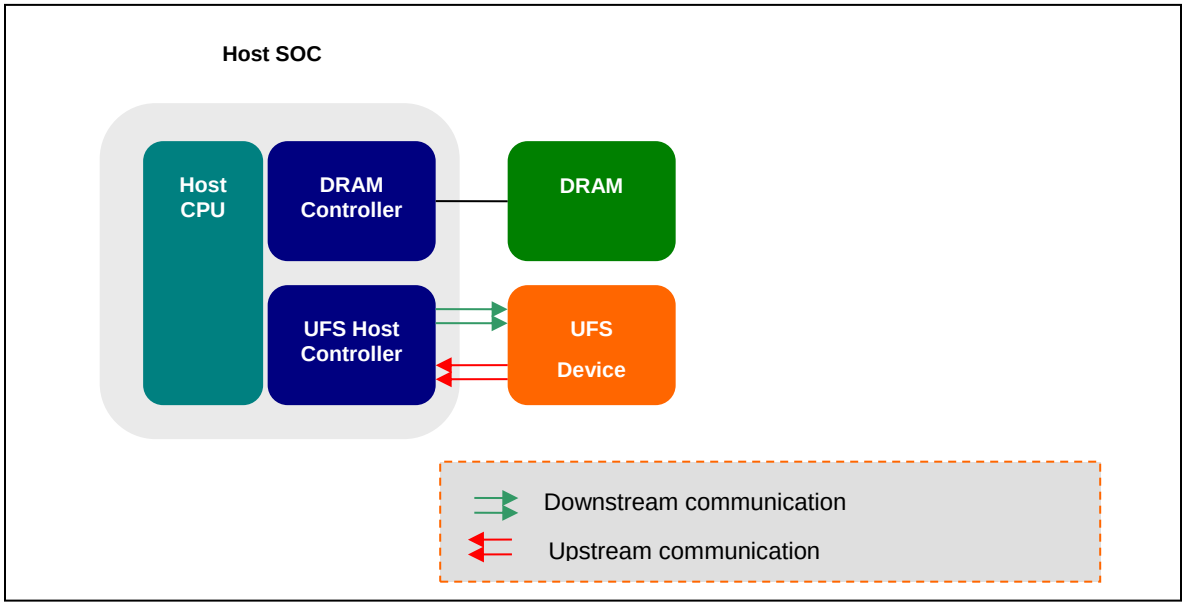


Figure 13-1 — UFS System Diagram

Two logical units (Boot LU A, Boot LU B) can be used to store the boot code, but only one of them will be active during the boot process. Any logical unit can be configured as “Boot LU A” or “Boot LU B”. No more than one logical unit may be configured as “Boot LU A”, and no more than one logical unit may be configured as “Boot LU B”. The logical unit that is active during boot is mapped onto the Boot well known logical unit (W-LUN = 30h) for read access. In this way it is maintained a fix logical unit number when the active logical unit is swapped from A to B or vice versa, when the host updates the boot code.

13.1.2 Boot Configuration (cont'd)

Several configurable fields of the Device Descriptor and the Unit Descriptors determine the device behavior during boot. Device Descriptor and Unit Descriptors are configured by writing the Configuration Descriptor.

For a UFS bootable device, the boot feature is enabled if bBootEnable field in the Device Descriptor is set to 01h.

The characteristics of the logical units used during boot are configured setting the corresponding fields of the Configuration Descriptor; see 14.1.4.3, Configuration Descriptor, for details.

In particular, the number of allocation units (dNumAllocUnits) field configures the logical unit size, and the boot logical unit ID (bBootLunID) field allows to designate the logical unit as being "Boot LU A" or "Boot LU B".

The logical unit active during the boot shall be configured by writing the bBootLunEn attribute, as described in Table 13-1.

Table 13-1 — bBootLunEn Attribute

bBootLunEn	Description
00h	Boot LU A = disabled Boot LU B = disabled
01h	Boot LU A = enabled Boot LU B = disable
02h	Boot LU A = disable Boot LU B = enabled
Others	Reserved

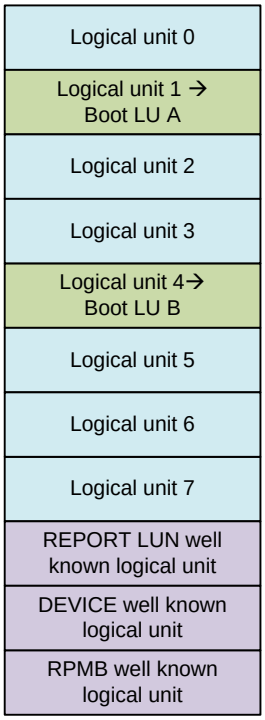
The host should not attempt to set bBootLunEn to 'Reserved' values, and UFS device shall generate an error in case of an attempt to set 'Reserved' values and not execute the request.

When bBootLunEn attribute is 00h the boot feature is disabled, the device behaves as if bBootEnable would be equal to zero.

The active boot logical unit will be mapped onto the Boot well known boot logical unit (W-LUN = 30h) once the bBootLunEn has been properly configured.

Figure 13-2 shows an example of a UFS device having eight logical units: LU 1 and LU 4 are configured, respectively, as "Boot LU A" and "Boot LU B". In particular, LU 1 is the active one (bBootLunEn = 01h).

13.1.2 Boot Configuration (cont'd)



bBootLunID in Unit Descriptors		
LU	Value	Description
0	00h	Disabled
1	01h	Boot LU A
2	00h	Disabled
3	00h	Disabled
4	02h	Boot LU B
5	00h	Disabled
6	00h	Disabled
7	00h	Disabled

bBootLunEn = 01h. "Boot LU A" is the active logical unit for boot. Therefore, the logical unit 1 is mapped onto the Boot well known logical unit (W-LUN=30h, LUN field in UPIU = B0h).

Figure 13-2 — Example of UFS Device Memory Organization for Boot

13.1.3 Initialization and boot code download process

The initialization and boot code download process is made up of the following phases: partial initialization, boot transfer and initialization completion.

13.1.3.1 Partial initialization

The partial initialization phase starts after power on, or hardware reset, or EndPointReset and involves the entire UFS stack. At the end of this phase, the UniPro boot sequence shall be completed, and the UTP layer shall be capable of accessing Device Descriptor (if the bDescrAccessEn field of the Device Descriptor is ‘01h’) and exchanging UPIU for READ command and TEST UNIT READY command. If the bDescrAccessEn field is ‘00h‘ descriptors will be accessible only after the initialization completion phase.

Each single layer in the UFS protocol stack executes the initialization process on both UFS host and UFS device sides.

a) Physical Layer (M-PHY)

After reset events, the physical layer will move from DISABLED state to HIBERN8 state.

b) Link Layer (UniPro)

On host and device side UniPro boot sequence takes place:

1) The UniPro stack is reset using the DME_RESET.req primitive.
2) Wait until the reset completion is indicated by the DME_RESET.cnf_L primitive.
3) The UniPro stack is enabled using the DME_ENABLE.req primitive.
4) Wait until the enable completion is indicated by the DME_ENABLE.cnf_L primitive.
5) The UniPro Link StartUp sequence is initiated using the DME_LINKSTARTUP.req primitive. The UniPro Link Startup consists of a series of multiphase handshakes to establish initial link communication in both directions between UFS host and device.
6) Wait until the link startup completion is indicated by the DME_LINKSTARTUP.cnf_L primitive.

c) UFS Transport Layer (UTP)

At the end of the UFS Interconnect Layer initialization on both host and device side, the host shall send a NOP OUT UPIU to verify that the device UTP Layer is ready.

For some implementations, the device UTP layer may not be initialized yet, therefore the device may not respond promptly to NOP OUT UPIU sending NOP IN UPIU.

The host waits until it receives the NOP IN UPIU from the device. When the NOP IN UPIU is received, the host is acknowledged that the UTP layer on the device is ready to execute UTP transactions.

d) Link Configuration

The host may configure the Link Attributes (i.e., Gear, HS Series, PWM Mode in Rx and Tx) by using DME primitives at UniPro level.

e) Device Descriptor Reading

The UFS host may optionally discover relevant device info for the boot process by accessing the Device Descriptor (i.e., Device Class/Subclass, Boot Enable, Boot LUs size, etc.). The UFS host is allowed to access the Device Descriptor only if the bDescrAccessEn is ‘01h’, otherwise this descriptor can be accessed only after the device has fully completed its initialization.

13.1.3.2 Boot transfer

The following steps can be executed only if bBootEnable field is set.

Boot code download

Firstly, the UFS host issues a TEST UNIT READY command to the Boot well known logical unit to verify if the latter can be accessed. If the command succeeds, the UFS host reads the Boot well known logical unit by issuing SCSI READ commands and the UFS device will start to send the boot code on the Upstream Link. During this phase only the Boot well known logical unit is accessible: this logical unit shall accept read commands, while other logical units may not be ready.

13.1.3.3 Initialization completion

After the host has completed the boot code download from the Boot well known logical unit, the initialization process proceeds as described in the following. The host sets the fDeviceInit flag to “01h” to communicate to the UFS device that it can complete its initialization. The device shall reset the fDeviceInit flag when the initialization is complete. The host polls the fDeviceInit flag to check the completion of the process. When the fDeviceInit is reset, the device is ready to accept any command.

13.1.3.3 Initialization completion (cont'd)

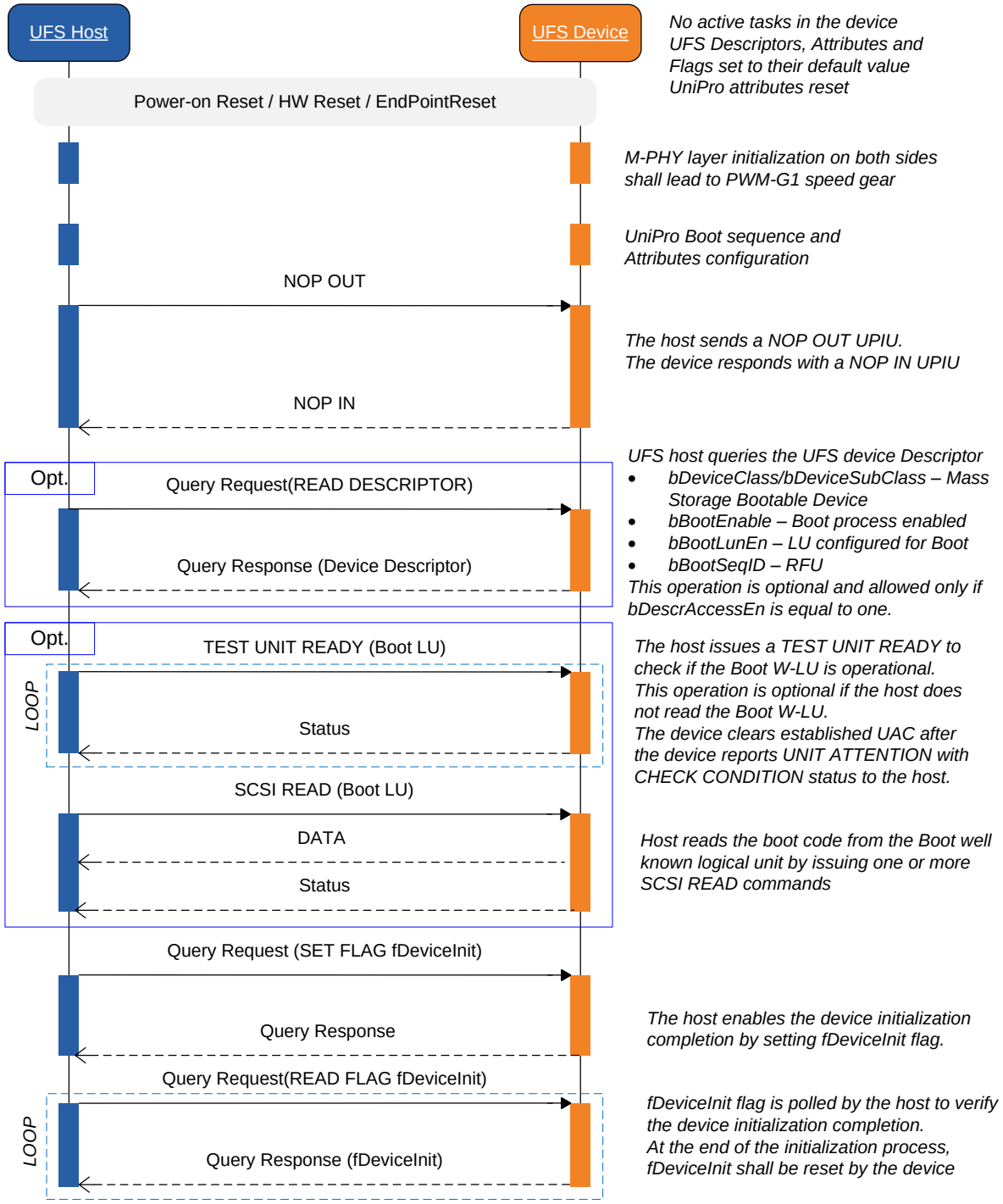


Figure 13-3 — Device Initialization and Boot Procedure Sequence Diagram

13.1.3.3 Initialization completion (cont'd)

Table 13-2 — Valid UPIUs and SCSI Commands for Each Initialization Phase

Phase	Event	Valid UPIU	Valid SCSI command
Before Initialization	Power On Reset / HW Reset / EndPointReset	None	None
UIC Layer Initialization Phase	M-PHY layer initialization UniPro Boot sequence and UIC Layer Attributes configuration	None	None
UTP Layer Initialization Phase	Receive a single NOP OUT UPIU Send NOP IN UPIU for response to NOP OUT UPIU	NOP OUT UPIU NOP IN UPIU	None
Boot W-LU Ready Phase (Optional)	Read Device Descriptor (Optional) (1)	QUERY REQUEST UPIU (READ DESCRIPTOR Device Descriptor)	None
	Boot Transfer (Optional) (2)	COMMAND UPIU for Boot W-LU	INQUIRY, REQUEST SENSE, TEST UNIT READY, READ (6), READ (10), READ (16) (3)
Application Layer Initialization Phase	Receive QUERY REQUEST UPIU (SET FLAG fDeviceInit to '01h') Send QUERY RESPONSE UPIU with fDeviceInit = '00h'	QUERY REQUEST UPIU (SET FLAG fDeviceInit to '01h') QUERY REQUEST UPIU (READ FLAG fDeviceInit) QUERY RESPONSE UPIU	None
Device initialization completed Phase	Any normal operation	Any supported UPIU	Any supported SCSI commands

NOTE1 Device Descriptor may be read only if bDescrAccessEn is set to '01h'.

NOTE2 Boot well-known logical unit may be read if bBootEnable is set to '01h', at least one logical unit is configured for boot (bBootLunEn) and bBootLunID selects the desired boot logical unit.

NOTE3 READ (16) command support is optional.

13.1.4 Initialization process without boot code download

If the boot process is not enabled on the UFS device, or it is not supported by the device class, or the host does not need to transfer the boot code, the host executes the initialization process as described in 13.1.3, omitting the boot transfer phase.

13.1.5 Boot Logical Unit Operations

The Boot well known logical unit is read only, therefore the boot code can be stored only writing the boot logical units (A or B).

Boot logical units are written to store the boot code during the system manufacturing phase and they may be also updated during the system lifecycle. These logical units can be read to verify their content.

Therefore the following operations are permitted on the Boot logical units:

1. boot code write – for boot code upload/update
2. boot code read – to verify the content programmed
3. boot code removal – to remove the content of the Boot logical unit

These operations can be executed regardless the bBootEnable field value in the Device Descriptor.

Boot logical units (A or B) can be write protected using the methods described in 12.3, Device Data Protection.

13.1.6 Configurability

The boot process is configurable through several parameters in the Configuration Descriptor (see 14.1.4.3) to adapt it to different usage models and system features.

The following parameters refer to boot capabilities.

- Device Descriptor parameters:
  - bBootEnable (Boot Enable)
  - bDescrAccessEn (Descriptor Access Enable)
  - bInitPowerMode (Initial Power Mode)
  - bInitActiveICCLevel (Initial Active ICC Level)
- Unit Descriptor parameters for Boot LU A and Boot LU B:
  - bLUEnable (Logical Unit Enable)
  - bBootLunID (Boot LUN ID)
  - bLUWriteProtect (Logical Unit Write Protect)
  - bMemoryType (Memory Type)
  - dNumAllocUnits (Number of Allocation Units)
  - bDataReliability (Data Reliability)
  - bLogicalBlockSize (Logical Block Size)
  - bProvisioningType (Provisioning Type)

NOTE These parameters are non volatile and they may be programmed during the system manufacturing phase.

In addition to the parameters mentioned, the following attributes are relevant for device initialization and boot

- bBootLunEn (Boot LUN Enable)
- bRefClkFreq (Reference Clock Frequency value)

13.1.7 Security

13.1.7.1 Boot Area Protection

Boot areas might be protected in order to avoid boot code alteration by a third party: the write protection mechanism for the boot logical units can be defined configuring the corresponding bLUWriteProtect parameter of the Unit Descriptor.

In particular, the boot logical units may be permanently write protected or power-on write protected. In case of power-on write protection, the boot logical units can be written only when the fPowerOnWPEn flag is equal to zero.

13.2 Logical Unit Management

13.2.1 Introduction

The functionality aims to provide a mechanism to let an external application define and use a virtual memory organization which could easily fit different usage models in a versatile way.

Besides segmenting the available addressable space, the mechanism introduces the possibility of differentiating each logical unit through dedicated functionalities and features.

This section describes the procedure to configure the UFS device in terms of: number of logical units, logical unit size, logical unit memory type, etc. Security features can be configured as described in clause 12, UFS Security.

A UFS device can be organized in different logical units. Each one represents an autonomous computing entity with independent logical address ranges and singularly accessible.

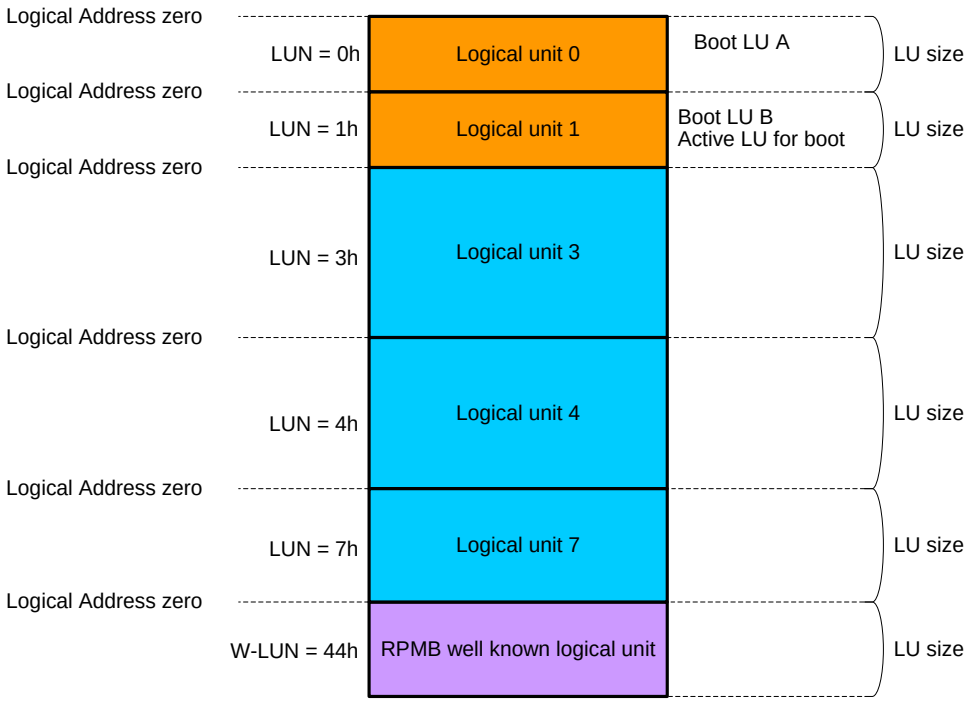
Moreover, each logical unit can be defined for a specified use and with peculiar attributes (i.e., memory type) in order to be adapted to different UFS host usage models and operating systems requirements.

13.2.2 Logical Unit features

UFS device address space is organized in several memory areas configurable by the user. In particular, such memory areas are denoted as logical units and characterized by the fact that they have independent logical addressable spaces starting from the logical address zero.

In addition to the logical units, the UFS device supports the following well known logical units for specific purposes: UFS Device, REPORT LUNS, Boot and RPMB. Logical units are addressed by the LUN (logical unit number), while well known logical unit are addressed by the W-LUN (well known logical unit number).

13.2.2 Logical Unit features (cont'd)



NOTE 1 The figure shows an example of device configuration in which LU 0 and LU 1 are used as boot logical units, and the logical units 3, 4 and 7 for code and data storage. LU 1 is the boot active logical unit and it may be accessed in read using the W-LUN = 30h (LUN field in UPIU = B0h).

Figure 13-4 — Example of UFS Device Memory Organization

Each logical unit will have a physical implementation on the non-volatile storage media.

In particular, the UFS device shall support:

- The number of logical units specified by bMaxNumberLU. Each of them configurable as boot logical units with a maximum of two.
- One RPMB well known logical unit (W-LUN = 44h, LUN field in UPIU = C4h)

Two logical units can be configured as boot logical unit, with only one of them active and readable through the Boot well known logical unit (W-LUN = 30h) for the execution of the system boot (see 13.1, UFS Boot). The RPMB well known logical unit is accessed by authenticated operations by a well defined security algorithm (see 12.4, RPMB). The other logical units will be used to fulfill other use cases.

13.2.2 Logical Unit features (cont'd)

Common features of each logical unit are:

- independent logical addressable spaces (starting from logical address zero up to the logical unit size),
- Configurable logical unit size.

The size of each logical unit is determined by the number of allocation units assigned to it: dNumAllocUnits parameter value of the Configuration Descriptor. The dNumAllocUnits is expressed in terms of allocation unit size (bAllocationUnitSize).

Moreover each logical unit is characterized by the memory type parameter which can be configured. Examples of memory types to differentiate logical unit properties are the following ones:

- Default type – regular memory characteristic
- System Code type – a logical unit that is rarely updated (e.g., system files or binary code executable files or the host operating system image and other system data structures)
- Non-Persistent type – a logical unit that is used for temporary information (e.g., swap file extend the host virtual memory space)
- Enhanced Memory type – vendor specific attribute

The definition of the Enhanced Memory type is left open in order to accomplish different needs and vendor specific implementations.

Mechanisms of write protection can be configured for each logical unit. Write protection feature types are:

- Permanent write protection (permanent read only)
- Power on write protection (write protection can be cleared with a power cycle or a hardware reset event)

Write protection is not available in the RPMB well known logical unit.

13.2.3 Logical Unit Configuration

The user shall configure the logical units of the UFS device according to the following rules:

- Maximum number of logical units is specified by bMaxNumberLU.
- One or two logical units can be configured as boot logical units.

When a UFS device is shipped only the following well known logical units will be available: UFS Device, REPORT LUNS and the RPMB. All other logical units shall be configured before they can be accessed.

NOTE The RPMB well known logical unit will be configured by the device manufacturer before shipping the device.

Logical units may be configured writing the Configuration Descriptors, see 14.1.3, Accessing Descriptors and Device Configuration, for details.

The configuration of each logical unit can be retrieved by reading the corresponding Unit Descriptor.

It is recommended to execute logical unit configuration during the system manufacturing phase.

Table 13-3 summarizes the configurable parameters per logical unit. See 14.1.4, Descriptor Definitions, for details about these parameters.

Table 13-3 — Logical unit configurable parameters

Configurable parameters		Logical Unit
Name	**Description**	
bLUEnable	Logical Unit Enable	LU 0, …, Maximum LU specified by bMaxNumberLU
bBootLunID	Boot LUN ID	LU 0, …, Maximum LU specified by bMaxNumberLU
bLUWriteProtect	Logical Unit Write Protect	LU 0, …, Maximum LU specified by bMaxNumberLU
bMemoryType	Memory Type	LU 0, …, Maximum LU specified by bMaxNumberLU
dNumAllocUnits	Number of allocation units assigned to the logical unit. The value shall be calculated considering the capacity adjustment factor of the selected memory type.	LU 0, …, Maximum LU specified by bMaxNumberLU
bDataReliability	Data Reliability	LU 0, …, Maximum LU specified by bMaxNumberLU
bLogicalBlockSize	Logical Block Size	LU 0, …, Maximum LU specified by bMaxNumberLU
bProvisioningType	Provisioning Type	LU 0, …, Maximum LU specified by bMaxNumberLU

13.2.3 Logical Unit Configuration (cont'd)

The following Geometry Descriptor parameters provide relevant information for configuring the logical units:

- qTotalRawDeviceCapacity (total raw device density in unit of 512 bytes)
- dSegmentSize
- bAllocationUnitSize (Allocation Unit Size, value expressed in number of segments)
- wSupportedMemoryTypes
- Maximum number of allocation unit for each memory type (dSystemCodeMaxNAllocU, dNonPersistMaxNAllocU, etc.)
- Capacity Adjustment Factor for each memory type (wSystemCodeCapAdjFac, wNonPersistCapAdjFac, etc.)
- bMinAddrBlockSize (this parameter indicates a value equal or greater than 4Kbyte)
- bOptimalReadBlockSize and bOptimalWriteBlockSize
- bMaxInBufferSize
- bMaxOutBufferSize

To enable the access to a logical unit, the user shall configure Unit Descriptor parameters as described in the following.

- bLUEnable
  bLUEnable shall be set to 01h to enable the logical unit. If bLUEnable is equal to 00h the logical unit is disabled and all Unit Descriptor parameters are don't care.
- bMemoryType
  bMemoryType shall be set to value corresponding to the desired memory type. The wSupportedMemoryTypes parameter in the Geometry Descriptor indicates which memory types are supported by the device.
- bLogicalBlockSize
  bLogicalBlockSize value shall adhere to the following rules:
  - $2^{bLogicalBlockSize} \geq bMinAddrBlockSize \times 512$,
  - $2^{bLogicalBlockSize} \leq bMaxInBufferSize \times 512$,
  - $2^{bLogicalBlockSize} \leq bMaxOutBufferSize \times 512$.

  To optimize the device performance, it is recommended to configure the logical block size (bLogicalBlockSize) to represent the value indicated by dOptimalLogicalBlockSize for the specific logical unit memory type.

  Supported bLogicalBlockSize values are device specific, refer to the vendor datasheet for further information.

13.2.3 Logical Unit Configuration (cont'd)

- dNumAllocUnits

  dNumAllocUnits determines the size of the logical unit. If LUCapacity is the desired logical unit size expressed in bytes, the dNumAllocUnits value shall be calculated using the following equation:

$$\text{dNumAllocUnits} = \text{CEILING}\left(\frac{\text{LUCapacity} \times \text{CapacityAdjFactor}}{\text{bAllocationUnitSize} \times \text{dSegmentSize} \times 512}\right)$$

  where:

  - CapacityAdjFactor = Capacity Adjustment Factor of the particular memory type

    The Capacity Adjustment Factor value for Normal memory type is one.

    The following example shows dNumAllocUnits calculation for two logical units (LU 1 and LU 4) with the characteristics:

  - LU 1: 12 Gbyte, Normal memory type
  - LU 4: 32Mbyte, Enhanced memory type 1

  Assuming that the medium of the UFS device is composed by NAND flash memories which support 2 bit-per-cell and 1 bit-per-cell operation modes. The 2 bit-per-cell operation mode may be associated with the Normal memory type, while the 1 bit-per-cell operation mode may be associated with the Enhanced memory type 1.

  The Capacity Adjustment Factor for the Enhanced memory type 1 will be equal to 2.

  If

  - dSegmentSize = 1024
  - bAllocationUnitSize = 8

  Then dNumAllocUnits for LU 1 and LU 4 are:

$$\text{dNumAllocUnits LU 1} = \text{CEILING}\left(\frac{12\ \text{Gbyte} \times 1}{8 \times 1024 \times 512\ \text{byte}}\right) = \text{CEILING}\left(\frac{12\ \text{Gbyte}}{4\ \text{Mbyte}}\right) = 3072$$

$$\text{dNumAllocUnits LU 4} = \text{CEILING}\left(\frac{32\ \text{Mbyte} \times 2}{8 \times 1024 \times 512\ \text{byte}}\right) = \text{CEILING}\left(\frac{64\text{Mbyte}}{4\ \text{Mbyte}}\right) = 16$$

  The logical unit capacity can be retrieved by either reading the qLogicalBlockCount parameter in the Unit Descriptor or issuing the READ CAPACITY command.

  In particular, the relations between the parameters returned by READ CAPACITY (RETURNED LOGICAL BLOCK ADDRESS and LOGICAL BLOCK LENGTH IN BYTES), and bLogicalBlockSize and qLogicalBlockCount parameters in Unit Descriptors are:

  - RETURNED LOGICAL BLOCK ADDRESS = qLogicalBlockCount – 1,
  - LOGICAL BLOCK LENGTH IN BYTES = $2^{\text{bLogicalBlockSize}}$

13.2.3 Logical Unit Configuration (cont'd)

- bBootLunID
  bBootLunID shall be set as described in the following:
  - 00h: if the logical unit is not a boot logical unit,
  - 01h: to configure the logical unit as "Boot LU A",
  - 02h: to configure the logical unit as "Boot LU B",

  NOTE The 01h value and 02h value shall be assigned to no more than one logical unit.
- bLUWriteProtect
  bLUWriteProtect shall be set as described in the following:
  - 00h: if the logical unit is not write protected,
  - 01h: to configure power on write protection,
  - 02h: to configure permanent write protection.
- bDataReliability
  bDataReliability shall be set to configure the device behavior when a power failure occurs during a write operation to the logical unit:
  - 00h: logical unit is not protected. Logical unit's entire data may be lost as a result of a power failure during a write operation,
  - 01h: logical unit is protected. Logical unit's data is protected against power failure.
- bProvisioningType
  bProvisioningType shall be set to configure the logical unit provisioning type:
  - 00h: to disable thin provisioning,
  - 02h: to enable thin provisioning with TPRZ = 0,
  - 03h: to enable thin provisioning with TPRZ = 1.

13.3 Logical Block Provisioning

13.3.1 Overview

Logical Block Provisioning is the concept that describes the relationship between the logical block address space and the physical memory resources that supports the logical address space.

Logical units in a UFS device shall be either a Full Provisioned LU or a Thin Provisioned LU.

13.3.2 Full Provisioning

Every LBA in a fully provisioned logical unit is mapped.

A logical unit that is fully provisioned shall provide enough LBA mapping resources to contain all logical blocks for the logical unit's capacity as reported by the device server in response to a READ CAPACITY command.

The device server shall not cause any LBA on a fully provisioned logical unit to become unmapped.

A fully provisioned logical unit does not support logical block provisioning management – i.e., does not support UNMAP command.

13.3.3 Thin Provisioning

In thin provisioning there is no requirement that the available physical memory resources match the size of the logical address space.

A thin provisioned logical unit is not required to provide LBA mapping resources sufficient to contain all logical blocks for the logical unit's capacity as reported by the device server in response to a READ CAPACITY command.

In UFS device, a thin provisioned logical unit shall have sufficient physical memory resources for all addressable logical blocks when the logical unit is configured by writing the Configuration Descriptor: the number of LBAs reported in READ CAPACITY shall not exceed the number of physical memory blocks available.

Logical address to physical resource allocation is managed by Logical Block Provisioning Management. Every LBA in a thin provisioned logical unit shall be either mapped or deallocated.

In UFS device, a thin provisioned logical unit shall support the Mapped and Deallocated states in the Logical Block Provisioning State Machine. An unmapped LBA shall be a deallocated LBA.

UFS device shall support Thin Provisioned logical unit including: Logical Block Provisioning Management, UNMAP command, erased, discard and purge functionalities as described in clause 12, UFS Security.

13.4 Host Device Interaction

13.4.1 Overview

This sub-section describes several UFS features conceived to improve performance and/or reliability. Some of them are related directly to commands present in the logical unit queues (e.g. inter-LU priority, system data tag, context management), others are related to the device state (e.g. background operations, dynamic device capacity) or focused on reliability (e.g. data reliability, real-time clock information).

13.4.2 Applicable Devices

All the features described 13.4 should be implemented on UFS devices. The extent to which the features are implemented will be up to the device manufacturer. It is expected that poor implementations will result in lower device performance or reliability and higher max power consumption in the low power modes.

13.4.3 Command Queue: Inter-LU Priority

The specific details of how commands interact in a queue of a specific logical unit are handled in other chapters. This section outlines how the queues within a device interact with each other. There can be many different implementations of a UFS device. For example, it may be implemented as a single or multithreaded processor. In order to make the UFS spec implementation agnostic this section outlines a parameter that allow the host to communicate to the device its priorities so that the device can take this into account when executing commands from the host.

In the implementation case where several queues are serviced from a single execution unit, it is necessary for the host to either designate all queues as having the same priority or designate a single Queue as having a higher priority. A parameter value shall be defined to allow the host to designate a queue as having high or no priority. In the case where a queue is designated as having a higher priority, whenever a command enters the queue with the high priority it will be executed as soon as possible resulting in commands from other queues being stalled.

One example of where a host may want to take advantage of this feature is when the host has allocated one logical unit to be a code execution unit and another unit to be a mass storage unit. In this example the execution of code takes priority over mass storage transfers, so the host would set the parameter bit associated with the code logical unit to be high priority and leave the remaining queues as lower priority.

The logical unit supports two level of queue priorities:

1. High priority – This is used for logical unit with high priority requests. Any commands sent to this logical unit will have higher priority than commands sent to logical units with lower priority. For example, when servicing demand-paging applications, read commands would have this priority.
2. No priority – This is used for all regular logical units which do not belong to priority #1

In addition to the execution of commands explicitly issued by the host, the device may execute background operations for device house-keeping. In general those operations have a much lower priority than the commands sent by the host and are implemented using a specific method described in 13.4.4, Background Operations Mode.

13.4.3.1 Implementation

The bHighPriorityLUN parameter in the Device Descriptor shall be set to configure which logical unit has the command queue with the higher priority.

bHighPriorityLUN shall be set to:

- 7Fh: if all logical units have the same priority,
- LUN of the higher priority logical unit.

Name	Description	Valid Values	Default(1)
bHighPriorityLUN	bHighPriorityLUN defines the high priority logical unit. If this parameter value is 7Fh all logical units have the same priority.	0 to the number of LU specified by bMaxNumberLU, and 7Fh	7Fh
NOTE 1 The column "Default" defines the parameter value set by the device manufacturer.			

13.4.4 Background Operations Mode

13.4.4.1 Introduction

A managed device requires time to execute management tasks. The background operations mode grants the device time to execute the commands associated with these flash management tasks. Flash management operations may include, but are not limited to, wear leveling, bad block management, wipe and garbage collection. The operations completed during the background operations period are determined by the device manufacturer and are not covered by the UFS spec.

13.4.4.2 Purpose

Performance Improvement

The intent of this mode is to improve a device response to host commands by allowing the device to postpone device management activities that occur as a result of host initiated operations to periods when the host is not using the device.

Systems that will use a UFS device tend to have peak periods of activity, in which the best response possible is needed from the UFS device. These peak periods are followed by idle periods, where the host can allow the device to do device management operations. Allowing better communication between the host and the device on when these idle periods will occur allows the UFS device to perform more optimally in the system.

The device will still be permitted to do device management when the host initiates operations, however the downside of doing this may be poorer device performance. This mode will just give device manufacturers the option to delay device management operations to improve performance. It is recommended that devices postpone as many tasks as possible to take full advantage of the possible performance improvements associated with this feature.

Host Power Management

This mode will also allow the host to control when the device uses power to perform management activities. The host will have more knowledge about the power consumed by the system and can make the appropriate tradeoffs about when to use system power and when to conserve it.

An example of where host control over power consumed by the device could be an advantage would be the case where the system has very little battery power and the UFS device has a lot of unused memory.

13.4.4.2 Purpose (cont'd)

In this case the host may not wish for the device to perform clean up operations but conserve the power for more critical system functions.

Allowing the host to communicate with the device on when activities can be performed will allow better system power management, which can be controlled by the host.

13.4.4.3 Background Operations Status

The device signals to the host that the device has a need for background operations using the Exception Event mechanism, and in particular the URGENT_BKOPS bit in wExceptionEventStatus

- '0': No immediate need to execute background operation (corresponds to no operations or non-critical)
- '1': Immediate need to execute background operation (corresponds to performance being impacted or critical)

When the host detects a request for executing background operations (URGENT_BKOPS set to one) it may read the bBackgroundOpStatus attribute to find out the need level, as follows:

- 00h : No operations required
- 01h : Operations outstanding (non-critical)
- 02h : Operations outstanding (performance being impacted)
- 03h : Operations outstanding (critical)

The URGENT_BKOPS bit is set to zero if the bBackgroundOpStatus is set to zero or one, otherwise it is set to one.

It is expected that the host will respond as soon as possible when the status changes to service, since if the background operations are not properly managed, then the device could fail to operate in an optimal way.

In the case where the device status is operations outstanding (critical) this will mean that the device will only respond to mode sense and mode select commands. The host should put in all possible measures to ensure the device never reaches this state since it means the device is no longer able to operate.

The point at which the device enters each of these states is up to the manufacturer and is not defined in this standard.

13.4.4.4 Operation Initiation

There is no explicit command to start the background operations. A mode enable bit indicates whether the device is allowed to execute background operations. If the background operations enable bit is set and the device is in Active power mode or Idle power mode, then the device is allowed to execute any internal operations.

When the device receives a command which requires a data transfer and if all command queues are empty, then the device shall start the data transfer sending DATA IN UPIU or RTT UPIU within the time declared in bBackgroundOpsTermLat. The host can minimize the device response time by disabling background operations mode during critical performance times.

In the case where the background operations status is "operations outstanding (critical)", the aforementioned latency limit does not apply.

13.4.4.5 Power Failure

It is the device's responsibility to ensure that the data in the device is not corrupted if a power failure occurs during a background operation.

13.4.4.6 Implementation

13.4.4.6.1 Background Operations Enable

fBackgroundOpsEn is the Flag to be used to enable or disable the execution of background operations. This Flag is defined as follows:

- 0 = Device is not permitted to run background operations
- 1 = Device is permitted to run background operations.

The default value of this Flag is one: background operations permitted. The device shall terminate ongoing background operations when this Flag is cleared by the host. For more details see Table 14-25, Flags.

13.4.4.6.2 Background Operations Status

bBackgroundOpStatus is an attribute defined as follows:

- 00h = not required
- 01h = required, not critical
- 02h = required, performance impact
- 03h = critical.

For more details see 14.3, Attributes.

13.4.4.6.3 Background Operations Termination Latency

bBackgroundOpsTermLat defines the maximum latency for starting data transmission when background operations are ongoing. The termination latency limit applies to two cases:

- When the device receives a COMMAND UPIU with a transfer request. The device shall start the data transfer and send a DATA IN UPIU or a RTT UPIU within the latency limit.
- When the device receives QUERY REQUEST UPIU clearing fBackgroundOpsEn Flag. The device is expected to terminate background operations within the latency limit.

The termination limit does not apply in the case where the background operations status is "operations outstanding (critical)".

bBackgroundOpsTermLat is a parameter of the Device Descriptor and its granularity is 10msec. It is expected that transitions between link states (e.g., HIBERNATE to ACTIVE) or temporary congestion on the link occur in shorter timescales and are therefore negligible in comparison to the background operations timescale.

13.4.5 Power Off Notification

A UFS host will notify the device when it is going to power the device off by requesting the device to move to UFS-PowerDown power mode. This will give the device time to cleanly complete any ongoing operations. The device will respond to the host when it is ready for power off, meaning that the device entered the UFS-PowerDown power mode. Host can then power off the device without the risk of data loss.

13.4.6 Dynamic Device Capacity

Common storage devices assume a fixed capacity. This presents a problem as the device ages and gets closer to its end of life. When some blocks of the device become too old to be used reliably, spare blocks are reallocated to replace them. A device contains some spare blocks for this purpose, as well as for some housekeeping operations. However, when all spare blocks are consumed, the device can no longer meet its fixed capacity definition and it stops being functional (some become read-only, some stop responding completely).

A simple solution to enable the host to continue operation at the end of the device lifetime, would be to allow the capacity to be dynamic. If the device is allowed to reduce its reported capacity, it can reallocate blocks that were used to store data as new spare blocks to compensate for aging. To support this, the device needs to report to the host how much more spare blocks it needs for each logical unit, and then the host needs to relinquish some used blocks to let the device reallocate them as spares.

A device may implement a spare blocks resource management policy either per logical unit, allocating a fixed amount of spare blocks per logical unit, or per memoty type, allocating a fixed amount of spare blocks for all logical units with the same memory type (bMemoryType).

bDynamicCapacityResourcePolicy parameter in the Geometry Descriptor indicates which spare blocks resource management policy is implemented.

13.4.6.1 Implementation

13.4.6.1.1 Initial Device Requirements

- Only logical units that support thin provisioning and logical block provisioning management functions can be involved in the dynamic device capacity process. The host can discover if thin provisioning is enabled and if a logical unit supports logical block provisioning management functions at UTP level, reading the bProvisioningType parameter in the Unit Descriptor of each logical unit, or at SCSI level through the TPE bit in the READ CAPACITY (16) parameter data.
  - bProvisioningType shall be either 02h or 03h.
  - TPE bit shall be set to one.
- The Unit Descriptor of each logical unit includes the following two parameters: qLogicalBlockCount and qPhyMemResourceCount.
  - The qLogicalBlockCount is equal to the total number of addressable logical blocks in the logical unit. Its value is established when the logical unit is configured and never changes during the device life time. In particular, the qLogicalBlockCount value shall be equal to RETURNED LOGICAL BLOCK ADDRESS + 1 (RETURNED LOGICAL BLOCK ADDRESS is a field included in the Read Capacity Parameter Data and corresponds to the last addressable block on medium under control of logical unit).
  - The qPhyMemResourceCount is equal to the total physical memory resources available in the logical unit, expressed in $2^{bLogicalBlockSize}$ unit. Its value decreases with the execution of dynamic capacity process.
- UFS requires that there shall be sufficient resource in the physical memory resources pool to support the logical addressable memory space reported in READ CAPACITY when the device is first configured. Therefore, qPhyMemResourceCount shall be equal to qLogicalBlockCount in the Unit Descriptor initially.
- Dynamic device capacity feature does not involve the following logical units: RPMB well known logical unit, power-on write protected logical units (independently of fPowerOnWPEn flag value), permanently write protected logical units (independently of fPermanentWPEn flag value), or logical units configured as boot logical unit (Boot LUN ID = 01h or 02h, independently of bBootLunEn value).

13.4.6.1.2 Dynamic Capacity Procedural Flow

1) When the physical memory resources necessary for proper operation in a logical unit has been drawn down, the device may request to the host to remove some resources from the physical memory resources pool serving the logical address space of the logical units. This is achieved setting to one the DYNCAP_NEEDED bit in the wExceptionEventStatus attribute. If DYNCAP_EVENT_EN bit in wExceptionEventControl attribute is one, then the EVENT_ALERT bit of Device Information field included in the RESPONSE UPIU will be set to one.

2) Device shall indicate the amount of physical memory to be removed from the resource pool in dDynCapNeeded attribute. Each element of the dDynCapNeeded[LUN] shall be equal to the amount of bytes to be removed divided by the optimal write block size (bOptimalWriteBlockSize is a parameter in the Geometry Descriptor). Therefore, the host can calculate the amount of physical memory to be removed from the resource pool for each logical unit multiplying the dDynCapNeeded[LUN] attribute by bOptimalWriteBlockSize parameter. UFS device shall not request to remove physical memory from the resource pool of logical units which are write protected or configured as boot logical unit (dDynCapNeeded[LUN] shall be zero).

3) The host ensures that all outstanding tasks in the device queues have been completed or aborted.

4) If the device spare blocks resource management policy is per memory type (bDynamicCapacityResourcePolicy = 01h), then the host should ensure that the amount of LBAs in the deallocated state in all logical units with the same memory type (bMemoryType) is equal to or greater than the amount of requested physical memory resources from all logical units with the same memory type (bMemoryType).

   For example, assuming that bDynamicCapacityResourcePolicy = 01h and the device asks to remove logical blocks from the resource pool of a single logical unit, the host may remove blocks from one or more logical units having that particular memory type as long as the total amount of logical blocks in a deallocated state results be greater than or equal to the total amount of logical blocks specified by the device.

   However, if the device spare blocks resource management policy is per logical unit (bDynamicCapacityResourcePolicy = 00h), then host ensures that the amount of LBAs in the deallocated state is equal or greater than the amount of requested physical memory resources for each logical unit.

   The host deallocates LBAs sending one or more UNMAP commands.

   a) The range(s) of LBA in the deallocate state shall be aligned to a bOptimalWriteBlockSize boundary, and their size shall be an integer multiple of bOptimalWriteBlockSize .

   b) The LBA range(s) shall be equal or greater than the memory resources amount that has been requested by the device for each logical unit.

   c) The LBA range(s) does not need to be at the end of the logical address space.

5) The host sets fPhyResourceRemoval flag to one and triggers an EndPointReset or a device hardware reset to initiate the dynamic capacity operation.

6) The host may either execute the complete boot process as described in 13.1, UFS Boot, or may skip reading the boot logical unit and set fDeviceInit flag to one to start the device initialization. During this phase the device will execute the dynamic capacity operation. The host waits until the device clears fDeviceInit flag. The device initialization may take a time longer than normal initialization due to internal processes necessary to re-arrange physical memory resources. If a power loss occurs, the dynamic capacity operation will proceed at the next power up during the device initialization.

13.4.6.1.2 Dynamic Capacity Procedural Flow (cont'd)

7) When the dynamic capacity operation is completed, the device will clear fDeviceInit flag and the fPhyResourceRemoval flag. If the operation is completed successfully, the DYNCAP_NEEDED bit is cleared too, therefore the EVENT_ALERT bit in Device Information will return to zero, if no other exception events are active. The qPhyMemResourceCount parameter in the Unit Descriptors is updated with a new value reflecting the amount of physical memory resources remaining in the resource pool.

   NOTE The qLogicalBlockCount parameter in the Unit Descriptors and the RETURNED LOGICAL BLOCK ADDRESS parameter in Read Capacity Parameter Data will stay the same as initially configured. Therefore, the updated qPhyMemResourceCount value will be less than those two parameters after dynamic capacity operation.

8) If the dynamic capacity operation does not succeed, for example because LBA range(s) does not meet one or more of the requirements described in point 4, the DYNCAP_NEEDED bit shall remain set to one. Therefore, the EVENT_ALERT bit in the Device Information field of the RESPONSE UPIU will remain set to one to notify the host that further dynamic capacity operation is needed.

   NOTE The dDynCapNeeded[LUN] attribute value may have been updated.

9) To account for the reduction of the physical memory resources pool after dynamic capacity operation, the host maintains a range(s) of LBA's in deallocated state that are aligned in address and size to integer multiples of bOptimalWriteBlockSize, with the total equal to or greater than qLogicalBlockCount minus qPhyMemResourceCount. Otherwise, a write error will result when the host attempts to write (map) more LBA's than the available physical memory resources.

   NOTE The host may change the LBA range(s) that are in deallocate state during the use of the device.

13.4.6.1.3 Dynamic Device Capacity Notification

The dynamic device capacity is one of the exception events that may set the EVENT_ALERT bit of the Device Information field included in the RESPONSE UPIU.

To enable the setting of this bit, the DYNCAP_EVENT_EN bit in the wExceptionEventControl attribute shall be set to one.

When the host detects that the EVENT_ALERT bit is set to one, it should read the wExceptionEventStatus attribute to discover if the source of this event is a request for reducing the device capacity. In particular, if DYNCAP_NEEDED bit is set to one, the host should process the request as described in this standard.

The DYNCAP_EVENT_EN bit shall be set to zero if the host is not capable or does not intend to use the dynamic device capacity feature. Otherwise, a dynamic capacity request event will set the EVENT_ALERT bit to one, masking out notification of other exception events.

13.4.6.1.4 Error handling

- Success/failure to unmap (release) LBA's is as defined in UNMAP command.
- If the host ignores the dynamic device capacity notification and continues to write to the device without unmapping LBA's to free up physical memory, the device may become non-writeable over time and cause a WRITE command to fail. In which case, the device server shall return the following error condition in response to the WRITE command.
  - The device server shall terminate the command requesting the operation with CHECK CONDITION status with the sense key set to DATA PROTECT and the additional sense code set to SPACE ALLOCATION FAILED WRITE PROTECT.
- When the qPhyMemResourceCount is less than qLogicalBlockCount in Unit Descriptors, it indicates that the physical memory resources pool is smaller than the logical addressable memory space (LBA's) in the logical unit. It is the host's responsibility to keep track of the amount of physical memory available. If the host attempts to write more data than the available physical memory resources and the device is unable to complete the write operation successfully, the device shall return the following error condition.
  - The device server shall terminate the command requesting the operation with CHECK CONDITION status with the sense key set to DATA PROTECT and the additional sense code set to SPACE ALLOCATION FAILED WRITE PROTECT.

13.4.6.1.5 Physical Memory Resource State Machine

Figure 13-5 shows the state machine for the physical memory resources. In addition to the "Mapped" state and the "Deallocated" state, which are defined in logical block provisioning state machine too, there is the "Removed from the Resource Pool" state.

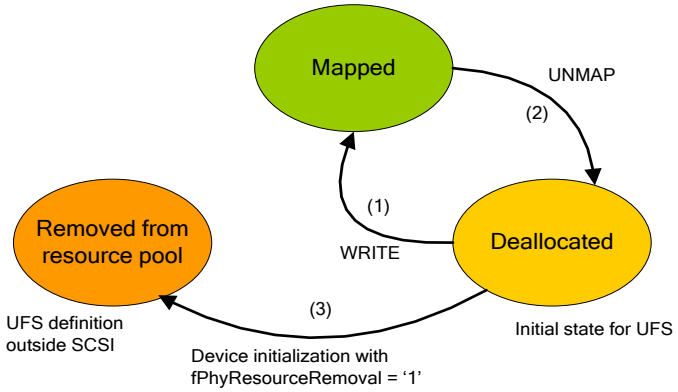


Figure 13-5 — Physical Memory Resource State Machine

1) Write operation: physical memory resource from the resource pool is mapped to LBA containing valid data.
2) UNMAP operation for erase/discard:
   a) Physical memory resource is unmapped (deallocated) from LBA and returned to the resource pool.
   b) Residual data in unmapped physical memory resource is not valid.
3) Device re-initialization with fPhyResourceRemoval flag previously set to one causes some physical memory resources to be removed from the resource pool servicing the logical address space. After conversion the qPhyMemResourceCount is updated by UFS device to indicate the amount of physical memory resources remaining in the resource pool of each logical unit.

13.4.7 Data Reliability

13.4.7.1 Description

The UFS host has the ability to define the level of data reliability during normal operation and power failure per logical unit.

There are two components to the data reliability that are defined for UFS. The first component deals with the data currently being written and the second deals with the data that has previously been stored in the medium.

The first component, also known as Reliable Write, means that if the device losses power during a write operation to the medium (when executing a SCSI write command, a host initiated flush of data to the medium, etc.), the data in the range affected by the operation will be either the old data or the new written data when the device recovers from power failure.

Keeping either the old or new data is important for the file system to recover after power loss. The file system may keep its structures segmented and signed with CRC or equivalent mechanism. The device insures that a large enough granularity of data is authentic either with an old or new copy, to make sure the file system can validate or invalidate these segments.

The resolution for the old and new data shall be aligned to the logical block size. The figure below shows some of the possible scenarios that the host will see when it recovers from power failure during a 32 Kbyte write operation in a logical unit with 4 Kbyte logical block size.

For RPMB well known logical unit, the data reliability granularity shall be equal to bRPMB_ReadWriteSize × 256 bytes.

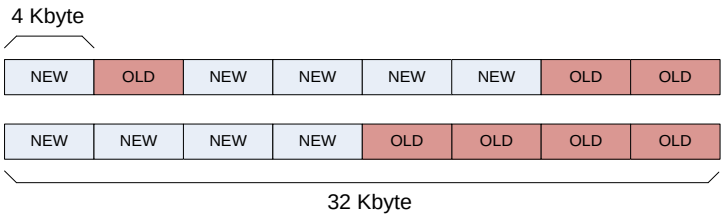


Figure 13-6 — Example of data status after a power failure during reliable write operation

The second component, also known as Data Reliability, allows the host to define the level of protection to be applied to existing data on the device. In some technologies that will be used to implement UFS devices, the device has the option to potentially sacrifice some of the existing data on a device during a power failure in order to provide better write performance. Depending on the application for the end device the host can select per logical unit whether the data in that logical unit shall be protected during power failure, which may have a performance impact, or to select device performance with the risk of losing data during a power failure.

Data Reliability feature for each logical unit can be set when the device is configured during system integration.

In particular, if Data Reliability is enabled, the logical unit will execute Reliable Write operation and the data already stored in the medium will not be corrupted by a power failure occurred during the execution of a write operation to the medium.

The data reliability feature will be configurable only for logical units and not for well known logical units.

The RPMB well known logical unit will automatically select data reliability.

13.4.7.2 Implementation

bDataReliability parameter in the Unit Descriptor shall be used by the host to configure the logical unit data reliability.

bDataReliability values are defined as in the following:

00h: Data reliability disabled
Corruption of the existing data in the medium of the specific logical unit may occur if a power loss happens during device activity like writing of data to medium.

01h: Data reliability enabled
The existing data stored in the medium of the specific logical unit shall not be corrupted if a power loss occurs, and the memory range that was accessed by the interrupted write command shall contain the old data or new data (or a mixture of old and new data as explained in 13.4.7.1) once power is restored.

Table 13-4 — Parameter for controlling logical unit data reliability

Parameter Name	Description
bDataReliability	bDataReliability defines the device behavior when a power failure occurs when writing data to the medium 00h: Data reliability disabled 01h: Data reliability enabled Others: Reserved

13.4.8 Real-Time Clock Information

Providing Real Time Clock (RTC) information to a storage device could be useful for the device internal maintenance operations (execution of RTC internal operations is not affected by fBackgroundOpsEn flag value).

Host may provide either absolute time if available, or relative time information. This feature provides a mechanism for the host to update either absolute or relative time.

The host sends RTC information when a wPeriodicRTCUpdate has passed since the last RTC information update. In case the device is not powered up or asleep when the period has expired, the host wakes it up and update RTC information.

NOTE When device is configured with wPeriodicRTCUpdate as ‘undefined’ (see Device Descriptor) a wPeriodicRTCUpdate could not expire and the host may update RTC information considering vendor recommendations.

It is recommended to provide RTC information after transitioning from UFS-Sleep power mode or UFS-PowerDown power mode to Active power mode.

Updating RTC information is done by writing to dSecondsPassed attribute.

While the device is busy handling an RTC update event and the related background operation, the device may keep the fBusyRTC flag is set to one.

The device may perform operations in the background as a result of receiving RTC update. In order to allow optimized efficiency of these background operations, it is recommended that the host refrains from sending commands, other than Query Request to the device, and keep the device powered and in Active power mode as long as fBusyRTC flag is set to one. The device may consume active power during that time. Therefore, it would be advisable to update RTC in times where the system is usually idle and has no specific power limitations, e.g., at night time when battery is being charged.

13.4.9 Context Management

To better differentiate between large sequential operations and small random operations and to improve multitasking support, contexts can be associated with groups of read or write commands. Associating a group of commands with a shared context allows the device to optimize data handling.

A context can be seen as an active session, configured for a specific read/write pattern (e.g., sequential in some granularity). Multiple read or write commands are associated with this context to create some logical association between them, to allow device performance optimization. For example, a large sequential write pattern may have better performance by allowing the device to improve internal locality and reduce some overhead (e.g., if some large unit of data is allowed to be affected by a power failure as a whole while it is being written, all of the commands that fill this unit can work faster because they can reduce the overhead usually required to protect each write individually in case of a power failure). Furthermore, handling of multiple concurrent contexts allows the device to better recognize each of the write patterns when they are all mixed together.

The maximum number of contexts the device can support is reported in the bMaxContexIDNumber field of the Geometry Descriptor. When the host configures the device, it divides this number across the created LUs and writes the number of contexts to be supported in each LU to wContextCapabilities field in the Configuration Descriptor.

To use a context, an available Context ID shall be picked. Then, it shall be initialized by writing the configuration attribute of the relevant LU (wContextConf). Then, data can be read/written associated to the context by specifying the Context ID in GROUP NUMBER field of the CDB of the read/write command. When the context is no longer used, the configuration attribute (wContextConf) should be written as all ‘00’ by the host to close the context. A context shall be closed prior to re-configuring it for another configuration/use.

No ContextID shall be open after power cycle.

13.4.9.1 Context configuration

Before any context can be used it shall be configured.

Configuration is done by setting the context configuration attribute of the relevant LU (setting wContextConf with INDEX equal to LUN and SELECTOR equal to ContextID, SELECTOR ‘0’ is reserved.). Then, all read commands or write commands that are associated with this ContextID shall be sent using GROUP NUMBER set to ContextID.

When the context is no longer needed, it should be closed by writing a zero byte to the configuration attribute.

To configure a specific ContextID for a specific LU, the following fields shall be written to the context configuration attribute of the specific context needed:

- Activation and direction mode (read-only, write-only or read/write)
  The direction indicates if all following accesses to this context would be either read-only, write-only or both read/write. Writing a non-zero direction to this field is the ‘activation code’ for the context. A zero in this field means a closed context which can no longer be addressed by its ID until re-configured.
- Large Unit context flag
  This indicates if the context is following Large Unit rules or not
  - If Large Unit context flag is set, then the Large Unit multiplier is used to specify the larger unit size.
- Reliability mode
  Controls how data written to a context should respond to interruptions.

13.4.9.2 Activation and direction mode

A non-zero context can be configured as a read-only context, a write-only context or a read/write context.

Any read command associated with any context to an address that is part of an active write-only context is not allowed, and may either fail the command or return dummy data.

Any write command associated with any context to an address that is part of an active read-only context is not allowed, and may either fail the command, ignore the data written or cause unexpected data to return in the read commands.

A context that is configured as read/write context may be read or written, as long as the writing follows the context rules.

NOTE read/write context may have reduced performance compared to read-only or write-only contexts.

13.4.9.3 Large-Unit mode

The Large Unit is the smallest unit that can be used for large sequential read/write operations, in order to reduce internal overhead and improve performance.

Accessing a Large Unit (both read and write) shall:

- Use a ContextID configured to operate in Large Units
- Always access a full Large Unit, in order and from beginning to end
    - Multiple read/write commands with TRANSFER LENGTH smaller than the Large Unit size may be used to read or write the Large Unit. Read/write commands may be interleaved with other accesses and commands, as long as the specific Large Unit is being accessed with its own separate Context ID
    - A Large Unit that is being written shall not be modified outside the scope of the context (e.g., no other writes from other contexts to the address range of the Large Unit shall be used, no erases/trims to that range, etc.)
    - Different Large Units belonging to the same context may be located in non consecutive addresses on the media, as long as alignment is kept (a Large Unit shall be accessed in order from beginning to end, but only within the range of the specific Large Unit – the next Large Unit can be non-consecutive and even in a lower address)
- When writing a Large Unit context, data shall always be aligned and in multiples of bOptimalWriteBlockSize

When writing a Large Unit context, the last Large Unit before closing the context may be partially written, as long as it is written from the beginning, in order and up to a specific point where it is closed. The rest of the Large Unit may be padded by the device to the end of the Large Unit with random data.

13.4.9.4 Reliability mode

In case a write command to a Context ID is interrupted, the device behaves as if all the writes to the context from its configuration were written in one large write command.

In case of a power failure or software reset before closing an active context – even if not in the middle of a write command to the specific context (even if not in the middle of any command) – is considered as if the event occurred during writing the entire context.

A context behavior is determined as part of its configuration and is applied to all writes to this context until it is closed. Interruption during any of the writes may cause some of the data not to be fully programmed on the device. Still no partial Logical Block (of bLogicalBlockSize size) shall exist – any Logical Block written as part of the context shall contain either the new data written or its old data before the context was configured. The scope of data that may be affected by the interruption depends on the mode configured:

- For non-Large Unit contexts:
  - MODE0 – Normal mode – Any data written to the context from the time it was configured may be affected
  - MODE1 – Non-Large Unit, reliable mode – Only data written by a specifically interrupted write command may be affected. Any previously completed write to the context shall not be changed because of any interruption.
- For Large Unit contexts:

  NOTE In the cases below, the unit N refers to the current Large Unit which is being written when interruption occurs, unit N-1 refers to the last Large Unit of the context that was written completely before the current one and unit N-2 and earlier are Large Units that were completed before the N-1 unit.

  - MODE0 – Normal mode – Any unit may be affected: Any data written to the context from the time it was configured may be affected.
  - MODE1 – Large Unit, unit-by-unit mode – Unit N may be affected, units N-1 and earlier are not: Any data written to a Large Unit context may affect the entire specific Large Unit accessed. Any previously completed Large Units in the context shall not be changed because of any interruption.
  - MODE2 – Large Unit, one-unit-tail mode – Unit N and N-1 may be affected, units N-2 and earlier are not: Any data written to a Large Unit context may affect the entire specific Large Unit accessed and the entire completed Large Unit that was accessed before the current one. Any other completed Large Units in the context shall not be changed because of any interruption.

In case the host sends a Task Management Request to abort a write command to a non-zero context or the write command fails with an error, the write may still be interrupted like any context-less write. In case these scenarios are interrupting a write to a Large Unit context, the device shall always stop writing on a bOptimalWriteBlockSize boundary.

13.4.9.5 Large-Unit Multiplier

In order to allow increased performance by parallelism, the device may allow reading or writing in multiples of the Large Unit granularity.

The granularity of Large Unit size is provided by bLargeUnitGranularity_M1 parameter in the Unit Descriptor as indicated in the following:

Large Unit size granularity = 1 Mbyte × (bLargeUnitGranularity_M1+1)

The device reports through a Unit Descriptor parameter (wContextCapabilities) the maximum multiplier that is supported by the logical unit.

The Large Unit size is configured setting the Large Unit Multiplier as defined in the following:

Large Unit size = Large Unit size granularity × Large Unit Multiplier =

= 1 Mbyte × (bLargeUnitGranularity_M1+1) × Large Unit Multiplier.

For example, if bLargeUnitGranularity_M1 = 0 and Large Unit Multiplier = 2, then the Large Unit granularity is 1 Mbyte and the Large Unit size is 2 Mbyte.

13.4.10 System Data Tag Mechanism

The System Data Tag mechanism enables the host to notify the device when System Data is sent for storage (for instance file system metadata, operating system data, time stamps, configuration parameters, etc.). This notification (using GROUP NUMBER field in the CDB) would guide the device to handle the System Data optimally. By matching storage characteristics to the System Data characteristics the device could improve access rate of read and update operations and offer a more reliable and robust storage characteristics.

A UFS device has a limited amount of System Data area, a storage area with special characteristics which are tailored to the characteristics and needs of system data. When receiving System Data Tag notification along with the write command, the device will store the system data in the System Data area. In case the capacity available for storing System Data is completely consumed, the device will store the System Data in regular storage and the SYSPOOL_EXHAUSTED bit in the wExceptionEventStatus attribute shall be set to one. Additionally, if SYSPOOL_EVENT_EN bit is equal to one, then the EVENT_ALERT bit of Device Information field present in the RESPONSE UPIU will be forced to one

The SYSPOOL_EVENT_EN bit is included in the wExceptionEventControl attribute.

The host may free up System Data area by unmapping LBAs that were previously written with system data tag characteristics.

The device handles the System Data Tag mechanism in units of system data, the size of system data unit is device specific and can be retrieved reading the bSysDataTagUnitSize parameter in the Geometry Descriptor.

The total available capacity for System Data is indicated by the bSysDataTagResSize parameter of the Geometry descriptor (see 14.1.4.4, Geometry Descriptor, for details).

When a host tags system data during a write operation, an entire storage area of bSysDataTagUnitSize size is handled by the device as system data area even if the size of the data being written is less than bSysDataTagUnitSize. In addition, any command (Write, Unmap, etc.) which updates a system data unit with data not tagged as System Data will change the entire system data unit storage characteristics to regular data. Therefore, it is recommended to handle system data in full units of bSysDataTagUnitSize size.

System Data areas are available only in Normal memory type logical units.

13.4.11 Exception Events Mechanism

The Exception Events Mechanism is used by the device to report occurrence of certain events to the host.

It consists of three components EVENT_ALERT bit, the wExceptionEventStatus attribute and wExceptionEventControl attribute:

- A bit in wExceptionEventStatus attribute is assigned to each exception event. The device shall set the wExceptionEventStatus bits to one when the corresponding exception events are active, otherwise they shall be set to zero.
- A bit in wExceptionEventControl attribute is assigned to each exception event.
  EVENT_ALERT bit shall be set if there is at least one wExceptionEventStatus bit and wExceptionEventControl bit pair set to one.
  The setting of an wExceptionEventStatus bit to one will not force the EVENT_ALERT bit to one if the corresponding bit in the wExceptionEventControl is zero.
- The EVENT_ALERT is a bit in the Device Information field of the RESPONSE UPIU which is the logical OR of all bits in the wExceptionEventStatus masked by the bits of the wExceptionEventControl. The EVENT_ALERT bit is set to one when at least one bit in the wExceptionEventStatus is set and the corresponding wExceptionEventControl bit is one. The EVENT_ALERT is set to zero if all exception events that are enabled in the wExceptionEventControl are not active.

There are three defined exception events: dynamic device capacity, system pool exhausted and background operation. The bits in the wExceptionEventStatus associated with those exception events are described in the following:

- DYNCAP_NEEDED – the device requests a Dynamic Capacity operation (see 13.4.6, Dynamic Device Capacity). This bit is cleared once a Dynamic Capacity operation has completed successfully, releasing the entire capacity that the device had requested to release.
- SYSPOOL_EXHAUSTED – the device ran out of resources to treat further host data as System Data (see 13.4.10, System Data Tag Mechanism). This bit is cleared once the host has turned enough memory areas that were previously handled as System data areas, to non-system data areas.
- URGENT_BKOPS – the device requests host attention for the level of need in Background Operations (see 13.4.4, Background Operations Mode). This bit is cleared once bBackgroundOpStatus returns to 00h or 01h.

In the Device Information field of the RESPONSE UPIU, the device will only indicate the events that were enabled by the host through writing to the wExceptionEventControl attribute. The event bits in the wExceptionEventStatus attribute and in the Device Information field of the Response UPIU are cleared by the device when the clear conditions are met.

13.4.12 Queue Depth Definition

Each logical unit is responsible for managing its own task set. Independently from the task set, the resources used for queueing tasks may either be statically allocated to each LU, so that the LU is capable of queueing new tasks up to a certain depth, or be shared by all LUs, so that queueing resources are dynamically allocated to LUs, depending on tasks received.

A device may implement one of the two queueing architectures described above. The device informs the host software on the policy implemented using read only parameters in the Device Descriptor and the Unit Descriptors.

The depth of a queue is defined as the number of pending commands which can be stored in the queue.

13.4.12.1 Shared Queue

In the shared queue architecture, the device has a fixed-depth queue where tasks are queued as they are received, regardless of their LUN designation.

When a COMMAND UPIU is received, resources are allocated from the shared queue and the command is accounted towards the queue depth limit. The host is expected to track the queue depth and not issue more commands than can be stored in the queue. If queue resources are unavailable, the device shall return a response with TASK SET FULL status.

QUERY REQUEST UPIUs, NOP OUT UPIUs, and TASK MANAGEMENT REQUEST UPIUs are not stored in the shared queue.

When this queueing architecture is implemented, the parameter bQueueDepth in the Device Descriptor shall indicate the depth of the queue. The value of bQueueDepth shall be equal to, or larger than, 1. The bLUQueueDepth parameters in all Unit Descriptors shall all be equal to 0.

13.4.12.2 Per-Logical Unit Queues

In the per-LU queueing architecture, the device implements separate fixed-depth queues, one queue for each LU, or, in other words, allocates a fixed number of queueing resources for each LU.

When a COMMAND UPIU is received, resources are allocated from the queue associated with its logical unit, as indicated by the LUN field in the UPIU Header. The command is accounted towards the depth limit of the respective queue. The host is expected to track the queue depths and not issue more commands than can be stored in their designated queue. If resources are unavailable for the desginated LU, the device shall return a response with TASK SET FULL status (even if queueing resources are available for other LUs).

QUERY REQUEST UPIUs, NOP OUT UPIUs, and TASK MANAGEMENT REQUEST UPIUs are not stored in the LU queues.

When this queueing architecture is implemented, the bLUQueueDepth parameters in Unit Descriptors shall indicate the depth of the queue of each logical unit. If a logical unit is enabled, the value of bLUQueueDepth in its Unit Descriptor shall be equal to, or larger than, 1.

bQueueDepth parameter in the Device Descriptor shall be equal to 0.

NOTE For backward compatibility with previous revisions of the standard, if bLUQueueDepth for an LU is 0, and bQueueDepth is also 0, the queue depth should be treated by the host as unknown. The host is expected to infer the queue depth using software algorithms.

13.4.12.3 RPMB Well Known Logical Unit Queue

RPMB logical unit may use shared queue resources or may use its own separate queue, as implemented by the device manufacturer.

If the RPMB logical unit uses the shared queue resources, its bLUQueueDepth parameter shall be equal to 0. When a COMMAND UPIU is received, resources are allocated from the shared queue, and the command is accounted towards the queue depth limit. The host is expected to track the queue depth and not issue more commands than can be stored in the queue. If queue resources are unavailable, the device shall return a response with TASK SET FULL status.

If the RPMB logical unit uses a separate queue, its bLUQueueDepth parameter shall be equal to 1. The host is expected to not issue more than one command to RPMB LU at any given time. If more than one command is issued to RPMB LU, the device shall return a response with TASK SET FULL status.

It should be noted that, unlike other LUs, it is permitted that RPMB LU has a fixed depth queue while other LUs use a shared queue.

13.4.13 Device Life Span Mode

The intent of this mode is to improve the device life span by increasing the device endurance. Devices use mechanism like wear leveling etc. for improving the device life time which is limited to P/E cycle count given by a memory vendor. In this mode, device may use technology like lower programming voltage etc. for operations to increase the P/E cycle count and result in improving the device life.

Read and write operation performance in UFS are very high, However maximum operation speed is not required always e.g. when user is sleeping, device is in screen-off mode, downloading large size files/video and so on. During such scenarios, UFS host may indicate to device to use technology like lower programming voltage etc. for operations by enabling fDeviceLifeSpanModeEn flag. On disabling this flag, device uses normal voltage for operations. It is expected that the device will respond normally as soon as the flag is disabled.

There may be performance degradation in this mode, therefore fDeviceLifeSpanModeEn should be set only when the device is not actively used.

The improvement of device life span is dependent on device implementation.

13.4.13.1 Implementation

Device Life Span Mode Enable

fDeviceLifeSpanModeEn is the Flag to be used to enable or disable the execution of technology like lower programming voltage etc. for operations.

0 = Device Life Span Mode is disabled.

1 = Device Life Span Mode is enabled.

The default value of this flag is zero. Host can enable this flag depending on scenarios like screen-off, downloading large files etc.

13.5 UFS Cache

Cache is a temporary storage space in a UFS device. The cache should in typical case reduce the access time (compared to an access to the medium) for both write and read. The cache is not directly accessible by the host but is a separate element in a UFS device. This temporary storage space may be utilized also for some implementation specific operations like as an execution memory for the memory controller and/or as storage for an address mapping table etc. but which definition is out of scope of this standard.

The implementation of the cache is optional for the UFS devices but the related commands shall be implemented so that compatibility with host software driver is seamless independent from the implementation. Devices may explicitly indicate that cache is supported by setting INQUIRY Data VPD page (see [SPC]) parameter V_SUP=1b. V_SUP=0b means that there may or may not be cache implemented. INQUIRY Data VPD page parameter NV_SUP shall be set to 0b.

The cache is a device level cache and applies for all LUs generically. Data written to and read from a Boot W-LU and RPMB W-LU shall not be cached due to the specific nature of these LUs.

The cache is expected to be volatile by nature. Data in the cache is not expected to remain data valid over power cycles or HW/SW resets. The UFS device is expected to manage the cache so that it shall not be possible to read stale data from the device.

While the cache is implemented the device server may utilize the cache during write and read operations for storing data which an application client may request later. The algorithm to manage the cache is out of scope of this standard and is left for the implementation. There are parameters related to the management of the cache defined in CDBs (see bullets) and in 11.4.2.3, Caching Mode Page.

- The disable page out (DPO) bit in the CDB of write, read and verify commands allows the application client to influence the replacement of the logical blocks in the cache (e.g., in case the cache is full). Setting the DPO bit to 1 means that the device server should not replace the existing logical blocks in the cache with the new logical blocks written or read. When the DPO and FUA bits are set to one, write and read operations effectively bypass the cache.

- The force unit access (FUA) bit in the CDB of write and read commands enables the application client to access the medium. Setting the FUA bit to 1 means that the device server shall perform the write to the medium before completing the command and to read logical blocks from the medium (not from the cache).

During write operations the device server may use the cache to store data that is to be written to the medium at a later time (write-back caching) and thus the command may complete prior to logical blocks being written to the medium. This means also that such data may get lost if sudden power loss or reset occurs. There is also possibility of an error occurring during the actual write operation to the medium later. If an error occurred during such write operation it may be reported as a deferred error on a later command.

An UNMAP operation or any other operation which affects data of a logical block in the medium shall cause the device server to update potential data related to the logical block in the cache accordingly.

It is recommended that the host synchronizes the cache before initiating a PURGE operation.

When a VERIFY command is processed both force unit access and synchronize cache operation are implied.

13.5 UFS Cache (cont'd)

Following commands shall be implemented by the device server to enable application client to control the behavior of the cache:

• PRE-FETCH commands: see 11.3.19 and 11.3.20

• SYNCHRONIZE CACHE commands: see 11.3.24 and 11.3.25

13.6 Production State Awareness (PSA)

13.6.1 Introduction

UFS device can utilize knowledge about its production status and adjust internal operations accordingly.

For example, content which was loaded into the storage device prior to device soldering might be corrupted, at a higher probability than in regular mode. The UFS device could use "special" internal operations for loading content prior to device soldering which would reduce production failures and use "regular" operations post-soldering.

The sensitivity for device soldering is a property of the logical unit, some logical units may be sensitive to device soldering while some logical units may not be sensitive to it. Before loading the data to the device the host should read bPSASensitive, to identify the LU which are sensitive to device soldering.

Pre-loaded data is data which is loaded on the device after device configuration is completed (bConfigDescrLock is set and reset of the device), and before device soldering to the host platform. The combined maximum amount of data which could be pre-loaded to all sensitive LUs is device specific and defined by dPSAMaxDataSize attribute.

13.6.2 PSA flow

PSA feature is based on the device capability to uniquely identify which data is written before the soldering process. The PSA flow may be initiated only if all LBAs in logical units with bPSASensitive = 01h are unmapped. In case the host does not know if LBAs are unmapped, then it should set bPSAState 'Off', send an UNMAP command for the entire LBA range of each LU with bPSASensitive set to 01h, to unmap that data and re-start the PSA flow as described below.

Depicted in Figure 13-7 is the PSA flow. To start the PSA flow, the host first checks if PSA feature is supported by the device (see bUFSFeaturesSupport parameter in Device descriptor).

The host is expected to set dPSADataSize to indicate amount of data it plans to pre-load in all logical units with bPSASensitive = 01h. In case the host tries to set dPSADataSize > dPSAMaxDataSize, then the device shall return a General failure error.

The host writes bPSAState attribute to 'Pre-soldering' and then pre-loads the data in the logical units through WRITE commands.

The host should count towards dPSAMaxDataSize limit only data written through a WRITE command to a LU with bPSASensitive descriptor set to '1'. During PSA flow the host is not expected to write data to the same LBA more than once, in such case device behavior may be undefined.

Once the host finishes pre-loading all LU (wrote total of dPSADataSize amount of data) the host is expected to change the state of bPSAState from 'Pre-soldering' to 'Loading Complete' to indicate to the device that pre-loading of data is complete.

Prior to soldering, at 'Loading Complete' bPSAState state, device may stop using special internal operations and resume regular operations. Therefore, the host should not write data to the device as data may be corrupted during soldering; a WRITE Command in this situation may result in an error.

13.6.2 PSA flow(cont'd)

After the setting of bPSAState to 'Loading Complete', the device may be soldered. The device shall set bPSAState to 'Soldered' during the processing of the first WRITE command after a power-up occurred with bPSAState = 'Loading Complete'.

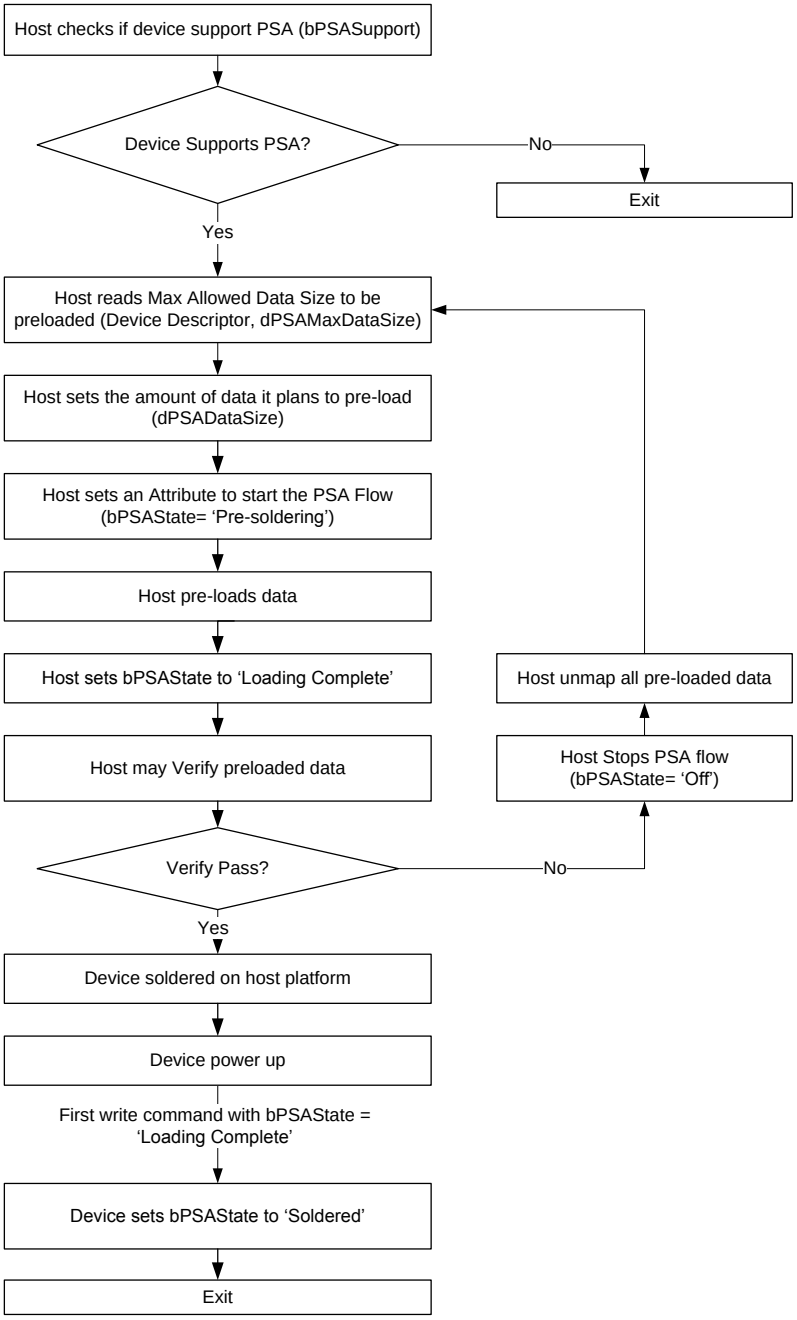


Figure 13-7 — PSA flow

13.6.2 PSA flow(cont'd)

Depicted in Figure 13-8 is the PSA state machine which describes the different states of the bPSAState attribute and the transitions between its states.

Host misbehavior of writing, during pre-soldering phase, more data than indicated by dPSADataSize may result in data corruption during device soldering.

At any time before setting bPSAState to 'Soldered' the host may re-start the PSA flow by switching bPSAState to 'Off', unmap all sensitive data and set bPSAState to 'Pre-soldering'.

A change in bPSAState attribute may involve additional operations by the device which may require some time. bPSAStateTimeout indicates the maximum allowed timeout in which the device may return a response.

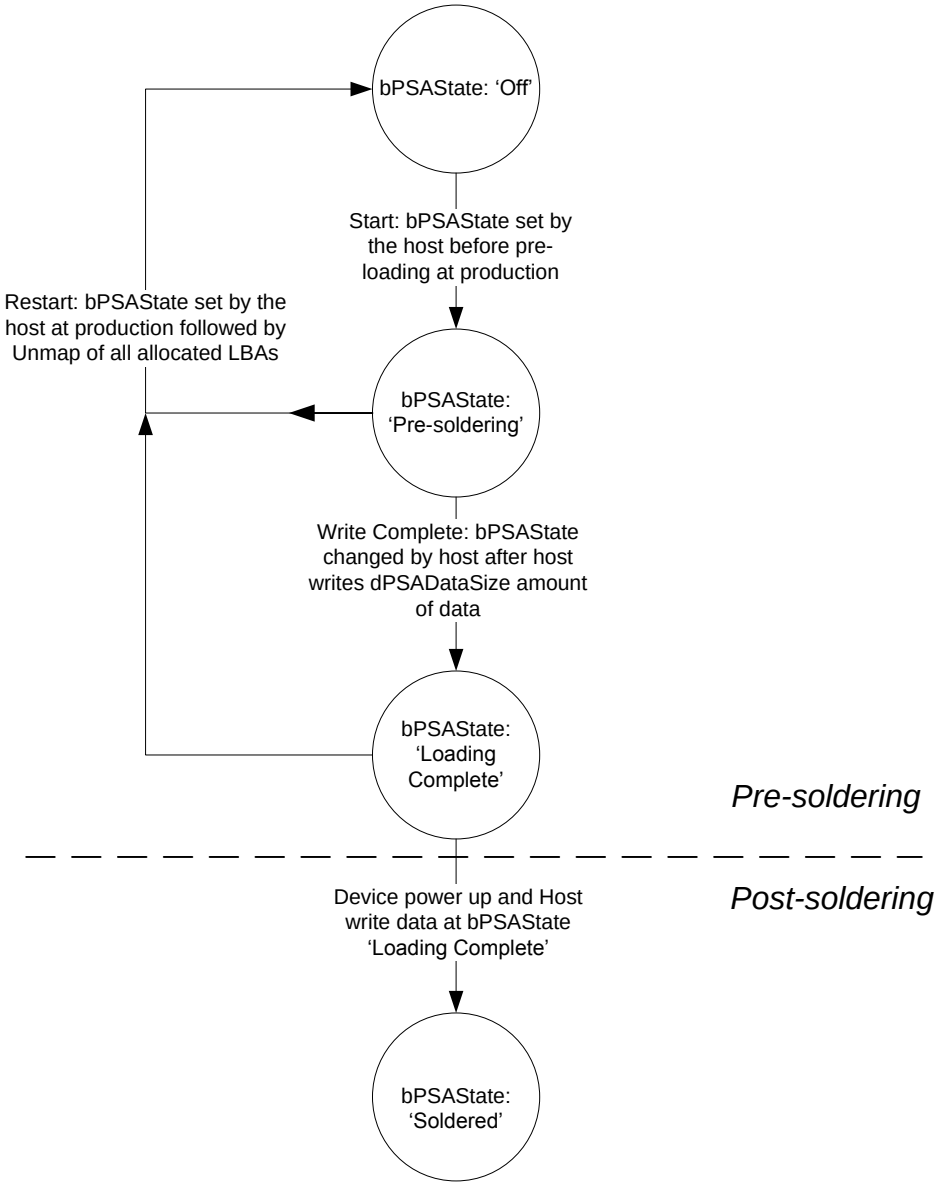


Figure 13-8 — PSA state machine

14 UFS DESCRIPTORS, FLAGS AND ATTRIBUTES

14.1 UFS Descriptors

A descriptor is a data structure with a defined format. Descriptors are accessed via QUERY REQUEST UPIU packets. Descriptors are independently addressable data structures. Descriptors may be stand alone and unique per device or they may be interrelated and linked in a hierarchical fashion to other descriptors by parameters defined within the top-level descriptor. Descriptors can range in size from 2 bytes through 255 bytes. A 2 byte descriptor is an empty descriptor. All descriptors have a length value as their first element. This length represents the entire length of the descriptor, including the length byte. All descriptors have a type identification as their second byte. A descriptor can be partially read, but starting point is always the offset 00h.

A Descriptor is a block or page of parameters that describe something about a Device. For example, there are Device Descriptors, Configuration Descriptors, Unit Descriptors, etc.

In general, all Descriptors are readable, some may be write once, others may have a write protection mechanism.

The Configuration Descriptor is writeable and allows modification of the device configuration set by the manufacturer.

Table 14-1 specifies the descriptor identification values.

Table 14-1 — Descriptor identification values

Descriptor IDN	Descriptor Type
00h	DEVICE
01h	CONFIGURATION
02h	UNIT
03h	Reserved
04h	INTERCONNECT
05h	STRING
06h	Reserved
07h	GEOMETRY
08h	POWER
09h	DEVICE HEALTH
0Ah ... FFh	Reserved

14.1.1 Descriptor Types

Descriptors are classified into types, as indicated in Table 14-1. Some descriptors are singular entities, such as a Device descriptor. Others can have multiple copies, depending upon the quantity defined, such as UNIT descriptors. See Figure 14-1.

14.1.2 Descriptor Indexing

Each descriptor type has an index number associated with it. The index value starts at zero and increments by one for each additional descriptor that is instantiated. For example, the first and only Device Descriptor has an index value of 0. The first String Descriptor will have an index value of 0. The Nth String Descriptor will have an index value of N-1.

14.1.3 Accessing Descriptors and Device Configuration

Descriptors are accessed by requesting a descriptor IDN and a descriptor INDEX. For example, to request the fifth Unit descriptor, the request would reference TYPE UNIT (numeric value = 2) and INDEX 4 (numeric value = N-1).

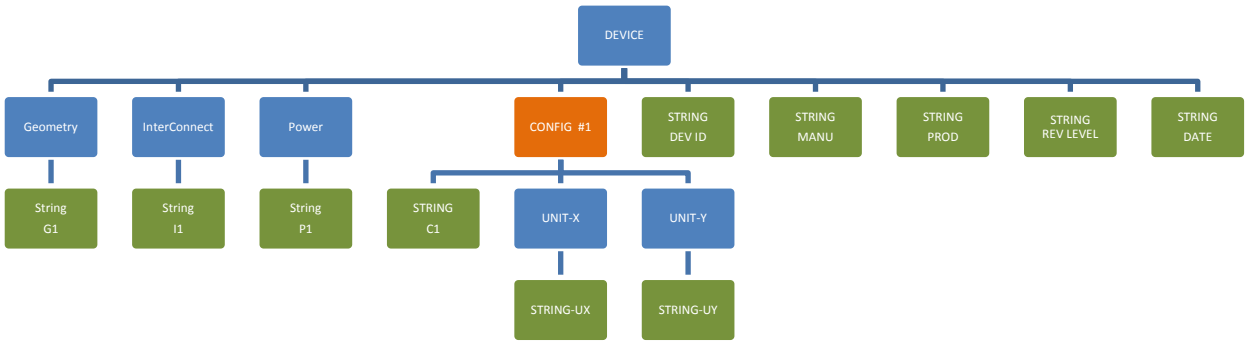


Figure 14-1 — Descriptor Organization

All descriptors are read-only, except the Configuration Descriptors and the OEM_ID String Descriptor which are: readable, writeable if bConfigDescrLock attribute value is equal to 00h.

In particular, the Configuration Descriptors allow modification of the device configuration set by the manufacturer. Parameter settings in the Configuration Descriptors are used to calculate and populate the parameter fields in the Device Descriptor and the Unit Descriptors – an internal operation by the device.

The host may write the Configuration Descriptors multiple times if bConfigDescrLock attribute is equal to zero.

A device that supports 8 logical units has only one Configuration Descriptor, while a device that supports 32 logical units has four Configuration Descriptors.The host may write only the Configuration Descriptors related to the logical units to be configured, and avoid to send write descriptor query requests for the Configuration Descriptors that do not need to be changed.

The bConfDescContinue parameter in Configuration Descriptor indicates the end of a sequence of write descriptor query requests during device configuration.In particular, if bConfDescContinue is set to one, the current query request will be followed by another one, and the device shall not start internal operations to implement the new configuration.If bConfDescContinue is set to zero, the current query request is the last one, and the device shall start internal operations to implement the new configuration.

14.1.3 Accessing Descriptors and Device Configuration (cont'd)

For example, to configure LU 0, LU 1, LU 2, LU 18 and LU 20 the following write descriptor query request may be sent.

1) write descriptor query request with INDEX = 0, bConfDescContinue = 01h, Device Descritpor parameters, logical unit parameters from 0 to 7

2) write descriptor query request with INDEX = 2, bConfDescContinue = 00h, Device Descritpor parameters, logical unit parameters from 16 to 23

The latency of a write descriptor request with bConfDescContinue is set to zero may be significantly longer than a query request with bConfDescContinue set to one.If bConfDescContinue = 00h, then the device shall send a QUERY RESPONSE UPIU only after completing the configuration process.

If bConfDescContinue = 0h, then the Query Response field in the QUERY RESPONSE UPIU shall be set to "Success" only if

- parameters in Device Descriptor and Unit Descriptors have been updated successfully
- logical units configuration has been completed successfully
- logical units are ready for operation (read, write, etc.)

If the query request fails, it shall be possible to repeat the configuration procedure sending a new sequence of write descriptor query requests.

The Configuration Descriptor for same index may be sent more than once by host (eg, Configuration Descriptor with Index = 00h) with bConfDescContinue = 01h. The last received values of each Configuration Descriptor index before bConfDescContinue set to zero, shall be considered as valid configuration.

If power cycle happens while bConfDescContinue is 01h, then all the configuration descriptor values shall be reset to previous successful configuration value or if there is no previous configuration then they shall be reset to MDV values as per UFS Specification.

Once the device has been configured, the setting of bConfigDescrLock to one permanently locks the device configuration. Some devices may be configured multiple times during system setup or system development. The details and restrictions related to multiple device configurations are vendor specific and out of scope for this standard.

NOTE This version of the standard does not require a power cycle to complete the device configuration.

14.1.3.1 Read Descriptor

A Query Request operation with READ DESCRIPTOR opcode is sent by the host to the device. The host builds a QUERY REQUEST UPIU places a READ DESCRIPTOR opcode within the UPIU, sets the appropriate values in the required fields and sends that UPIU to the target device.

Upon reception of the QUERY REQUEST UPIU the device will decode the READ DESCRIPTOR opcode field and retrieve the descriptor indicated by the DESCRIPTOR IDN field, the INDEX field and the SELECTOR field . When the device is ready to return data to the host the device will construct a QUERY RESPONSE UPIU, set the appropriate fields and place the entire retrieved descriptor within a single Data Segment area of the QUERY RESPONSE UPIU.

Upon transmission of the QUERY RESPONSE UPIU the device will consider the Query Request operation complete.

Upon reception of the QUERY RESPONSE UPIU the host will retrieve the requested descriptor from the Data Segment area, and decode the Status and other fields within the UPIU and take the appropriate completion action. Upon reception of the QUERY RESPONSE UPIU the host will consider that the Query Request operation is complete.

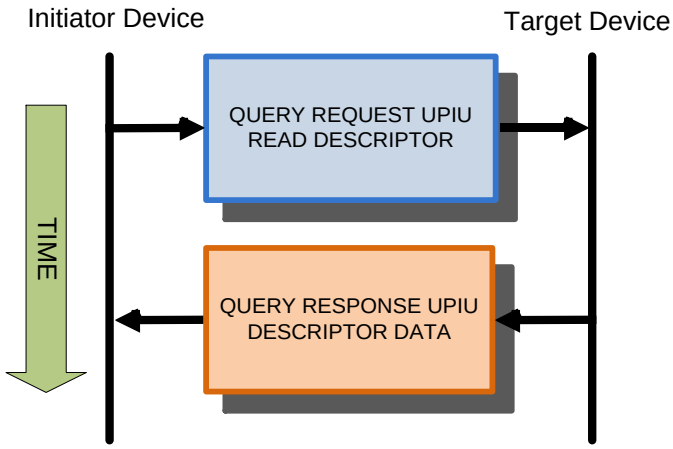


Figure 14-2 — Read Request Descriptor

14.1.3.2 Write Descriptor

A Query Request operation with WRITE DESCRIPTOR opcode is sent by the host to the device. The host builds a QUERY REQUEST UPIU places a WRITE DESCRIPTOR opcode within the UPIU, sets the appropriate values in the required fields and sends that UPIU to the target device. A QUERY REQUEST UPIU with a WRITE DESCRIPTOR opcode will additionally include a data segment that contains the descriptor data the host is sending to the device.

Upon reception of the QUERY REQUEST UPIU the device will decode the WRITE DESCRIPTOR opcode field and access the descriptor indicated by the DESCRIPTOR IDN field, the INDEX field and the SELECTOR field. The device will extract the descriptor data within the data segment of the UPIU and will internally overwrite the addressed descriptor with the extracted data. When the device has completed this process it will then notify the host of the success or failure of this operation by returning to the host a QUERY RESPONSE UPIU with the RESPONSE field containing the response code.

After the transmission of the QUERY RESPONSE UPIU the device will consider the operation complete.

Upon reception of the QUERY RESPONSE UPIU the host will decode the RESPONSE field and other fields within the UPIU and take the appropriate completion action. Upon reception of the QUERY RESPONSE UPIU the host will consider that the operation is complete.

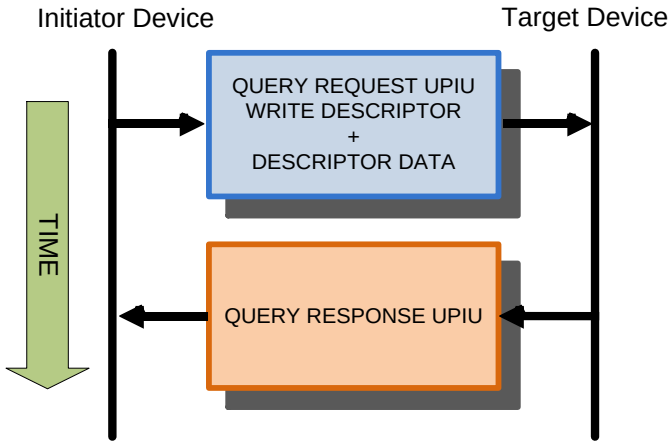


Figure 14-3 — Write Request Descriptor

14.1.4 Descriptor Definitions

14.1.4.1 Generic Descriptor Format

The format of all descriptors begins with a header which contains the length of the descriptor and a type value that indentifies the specific type of descriptor. The length value includes the length of the header plus any additional data.

Table 14-2 — Generic Descriptor Format

DEVICE DESCRIPTOR			
Offset	**Size**	**Name**	**Description**
0	1	bLength	Size of this descriptor inclusive = N
1	1	bDescriptorIDN	Descriptor Type Identifier
2 … N-1	N-2	DATA	Descriptor Information

For unit descriptors, the format includes an indication for the unit being addressed.

Table 14-3 — Logical Unit Descriptor Format

UNIT DESCRIPTOR			
Offset	**Size**	**Name**	**Description**
0	1	bLength	Size of this descriptor inclusive = N
1	1	bDescriptorIDN	Descriptor Type Identifier
2	1	bUnitIndex	Unit index - 00h to the number of LU specified by bMaxNumberLU
3 ... N-1	N-3	DATA	Descriptor Information

6226 **14.1.4.2 Device Descriptor**

6227 This is the main descriptor and should be the first descriptor retrieved as it specifies the device class and
6228 sub-class and the protocol (command set) to use to access this device and the maximum number of logical
6229 units contained within the device. The Device Descriptor is read only, some of its parameters may be
6230 changed writing the corresponding parameter of the Configuration Descriptor.

6231 In a QUERY REQUEST UPIU, the Device Descriptor is addressed setting: DESCRIPTOR IDN = 00h,
6232 INDEX = 00h and SELECTOR = 00h.

6233 **Table 14-4 — Device Descriptor**

DEVICE DESCRIPTOR					
Offset	**Size**	**Name**	**MDV (1)**	**User Conf.**	**Description**
00h	1	bLength	40h	No	Size of this descriptor
01h	1	bDescriptorIDN	00h	No	Device Descriptor Type Identifier
02h	1	bDevice	00h	No	Device type 00h: Device Others: Reserved
03h	1	bDeviceClass	00h	No	UFS Device Class 00h: Mass Storage Others: Reserved
04h	1	bDeviceSubClass	Device specific	No	UFS Mass Storage Subclass Bits (0/1) specify as follows: Bit 0: Bootable / Non-Bootable Bit 1: Embedded / Removable Bit 2: Reserved (for JESD220-1 (UME)) Others: Reserved Examples: 00h: Embedded Bootable 01h: Embedded Non-Bootable 02h: Removable Bootable 03h: Removable Non-Bootable
05h	1	bProtocol	00h	No	Protocol supported by UFS Device 00h: SCSI Others: Reserved
06h	1	bNumberLU	00h	Yes(3)	Number of Logical Units bNumberLU does not include well known logical units.
07h	1	bNumberWLU	04h	No	Number of Well known Logical Units
08h	1	bBootEnable	00h	Yes	Boot Enable Indicate whether the device is enabled for boot. 00h: Boot feature disabled 01h: Bootable feature enabled Others: Reserved

DEVICE DESCRIPTOR					
Offset	**Size**	**Name**	**MDV (1)**	**User Conf.**	**Description**
09h	1	bDescrAccessEn	00h	Yes	Descriptor Access Enable Indicate whether the Device Descriptor can be read after the partial initialization phase of the boot sequence 00h: Device Descriptor access disabled 01h: Device Descriptor access enabled Others: Reserved
0Ah	1	bInitPowerMode	01h	Yes	Initial Power Mode bInitPowerMode defines the Power Mode after device initialization or hardware reset 00h: UFS-Sleep Mode 01h: Active Mode Others: Reserved
0Bh	1	bHighPriorityLUN	7Fh	Yes	High Priority LUN bHighPriorityLUN defines the high priority logical unit. Valid values are: from 0 to the number of LU specified by bMaxNumberLU, and 7Fh. If this parameter value is 7Fh all logical units have the same priority.
0Ch	1	bSecureRemovalType	00h	Yes	Secure Removal Type 00h: information removed by an erase of the physical memory 01h: information removed by overwriting the addressed locations with a single character followed by an erase. 02h: information removed by overwriting the addressed locations with a character, its complement, then a random character. 03h: information removed using a vendor define mechanism. Others: Reserved
0Dh	1	bSecurityLU	01h	No	Support for security LU 00h: not supported 01h: RPMB Others: Reserved

DEVICE DESCRIPTOR					
Offset	**Size**	**Name**	**MDV (1)**	**User Conf.**	**Description**
0Eh	1	bBackgroundOpsTermLat	Device specific	No	Background Operations Termination Latency bBackgroundOpsTermLat defines the maximum latency for the termination of ongoing background operations. When the device receives a COMMAND UPIU with a transfer request, the device shall start the data transfer and send a DATA IN UPIU or a RTT UPIU within the latency declared in bBackgroundOpsTermLat. The latency is expressed in units of 10 ms (e.g., 01h= 10ms, FFh= 2550ms). The latency is undefined if the value of this parameter is zero.
0Fh	1	bInitActiveICCLevel	00h	Yes	Initial Active ICC Level bInitActiveICCLevel defines the bActiveICCLevel value after power on or reset. Valid range from 00h to 0Fh.
10h	2	wSpecVersion	0210h	No	Specification version Bits[15:8] = Major version in BCD format Bits[7:4] = Minor version in BCD format Bits[3:0] = Version suffix in BCD format Example: 3.21=0321h
12h	2	wManufactureDate	Device specific	No	Manufacturing Date BCD version of the device manufacturing date, i.e. August 2010 = 0810h
14h	1	iManufacturerName	Device specific	No	Manufacturer Name Index to the string which contains the Manufacturer Name.
15h	1	iProductName	Device specific	No	Product Name Index to the string which contains the Product Name.
16h	1	iSerialNumber	Device specific	No	Serial Number Index to the string which contains the Serial Number.
17h	1	iOemID	Device specific	No	OEM ID Index to the string which contains the OEM ID.

DEVICE DESCRIPTOR					
Offset	**Size**	**Name**	**MDV (1)**	**User Conf.**	**Description**
18h	2	wManufacturerID	Device specific	No	Manufacturer ID Manufacturer ID as defined in JEDEC JEP106, Standard Manufacturer's Identification Code.
1Ah	1	bUD0BaseOffset	10h	No	Unit Descriptor 0 Base Offset Offset of the Unit Descriptor 0 configurable parameters within the Configuration Descriptor.
1Bh	1	bUDConfigPLength	10h	No	Unit Descr. Config. Param. Length Total size of the configurable Unit Descriptor parameters.
1Ch	1	bDeviceRTTCap	Device specific	No	RTT Capability of device Maximum number of outstanding RTTs supported by device. The minimum value is 2.
1Dh	2	wPeriodicRTCUpdate	0000h	Yes	Frequency and method of Real-Time Clock update. Bits [15:10] Reserved Bits [9] TIME_BASELINE 0b: Time elapsed from the previous dSecondsPassed update. 1b: Absolute time elapsed from January 1st 2010 00:00. NOTE If the host device has a Real Time Clock it should use TIME BASELINE = '1'. If the host device has no Real Time Clock it should use TIME BASELINE = '0'. Bits [8:6] TIME_UNIT 000b = Undefined 001b = Months 010b = Weeks 011b = Days 100b = Hours 101b = Minutes 110b = Reserved 111b = Reserved Bits [5:0] TIME_PERIOD If TIME_UNIT is 0 TIME_PERIOD is ignored and the period between RTC update is not defined. All fields are configurable by the host.

DEVICE DESCRIPTOR					
Offset	**Size**	**Name**	**MDV (1)**	**User Conf.**	**Description**
1Fh	1	bUFSFeaturesSupport	Device specific	No	UFS Features Support This field indicates which features are supported by the device. A feature is supported if the related bit is set to one. bit[0]: Field Firmware Update (FFU) bit[1]: Production State Awarness (PSA) bit[2]: Device Life Span Others: Reserved Bit 0 shall be set to one.
20h	1	bFFUTimeout	Device specific	No	Field Firmware Update Timeout The maximum time, in seconds, that access to the device is limited or not possible through any ports associated due to execution of a WRITE BUFFER command. A value of zero indicates that no timeout is provided.
21h	1	bQueueDepth	Device specific	No	Queue Depth 0: The device implements the per-LU queueing architecture. 1.. 255: The device implements the shared queueing architecture. This parameter indicates the depth of the shared queue. If bLUQueueDepth>0 for any LU (except RPMB LU), then bQueueDepth shall be 0.
22h	2	wDeviceVersion	Device specific	No	Device Version This field provides the device version.
24h	1	bNumSecureWPArea	Device specific	No	Number of Secure Write Protect Areas This value specifies the total number of Secure Write Protect Areas supported by the device. The value shall be equal to or greater than bNumberLU and shall not exceed 32 (bNumberLU ≤ bNumSecureWPArea ≤ 32).
25h	4	dPSAMaxDataSize	Device specific	No	PSA Maximum Data Size This parameter specifies the maximum amount of data that may be written during the pre-soldering phase of the PSA flow. The value indicates the total amount of data for all logical units with bPSASensitive = 01h. Value expressed in units of 4 Kbyte.

DEVICE DESCRIPTOR					
Offset	**Size**	**Name**	**MDV (1)**	**User Conf.**	**Description**
29h	1	bPSAStateTimeout	Device specific	No	PSA State Timeout This parameter specifies the command maximum timeout for a change in bPSAState state. 00h means undefined. Otherwise, the formula to calculate the max timeout value is: Production State Timeout = 100us * 2^bPSAStateTimeout For example: 01h means 100us x 2^1 = 200us 02h means 100us x 2^2 = 400us 17h means 100us x 2^23 = 838.86s
2Ah	1	iProductRevisionLevel	Device specific	No	Product Revision Level Index to the string which contains the Product Revision Level
2Bh	5	Reserved			Reserved
30h	16	Reserved			Reserved for Unified Memory Extension standard

NOTE 1 The column “MDV” (Manufacturer Default Value) specifies parameter values after device manufacturing. Some parameters may be configured by the user writing the Configuration Descriptor.

NOTE 2 “User Conf.” column specifies which fields can be configured by the user writing the Configuration Descriptor: “Yes” means that the field can be configured, “No” means that the field is a capability of the device and cannot be changed by the user. The desired value shall be set in the equivalent parameter of the Configuration Descriptor.

NOTE 3 bNumberLU field value is calculated by the device based on bLUEnable field value in the Unit Descriptors.

6234

14.1.4.2 Device Descriptor (cont'd)

a) wManufacturerID

This parameter contains manufacturer identification information for the device manufacturer. The Manufacturer ID is defined by JEDEC in Standard Manufacturer's identification code [JEP106]. The wManufacturerID consists of two fields: Manufacturer ID Code and Bank Index.

Table 14-5 — wManufacturerID definition

Bit / Byte	7	6	5	4	3	2	1	0
0	Bank Index							
1	Manufacturer ID Code							

b) Bank Index

This field contains an index value of the bank that contains the manufacturer identification code. The Bank Index value shall be equal to the number of the continuation fields that precede the manufacturer identification code as specified by [JEP106].

c) Manufacturer ID Code

Manufacturer identification code as defined by JEDEC in Standard Manufacturer's identification code [JEP106].

14.1.4.3 Configuration Descriptor

The device configuration set by the manufacturer can be modified by writing the Configuration Descriptor. In particular, the Configuration Descriptor allows to configure parameters included in the Device Descriptor and in the Unit Descriptors. The Configuration Descriptor can be written if bConfigDescrLock attribute value is equal to 00h. If bConfigDescrLock attribute value is 01h the Configuration Descriptor is locked and a write request shall fail.

There are up to four Configuration Descriptors.

The first Configuration Descriptor is addressed by setting DESCRIPTOR IDN = 01h, INDEX = 00h and SELECTOR = 00h in a QUERY REQUEST UPIU, and used to change configurable parameters of Device Descriptor and the first eight Unit Descriptors (LU 0 to LU 7).

The second Configuration Descriptor is addressed by setting DESCRIPTOR IDN = 01h, INDEX = 01h and SELECTOR = 00h in a QUERY REQUEST UPIU, and used to change configurable parameters of the next eight Unit Descriptors (LU 8 to LU 15).

The third Configuration Descriptor is addressed by setting DESCRIPTOR IDN = 01h, INDEX = 02h and SELECTOR = 00h in a QUERY REQUEST UPIU, and used to change configurable parameters of the next eight Unit Descriptors (LU 16 to LU 23).

The fourth Configuration Descriptor is addressed by setting DESCRIPTOR IDN = 01h, INDEX = 03h and SELECTOR = 00h in a QUERY REQUEST UPIU, and used to change configurable parameters of the last eight Unit Descriptors (LU 24 to LU 31).

14.1.4.3 Configuration Descriptor (cont'd)

Table 14-6 shows the Configuration Descriptor format with DESCRIPTOR IDN = 01h, INDEX = 00h and SELECTOR = 00h: the lower address space is used for Configuration Descriptor header and user configurable parameters included in the Device Descriptor, then there are eight address spaces for user configurable parameters included in the Unit Descriptors of LU 0 to LU 7.

Table 14-6 — Configuration Descriptor Format (INDEX = 00h)

Offset	Description
00h … (B-1)h	Configuration Descriptor header and Device Descriptor configurable parameters
(B)h … (B+L-1)h	Unit Descriptor 0 configurable parameters
(B+L)h … (B+2*L-1)h	Unit Descriptor 1 configurable parameters
…	…
(B+7*L)h … (B+8*L-1)h	Unit Descriptor 7 configurable parameters

Table 14-7 shows the Configuration Descriptor format with DESCRIPTOR IDN = 01h, INDEX = 01h and SELECTOR = 00h: the lower address space is used for Configuration Descriptor header, then there are eight address spaces for user configurable parameters included in the Unit Descriptors of LU 8 to LU 15.

Table 14-7 — Configuration Descriptor Format (INDEX = 01h)

Offset	Description
00h … (B-1)h	Configuration Descriptor header
(B)h … (B+L-1)h	Unit Descriptor 8 configurable parameters
(B+L)h … (B+2*L-1)h	Unit Descriptor 9 configurable parameters
…	…
(B+7*L)h … (B+8*L-1)h	Unit Descriptor 15 configurable parameters

Table 14-8 shows the Configuration Descriptor format with DESCRIPTOR IDN = 01h, INDEX = 02h and SELECTOR = 00h: the lower address space is used for Configuration Descriptor header, then there are eight address spaces for user configurable parameters included in the Unit Descriptors of LU 16 to LU 23.

Table 14-8 — Configuration Descriptor Format (INDEX = 02h)

Offset	Description
00h … (B-1)h	Configuration Descriptor header
(B)h … (B+L-1)h	Unit Descriptor 16 configurable parameters
(B+L)h … (B+2*L-1)h	Unit Descriptor 17 configurable parameters
…	…
(B+7*L)h … (B+8*L-1)h	Unit Descriptor 23 configurable parameters

14.1.4.3 Configuration Descriptor (cont'd)

Table 14-9 shows the Configuration Descriptor format with DESCRIPTOR IDN = 01h, INDEX = 03h and SELECTOR = 00h: the lower address space is used for Configuration Descriptor header, then there are eight address spaces for user configurable parameters included in the Unit Descriptors of LU 24 to LU 32.

Parameter offsets are defined based on:

- Offset of the Unit Descriptor 0 configurable parameters within the Configuration Descriptor (B).
- Length of Unit Descriptor configurable parameters (L)

Values for B and L are stored in the Device Descriptor parameters bUD0BaseOffset and bUDConfigPLength respectively.

Table 14-9 — Configuration Descriptor Format (INDEX = 03h)

Offset	Description
00h … (B-1)h	Configuration Descriptor header
(B)h … (B+L-1)h	Unit Descriptor 24 configurable parameters
(B+L)h … (B+2*L-1)h	Unit Descriptor 25 configurable parameters
…	…
(B+7*L)h … (B+8*L-1)h	Unit Descriptor 31 configurable parameters
NOTE 1 B is offset of the Unit Descriptor 0/8/16/24 configurable parameters within the Configuration Descriptor, L is the total size of the configurable Unit Descriptor parameters.	

6299 **14.1.6.3 Configuration Descriptor (cont'd)**

6300 Table 14-10 defines Configuration Descriptor header and Device Descriptor configurable parameters
6301 within the Configuration Descriptor with INDEX = 00h.

6302 See 14.1.4.2, Device Descriptor, for details about configurable parameters.

6303 **Table 14-10 — Configuration Descr. Header and Device Descr. Conf. parameters (INDEX = 00h)**

Configuration Descriptor Header and Device Descriptor configurable parameters				
Offset	**Size**	**Name**	**MDV (1,2)**	**Description**
00h	1	bLength	90h	Size of this descriptor
01h	1	bDescriptorIDN	01h	Configuration Descriptor Type Identifier
02h	1	bConfDescContinue	00h	00h : This value indicates that this is the last Configuration Descriptor in a sequence of write descriptor query requests. Device shall perform internal configuration based on received Configuration Descriptor(s). 01h : This value indicates that this is not the last Configuration Descriptor in a sequence of write descriptor query requests. Other Configuration Descriptors will be sent by host. Therefore the device should not perform the internal configuration yet.
03h	1	bBootEnable		Boot Enable Enables to boot feature.
04h	1	bDescrAccessEn		Descriptor Access Enable Enables access to the Device Descriptor after the partial initialization phase of the boot sequence.
05h	1	bInitPowerMode		Initial Power Mode Configures the power mode after device initialization or hardware reset.
06h	1	bHighPriorityLUN		High Priority LUN Configures the high priority logical unit.
07h	1	bSecureRemovalType		Secure Removal Type Configures the secure removal type.
08h	1	bInitActiveICCLevel		Initial Active ICC Level Configures the ICC level in Active mode after device initialization or hardware reset
09h	2	wPeriodicRTCUpdate		Frequency and method of Real-Time Clock update (see Device Descriptor).
0Bh:0Fh	5	Reserved		

NOTE 1 The column "MDV" (Manufacturer Default Value) specifies parameter values after device manufacturing.

NOTE 2 See Table 14-4, Device Descriptor, for the default parameters value set by the device manufacturer.

6304

6305 **14.1.6.3 Configuration Descriptor (cont'd)**

6306 Table 14-11 defines Configuration Descriptor header within the Configuration Descriptor with INDEX =
6307 01h, 02h, or 03h.

6308 **Table 14-11— Configuration Descr. Header with INDEX = 01h/02h/03h**

Configuration Descriptor Header				
Offset	**Size**	**Name**	**MDV (1)**	**Description**
00h	1	bLength	90h	Size of this descriptor
01h	1	bDescriptorIDN	01h	Configuration Descriptor Type Identifier
02h	1	bConfDescContinue	00h	00h : This value indicates that this is the last Configuration Descriptor in a sequence of write descriptor query requests. Device shall perform internal configuration based on received Configuration Descriptor(s). 01h : This value indicates that this is not the last Configuration Descriptor in a sequence of write descriptor query requests. Other Configuration Descriptors will be sent by host. Therefore the device should not perform the internal configuration yet.
02h:0Fh	14	Reserved		
NOTE 1 The column "MDV" (Manufacturer Default Value) specifies parameter values after device manufacturing.				

6309 Table 14-12 defines the Unit Descriptors user configurable parameters within the Configuration
6310 Descriptor. See 14.1.4.5, Unit Descriptor.

6311 **Table 14-12 — Unit Descriptor configurable parameters**

Unit Descriptor configurable parameters			
Offset	**Size**	**Name**	**Description**
00h	1	bLUEnable	Logical Unit Enable
01h	1	bBootLunID	Boot LUN ID
02h	1	bLUWriteProtect	Logical Unit Write Protect
03h	1	bMemoryType	Memory Type
04h	4	dNumAllocUnits	Number of Allocation Units Number of allocation units assigned to the logical unit. The value shall be calculated considering the capacity adjustment factor of the selected memory type
08h	1	bDataReliability	Data Reliability
09h	1	bLogicalBlockSize	Logical Block Size
0Ah	1	bProvisioningType	Provisioning Type
0Bh	2	wContextCapabilities	Context Capabilities
0Dh:0Fh	3	Reserved	

6312 **14.1.4.4 Geometry Descriptor**

6313 The Geometry Descriptor describes the device geometric parameters. In a QUERY REQUEST UPIU, the
6314 Geometry Descriptor is addressed setting: DESCRIPTOR IDN = 07h, INDEX = 00h and SELECTOR =
6315 00h.

6316 **Table 14-13 — Geometry Descriptor**

GEOMETRY DESCRIPTOR				
Offset	**Size**	**Name**	**Value**	**Description**
00h	1	bLength	48h	Size of this descriptor
01h	1	bDescriptorIDN	07h	Geometry Descriptor Type Identifier
02h	1	bMediaTechnology	00h	Reserved
03h	1	Reserved	00h	Reserved
04h	8	qTotalRawDeviceCapacity	Device specific	Total Raw Device Capacity Total memory quantity available to the user to configure the device logical units (RPMB excluded). It is expressed in unit of 512 bytes
0Ch	1	bMaxNumberLU	Device specific	Maximum number of Logical Unit supported by the UFS device 00h: 8 Logical units 01h: 32 Logical units Others: Reserved
0Dh	4	dSegmentSize	Device specific	Segment Size Value expressed in unit of 512 bytes
11h	1	bAllocationUnitSize	Device specific	Allocation Unit Size Value expressed in number of Segments Each logical unit can be allocated as a multiple of Allocation Units.
12h	1	bMinAddrBlockSize	Device specific	Minimum addressable block size Value expressed in unit of 512 bytes. Its minimum value is 08h, which corresponds to 4 Kbyte.
13h	1	bOptimalReadBlockSize	Device specific	Optimal Read Block Size Value expressed in unit of 512 bytes. This is an optional parameter. A value of zero indicates that this information is not available. If not zero, bOptimalReadBlockSize shall be equal to or greater than bMinAddrBlockSize.
14h	1	bOptimalWriteBlockSize	Device specific	Optimal Write Block Size Value expressed in unit of 512 bytes. bOptimalWriteBlockSize shall be equal to or greater than bMinAddrBlockSize.

GEOMETRY DESCRIPTOR				
Offset	**Size**	**Name**	**Value**	**Description**
15h	1	bMaxInBufferSize	Device specific	Max. data-in buffer size Value expressed in unit of 512 bytes. Its minimum value is 08h, which corresponds to 4 Kbyte
16h	1	bMaxOutBufferSize	Device specific	Max. data-out buffer size Value expressed in unit of 512 bytes. Its minimum value is 08h, which corresponds to 4 Kbyte
17h	1	bRPMB_ReadWriteSize	Device specific	Maximum number of RPMB frames (256-byte of data) allowed in Security Protocol In and Security Protocol Out (i.e., associated with a single command UPIU) If the data to be transferred is larger than bRPMB_ReadWriteSize x 256 bytes, the host will transfer it using multiple Security Protocol In/Out commands
18h	1	bDynamicCapacityResourcePolicy	Device specific	Dynamic Capacity Resource Policy This parameter specifies the device spare blocks resource management policy: 00h: Spare blocks resource management policy is per logical unit. The host should release amount of logical blocks from each logical unit as asked by the device. 01h: Spare blocks resource management policy is per memory type. The host may deallocate the required amount of logical blocks from any logical units with the same bMemoryType.
19h	1	bDataOrdering	Device specific	Support for out-of-order data transfer 00h: out-of-order data transfer is not supported by the device, in-order data transfer is required. 01h: out-of-order data transfer is supported by the device All others: Reserved
1Ah	1	bMaxContexIDNumber	Device specific	Max. available number of contexts which are supported by the device. Minimum number of supported contexts shall be 5.
1Bh	1	bSysDataTagUnitSize	Device specific	bSysDataTagUnitSize provides system data tag unit size, which can be calculated as in the following (in bytes) Tag Unit Size = $2^{bSysDataTagUnitSize}$ x bMinAddrBlockSize x 512

GEOMETRY DESCRIPTOR				
Offset	**Size**	**Name**	**Value**	**Description**
1Ch	1	bSysDataTagResSize	Device specific	This field is defined to inform the host about the maximum storage area size in bytes allocated by the device to handle system data by the tagging mechanism: $\text{System Data Tag Resource Size} = \text{Tag Unit Size} \times floor\left(\frac{\text{qTotalRawDeviceCapacity x } 2^{\text{bSysDataTagResSize}-10}}{Tag\ Unit\ Size}\right)$ The range of valid bSysDataTagResSize values is from 0 to 6. Values in range of 07h to FFh are reserved. The formula covers a range from about 0.1% to 6.25% of the device capacity
1Dh	1	bSupportedSecRTypes	Device specific	Supported Secure Removal Types Bit map which represents the supported Secure Removal types. bit 0: information removed by an erase of the physical memory bit 1: information removed by overwriting the addressed locations with a single character followed by an erase. bit 2: information removed by overwriting the addressed locations with a character, its complement, then a random character. bit 3: information removed using a vendor define mechanism. Others: Reserved A value of one means that the corresponding Secure Removal type is supported.

GEOMETRY DESCRIPTOR				
Offset	**Size**	**Name**	**Value**	**Description**
1Eh	2	wSupportedMemoryTypes	Device specific	Supported Memory Types Bit map which represents the supported memory types. bit 0: Normal memory type bit 1: System code memory type bit 2: Non-Persistent memory type bit 3: Enhanced memory type 1 bit 4: Enhanced memory type 2 bit 5: Enhanced memory type 3 bit 6: Enhanced memory type 4 bit 7: Reserved … bit 14: Reserved bit 15: RPMB memory type A value one means that the corresponding memory type is supported. Bit 0 and bit 15 shall be one for all UFS device.
20h	4	dSystemCodeMaxNAllocU	Device specific	Max Number of Allocation Units for the System Code memory type Maximum available quantity of System Code memory type for the entire device. Value expressed in number of Allocation Unit
24h	2	wSystemCodeCapAdjFac	Device specific	Capacity Adjustment Factor for the System Code memory type This parameter is the ratio between the capacity obtained with the Normal memory type and the capacity obtained with the System Code memory type for the same amount of allocation units. $CapacityAdjFactor = Capacity_{NormalMem} / Capacity_{SystemCode}$ wSystemCodeCapAdjFac = INTEGER(256 × CapactiyAdjFactor)
26h	4	dNonPersistMaxNAllocU	Device specific	Max Number of Allocation Units for the Non-Persistent memory type Maximum available quantity of Non-Persistent memory type for the entire device. Value expressed in number of Allocation Unit

GEOMETRY DESCRIPTOR				
Offset	**Size**	**Name**	**Value**	**Description**
2Ah	2	wNonPersistCapAdjFac	Device specific	Capacity Adjustment Factor for the Non-Persistent memory type This parameter is the ratio between the capacity obtained with the Normal memory type and the capacity obtained with the Non-Persistent memory type for the same amount of allocation units. CapacityAdjFactor = $Capacity_{NormalMem}$ / $Capacity_{NonPersist}$ wNonPersistCapAdjFac = INTEGER(256 × CapactiyAdjFactor)
2Ch	4	dEnhanced1MaxNAllocU	Device specific	Max Number of Allocation Units for the Enhanced memory type 1 Maximum available quantity of Enhanced memory type 1 for the entire device Value expressed in number of Allocation Unit
30h	2	wEnhanced1CapAdjFac	Device specific	Capacity Adjustment Factor for the Enhanced memory type 1 This parameter is the ratio between the capacity obtained with the Normal memory type and the capacity obtained with the Enhanced memory type 1 for the same amount of allocation units. CapacityAdjFactor = $Capacity_{NormalMem}$ / $Capacity_{Enhanced1}$ wEnhanced1CapAdjFac = INTEGER(256 × CapacityAdjFactor)
32h	4	dEnhanced2MaxNAllocU	Device specific	Max Number of Allocation Units for the Enhanced memory type 2 Maximum available quantity of Enhanced memory type 2 for the entire device Value expressed in number of Allocation Unit
36h	2	wEnhanced2CapAdjFac	Device specific	Capacity Adjustment Factor for the Enhanced memory type 2 This parameter is the ratio between the capacity obtained with the Normal memory type and the capacity obtained with the Enhanced memory type 2 for the same amount of allocation units. CapacityAdjFactor = $Capacity_{NormalMem}$ / $Capacity_{Enhanced2}$ wEnhanced2CapAdjFac = INTEGER(256 × CapacityAdjFactor)

GEOMETRY DESCRIPTOR				
Offset	**Size**	**Name**	**Value**	**Description**
38h	4	dEnhanced3MaxNAllocU	Device specific	Max Number of Allocation Units for the Enhanced memory type 3 Maximum available quantity of Enhanced memory type 3 for the entire device Value expressed in number of Allocation Unit
3Ch	2	wEnhanced3CapAdjFac	Device specific	Capacity Adjustment Factor for the Enhanced memory type 3 This parameter is the ratio between the capacity obtained with the Normal memory type and the capacity obtained with the Enhanced memory type 3 for the same amount of allocation units. $CapacityAdjFactor = Capacity_{NormalMem} / Capacity_{Enhanced3}$ wEnhanced3CapAdjFac = INTEGER(256 × CapacityAdjFactor)
3Eh	4	dEnhanced4MaxNAllocU	Device specific	Max Number of Allocation Units for the Enhanced memory type 4 Maximum available quantity of Enhanced memory type 4 for the entire device Value expressed in number of Allocation Unit
42h	2	wEnhanced4CapAdjFac	Device specific	Capacity Adjustment Factor for the Enhanced memory type 4 This parameter is the ratio between the capacity obtained with the Normal memory type and the capacity obtained with the Enhanced memory type 4 for the same amount of allocation units. $CapacityAdjFactor = Capacity_{NormalMem} / Capacity_{Enhanced4}$ wEnhanced4CapAdjFac = INTEGER(256 × CapacityAdjFactor)

GEOMETRY DESCRIPTOR				
Offset	**Size**	**Name**	**Value**	**Description**
44h	4	dOptimalLogicalBlockSize	Device specific	Optimal Logical Block Size bit [3:0]: Normal memory type bit [7:4]: System code memory type bit [11: 8]: Non-Persistent memory type bit [15:12]: Enhanced memory type 1 bit [19:16]: Enhanced memory type 2 bit [23:20]: Enhanced memory type 3 bit [27:24]: Enhanced memory type 4 bit [31:28]:Reserved The optimal logical block size for each memory type can be calculated from the related dOptimalLogicalBlockSize field as indicated in the following: Optimal Logical Block Size = 2^ (dOptimalLogicalBlockSize field) x bMinAddrBlockSize x 512 byte
NOTE 1 The Capacity Adjustment Factor value for Normal memory type is one.				

6317

6318 **14.1.4.5 Unit Descriptor**

6319 This page describes specific characteristics and capabilities of an individual logical unit, for example
6320 geometry of the device and maximum addressable item. There are up to thirty two unit descriptors. In a
6321 QUERY REQUEST UPIU, an Unit Descriptor is addressed setting: DESCRIPTOR IDN = 02h, INDEX =
6322 unit index, and SELECTOR = 00h.

6323 **Table 14-14 — Unit Descriptor**

UNIT DESCRIPTOR					
Offset	**Size**	**Name**	**MDV (1)**	**User Conf. (2)**	**Description**
00h	1	bLength	23h	No	Size of this descriptor
01h	1	bDescriptorIDN	02h	No	Unit Descriptor Type Identifier
02h	1	bUnitIndex	00h to the number of LU specified by bMaxNumberLU	No	Unit Index
03h	1	bLUEnable	00h	Yes	Logical Unit Enable 00h: Logical Unit disabled 01h: Logical Unit enabled Others: Reserved
04h	1	bBootLunID	00h	Yes	Boot LUN ID 00h: Not bootable 01h: Boot LU A 02h: Boot LU B Others: Reserved.
05h	1	bLUWriteProtect	00h	Yes	Logical Unit Write Protect 00h: LU not write protected 01h: LU write protected when fPowerOnWPEn =1 02h: LU permanently write protected when fPermanentWPEn =1 03h: Reserved (for UFS Security Extension standard) Others: Reserved
06h	1	bLUQueueDepth	Device specific	No	Logical Unit Queue Depth 0 : LU queue not available (shared queuing is used) [1.. 255] : LU queue depth If any bQueueDepth>0, bLUQueueDepth shall be 0.
07h	1	bPSASensitive	Device specific	No(3)	00h: LU is not sensitive to soldering 01h: LU is sensitive to soldering Others: Reserved

UNIT DESCRIPTOR					
Offset	**Size**	**Name**	**MDV (1)**	**User Conf. (2)**	**Description**
08h	1	bMemoryType	00h	Yes	Memory Type bMemoryType defines logical unit memory type. 00h: Normal Memory 01h: System code memory type 02h: Non-Persistent memory type 03h: Enhanced memory type 1 04h: Enhanced memory type 2 05h: Enhanced memory type 3 06h: Enhanced memory type 4 Others: Reserved
09h	1	bDataReliability	00h	Yes	Data Reliability bDataReliability defines the device behavior when a power failure occurs during a write operation to the logical unit 00h: the logical unit is not protected. Logical unit's entire data may be lost as a result of a power failure during a write operation 01h: logical unit is protected. Logical unit's data is protected against power failure. Others: Reserved
0Ah	1	bLogicalBlockSize	0Ch	Yes	Logical Block Size The size of addressable logical blocks is equal to the result of exponentiation with as base the number two and as exponent the bLogicalBlockSize value: $2^{bLogicalBlockSize}$ (i.e., bLogicalBlockSize = 0Ch corresponds to 4 Kbyte Logical Block Size). Its minimum value is 0Ch, which corresponds to 4 Kbyte
0Bh	8	qLogicalBlockCount	00h	Yes(4)	Logical Block Count Total number of addressable logical blocks in the logical unit
13h	4	dEraseBlockSize	00h(5)	No	Erase Block Size In number of Logical Blocks
17h	1	bProvisioningType	00h	Yes	Provisioning Type 00h:Thin Provisioning is disabled (default) 02h:Thin Provisioning is enabled and TPRZ = 0 03h:Thin Provisioning is enabled and TPRZ = 1 Others: Reserved

UNIT DESCRIPTOR					
Offset	**Size**	**Name**	**MDV (1)**	**User Conf. (2)**	**Description**
18h: 1Fh	8	qPhyMemResourceCount	Device specific	No	Physical Memory Resource Count Total physical memory resources available in the logical unit. Value expressed in units of Logical Block Size.
20h	2	wContextCapabilities	00h	Yes	Bits [3:0]: MaxContextID is the maximum amount of contexts that the LU supports simultaneously. The sum of all MaXContextID must not exceed bMaxContexIDNumber. Bits [6:4]: LARGE_UNIT_MAX_MULTIPLIER_M1 is the highest multiplier that can be configured for Large Unit contexts, minus one. Large Unit contexts may be configured to have a multiplier in the range: 1 ≤ multiplier ≤ (LARGE_UNIT_MAX_MULTIPLIER_M1 + 1) This field is read only. Bit [15:7]: Reserved.
22h	1	bLargeUnitGranularity_M1	Device specific	No	Granularity of the Large Unit, minus one. Large Unit Granularity = 1MB * (bLargeUnitGranularity_M1 + 1)

NOTE 1 The column “MDV” (Manufacturer Default Value) specifies parameters value after device manufacturing. Some fields may be configured by the user writing the Configuration Descriptor.

NOTE 2 “User Conf.” column specifies which fields can be configured by the user writing the Configuration Descriptor: “Yes” means that the field can be configured, “No” means that the field is a capability of the device and cannot be changed by the user. The desired value shall be set in the equivalent parameter of the Configuration Descriptor.

NOTE 3 bPSASensitive value is updated automatically by the device after device configuration.

NOTE 4 qLogicalBlockCount can be configured setting the dNumAllocUnits parameter of the Configuration Descriptor.

NOTE 5 dEraseBlockSize value is updated automatically by the device after device configuration.

6324

6325 **14.1.4.6 RPMB Unit Descriptor**

6326 In a QUERY REQUEST UPIU, the RPMB Unit Descriptor is addressed setting: DESCRIPTOR IDN =
6327 02h, INDEX = C4h, and SELECTOR = 00h.

6328 **Table 14-15 — RPMB Unit Descriptor**

RPMB UNIT DESCRIPTOR					
Offset	**Size**	**Name**	**Value**	**User Conf.**	**Description**
00h	1	bLength	23h	No	Size of this descriptor
01h	1	bDescriptorIDN	02h	No	Unit Descriptor Type Identifier
02h	1	bUnitIndex	C4h	No	Unit Index
03h	1	bLUEnable	01h	No	Logical Unit Enable 01h: Logical Unit enabled
04h	1	bBootLunID	00h	No	Boot LUN ID 00h: Not bootable
05h	1	bLUWriteProtect	00h	No	Logical Unit Write Protect 00h: LU not write protected
06h	1	bLUQueueDepth	Device specific	No	Logical Unit Queue Depth 0: RPMB LU queue not available (shared queuing is used) 1: Queue depth available in RPMB LU. Only 1 task may be queued at any given time
07h	1	bPSASensitive	Device specific	No	00h: LU is not sensitive to soldering 01h: LU is sensitive to soldering Others: Reserved
08h	1	bMemoryType	0Fh	No	Memory Type 0Fh: RPMB Memory Type
09h	1	Reserved	00h	No	Reserved
0Ah	1	bLogicalBlockSize	08h	No	Logical Block Size The size of addressable logical blocks is equal to the result of exponentiation with as base the number two and as exponent the bLogicalBlockSize value: $2^{bLogicalBlockSize}$ (i.e., bLogicalBlockSize = 08h corresponds to 256 byte Logical Block Size)
0Bh	8	qLogicalBlockCount	Device specific	No	Logical Block Count Total number of addressable logical blocks in the RPMB LU. For RPMB, Logical Block Count shall be a multiple of 512 (i.e., 128 Kbyte)
13h	4	dEraseBlockSize	00h	No	Erase Block Size In number of Logical Blocks. For RPMB, Erase Block Size is ignored; set to ‘0’

RPMB UNIT DESCRIPTOR					
Offset	**Size**	**Name**	**Value**	**User Conf.**	**Description**
17h	1	bProvisioningType	00h	No	Provisioning Type 00h:Thin Provisioning is disabled
18h: 1Fh	8	qPhyMemResourceCount		No	Physical Memory Resource Count Total physical memory resources available in the logical unit. Value expressed in units of bLogicalBlockSize. The dynamic device capacity feature does not apply to the RPMB well known logical unit therefore qPhyMemResourceCount value is always equal to qLogicalBlockCount value
20h: 22h	3	Reserved	0000h		

6329

14.1.4.7 Power Parameters Descriptor

This descriptor contains information about the power capabilities and power states of the device. In a QUERY REQUEST UPIU, the Power Parameters Descriptor is addressed setting: DESCRIPTOR IDN = 08h, INDEX = 00h, and SELECTOR = 00h.

Table 14-16 — Power Parameters Descriptor

POWER PARAMETERS DESCRIPTOR				
Offset	**Size**	**Name**	**Value**	**Description**
00h	1	bLength	62h	Size of this descriptor
01h	1	bDescriptorIDN	08h	Power Parameters Descriptor Type Identifier
02h	2	wActiveICCLevelsVCC[0]	Device specific	Maximum VCC current value for bActiveICCLevel = 0
04h	2	wActiveICCLevelsVCC[1]	Device specific	Maximum VCC current value for bActiveICCLevel = 1
…	…	…	…	…
20h	2	wActiveICCLevelsVCC[15]	Device specific	Maximum VCC current value for bActiveICCLevel = 15
22h	2	wActiveICCLevelsVCCQ[0]	Device specific	Maximum VCCQ current value for bActiveICCLevel = 0
24h	2	wActiveICCLevelsVCCQ[1]	Device specific	Maximum VCCQ current value for bActiveICCLevel = 1
…	…	…	…	…
40h	2	wActiveICCLevelsVCCQ[15]	Device specific	Maximum VCCQ current value for bActiveICCLevel = 15
42h	32	wActiveICCLevelsVCCQ2[0]	Device specific	Maximum VCCQ2 current value for bActiveICCLevel = 0
44h	2	wActiveICCLevelsVCCQ2[1]	Device specific	Maximum VCCQ2 current value for bActiveICCLevel = 1
…	…	…	…	…
60h	2	wActiveICCLevelsVCCQ2[15]	Device specific	Maximum VCCQ2 current value for bActiveICCLevel = 15

14.1.4.8 Interconnect Descriptor

The Interconnect Descriptor contains the MIPI M-PHY® specification version number and the MIPI UniPro℠ specification version number. In a QUERY REQUEST UPIU, the Interconnect Descriptor is addressed setting: DESCRIPTOR IDN = 04h, INDEX = 00h, and SELECTOR = 00h.

Table 14-17 — Interconnect Descriptor

INTERCONNECT DESCRIPTOR				
Offset	**Size**	**Name**	**Value**	**Description**
00h	1	bLength	06h	Size of this descriptor
01h	1	bDescriptorIDN	04h	Interconnect Descriptor Type Identifier
02h	2	bcdUniproVersion	0160h	MIPI UniPro℠ version number in BCD format (i.e., version 3.21=0321h)
04h	2	bcdMphyVersion	0300h	MIPI M-PHY® version number in BCD format (i.e., version 3.21=0321h)

14.1.4.9 Manufacturer Name String Descriptor

This descriptor contains the UNICODE, left justified, manufacturer name string.

The content of the descriptor shall be identical to the content of the “VENDOR IDENTIFICATION” field in Inquiry Response Data. The length of the descriptor shall be 12h (18 decimal), containing exactly 8 UNICODE characters, to match “VENDOR IDENTIFICATION” field in Inquiry Response Data.

In a QUERY REQUEST UPIU, the Manufacturer Name String Descriptor is addressed setting: DESCRIPTOR IDN = 05h, INDEX = iManufacturerName (Device Descriptor parameter), and SELECTOR = 00h.

Table 14-18 — Manufacturer Name String

MANUFACTURER NAME STRING				
Offset	**Size**	**Name**	**Value**	**Description**
00h	1	bLength	12h	Size of this descriptor
01h	1	bDescriptorIDN	05h	String Descriptor Type Identifier
02h	2	UC[0]		Unicode string character
04h	-	-		-
10h	2	UC[7]		Unicode string character

14.1.4.10 Product Name String Descriptor

This descriptor contains the UNICODE, left justified, product name string.

The content of the descriptor shall be identical to the content of the "PRODUCT IDENTIFICATION" field in Inquiry Response Data. The length of the descriptor shall be 22h (34 decimal), containing exactly 16 UNICODE characters, to match "PRODUCT IDENTIFICATION" field in Inquiry Response Data.

In a QUERY REQUEST UPIU, the Product Name String Descriptor is addressed setting: DESCRIPTOR IDN = 05h, INDEX = iProductName (Device Descriptor parameter), and SELECTOR = 00h.

Table 14-19 — Product Name String

PRODUCT NAME STRING DESCRIPTOR				
Offset	**Size**	**Name**	**Value**	**Description**
00h	1	bLength	22h	Size of this descriptor
01h	1	bDescriptorIDN	05h	String Descriptor Type Identifier
02h	2	UC[0]		Unicode string character
04h	-	-		-
20h	2	UC[15]		Unicode string character

14.1.4.11 OEM ID String Descriptor

This descriptor contains the UNICODE OEM ID string that may consist of up to 126 UNICODE characters. Number of UNICODE characters is calculated by $(LENGTH - 2) \div 2$.

In a QUERY REQUEST UPIU, the OEM ID String Descriptor is addressed setting: DESCRIPTOR IDN = 05h, INDEX = iOemID (Device Descriptor parameter), and SELECTOR = 00h.

OEM_ID String descriptor is: readable, writeable if bConfigDescrLock attribute value is equal to 00h.

Table 14-20 — OEM_ID String

OEM ID STRING DESCRIPTOR				
Offset	**Size**	**Name**	**Value**	**Description**
00h	1	bLength	LENGTH	Size of this descriptor
01h	1	bDescriptorIDN	05h	String Descriptor Type Identifier
02h	2	UC[0]		Unicode string character
04h	-	-		-
LENGTH-2	2	UC[(LENGTH-2)÷2-1]		Unicode string character

14.1.4.12 Serial Number String Descriptor

This descriptor contains the UNICODE serial number string that may consist of up to 126 UNICODE characters. Number of UNICODE characters is calculated by $(LENGTH - 2) \div 2$.

In a QUERY REQUEST UPIU, the Serial Number String Descriptor is addressed setting: DESCRIPTOR IDN = 05h, INDEX = iSerialNumber (Device Descriptor parameter), and SELECTOR = 00h.

Table 14-21 — Serial Number String Descriptor

SERIAL NUMBER STRING DESCRIPTOR				
Offset	**Size**	**Name**	**Value**	**Description**
00h	1	bLength	LENGTH	Size of this descriptor
01h	1	bDescriptorIDN	05h	String Descriptor Type Identifier
02h	2	UC[0]		Unicode string character
04h	-	-		-
LENGTH-2	2	UC[(LENGTH-2)÷2-1]		Unicode string character

14.1.4.13 Product Revision Level String Descriptor

This descriptor contains the UNICODE, left justified, product revision level string.

The content of the descriptor shall be identical to the content of the "PRODUCT REVISION LEVEL" field in Inquiry Response Data. The length of the descriptor shall be Ah, containing exactly 4 UNICODE characters, to match "PRODUCT REVISION LEVEL" field in Inquiry Response Data.

In a QUERY REQUEST UPIU, the Product Revision Level String Descriptor is addressed setting: DESCRIPTOR IDN = 05h, INDEX = iProductRevisionLevel (Device Descriptor parameter), and SELECTOR = 00h.

Table 14-22 — Product Revision Level String

PRODUCT REVISION LEVEL STRING DESCRIPTOR				
Offset	**Size**	**Name**	**Value**	**Description**
00h	1	bLength	0Ah	Size of this descriptor
01h	1	bDescriptorIDN	05h	String Descriptor Type Identifier
02h	2	UC[0]		Unicode string character
04h	-	-		-
08h	2	UC[3]		Unicode string character

6387 **14.1.4.14 Device Health Descriptor**

6388 Device Health Descriptor provides information related to the health of the device.

6389 In a QUERY REQUEST UPIU, the Device Health Descriptor is addressed by setting: DESCRIPTOR
6390 IDN = 09h, INDEX = 00h and SELECTOR = 00h.

6391 **Table 14-23 — Device Health Descriptor**

Offset	Size	Name	Value	Description
00h	1	bLength	25h	Size of this descriptor
01h	1	bDescriptorIDN	09h	Device Health Descriptor Type Identifier
02h	1	bPreEOLInfo	Device specific	Pre End of Life Information This field provides indication about device life time reflected by average reserved blocks. 00h: Not defined 01h: Normal 02h: Warning. Consumed 80% of reserved blocks. 03h: Critical. Consumed 90% of reserved blocks. Others: Reserved
03h	1	bDeviceLifeTimeEstA	Device specific	This field provides an indication of the device life time based on the amount of performed program/erase cycles. The calculation method is vendor specific and referred as method A.. 00h: Information not available 01h: 0% - 10% device life time used 02h: 10% - 20% device life time used 03h: 20% - 30% device life time used 04h: 30% - 40% device life time used 05h: 40% - 50% device life time used 06h: 50% - 60% device life time used 07h: 60% - 70% device life time used 08h: 70% - 80% device life time used 09h: 80% - 90% device life time used 0Ah: 90% - 100% device life time used 0Bh: Exceeded its maximum estimated device life time Others: Reserved

Offset	Size	Name	Value	Description
04h	1	bDeviceLifeTimeEstB	Device specific	This field provides an indication of the device life time based on the amount of performed program/erase cycles. The calculation method is vendor specific and referred as method B.. 00h: Information not available 01h: 0% - 10% device life time used 02h: 10% - 20% device life time used 03h: 20% - 30% device life time used 04h: 30% - 40% device life time used 05h: 40% - 50% device life time used 06h: 50% - 60% device life time used 07h: 60% - 70% device life time used 08h: 70% - 80% device life time used 09h: 80% - 90% device life time used 0Ah: 90% - 100% device life time used 0Bh: Exceeded its maximum estimated device life time Others: Reserved
05h	32	VendorPropInfo	Device specific	Reserved for Vendor Proprietary Health Report

6392

14.2 Flags

A flag is a single Boolean value that represents a TRUE or FALSE, ‘0’ or ‘1’, ON or OFF type of value. A flag can be cleared or reset, set, toggled or read. Flags are useful to enable or disable certain functions or modes or states within the device.

Read access property (read or write only) and write access property (read only, write only, persistent, etc.) are defined for each flag. Table 14-24 describes the supported access properties for flags.

Table 14-24 — Flags access properties

Access Property		Description
Read	Read	The flag can be read.
	Write only	The flag cannot be read.
Write	Read only	The flag cannot be written.
	Write once	The flag can be written only one time, the value is kept after power cycle or any type of reset event. Write operation includes: set, clear and toggle.
	Persistent	The flag can be written multiple times, the value is kept after power cycle or any type reset event.
	Volatile	The flag can be written multiple times. The flag is set to the default value after power cycle or any type of reset event.
	Set only	The flag can only be set to one (or zero) by the host and cleared to zero (or one) by the device. The flag is cleared after power cycle or any type of reset event
	Power on reset	The flag is set to the default value after power cycle or hardware reset event. The flag can be set, cleared or toggled only one time after a power cycle or hardware reset, and it cannot be re-written until the next power cycle or hardware reset.

6401 **14.2 Flags (cont'd)**

6402 **Table 14-25 — Flags**

FLAGS					
IDN	**Name**	**Type**	**Type[1] / # Ind.[2] / # Sel.[3]**	**Default**	**Description**
00h	Reserved				
01h	fDeviceInit	Read / Set only	D	0	Device Initialization Host set fDeviceInit flag to initiate device initialization after boot process is completed. Device resets flag when device initialization is completed. 0b: Device initialization completed or not started yet. 1b: Device initialization in progress.
02h	fPermanentWPEn	Read / Write once	D	0	Permanent Write Protection Enable fPermanentWPEn enables permanent write protection on all logical units configured as permanent protected; it cannot be toggled or cleared once it is set. 00h: Permanent write protection disabled 01h: Permanent write protection enabled
03h	fPowerOnWPEn	Read / Power on reset	D	0	Power On Write Protection Enable fPowerOnWPEn enables the write protection on all logical units configured as power on write protected. If fPowerOnWPEn is equal to one and the device receives a Query Request to clear or toggle this flag, the Query Request shall fail and Response field shall be set to "F8h" (Parameter already written). The device shall set fPowerOnWPEn to zero in the event of power cycle or hardware reset. 0b: Power on write protection disabled. 1b: Power on write protection enabled.
04h	fBackgroundOpsEn	Read / Volatile	D	1	Background Operations Enable 0b: Device is not permitted to run background operations. 1b: Device is permitted to run background operations.

FLAGS					
IDN	**Name**	**Type**	**Type[1] / # Ind.[2] / # Sel.[3]**	**Default**	**Description**
05h	fDeviceLifeSpanModeEn	Read / Volatile	D	0	Device Life Span Mode 0b: Device Life Span Mode is disabled. 1b: Device Life Span Mode is enabled. For more details see 13.4.13, Device Life Span Mode.
06h	fPurgeEnable	Write only / Volatile	D	0	Purge Enable 0b: Purge operation is disabled. 1b: Purge operation is enabled. This flag shall only be set when the command queue of all logical units are empty and the bPurgeStatus is 00h (Idle). fPurgeEnable is automatically cleared by the UFS device when the operation completes or an error condition occurs. fPurgeEnable can be cleared by the host to interrupt an ongoing purge operation.
07h	Reserved	-	-	-	Reserved
08h	fPhyResourceRemoval	Read / Persistent	D	0	Physical Resource Removal The host sets this flag to one to indicate that the dynamic capacity operation shall commence upon device EndPointReset or hardware reset. The device shall reset this flag to zero after completion of dynamic capacity operation. The host cannot reset this flag.
09h	fBusyRTC	Read Only	D	0	Busy Real Time Clock 0b : Device is not executing internal operation related to RTC 1b: Device is executing internal operation related to RTC When this flag is set to one, it is recommended for the host to not send commands to the device
0Ah	Reserved	-	-	-	Reserved for Unified Memory Extension standard
0Bh	fPermanentlyDisableFwUpdate	Read / Write once	D	0	Permanently Disable Firmware Update 0b: The UFS device firmware may be modified 1b: The UFS device shall permanently disallow future firmware updates to the UFS device

FLAGS					
IDN	**Name**	**Type**	**Type[1] / # Ind.[2] / # Sel.[3]**	**Default**	**Description**
0Ch	Reserved	-	-	-	Reserved for Unified Memory Extension standard
0Dh	Reserved	-	-	-	Reserved for Unified Memory Extension standard

NOTE 1 The type “D” identifies a device level flag, while the type “A” identifies an array of flags. If Type = “D”, the flag is addressed setting INDEX = 00h and SELECTOR = 00h.

NOTE 2 For array of flags, ““**#** Ind.” specifies the amount of valid values for the INDEX field in QUERY REQUEST/RESPONSE UPIU. If # Ind = 0, the flag is addressed setting INDEX = 00h.

NOTE 3 For array of flags, ““**#** Sel.” specifies the amount of valid values for the SELECTOR field in QUERY REQUEST/RESPONSE UPIU. If # Sel = 0, the flag is addressed setting SELECTOR = 00h.

6403

14.3 Attributes

An Attribute is a parameter that represents a specific range of numeric values that can be written or read. For example, the maximum Data In data packet size would be an attribute. Attribute size can be from 1-bit to 32-bit. Attributes of the same type can be organized in arrays, each element of them identified by an index. For example, in case of parameter that is logical unit specific, the LUN would be used as index.

Read access property (read or write only) and write access property (read only, write once, persistent, etc.) are defined for each attribute. Table 14-26 describes the supported access properties for attributes.

Table 14-26 — Attributes access properties

Access Property		Description
Read	Read	The attribute can be read.
	Write only	The attribute cannot be read.
Write	Read only	The attribute cannot be written.
	Write once	The attribute can be written only one time, the value is kept after power cycle or any type of reset.
	Persistent	The attribute can be written multiple times, the value is kept after power cycle or any type of reset event.
	Volatile	The attribute can be written multiple times. The attribute is set to the default value after power cycle or any type of reset event.
	Power on reset	The attribute is set to the default value after power cycle or hardware reset event. The attribute can written only one time after a power cycle or hardware reset, and it cannot be re-written until the next power cycle or hardware reset.

6414 **14.3 Attributes (cont'd)**

6415 **Table 14-27 — Attributes**

ATTRIBUTES							
IDN	**Name**	**Access Property**	**Size**	**Type[1]** / **# Ind.[2]** / **# Sel.[3]**	**MDV[4]**	**Description**	**Notes**
00h	bBootLunEn	Read / Persistent	1 byte	D	00h	Boot LUN Enable 00h: Boot disabled 01h: Enabled boot from Boot LU A 02h: Enabled boot from Boot LU B All others: Reserved When bBootLunEn = 00h the boot feature is disabled, the device behaves as if bBootEnable would be equal to 0	
01h	Reserved	-	-	-	-		
02h	bCurrentPowerMode	Read only	1 byte	D	see Note 5	Current Power Mode 00h: Idle power mode 10h: Pre-Active power mode 11h: Active power mode 20h: Pre-Sleep power mode 22h: UFS-Sleep power mode 30h: Pre-PowerDown power mode 33h: UFS-PowerDown power mode Others: Reserved	5
03h	bActiveICCLevel	Read / Volatile	1 byte	D	see Note 6	Active ICC Level bActiveICCLevel defines the maximum current consumption allowed during Active Mode. 00h: Lowest Active ICC level … 0Fh: Highest Active ICC level Others: Reserved Valid range from 00h to 0Fh.	6
04h	bOutOfOrderDataEn	Read / Write once	1 byte	D	00h	Out of Order Data transfer Enable 00h: Out-of-order data transfer is disabled. 01h: Out-of-order data transfer is enabled. Others: Reserved This bit shall have effect only when bDataOrdering = 01h	

ATTRIBUTES							
IDN	**Name**	**Access Property**	**Size**	**Type[1]** **# Ind.[2]** **# Sel.[3]**	**MDV[4]**	**Description**	**Notes**
05h	bBackgroundOpStatus	Read only	1 byte	D	00h	Background Operations Status Device health status for background operation 00h: Not required 01h: Required, not critical 02h: Required, performance impact 03h: Critical. Others: Reserved	
06h	bPurgeStatus	Read only	1 byte	D	00h	Purge Operation Status 00h: Idle (purge operation disabled) 01h: Purge operation in progress 02h: Purge operation stopped prematurely 03h: Purge operation completed successfully 04h: Purge operation failed due to logical unit queue not empty 05h: Purge operation general failure. Others: Reserved When the bPurgeStatus is equal to the values 02h, 03h, 04h or 05h, the bPurgeStatus is automatically cleared to 00h (Idle) the first time that it is read.	
07h	bMaxDataInSize	Read / Persistent	1 byte	D	see Note 7	Maximum Data In Size Maximum data size in a DATA IN UPIU. Value expressed in number of 512-byte units. bMaxDataInSize shall not exceed the bMaxInBufferSize parameter. bMaxDataInSize = bMaxInBufferSize when the UFS device is shipped. This parameter can be written by the host only when all LU task queues are empty.	7, 8

ATTRIBUTES							
IDN	**Name**	**Access Property**	**Size**	**Type[1]** **# Ind.[2]** **# Sel.[3]**	**MDV[4]**	**Description**	**Notes**
08h	bMaxDataOutSize	Read / Persistent	1 byte	D		Maximum Data-Out Size The maximum number of bytes that can be requested with a READY TO TRANSFER UPIU shall not be greater than the value indicated by this attribute. Value expressed in number of 512-byte units. bMaxDataOutSize shall not exceed the bMaxOutBufferSize parameter. bMaxDataOutSize = bMaxOutBufferSize when the UFS device is shipped. This parameter can be written by the host only when all LU task queues are empty.	8
09h	dDynCapNeeded	Read only	4 bytes	A Number of LU specified by bMaxNumberLU (LUN) 0	0000 0000h	Dynamic Capacity Needed The amount of physical memory needed to be removed from the physical memory resources pool of the particular logical unit, in units of bOptimalWriteBlockSize.	9
0Ah	bRefClkFreq	Read / Persistent	1 byte	D	01h	Reference Clock Frequency value 0h:19.2MHz 1h: 26MHz 2h: 38.4MHz 3h: 52MHz Others: Reserved	10
0Bh	bConfigDescrLock	Read / Write once	1 byte	D	00h	Configuration Descriptor Lock 0h: Configuration Descriptor not locked 1h: Configuration Descriptor locked Others: Reserved	

ATTRIBUTES							
IDN	**Name**	**Access Property**	**Size**	**Type[1]** **# Ind.[2]** **# Sel.[3]**	**MDV[4]**	**Description**	**Notes**
0Ch	bMaxNumOfRTT	Read / Persistent	1 byte	D	02h	Maximum current number of outstanding RTTs in device that is allowed. bMaxNumOfRTT shall not exceed the bDeviceRTTCap parameter. This parameter can be written by the host only when all LU task queues are empty.	
0Dh	wExceptionEventControl	Read / Volatile	2 bytes	D	0000h	Exception Event Control This attribute enables the setting of the EVENT_ALERT bit of Device Information field, which is contained in the RESPONSE UPIU. EVENT_ALERT is set to one if at least one exception event occured (wExceptionEventStatus[i]) and the corresponding bit in this attribute is one (wExceptionEventControl[i]). Bit 0: DYNCAP_EVENT_EN Bit 1: SYSPOOL_EVENT_EN Bit 2: URGENT_BKOPS_EN Bit 3 -15: Reserved	
0Eh	wExceptionEventStatus	Read only	2 bytes	D	0000h	Each bit represents an exception event. A bit will be set only if the relevant event has occurred (regardless of the wExceptionEventControl status). Bit 0: DYNCAP_NEEDED Bit 1: SYSPOOL_EXHAUSTED Bit 2: URGENT_BKOPS Bit 3 -15: Reserved	
0Fh	dSecondsPassed	Write only	4 bytes	D	0000 0000h	Bits[31:0]: Seconds passed from TIME BASELINE (see wPeriodicRTCUpdate in Device Descriptor)	

ATTRIBUTES							
IDN	**Name**	**Access Property**	**Size**	**Type[1]** **# Ind.[2]** **# Sel.[3]**	**MDV[4]**	**Description**	**Notes**
10h	wContextConf	Read / Volatile	2 bytes	A Number of LU specified by bMaxNumberLU (LUN) 15 (ID)	0000h	INDEX specifies the LU number. SELECTOR specifies the Context ID within the LU. Valid values are 01h – Fh. **Bit[15:8]: Reserved** **Bit[7:6]: Reliability mode** 00b: MODE0 (normal) 01b: MODE1 (non-Large Unit, reliable mode or Large Unit unit-by-unit mode) 10b: MODE2 (Large Unit, one-unit-tail mode) 11b: Reserved **Bit[5:3]: Large Unit multiplier** If Large Unit context is set, this field defines the Large Unit size, else it is ignored **Bit[2]: Large Unit context** 0b: Context is not following Large Unit rules 1b: Context follows Large Unit rules **Bit [1:0]: Activation and direction mode** 00b: Context is closed and it is no longer active 01b: Context is configured and activated as a write-only context. 10b: Context is configured and activated as a read-only context 11b: Context is configured and activated as a read/write context	
11h	Obsolete	-	-	-	-	-	
12h	Reserved	-	-	-	-	Reserved for Unified Memory Extension standard	
13h	Reserved	-	-	-	-	Reserved for Unified Memory Extension standard	

ATTRIBUTES							
IDN	**Name**	**Access Property**	**Size**	**Type[1]** **# Ind.[2]** **# Sel.[3]**	**MDV[4]**	**Description**	**Notes**
14h	bDeviceFFUStatus	Read Only	1 byte	D	00h	Device FFU Status 00h: No information 01h: Successful microcode update 02h: Microcode corruption error 03h: Internal error 04h: Microcode version mismatch 05h-FEh: Reserved 0FFh: General Error	11
15h	bPSAState	Read / Persistent	1 byte	D	Device specific	00h: 'Off'. PSA feature is off. 01h: 'Pre-soldering'. PSA feature is on, device is in the pre-soldering state. 02h: 'Loading Complete' PSA feature is on. The host will set to this value after the host finished writing data during pre-soldering state. 03h: 'Soldered'. PSA feature is no longer available. Set by the Device to indicate it is in post-soldering state. This attribute unchangeable after it is in 'Soldered' state.	
16h	dPSADataSize	Read / Persistent	8 bytes	D	00 … 00h	The amount of data that the host plans to load to all logical units with bPSASensitive set to 1.	

NOTE 1 The type "D" identifies a device level attribute, while the type "A" identifies an array of attributes. If Type = "D", the attribute is addressed setting INDEX = 00h and SELECTOR = 00h.

NOTE 2 For array of attributes, "# Ind." specifies the amount of valid values for the INDEX field in QUERY REQUEST/RESPONSE UPIU. If # Ind = 0, the attribute is addressed setting INDEX = 00h.

NOTE 3 For array of attributes, "# Sel." specifies the amount of valid values for the SELECTOR field in QUERY REQUEST/RESPONSE UPIU. If # Sel = 0, the attribute is addressed setting SELECTOR = 00h.

NOTE 4 The column "MDV" (Manufacturer Default Value) specifies attribute values after device manufacturing.

NOTE 5 bCurrentPowerMode value after device initialization may be: 20h (Pre-Sleep mode) or 22h (UFS-Sleep mode) if bInitPowerMode = 00h, or 11h (Active Mode) if bInitPowerMode = 01h.

NOTE 6 After power on or reset, bActiveICCLevel is equal to bInitActiveICCLevel parameter value included in the Device Descriptor. bInitActiveICCLevel is equal to 00h after device manufacturing and it can be configured by writing the Configuration Descriptor.

NOTE 7 bMaxDataInSize = bMaxInBufferSize when the UFS device is shipped.

NOTE 8 If the host attempts to write this Attribute when there is at least one logical unit with command queue not empty, the operation shall fail, and Response field in the QUERY RESPONSE UPIU shall be set to FFh ("General failure").

NOTE 9 dDynCapNeeded is composed by eight elements, one for each logical unit. The desired element shall be selected assigning the LUN to INDEX field of QUERY REQUEST UPIU.

NOTE 10 bRefClkFreq field had "Write once" Access Property up to UFS 2.0.

NOTE 11 bDeviceFFUStatus value is kept after power cycle, hardware reset or any other type of reset.This attribute may change value when a microcode activation event occurs.

ANNEX A - DYNAMIC CAPACITY HOST IMPLEMENTATION EXAMPLE (INFORMATIVE)

A.1 Overview

This standard defines the Dynamic Capacity feature to enable a UFS device to regain at least partial functionality at the end-of-life where the defect level of the storage medium has accumulated to the point where the device can no longer maintain normal functionality.

Dynamic Capacity operation allows the device, at the direction of the host system, to remove physical memory resources from the resource pool dedicated for data storage and re-task the resources for device internal utility, thus restoring device functionality.

The net result of the Dynamic Capacity operation is a reduction of the usable storage space in the physical medium while the logical address space remains the same as before – i.e., the physical storage is less than the logical address space. It is the responsibility of the host system to keep track of the reduction in physical storage to maintain normal operation in its file system.

This application note outlines a method for the host file system to account for the reduction in physical storage with minimal impact.

A.2 Method Outline

1. The host system receives notification from the device in the Device Information parameter in Response UPIU that the device requires physical memory to be freed up from the storage space to continue operation.

2. The host reads the bDynCapNeeded[LUN] attributes and the bOptimalWriteBlockSize parameter in the Device Descriptor to determine how much physical memory resources needs to be freed up in each logical unit.

3. The host identifies the logical block address range(s) in the file system where the data can be discarded/erased to free up the physical memory resources. The host then uses the UNMAP command to unmap (deallocate) the LBA range(s), and initiates the Dynamic Capacity operation by setting the fPhyResourceRemoval flag and resetting the UFS device.

4. The host can mark the particular LBA range(s) as unusable in its file system by the means of dummy file(s) to ensure these LBA’s will not be used in future write operations. The unusable LBA’s marked by dummy file(s) match the reduction of physical storage, therefore from the host system perspective, the file system is intact.

5. The host can further backup the unusable LBA information by storing the information in the system area in case the file system of the main data storage logical unit is corrupted.

ANNEX B (INFORMATIVE) DIFFERENCES BETWEEN JESD220C AND JESD220B

B.1 Changes between JESD220C and its predecessor JESD220B (September 2013)

B.1.1 New features or new definitions

The following items were added

- Multi-initiator, see section 10.6.2 “Basic Header Format”
- Command priority, see section 10.7.1 “COMMAND UPIU”
- Field Firmware Update, see section 11.3.28 “WRITE BUFFER Command”
- Vital product data parameters see section 11.5
- Secure write protection, see sections
  - 12.4 “RPMB”
  - 12.4.6.7 “Authenticated Secure Write Protect Configuration Block Write”
  - 12.4.6.8 “Authenticated Secure Write Protect Configuration Block Read”
- Device Life Span Mode, see section 13.4.13
- Production State Awareness, see section 13.6
- Product Revision Level String Descriptor, see section 14.1.4.13
- Device Health Descriptor, see section 14.1.4.14

B.1.2 Changes in section 2 “Normative Reference”

Added the following specifications:

- 1.2 V +/- 0.1V (Normal Range) and 0.8 - 1.3 V (Wide Range) Power Supply Voltage and Interface Standard for Nonterminated Digital Integrated Circuits
- JEDEC Recommended ESD Target Levels for HBM/MM Qualification, JEP155A.01, March
- JEDEC Recommended ESD-CDM Target Levels, JEP157, October 2009
- Eastlake, D. and T. Hansen, "US Secure Hash Algorithms (SHA and HMAC-SHA)", RFC

B.1.3 Changes in features already defined in UFS 2.0

Changes for features already defined in the previous version of the standard are summarized in the following:

- 3.2 “Terms and Definitions”, added: “application client”, “device server”; changed the definition for: “initiator device”, “target device”, “transaction”.
- 3.3 “Keywords”, added “obsolete”.
- 6.1 “UFS Signals” Table 6-1, added reference to ESD specifications.
- 6.3 “Power Supplies”, added figure for tPRUH, tPRUL and tPRUV.
- 6.4 “Reference Clock”, clarified reference clock details and bRefClkFreq attribute setting.
- 7.4.1.4 “UFS-Sleep Power Mode”, fixed sense key and additional sense code values for commands other than START STOP UNIT or REQUEST SENSE.

B.1.3 Changes in features already defined in UFS 2.0 (cont'd)

- 7.4.1.5 "Pre-Sleep Power Mode", fixed sense key value in pollable sense data.
- 7.4.1.7 "Pre-PowerDownPower Mode", fixed sense key value in pollable sense data.
- 8.4 "HS Burst" removed section 8.4.3 "Slew Rate Control".
- 8.6 "UFS PHY Attributes", extended the ranges of the following capability attributes: TX_Min_STALL_NoConfig_Time_Capability, RX_Min_STALL_NoConfig_Time_Capability, RX_HS_G1_SYNC_LENGTH_Capability, RX_HS_G2_SYNC_LENGTH_Capability, RX_HS_G3_SYNC_LENGTH_Capability
- 9.5 "UniPro/UFS Transport Protocol Address Mapping", added Table 9-1 "UFS Port IDs"
- 9.6.5 "UniPro Device Management Entity Transport Layer", substituted Table 9.1 "DME Service Primitives" with a text.
- 10.7.1 "COMMAND UPIU", added Flags.CP bit and IID field.
- 10.7.2 "RESPONSE UPIU", added requirements for the termination of a command with Data-Out data transfer, added IID field.
- 10.7.3 "DATA OUT UPIU", added: IID field, the requirement that Data Segment area shall contain an integer number of logical blocks and an example for Data out transfer.
- 10.7.4 "DATA IN UPIU", added: IID field, the requirement that Data Segment area shall contain an integer number of logical blocks and an example for Data in transfer.
- 10.7.5 "READY TO TRANSFER UPIU", added: IID field, the requirement to request the transfer of an integer number of logical blocks and an example for READY TO TRANSFER UPIU sequence.
- 10.7.6 "TASK MANAGEMENT REQUEST UPIU", added IID field, clarified behavior if more than one task management request are received, added IID field in Task Management Input Parameters.
- 10.7.7 "TASK MANAGEMENT RESPONSE UPIU", added IID field, added requirements for the termination of a command with Data-Out data transfer.
- 10.7.8.3 "Transaction Specific Fields", indicated QUERY FUNCTION value for each opcode in Table 10-30.
- 10.7.10 "REJECT UPIU", added IID field.
- 10.7.13 "Data out transfer rules", in UFS 2.0 this section was in chapter 13, while in UFS 2.1 it was moved in chapter 10, figures were updated.
- 10.8.5 "Well Known Logical Unit Defined in UFS", added FORMAT UNIT command in the list of commands supported by the Device well known logical unit.
- 10.9.8 "Task Management Function procedure calls", added requirements for the termination of a command with Data-Out data transfer.
- 11.3 "Universal Flash Storage SCSI Commands" changed WRITE BUFFER command support from optional to mandatory.
- 11.3.2.4 "Inquiry Response Data", specified the PERIPHERAL DEVICE TYPE value for well known logical unit (e.g. 1Eh).
- 11.3.18 "FORMAT UNIT Command", added description related to Device well known logical unit.

B.1.3 Changes in features already defined in UFS 2.0 (cont'd)

- 11.3.28 "WRITE BUFFER Command", added Field Firmware Update.
- 11.4.2.1 "Control Mode Page", changed EXTENDED SELF-TEST COMPLETION TIME field value from 0000h to device specific, BUSY TIMEOUT PERIOD field from changeable to not changeable, defined TST as not changeable.
- 12.2.3.4 "Wipe Device", added wipe device feature using Device well known logical.
- 12.4.5.1 "CDB format of SECURITY PROTOCOL IN/OUT", specified command response in case of invalid ALLOCATION LENGTH or TRANSFER LENGTH values.
- 13.2.3 "Logical Unit Configuration", added an example for dNumAllocUnits calculation, changed the recommendation for setting logical block size: in UFS 2.1 references dOptimalLogicalBlockSize instead of bOptimalWriteBlockSize or OptimalReadBlockSize.
- 13.4.6 "Dynamic Device Capacity", added Dynamic Capacity Resource Policy.
- 13.4.12 "Queue Depth Definition", new section which describes shared queue and per-logical unit queue implementations.
- 14.1.4.2 "Device Descriptor", changed bSecurityLU parameter value from "Device specific" to "01h", changed wSpecVersion parameter value from "0200h" to "0210h", added the following parameters: bUFSFeaturesSupport, bFFUTimeout, bQueueDepth, wDeviceVersion, bNumSecureWPArea, dPSAMaxDataSize, bPSAStateTimeout, iProductRevisionLevel.
- 14.1.6.3 "Configuration Descriptor", added three Configuration Descriptors and bConfDescContinue parameter.
- 14.1.4.4 "Geometry Descriptor", added the following parameters: bMaxNumberLU, bDynamicCapacityResourcePolicy, dOptimalLogicalBlockSize.
- 14.1.4.5 "Unit Descriptor", added bPSASensitive parameter.
- 14.1.4.6 "RPMB Unit Descriptor", added bPSASensitive parameter, changed bLUQueueDepth value from "00h" to "Device specific".
- 14.2 "Flags", added fDeviceLifeSpanModeEn flag.
- 14.3 "Attributes", changed bActiveICCLevel access property from "Persistent" to "Volatile", changed bRefClkFreq access property from "Write once" to "Persistent", defined as obsolete the dCorrPrgBlkNum (IDN=11h), added bDeviceFFUStatus, bPSAState and dPSADataSize.
- Global
    - Removed the use of "embedded UFS", " UFS Card"
    - Reviewed the use of "initiator device", "target device", "UFS host", "UFS device"
    - Added optional support for 32 logical units are related Configuration Descriptors
        - 14.1.3 "Accessing Descriptors and Device Configuration"
        - 14.1.6.3 "Configuration Descriptor"

Several clarifications were added and editorial changes were implemented in addition to what summarized in this annex.

B.2 Changes between JESD220B and its predecessor JESD220A (June 2012)

B.2.1 New features or new definitions

The following items were added

- HS-GEAR3 support (optional)
- Multi-lane support (two lanes, optional)
- New 6.5.1“HS Gear Rates”, Defined HS-BURST rates
- New 6.7 “Absolute Maximum DC Ratings”, Defined absolute maximum DC ratings for signal voltages, power supply voltages, and storage temperature.
- New 7.2 “Power up ramp”, Defined requirements for power up ramp.
- New 7.3 “Power off ramp”, Defined requirements power off ramp.
- Mode Page Policy VPD support (see 11.3.2.1 “VITAL PRODUCT DATA”)
- New 11.3.21 “SECURITY PROTOCOL IN Command”
- New 11.3.22 “SECURITY PROTOCOL OUT Command”

B.2.2 Changes in section 2 “Normative Reference”

- M-PHY^{SM} specification: from version 2.00.00 to version 3.0.
- Unified Protocol (UniPro^{SM}) specification: from version 1.41.00 to version 1.6.

B.2.3 Changes in features already defined in UFS 1.1

Changes for features already defined in the previous version of the standard are summarized in the following

- 4.1 “General Features”, Mandatory support for HS-GEAR2
- 5.4.3 “MIPI UniPro Related Attributes”, DME_DDBL1_ManufacturerID and DME_DDBL1_DeviceClass, Used Attribute names defined in [MIPI-UniPro] and fixed Attribute ID value.
- 5.5.1.1 “Client-Server Model”, Deleted the sentence: “Only one Task can be processed at a time within a Logical Unit. If a device contains multiple Logical Units, it could have the ability to process multiple Tasks simultaneously or concurrently if so designed.”.
- 7.3.1 “EndPointReset”, Deleted the sentence “Note that if the device has already completed the initialization phase before receiving the EndPointReset, it is not required to set fDeviceInit to ‘1‘ and wait until the device clears it.”.

B.2.3 Changes in features already defined in UFS 1.1 (cont'd)

- 7.4.1 "Device Power Modes", Clarified logical unit response to SCSI commands for each power mode. UFS-Sleep power mode: added the sentence: "VCC power supply should be restored before issuing START STOP UNIT command to request transition to Active power mode or PowerDown power mode.". Figure "Power Mode State Machine": added Powered On power mode, arrow from Active to Pre-Sleep with "bInitPowerMode=00h", and some notes. Defined all power mode transitions. Added the sentence: "The effects of concurrent power mode changes requested by START STOP UNIT commands with the IMMED bit set to one are vendor specific.". Added the sentence: "A START STOP UNIT command with the IMMED bit set to zero causing a transition to Active, Sleep, or PowerDown power modes shall not complete with GOOD status until the device reaches the power mode specified by the command.". Added 7.4.1.9 "Device Well Known Logical Unit Responses to SCSI commands"
- New 7.4.4 "Logical Unit Power Condition"
- 8.6 "UFS PHY Attributes", Added the following M-PHY[SM] Attributes: TX_Hibern8Time_Capability, TX_Advanced_Granularity_Capability, TX_Advanced_Hibern8Time_Capability, TX_HS_Equalizer_Setting_Capability, RX_Hibern8Time_Capability, RX_PWM_G6_G7_SYNC_LENGTH_Capability, RX_HS_G2_SYNC_LENGTH_Capability, RX_HS_G3_SYNC_LENGTH_Capability, RX_HS_G2_PREPARE_LENGTH_Capability, RX_HS_G3_PREPARE_LENGTH_Capability, RX_Advanced_Granularity_Capability, RX_Advanced_Hibern8Time_Capability, RX_Advanced_Min_ActivateTime_Capability.
- 9.6.6 "UniPro Attributes", Added the UniPro[SM] PA_MaxDataLanes constant, PA_AvailTxDataLanes and PA_AvailRxDataLanes Attributes.
- Table 10.17 — Sense Key, Deleted the sentence: "The UNIT ATTENTION condition will remain set until an explicit REQUEST SENSE has been issued to the target."
- 10.5.5 "DATA OUT UPIU", Added the sentence: "Note that in case out of order DATA OUT UPIUs, the last data portion may not be transmitted by the final UPIU.".
- 10.5.6 "DATA IN UPIU", Added the sentence: "Note that in case out of order DATA IN UPIUs, the final data portion may not be transmitted by the last UPIU."
- 10.5.7 "READY TO TRANSFER UPIU""
  Added the sentence: "The Data Buffer Offset shall be an integer multiple of four.".
  Added the sentence: "The Data Transfer Count field shall be always an integer multiple of four bytes except for RTT which requests the final portion of data in the transfer.".
- New 10.5.10.12 "NOP"
  Defined NOP OPCODE for QUERY REQUEST UPIU.
- 10.5.11 "QUERY RESPONSE UPIU", Added Device Information in byte 9 of QUERY RESPONSE UPIU. Defined Query Response value for invalid Query Function field and OPCODE field combinations.
- New 10.5.11.14 "NOP", Defined NOP OPCODE for QUERY RESPONSE UPIU.
- .3.2.4 "Inquiry Response Data", Added the sentence: "The 4-byte PRODUCT REVISION LEVEL in the Inquiry Response Data shall identify the firmware version of the UFS device and shall be uniquely encoded for any firmware modification implemented by the UFS device vendor."

B.2.3 Changes in features already defined in UFS 1.1 (cont'd)

- 11.4.2 "UFS Supported Pages", Defined default value and changeable fields for the following Mode Pages: Control Mode Page, Read-Write Error Recovery Mode Page, Caching Mode Page.
- 12.4.6 "RPMB Operations", Changed Command Set Type field value from 1h to 0h.
- 13.1.3.1 "Partial initialization", During device initializialization is no longer required to send a sequence of NOP OUT UPIU: the host sends only one NOP OUT UPIU.
- 13.1.3.3 "Initialization completion" , Added a table to clarify which are the valid UPIUs and SCSI commands for each initialization phase.
- 13.2.3 "Logical Unit Configuration"
  Added the sentence: "Supported bLogicalBlockSize values are device specific, refer to the vendor datasheet for further information".
  Added the sentence: "The Capacity Adjustment Factor value for Normal memory type is one."
- 13.4.4 "Background Operations Mode"
  Removed the requirement of having empty command queues for starting background operations.
  Added bBackgroundOpsTermLat (Background Operations Termination Latency) parameter in Device Descriptor.
  Defined URGENT_BKOPS = 0 if bBackgroundOpStatus = 1.
- 13.4.7 "Data Reliability"
  Increased data reliability granularity from 512 bytes to logical block size.
  Defined RPMB data reliability granularity (bRPMB_ReadWriteSize × 256 bytes)
- 13.4.12 "Data transfer rules related with RTT (Ready-to-Transfer)"
  Added the sentence: "RTT requests related to several write commands or from different logical units may be interleaved."
- 14.1.6.2 "Device Descriptor"
  bDeviceSubClass: reserved bit 2 for Unified Memory Extension standard.
  Changed bNumberLU manufacturer default value (MDV) from 01h to 00h.
  Added bBackgroundOpsTermLat parameter.
  Reserved bytes for Unified Memory Extension standard.
  Clarified wManufacturerID definition.
- 14.1.6.3 "Configuration Descriptor" Removed bNumberLU (because this parameter is no longer configurable).
- 14.1.6.5 "Unit Descriptor" bLUWriteProtect: reserved the value 03h for UFS Security Extension standard.
- 14.2 "Flags": Added fPermanentlyDisableFwUpdate (Permanently Disable Firmware Update) flag.
- 14.3 "Attributes": Changed bRefClkFreq manufacturer default value (MDV) from 00h to 01h.

Several clarifications were added and editorial changes were implemented in addition to what summarized above.

Standard Improvement Form JEDEC _____

The purpose of this form is to provide the Technical Committees of JEDEC with input from the industry regarding usage of the subject standard. Individuals or companies are invited to submit comments to JEDEC. All comments will be collected and dispersed to the appropriate committee(s).

If you can provide input, please complete this form and return to:

JEDEC
Attn: Publications Department
3103 North 10th Street
Suite 240 South
Arlington, VA 22201-2107

Fax: 703.907.7583

1. I recommend changes to the following:
   - ☐ Requirement, clause number _____
   - ☐ Test method number _____ Clause number _____

   The referenced clause number has proven to be:
   - ☐ Unclear ☐ Too Rigid ☐ In Error
   - ☐ Other _____

2. Recommendations for correction:

3. Other suggestions for document improvement:

Submitted by
Name: _____ Phone: _____
Company: _____ E-mail: _____
Address: _____
City/State/Zip: _____ Date: _____


JEDEC®