

# **JEDEC STANDARD**

---

**Universal Flash Storage (UFS)**

**Version 3.1**

---

**JESD220E**

(Revision of JESD220D, January 2018)

**JANUARY 2020**

---

**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](http://www.jedec.org) under Standards and Documents for alternative contact information.

Published by  
©JEDEC Solid State Technology Association 2020  
3103 North 10th Street  
Suite 240 South  
Arlington, VA 22201-2107

JEDEC retains the copyright on this material. By downloading this file the individual agrees not to charge for or resell the resulting material.

### PRICE: Contact JEDEC

Printed in the U.S.A.  
All rights reserved

**PLEASE!**

**DON'T VIOLATE  
THE  
LAW!**

This document is copyrighted by JEDEC and may not be  
reproduced without permission.

For information, contact:

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

or refer to [www.jedec.org](http://www.jedec.org) under Standards-Documents/Copyright Information.



## UNIVERSAL FLASH STORAGE (UFS), VERSION 3.1

### Contents

	Page
<b>1 Scope</b>	1
<b>2 Normative Reference</b>	1
<b>3 Terms and Definitions</b>	2
3.1 Acronyms	3
3.2 Conventions	4
3.3 Keywords	5
3.4 Abbreviations	5
<b>4 Introduction</b>	6
4.1 General Features	6
4.2 Interface Features	7
4.3 Functional Features	7
<b>5 UFS Architecture Overview</b>	8
5.1 UFS Top Level Architecture	8
5.2 UFS System Model	11
5.3 System Boot and Enumeration	11
5.4 UFS Interconnect (UIC) Layer	12
5.4.1 UFS Physical Layer Signals	12
5.4.2 MIPI UniPro	12
5.4.3 MIPI UniPro Related Attributes	13
5.5 UFS Transport Protocol (UTP) Layer	13
5.5.1 Architectural Model	14
5.6 UFS Application and Command Layer	18
5.7 Mechanical	19
<b>6 UFS Electrical: Clock, Reset, Signals And Supplies</b>	19
6.1 UFS Signals	19
6.2 Reset Signal	21
6.3 Power Supplies	21
6.4 Reference Clock	22
6.4.1 HS Gear Rates	25
6.4.2 Host Controller requirements for reference clock generation	26
6.5 External Charge Pump Capacitors (Optional)	27
6.6 Absolute Maximum DC Ratings and Operating Conditions	28
<b>7 Reset, Power-Up And Power-Down</b>	29
7.1 Reset	29
7.1.1 Power-on Reset	29
7.1.2 Hardware Reset	30
7.1.3 EndPointReset	31
7.1.4 Logical Unit Reset	32
7.1.5 Host UniPro Warm Reset	32
7.1.6 Summary of Resets and Device Behavior	33
7.2 Power up ramp	34
7.3 Power off ramp	35
7.4 UFS Device Power Modes and LU Power Condition	36
7.4.1 Device Power Modes	36
7.4.2 Power Management Command: START STOP UNIT	46
7.4.3 Power Mode Control	48
7.4.4 Logical Unit Power Condition	50
<b>8 UFS UIC Layer: MIPI M-PHY</b>	51
8.1 Termination	51
8.2 Drive Levels	51

**Contents (cont'd)**


---

8.3	PHY State machine	51
8.4	HS Burst	52
8.4.1	HS Prepare Length Control	52
8.4.2	HS Sync Length Control	52
8.5	PWM Burst	52
8.5.1	LS Prepare Length Control	52
8.6	Adapt	52
8.7	UFS PHY Attributes	52
8.8	Electrical characteristics	55
8.8.1	Transmitter Characteristics	55
8.8.2	Receiver Characteristics	55
<b>9</b>	<b>UFS UIC Layer: MIPI Unipro</b>	<b>56</b>
9.1	Overview	56
9.2	Architectural Model	56
9.3	UniPro/UFS Transport Protocol Interface (Data Plane)	57
9.4	UniPro/UFS Control Interface (Control Plane)	58
9.5	UniPro/UFS Transport Protocol Address Mapping	59
9.6	Options and Tunable Parameters of UniPro	60
9.6.1	UniPro PHY Adapter	60
9.6.2	UniPro Data Link Layer	61
9.6.3	UniPro Network Layer	61
9.6.4	UniPro Transport Layer	62
9.6.5	UniPro Device Management Entity Transport Layer	62
9.6.6	UniPro Attributes	63
<b>10</b>	<b>UFS Transport Protocol (UTP) Layer</b>	<b>64</b>
10.1	Overview	64
10.2	UTP and UniPro Specific Overview	65
10.2.1	Phases	65
10.2.2	Data Pacing	65
10.2.3	UniPro	65
10.3	UFS Transport Protocol Transactions Overview	66
10.4	Service Delivery Subsystem	66
10.5	UPIU Transactions	66
10.6	General UFS Protocol Information Unit Format	68
10.6.1	Overview	69
10.6.2	Basic Header Format	69
10.7	UFS Protocol Information Units	74
10.7.1	COMMAND UPIU	74
10.7.2	RESPONSE UPIU	77
10.7.3	DATA OUT UPIU	86
10.7.4	DATA IN UPIU	89
10.7.5	READY TO TRANSFER UPIU	92
10.7.6	TASK MANAGEMENT REQUEST UPIU	95
10.7.7	TASK MANAGEMENT RESPONSE UPIU	97
10.7.8	QUERY REQUEST UPIU	99
10.7.9	QUERY RESPONSE UPIU	112
10.7.10	REJECT UPIU	124
10.7.11	NOP OUT UPIU	126
10.7.12	NOP IN UPIU	128
10.7.13	Data out transfer rules	130
10.8	Logical Units	134
10.8.1	UFS SCSI Domain	134
10.8.2	UFS Logical Unit Definition	134

**Contents (cont'd)**

10.8.3	Well Known Logical Unit Definition	135
10.8.4	Logical Unit Addressing	135
10.8.5	Well Known Logical Unit Defined in UFS	136
10.8.6	Translation of 8-bit UFS LUN to 64-bit SCSI LUN Address	137
10.8.7	SCSI Write Command	138
10.8.8	SCSI Read Command	139
10.8.9	Unit Attention Condition	140
10.9	Application Layer and Device Manager Transport Protocol Services	141
10.9.1	UFS Initiator Port and Target Port Attributes	141
10.9.2	Execute Command procedure call transport protocol services	142
10.9.3	SCSI Command transport protocol service	143
10.9.4	SCSI Command Received transport protocol	144
10.9.5	Send Command Complete transport protocol service	145
10.9.6	Command Complete Received transport protocol service	146
10.9.7	Data transfer SCSI transport protocol services	147
10.9.8	Task Management Function procedure calls	151
10.9.9	Query Function transport protocol services	157
<b>11</b>	<b>UFS Application (UAP) Layer – SCSI Commands</b>	<b>160</b>
11.1	Universal Flash Storage Command Layer (UCL) Introduction	160
11.1.1	The Command Descriptor Block (CDB)	160
11.2	Universal Flash Storage Native Commands (UNC)	160
11.3	Universal Flash Storage SCSI Commands	161
11.3.1	General information about SCSI commands in UFS	162
11.3.2	INQUIRY Command	162
11.3.3	MODE SELECT (10) Command	165
11.3.4	MODE SENSE (10) Command	167
11.3.5	READ (6) Command	170
11.3.6	READ (10) Command	171
11.3.7	READ (16) Command	173
11.3.8	READ CAPACITY (10) Command	175
11.3.9	READ CAPACITY (16) Command	177
11.3.10	START STOP UNIT Command	181
11.3.11	TEST UNIT READY Command	182
11.3.12	REPORT LUNS Command	183
11.3.13	VERIFY (10) Command	188
11.3.14	WRITE (6) Command	190
11.3.15	WRITE (10) Command	192
11.3.16	WRITE (16) Command	195
11.3.17	REQUEST SENSE Command	198
11.3.18	FORMAT UNIT Command	200
11.3.19	PRE-FETCH (10) Command	202
11.3.20	PRE-FETCH (16) Command	205
11.3.21	SECURITY PROTOCOL IN Command	206
11.3.22	SECURITY PROTOCOL OUT Command	207
11.3.23	SEND DIAGNOSTIC Command	209
11.3.24	SYNCHRONIZE CACHE (10) Command	211
11.3.25	SYNCHRONIZE CACHE (16) Command	214
11.3.26	UNMAP Command	215
11.3.27	READ BUFFER Command	219
11.3.28	WRITE BUFFER Command	225
11.4	Mode Pages	229
11.4.1	Mode Page Overview	229
11.4.2	UFS Supported Pages	234

**Contents (cont'd)**

11.5	Vital product data parameters	241
11.5.1	Overview	241
11.5.2	VPD page format	241
11.5.3	Supported VPD Pages VPD page	242
11.5.4	Mode Page Policy VPD page	243
<b>12</b>	<b>UFS Security</b>	245
12.1	UFS Security Feature Support Requirements	245
12.2	Secure Mode	245
12.2.1	Description	245
12.2.2	Requirements	246
12.2.3	Implementation	247
12.3	Device Data Protection	252
12.3.1	Description and Requirements	252
12.3.2	Implementation	252
12.4	RPMB	253
12.4.1	Introduction	253
12.4.2	RPMB Well Known Logical Unit Description	253
12.4.3	Requirements	254
12.4.4	Implementation	263
12.4.5	SECURITY PROTOCOL IN/OUT Commands	264
12.4.6	RPMB Operations	269
12.5	Malware Protection	285
<b>13</b>	<b>UFS Functional Descriptions</b>	286
13.1	UFS Boot	286
13.1.1	Introduction	286
13.1.2	Boot Configuration	286
13.1.3	Initialization and boot code download process	289
13.1.4	Initialization process without boot code download	292
13.1.5	Boot Logical Unit Operations	293
13.1.6	Configurability	293
13.1.7	Security	294
13.2	Logical Unit Management	294
13.2.1	Introduction	294
13.2.2	Logical Unit features	294
13.2.3	Logical Unit Configuration	297
13.3	Logical Block Provisioning	303
13.3.1	Overview	303
13.3.2	Full Provisioning	303
13.3.3	Thin Provisioning	303
13.4	Host Device Interaction	304
13.4.1	Overview	304
13.4.2	Applicable Devices	304
13.4.3	Command Queue: Inter-LU Priority	304
13.4.4	Background Operations Mode	305
13.4.5	Power Off Notification	307
13.4.6	Dynamic Device Capacity	308
13.4.7	Data Reliability	312
13.4.8	Real-Time Clock Information	313
13.4.9	Context Management	314
13.4.10	System Data Tag Mechanism	317
13.4.11	Exception Events Mechanism	318
13.4.12	Queue Depth Definition	319
13.4.13	Device Life Span Mode	320

**Contents (cont'd)**

13.4.14	Refresh Operation	321
13.4.15	Temperature Event Notification	324
13.4.16	Performance Throttling Event Notification	324
13.4.17	WriteBooster	325
13.5	UFS Cache	330
13.6	Production State Awareness (PSA)	331
13.6.1	Introduction	331
13.6.2	PSA flow	331
<b>14</b>	<b>UFS Descriptors, Flags And Attributes</b>	334
14.1	UFS Descriptors	334
14.1.1	Descriptor Types	334
14.1.2	Descriptor Indexing	335
14.1.3	Accessing Descriptors and Device Configuration	335
14.1.4	Descriptor Definitions	339
14.2	Flags	374
14.3	Attributes	379
<b>15</b>	<b>UFS Mechanical Standard</b>	394
<b>Annex A (informative) Dynamic Capacity Host Implementation Example</b>		395
A.1	Overview	395
A.2	Method Outline	395
<b>Annex B (informative) Differences Between Specification Revisions</b>		396
B.1	Changes between JESD220E and its predecessor JESD220D (January 2018)	396
B.1.2	Changes in section 2 “Normative Reference”	396
B.1.3	Changes in features already defined in UFS 2.1	396
B.2	Changes between JESD220D and its predecessor JESD220C (March 2016)	397
B.2.1	New features or new definitions	397
B.2.2	Changes in section 2 “Normative Reference”	397
B.2.3	Changes in features already defined in UFS 2.1	397
B.3	Changes between JESD220C and its predecessor JESD220B (September 2013)	398
B.3.1	New features or new definitions	398
B.3.2	Changes in section 2 “Normative Reference”	398
B.3.3	Changes in features already defined in UFS 2.0	398
B.4	Changes between JESD220B and its predecessor JESD220A (June 2012)	401
B.4.1	New features or new definitions	401
B.4.2	Changes in section 2 “Normative Reference”	401
B.4.3	Changes in features already defined in UFS 1.1	401

**Figures**

Figure 5.1 — UFS Top Level Architecture .....	8
Figure 5.2 — Usage of UDM_SAP .....	9
Figure 5.3 — Usage of UIO_SAP .....	9
Figure 5.4 — UFS System Model .....	11
Figure 5.5 — SCSI Domain Class Diagram .....	15
Figure 5.6 — UFS Domain Class Diagram .....	16
Figure 6.1 — UFS Device Block Diagram.....	19
Figure 6.2 — Supply voltage power up timings .....	22
Figure 6.3 — bRefClkGatingWaitTime .....	24
Figure 6.4 — Clock input levels, rise time, and fall time .....	25
Figure 6.5 — Test Load Impedance .....	26
Figure 6.6 — Output driver and Input receiver levels .....	26
Figure 6.7 — Clock output levels, rise time and fall time .....	27
Figure 7.1 — Power-on Reset .....	29
Figure 7.2 — Hardware Reset .....	30

**Contents (cont'd)**


---

Figure 7.3 — Reset AC timings .....	30
Figure 7.4 — EndPointReset .....	31
Figure 7.5 — Logical Unit Reset.....	32
Figure 7.6 — Power up ramps .....	34
Figure 7.7 — Power off ramps .....	35
Figure 7.8 — Power Mode State Machine .....	41
Figure 8.1 — Simplified example for I/O termination .....	51
Figure 9.1 — UniPro internal layering view (a) and UniPro Black Box view (b).....	56
Figure 9.2 — Physical lane connections.....	61
Figure 10.1 — Data out transfer example.....	88
Figure 10.2 — Data in transfer example.....	91
Figure 10.3 — READY TO TRANSFER UPIU sequence example .....	94
Figure 10.4 — Example for data out transfer rule 1 .....	130
Figure 10.5 — Example for Data Transfer Count mismatch .....	131
Figure 10.6 — Example for data out transfer rule 2 .....	132
Figure 10.7 — Example for data out transfer rule 3 .....	133
Figure 10.8 — UFS SCSI domain .....	134
Figure 10.9 — Logical Unit Addressing .....	135
Figure 10.10 — SCSI Write .....	138
Figure 10.11 — SCSI Read .....	139
Figure 10.12 — Command without Data Phase .....	142
Figure 10.13 — Command + Read Data Phase 1/2 .....	148
Figure 10.14 — Command + Read Data Phase 2/2 .....	148
Figure 10.15 — Command + Write Data Phase ½ .....	150
Figure 10.16 — Command + Write Data Phase 2/2 .....	150
Figure 10.17 — Task Management Function .....	154
Figure 10.18 — UFS Query Function .....	157
Figure 11.1 — UFS Command Layer.....	160
Figure 12.1 — Purge operation state machine.....	249
Figure 12.2 — Request type message delivery.....	270
Figure 12.3 — Response Type Message delivery .....	272
Figure 12.4 — Authentication Key Programming Flow.....	274
Figure 12.5 — Read Counter Value Flow .....	275
Figure 12.6 — Authenticated Data Read Flow.....	280
Figure 12.7 —Authenticated Secure Write Protect Configuration Block Write Flow .....	283
Figure 12.8 —Authenticated Secure Write Protect Configuration Block Read Flow .....	285
Figure 13.1 — UFS System Diagram .....	286
Figure 13.2 — Example of UFS Device Memory Organization for Boot .....	288
Figure 13.3 — Device Initialization and Boot Procedure Sequence Diagram.....	291
Figure 13.4 — Example of UFS Device Memory Organization .....	295
Figure 13.5 — Physical Memory Resource State Machine .....	311
Figure 13.6 — Example of data status after a power failure during reliable write operation .....	312
Figure 13.7 — Concept of WriteBooster feature .....	325
Figure 13.8 — Example of “LU dedicated buffer” mode configuration .....	326
Figure 13.9 — Example of “shared buffer” mode configuration .....	327
Figure 13.10 — PSA flow .....	332
Figure 13.11 — PSA state machine .....	333
Figure 14.1 — Descriptor Organization .....	335
Figure 14.2 — Read Request Descriptor .....	337
Figure 14.3 — Write Request Descriptor .....	338

---

**Contents (cont'd)**


---

**Tables**

Table 5.1 — UFS Signals .....	12
Table 5.2 — ManufacturerID and DeviceClass Attributes .....	13
Table 6.1 — Signal Name and Definitions .....	20
Table 6.2 — Reset Signal Electrical Parameters .....	21
Table 6.3 — UFS Power Supply Parameters .....	21
Table 6.4 — Reference Clock parameters .....	23
Table 6.5 — HS-BURST Rates .....	25
Table 6.6 — Host controller reference clock parameters .....	26
Table 6.7 — Charge pump capacitors description .....	27
Table 6.8 — Charge pump related ball names .....	28
Table 6.9 — Absolute maximum DC ratings and Operating Conditions .....	28
Table 7.1 — Reset timing parameters .....	30
Table 7.2 — Reset States .....	33
Table 7.3 — UniPro Attributes, UFS Attributes and UFS Flags reset .....	33
Table 7.4 — Allowed SCSI commands and UPIU for each Power Mode .....	44
Table 7.5 — Device Well Known Logical Unit Responses to SSU command .....	45
Table 7.6 — Device Well Known Logical Unit Responses to commands other than SSU .....	46
Table 7.7 — Pollable Sense Data for each Power Modes .....	46
Table 7.8 — START STOP UNIT command .....	46
Table 7.9 — START STOP UNIT fields .....	47
Table 7.10 — Attribute for Power Mode Control .....	48
Table 7.11 — Device Descriptor parameters .....	48
Table 7.12 — Power Parameters Descriptor fields .....	49
Table 7.13 — Format for Power Parameter element .....	49
Table 7.14 — Logical Unit Response to SCSI command .....	50
Table 8.1 — UFS PHY M-TX Capability Attributes .....	53
Table 8.2 — UFS PHY M-RX Capability Attributes .....	54
Table 9.1 — UFS Initiator and Target Port Identifiers .....	60
Table 9.2 — UniPro Attribute .....	63
Table 10.1 — UPIU Transaction Codes .....	66
Table 10.2 — UPIU Transaction Code Definitions .....	67
Table 10.3 — General format of the UFS Protocol Information Unit .....	68
Table 10.4 — Basic Header Format .....	69
Table 10.5 — Transaction Type Format .....	69
Table 10.6 — UPIU Flags .....	70
Table 10.7 — Task Attribute definition .....	70
Table 10.8 — UTP Response Values .....	71
Table 10.9 — UPIU associated to a single task .....	71
Table 10.10 — Command Set Type .....	72
Table 10.11 — COMMAND UPIU .....	74
Table 10.12 — Flags definition for COMMAND UPIU .....	75
Table 10.13 — RESPONSE UPIU .....	77
Table 10.14 — Flags definition for RESPONSE UPIU .....	78
Table 10.15 — SCSI Status Values .....	79
Table 10.16 — Flags and Residual Count Relationship .....	81
Table 10.17 — SCSI fixed format sense data .....	83
Table 10.18 — Sense Key .....	85
Table 10.19 — DATA OUT UPIU .....	86
Table 10.20 — DATA IN UPIU .....	89
Table 10.21 — READY TO TRANSFER UPIU .....	92
Table 10.22 — Task Management Request UPIU .....	95
Table 10.23 — Task Management Function values .....	96

**Contents (cont'd)**


---

Table 10.24 — Task Management Input Parameters.....	96
Table 10.25 — Task Management Response UPIU .....	97
Table 10.26 — Task Management Output Parameters .....	98
Table 10.27 — Task Management Service Response.....	98
Table 10.28 — QUERY REQUEST UPIU .....	99
Table 10.29 — Query Function field values.....	101
Table 10.30 — Transaction specific fields .....	102
Table 10.31 — Query Function opcode values .....	102
Table 10.32 — Read descriptor .....	103
Table 10.33 — Write Descriptor .....	104
Table 10.34 — Read Attribute.....	105
Table 10.35 — Write Attribute.....	106
Table 10.36 — Read Flag.....	107
Table 10.37 — Set Flag.....	108
Table 10.38 — Clear Flag .....	109
Table 10.39 — Toggle Flag.....	110
Table 10.40 — NOP .....	111
Table 10.41 — QUERY RESPONSE.....	112
Table 10.42 — Query Response Code.....	113
Table 10.43 — Transaction Specific Fields.....	114
Table 10.44 — Read Descriptor .....	115
Table 10.45 — Write Descriptor .....	116
Table 10.46 — Read Attribute Response Data Format .....	117
Table 10.47 — Write Attribute.....	118
Table 10.48 — Read Flag Response Data Format.....	119
Table 10.49 — Set Flag.....	120
Table 10.50 — Clear Flag .....	121
Table 10.51 — Toggle Flag.....	122
Table 10.52 — NOP .....	123
Table 10.53 — Reject UPIU.....	124
Table 10.54 — Basic Header Status Description.....	125
Table 10.55 — E2E Status Definition .....	125
Table 10.56 — NOP OUT UPIU.....	126
Table 10.57 — NOP IN UPIU.....	128
Table 10.58 — Parameters related to data out transfer rules .....	133
Table 10.59 — Well known logical unit commands .....	136
Table 10.60 — Examples of logical unit representation format .....	137
Table 10.61 — Events for UAC establishment.....	140
Table 10.62 — Commands for exceptional behavior on UAC .....	140
Table 10.63 — UFS Initiator Port and Target Port Attributes.....	141
Table 10.64 — Send SCSI Command transport protocol service.....	143
Table 10.65 — SCSI Command Received transport protocol .....	144
Table 10.66 — Send Command Complete transport protocol service.....	145
Table 10.67 — Command Complete Received transport protocol service.....	146
Table 10.68 — Send Data-In transport protocol service .....	147
Table 10.69 — Data-In Delivered transport protocol service.....	147
Table 10.70 — Receive Data-Out transport protocol service .....	149
Table 10.71 — Data-Out Received transport protocol service .....	149
Table 10.72 — Task Management Function procedure calls .....	151
Table 10.73 — SCSI transport protocol service responses.....	151
Table 10.74 — Send Task Management Request SCSI transport protocol service request.....	155
Table 10.75 — Task Management Request Received SCSI transport protocol service indication .....	155
Table 10.76 — Task Management Function Executed SCSI transport protocol service response .....	155

---

**Contents (cont'd)**


---

Table 10.77 — Received Task Management Function Executed SCSI transport protocol service confirmation.....	156
Table 10.78 — Send Query Request UFS transport protocol service.....	157
Table 10.79 — Query Request Received UFS transport protocol service indication .....	158
Table 10.80 — Query Function Executed UFS transport protocol service response.....	158
Table 10.81 — Received Query Function Executed UFS transport protocol service confirmation .....	159
Table 11.1 — UFS SCSI Command Set.....	161
Table 11.2 — INQUIRY command.....	162
Table 11.3 — Standard INQUIRY data format .....	163
Table 11.4 — Standard INQUIRY Response Data .....	164
Table 11.5 — MODE SELECT (10) command.....	165
Table 11.6 — Mode Select Command Parameters .....	166
Table 11.7 — MODE SENSE (10) command .....	168
Table 11.8 — Mode Sense Command Parameters .....	168
Table 11.9 — Page Control Function .....	169
Table 11.10 — READ (6) command .....	170
Table 11.11 — READ (10) command .....	171
Table 11.12 — READ (16) command .....	173
Table 11.13 — READ CAPACITY (10) command .....	175
Table 11.14 — Read Capacity (10) Parameter Data.....	176
Table 11.15 — READ CAPACITY (16) command .....	177
Table 11.16 — Read Capacity (16) Parameter Data.....	179
Table 11.17 — START STOP UNIT command.....	181
Table 11.18 — TEST UNIT READY command .....	182
Table 11.19 — REPORT LUNS command.....	183
Table 11.20 — Report LUNS Command Parameters .....	184
Table 11.21 — SELECT REPORT field .....	184
Table 11.22 — Report LUNS Parameter Data Format .....	185
Table 11.23 — Single level LUN structure using peripheral device addressing method .....	185
Table 11.24 — Well Known Logical Unit Extended Addressing Format .....	186
Table 11.25 — Format of LUN field in UPIU .....	187
Table 11.26 — Well known logical unit numbers .....	187
Table 11.27 — VERIFY (10) command .....	188
Table 11.28 — Verify Command Parameters .....	189
Table 11.29 — WRITE (6) command .....	190
Table 11.30 — WRITE (10) command .....	192
Table 11.31 — WRITE (16) command .....	195
Table 11.32 — REQUEST SENSE command .....	198
Table 11.33 — FORMAT UNIT command .....	200
Table 11.34 — Format Unit Command Parameters .....	201
Table 11.35 — PRE_FETCH command.....	202
Table 11.36 — PRE-FETCH Command Parameters .....	203
Table 11.37 — PRE-FETCH (16) command .....	205
Table 11.38 — SECURITY PROTOCOL IN command .....	206
Table 11.39 — SECURITY PROTOCOL OUT command .....	207
Table 11.40 — SEND DIAGNOSTIC command .....	209
Table 11.41 — Send Diagnostic Parameters .....	209
Table 11.42 — SYNCHRONIZE CACHE (10) command .....	211
Table 11.43 — Synchronize Cache Command Parameters .....	212
Table 11.44 — SYNCHRONIZE CACHE (16) Command Descriptor Block .....	214
Table 11.45 — UNMAP command .....	215
Table 11.46 — UNMAP parameter list .....	216
Table 11.47 — UNMAP block descriptor .....	217
Table 11.48 — READ BUFFER command.....	219

**Contents (cont'd)**


---

Table 11.49 — Read Buffer Command Mode Field Values.....	220
Table 11.50 — Buffer ID Field for Error History Mode .....	221
Table 11.51 — Error history directory .....	222
Table 11.52 — EHS_SOURCE field.....	223
Table 11.53 — Error history directory entry .....	223
Table 11.54 — WRITE BUFFER command .....	225
Table 11.55 — Write Buffer Command Parameters.....	226
Table 11.56 — Write Buffer Command Mode Field Values.....	226
Table 11.57 — Mode page code usage .....	229
Table 11.58 — UFS Mode parameter list .....	230
Table 11.59 — UFS Mode parameter header (10).....	230
Table 11.60 — Mode Parameter Header Detail.....	231
Table 11.61 — Page_0 mode page format.....	232
Table 11.62 — Page 0 Format parameters.....	232
Table 11.63 — Sub_page mode page format.....	233
Table 11.64 — Subpage Format parameters .....	233
Table 11.65 — UFS Supported Pages .....	234
Table 11.66 — Control Mode Page default value .....	235
Table 11.67 — Control Mode Page Parameters .....	236
Table 11.68 — Read-Write Error Recovery Mode Page default value .....	237
Table 11.69 — Read-Write Error Recovery Parameters .....	238
Table 11.70 — Caching Mode Page default value .....	239
Table 11.71 — Caching Mode Page Parameters .....	240
Table 11.72 — VPD page format .....	241
Table 11.73— Supported VPD Pages VPD page .....	242
Table 11.74 — Mode Page Policy VPD page.....	243
Table 11.75 — Mode page policy descriptor.....	243
Table 11.76 — MODE PAGE POLICY field .....	244
Table 12.1 — Secure Write Protect Configuration Block .....	255
Table 12.2 — Secure Write Protect Entry .....	256
Table 12.3 — Write Protect Type field .....	257
Table 12.4 — RPMB Message Components .....	258
Table 12.5 — Request Message Types.....	259
Table 12.6 — Response Message Types .....	260
Table 12.7 — Result data structure .....	261
Table 12.8 — Result code definition .....	262
Table 12.9 — RPMB Message Data Frame.....	263
Table 12.10 — CDB format of SECURITY PROTOCOL IN/OUT commands .....	265
Table 12.11 — SECURITY PROTOCOL SPECIFIC field for protocol ECh.....	266
Table 12.12 — Security Protocol specific field values for protocol 00h .....	266
Table 12.13 — Supported security protocols SECURITY PROTOCOL IN parameter data .....	267
Table 12.14 — UFS Supported security protocols SECURITY PROTOCOL IN parameter data .....	267
Table 12.15 — Certificate data SECURITY PROTOCOL IN parameter data .....	268
Table 12.16 — UFS certificate data SECURITY PROTOCOL IN parameter data .....	268
Table 12.17 — Expected Data Transfer Length value for Request Type Messages.....	269
Table 12.18 — Expected Data Transfer Length value for Response Type Messages .....	271
Table 13.1 — bBootLunEn Attribute .....	287
Table 13.2 — Valid UPIUs and SCSI Commands for Each Initialization Phase .....	292
Table 13.3 — Logical unit configurable parameters .....	298
Table 13.4 — Parameter for controlling logical unit data reliability .....	313
Table 14.1 — Descriptor identification values .....	334
Table 14.2 — Generic Descriptor Format .....	339
Table 14.3 — Logical Unit Descriptor Format.....	339

---

**Contents (cont'd)**

Table 14.4 — Device Descriptor .....	340
Table 14.5 — wManufacturerID definition .....	348
Table 14.6 — Configuration Descriptor Format (INDEX = 00h) .....	349
Table 14.7 — Configuration Descriptor Format (INDEX = 01h) .....	349
Table 14.8 — Configuration Descriptor Format (INDEX = 02h) .....	349
Table 14.9 — Configuration Descriptor Format (INDEX = 03h) .....	350
Table 14.10 — Configuration Descr. Header and Device Descr. Conf. parameters (INDEX = 00h) .....	351
Table 14.11 — Configuration Descr. Header with INDEX = 01h/02h/03h .....	352
Table 14.12 — Unit Descriptor configurable parameters .....	352
Table 14.13 — Geometry Descriptor .....	353
Table 14.14 — Unit Descriptor .....	361
Table 14.15 — RPMB Unit Descriptor .....	365
Table 14.16 — Power Parameters Descriptor .....	367
Table 14.17 — Interconnect Descriptor .....	368
Table 14.18 — Manufacturer Name String .....	368
Table 14.19 — Product Name String .....	369
Table 14.20 — OEM_ID String .....	369
Table 14.21 — Serial Number String Descriptor .....	370
Table 14.22 — Product Revision Level String .....	370
Table 14.23 — Device Health Descriptor .....	371
Table 14.24 — Flags access properties .....	373
Table 14.25 — Flags .....	374
Table 14.26 — Attributes access properties .....	378
Table 14.27 — Attributes .....	379

---

## 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.

1                   **UNIVERSAL FLASH STORAGE (UFS), VERSION 3.1**

2       (From JEDEC Board Ballot JCB-19-31, formulated under the cognizance of the JC-64.1 Subcommittee  
3       on Electrical Specifications and Command Protocols, Item 135.99)

---

4                   **1       Scope**

---

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

---

12                  **2       Normative Reference**

---

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

- 18       [MIPI-M-PHY], *MIPI Alliance Specification for M-PHY<sup>®</sup>, Version 4.1, 28 March 2017*  
19       [MIPI-UniPro], *MIPI Alliance Specification for Unified Protocol (UniPro<sup>®</sup>), Version 1.8, 11 January*  
20       *2018*  
21       [MIPI-DDB], *MIPI Alliance Specification for Device Descriptor Block (DDB), Version 1.0*  
22       [MIPI-AP], *Application Note for UniPro<sup>®</sup> v1.8*  
23       [SAM], *INCITS T10 draft standard: SCSI Architecture Model – 5 (SAM–5), Revision 05, 19 May 2010*  
24       [SPC], *INCITS T10 draft standard: SCSI Primary Commands – 4 (SPC-4), Revision 27, 11 October 2010*  
25       [SBC], *INCITS T10 draft standard: SCSI Block Commands – 3 (SBC–3), Revision 24, 05 August 2010*  
26       [JESD8-12A], *1.2 V +/- 0.1V (Normal Range) and 0.8 - 1.3 V (Wide Range) Power Supply Voltage and*  
27       *Interface Standard for Nonterminated Digital Integrated Circuits*  
28       [HBM-MM], *JEDEC Recommended ESD Target Levels for HBM/MM Qualification, JEP155A.01,*  
29       *March 2012*  
30       [CDM], *JEDEC Recommended ESD-CDM Target Levels, JEP157, October 2009*  
31       [HMAC-SHA], *Eastlake, D. and T. Hansen, US Secure Hash Algorithms (SHA and HMAC-SHA), RFC*  
32       *4634, July 2006.*  
33       [JEP106], *Standard Manufacturer's identification code, JEP106*  
34       [JESD21C], *Multichip Packages (MCP) and Discrete eMMC, e2MMC, and UFS*  
35       [JESD220-3], *UFS Host Performance Booster (HPB) Extension Specification*

36    **3 Terms and Definitions**

---

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

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

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

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

45    **Device ID:** The bus address of a UFS device.

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

47    **Doubleword:** A 32-bit data value with most significant bit labeled as bit 31 and least significant bit as bit  
48    0.

49    **Dword:** 32-bit data value, a Doubleword.

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

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

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

55    **Kilobyte:** 1024 or  $2^{10}$  bytes.

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

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

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

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

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

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

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

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

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

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

74    **3 Terms and Definitions (cont'd)**

75    **Terabyte:** 1.099.511.627.776 or  $2^{40}$  bytes.

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

80    **Unit:** A bus device

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

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

84    **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
HPB	UFS Host Performance Booster
IID	Initiator ID
KB	Kilobyte
LBA	Logical Block Address
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

86    **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

87    **3.2 Conventions**

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

89 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  
90 in binary number representations to increase readability or delineate field boundaries (e.g., 0 0101 1010b).

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

96 A decimal number is represented in this standard by any sequence of digits consisting of only the  
97 Western-Arabic numerals 0 through 9 not immediately followed by a lower-case b or lower-case h (e.g.,  
98 25).

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

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

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

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

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

110    **3.3    Keywords**

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

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

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

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

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

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

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

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

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

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

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

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

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

144    **3.4    Abbreviations**

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

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

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

148 **4 Introduction**

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

152 **4.1 General Features**

- 153 • Target performance
  - 154 ○ High speed GEARs<sup>(1)</sup>
    - 155 ▪ Support for GEAR1 is mandatory
    - 156 ▪ Support for GEAR2 is mandatory
    - 157 ▪ Support for GEAR3 is mandatory
    - 158 ▪ Support for GEAR4 is mandatory
- 159 • Target host applications
  - 160 ○ Mobile phone, UMPC, DSC, PMP, MP3 and any other applications that require mass storage,  
161 bootable mass storage, and external card
- 162 • Target device types
  - 163 ○ External card
  - 164 ○ Embedded device
    - 165 ▪ Mass storage and bootable mass storage
  - 166 ○ Future expansion of device class types
    - 167 ▪ I/O devices, camera, wireless, . . . , etc.
- 168 • Topology
  - 169 ○ One device per UFS port.
- 170 • UFS Layering
  - 171 ○ UFS Command Set Layer (UCS)
    - 172 ▪ Simplified SCSI command set based on SBC and SPC. UFS will not modify these SBC and SPC  
173 Compliant commands. Option for defining UFS Native command and future extension exist.
  - 174 ○ UFS Transport Protocol Layer (UTP)
    - 175 ▪ JEDEC to define the supported protocol layer, i.e., UTP for SCSI. This does not exclude the  
176 support of other protocol in UFS Transport Protocol Layer.
  - 177 ○ UFS Interconnect Layer (UIC)
    - 178 ▪ MIPI UniPro® [MIPI-UniPro] is adopted for data link layer
    - 179 ▪ MIPI M-PHY® [MIPI-M-PHY ] is adopted for physical layer
- 180 NOTE 1 See 6.4.1 for details.

181   **4.2   Interface Features**

- 182   • Three power supplies
  - 183       ◦ VCCQ power supply: 1.2 V (nominal)
  - 184       ◦ VCCQ2 power supply: 1.8 V (nominal)
  - 185       ◦ VCC power supply: 2.5 V/3.3 V (nominal)
- 186   • Signaling as defined by [MIPI-M-PHY]
  - 187       ◦ 400 mVp (not terminated)
  - 188       ◦ 200 mVp (terminated)
- 189   • 8b10b line coding, as defined by MIPI M-PHY
- 190   • High reliability – BER under  $10^{-10}$
- 191   • Two signaling schemes
  - 192       ◦ Low-speed mode with PWM signaling scheme
  - 193       ◦ High-Speed burst mode
- 194   • Multiple gears defined for both Low-Speed and High-Speed mode
- 195   • Adapt (M-PHY® version 4.1)
  - 196       ◦ M-RX equalizer training to adapt to the channel characteristic

197   **4.3   Functional Features**

198   UFS functional features are NAND management features. These include

- 199   • Similar functional features as eMMC
- 200   • Boot Operation Mode
- 201   • Multiple logical units with configurable characteristics
- 202   • Replay Protected Memory Block (RPMB)
- 203   • Reliable write operation
- 204   • Background operations
- 205   • Secure operations, Purge and Erase to enhance data security
- 206   • Write Protection options, including Permanent and Power-On Write Protection
- 207   • Signed access to a Replay Protected Memory Block
- 208   • HW Reset Signal
- 209   • Task management operations
- 210   • Power management operations

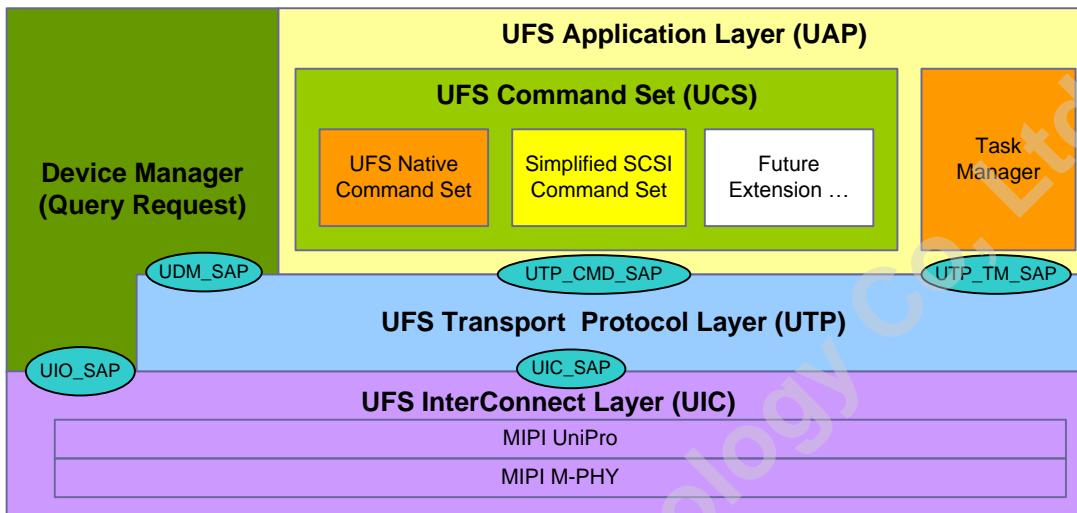
---

211    **5 UFS Architecture Overview**

---

212    **5.1 UFS Top Level Architecture**

213    Figure 5.1 shows the Universal Flash Storage (UFS) top level architecture.



214  
215    **Figure 5.1 — UFS Top Level Architecture**

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

218    **5.1.1 Application Layer**

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

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

226    **5.1.2 UFS Device Manager**

227    The device manager has the following two responsibilities:

- 228    • Handling device level operations.
- 229    • Managing device level configurations.

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

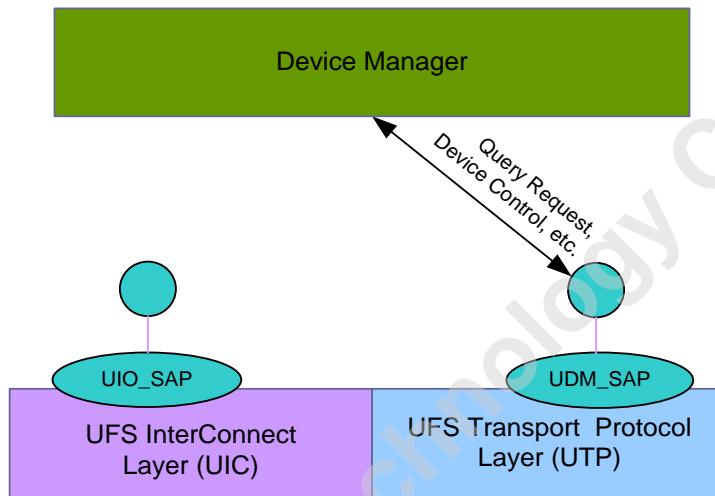
232    Device level configuration is managed by the device manager by maintaining and storing a set of  
233    descriptors. The device manager handles commands like query request which allow to modify or retrieve  
234    configuration information of the device.

235    **5.1.3 Service Access Points**

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

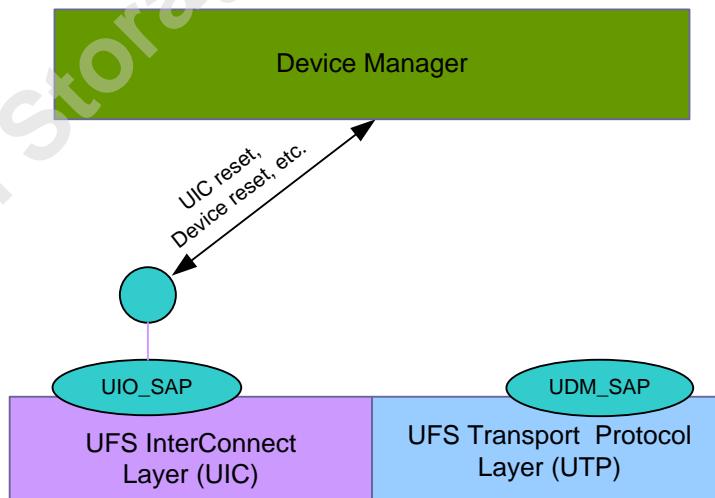
- 238    • UDM\_SAP  
239    • UIO\_SAP

240    UDM\_SAP is the service access point exposed by the UTP for the device manager to allow handling of  
241    device level operations and configurations. For example the handling of query request for descriptors  
242    would be done using this service access point. Figure 5.2 depicts the usage of the service access point.



243  
244    **Figure 5.2 — Usage of UDM\_SAP**

245  
246    UIO\_SAP is the service access point exposed by the UIC layer for the device manager to trigger the reset  
247    of the UIC layer and to transfer requests and responses related to UIC management functions. Figure 5.3  
248    depicts the usage of the service access point.



249  
250    **Figure 5.3 — Usage of UIO\_SAP**

251 **5.1.4 UIO\_SAP**

252 UIO\_SAP is the service access point exposed by the UIC layer. In UniPro, UIO\_SAP corresponds to  
253 DME\_SAP. The DME\_SAP provides service primitives including one for resetting the entire UniPro  
254 protocol stack and one for UFS device reset, etc.

- 255 • DME\_RESET : It is used when the UniPro stack has to be reset.  
256 • DME\_ENDIANRESET: It is used when UFS host wants the UFS device to perform a reset.

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

259 **5.1.5 UDM\_SAP**

260 UDM\_SAP is the service access point exposed by the UTP layer to the Device Manager for UFS device  
261 level functions. UDM\_SAP corresponds to the Query Request and Query Response functions defined by  
262 the UFS UTP layer.

263 For further details refer to the following subclauses: 10.9.9, Query Function transport protocol services,  
264 10.7.8, QUERY REQUEST UPIU, and 10.7.9, QUERY RESPONSE UPIU.

265 **5.1.6 UFS Transport Protocol Layer**

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

- 271 1) UFS Device Manager Service Access Point (UDM\_SAP) to perform the device level management  
272 like descriptor access.  
273 2) UTP Command Service Access Point (UTP\_CMD\_SAP) to transport commands.  
274 3) UTP Task Management Service Access Point (UTP\_TM\_SAP) to transport task-management  
275 function like “abort task” function.

276 **5.1.7 UFS Interconnect Layer**

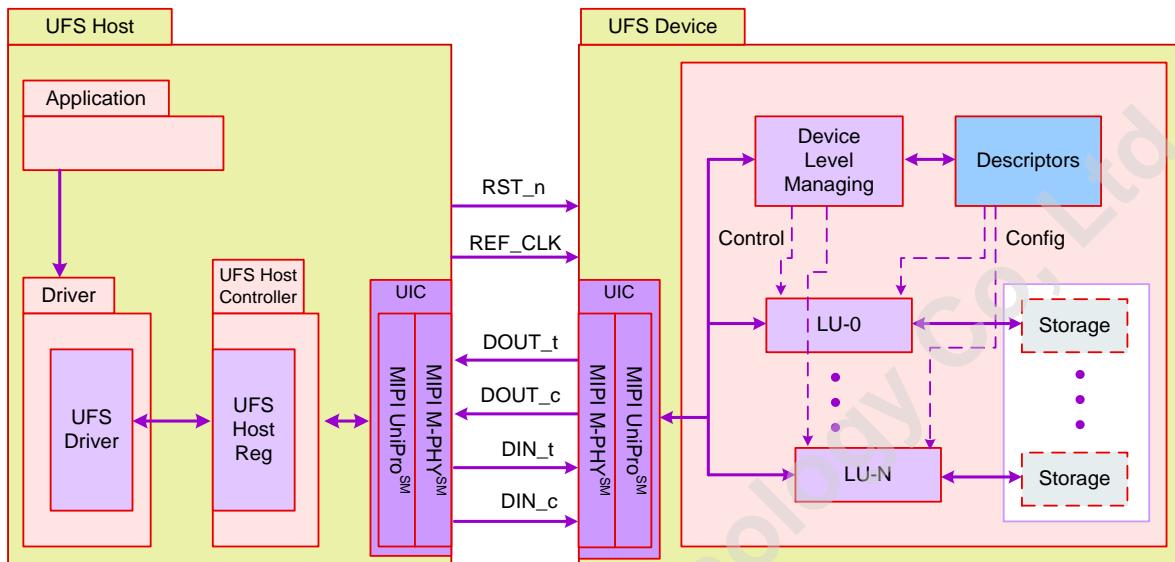
277 The lowest layer is UFS Interconnect Layer (UIC) which handles connection between UFS host and UFS  
278 device. UIC consists of MIPI UniPro and MIPI M-PHY. The UIC provides two service access points to  
279 upper layer. The UIC Service Access Point (UIC\_SAP) to transport UPIU between UFS host and UFS  
280 device. The UIC\_SAP corresponds to T\_SAP in UniPro. The UIC IO control Service Access Point  
281 (UIO\_SAP) to manage UIC. The UIO\_SAP corresponds to DME\_SAP in UniPro.

282 **5.1.8 UFS Topology**

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

285 **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.

302    5.3    System Boot and Enumeration

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

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

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

310 **5.4 UFS Interconnect (UIC) Layer**

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

313 **5.4.1 UFS Physical Layer Signals**

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

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

320 Table 5.1 summarizes the signals required for a UFS device. Only the single lane, per direction, per link,  
321 configuration is shown. See clause 6, UFS ELECTRICAL: CLOCK, RESET, SIGNALS AND  
322 SUPPLIES, and clause 8, UFS UIC Layer: MIPI M-PHY, for full details about UFS signals.

323 **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

324 **5.4.2 MIPI UniPro**

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

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

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

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

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

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

337 **5.4 UFS Interconnect (UIC) Layer (cont'd)**

338 **5.4.3 MIPI UniPro Related Attributes**

339 In general the UniPro related attributes, values and use of them are defined in the MIPI UniPro  
340 specification. The attributes may be generic for all UniPro applications and thus out of scope of this  
341 document. Following attributes are defined in this standard specifically for UFS application as indicated  
342 in Table 5.2.

343 **Table 5.2 — ManufacturerID and DeviceClass Attributes**

Attribute	AttributeID <sup>(1)</sup>	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]

344 **5.5 UFS Transport Protocol (UTP) Layer**

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

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

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

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

369   **5.5 UFS Transport Protocol (UTP) Layer (cont'd)**

370   **5.5.1 Architectural Model**

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

373   **5.5.1.1 Client-Server Model**

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

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

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

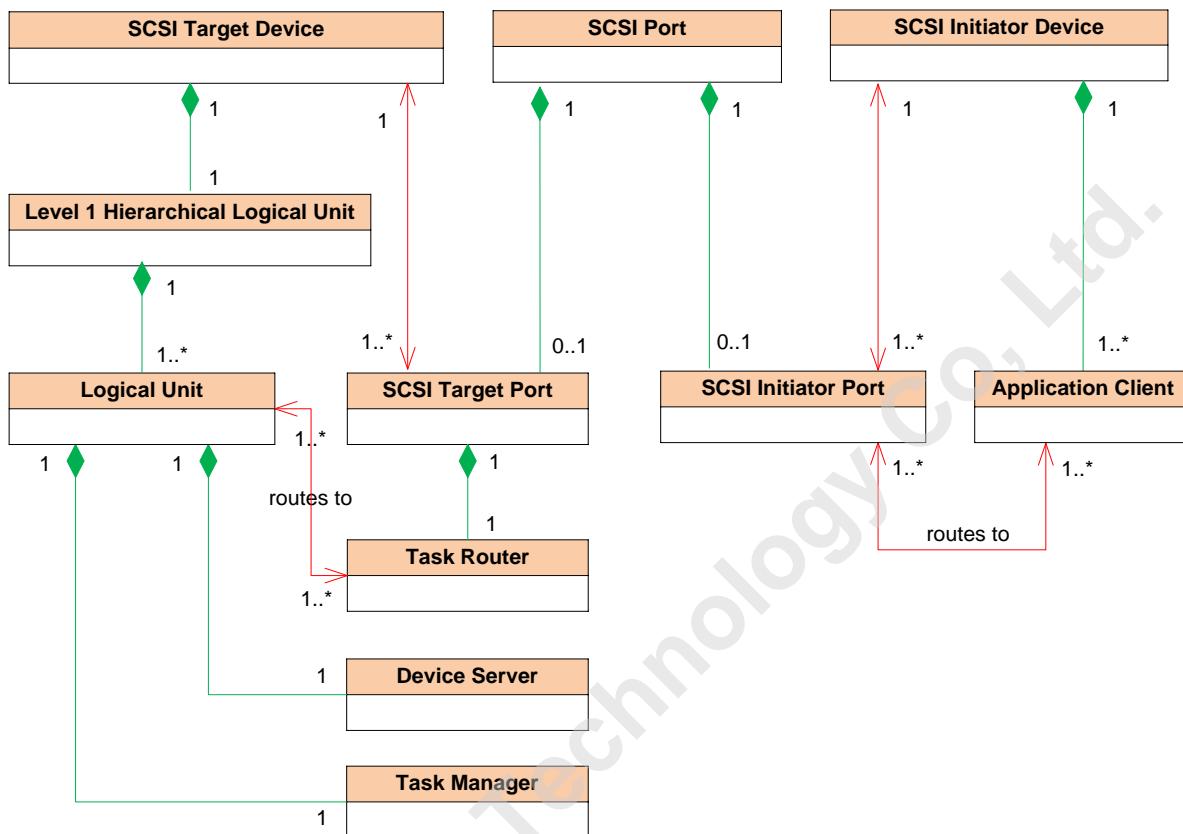
384   An Initiator request is directed to a single Logical Unit within a device. A Logical Unit will receive and  
385   process the client command or request. Each Logical Unit has an address within the Target device called a  
386   Logical Unit Number (LUN).

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

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

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

401 5.5.1.1 Client-Server Model (cont'd)

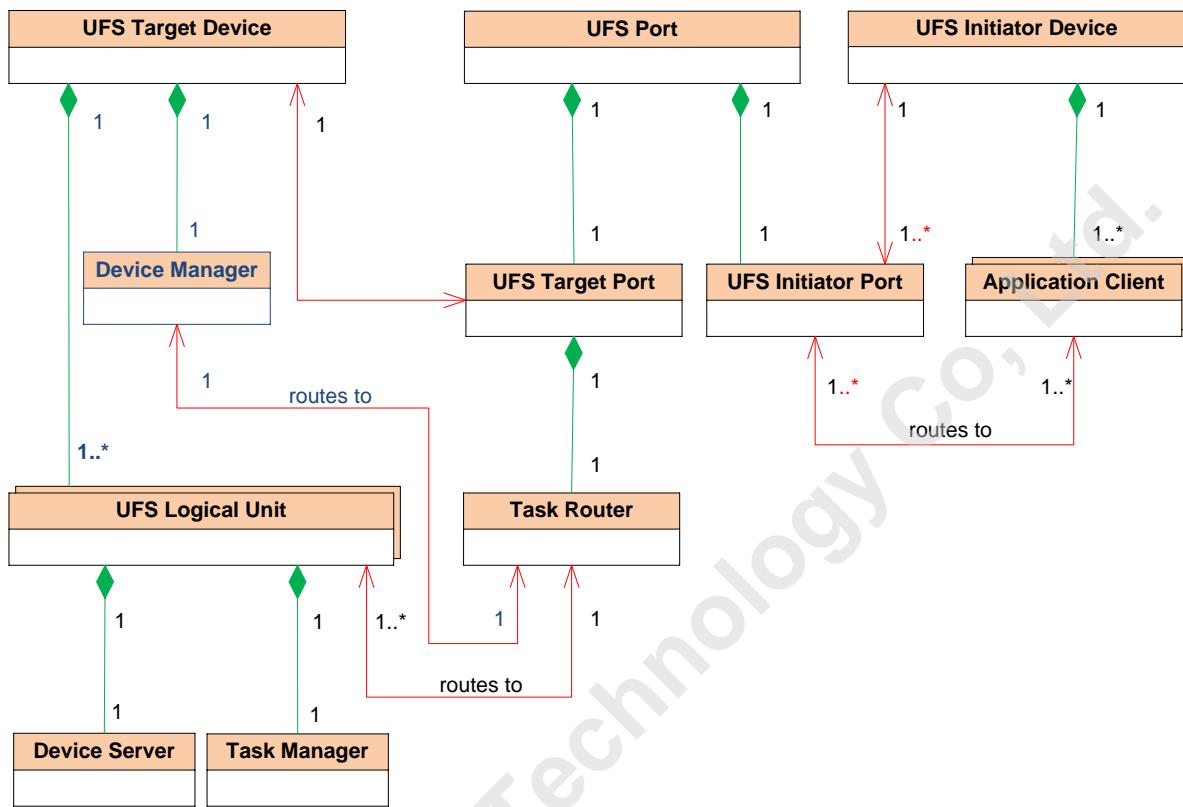


402  
403

404

Figure 5.5 — SCSI Domain Class Diagram

405    **5.5.1.1 Client-Server Model (cont'd)**



406  
407

408  
409

**Figure 5.6 — UFS Domain Class Diagram**

410 **5.5.1.2 CDB, Status, Task Management**

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

415 **5.5.1.3 Nexus**

416 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

418 There shall be at least one initiator device in the UFS definition. There shall only one target device, the  
419 UFS device. There shall be one or more logical units in a UFS device. The command identifier (i.e., the Q  
420 in an I\_T\_L\_Q nexus) is assigned by the Initiator device to uniquely identify one command in the context  
421 of a particular I\_T\_L nexus, allowing more than one command to be outstanding for the I\_T\_L nexus at  
422 the same time.

423 An overlapped command occurs when a task manager or a task router detects the use of a duplicate  
424 I\_T\_L\_Q nexus in a command before that I\_T\_L\_Q nexus completes its command lifetime (see [SAM]).

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

427 **5.5.1.4 SCSI Command Model**

428 All command requests originate from application clients in an Initiator device. An application Client  
429 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  
431 Size], [Data-Out Buffer], [Data-Out Buffer Size], [CRN], [Command Priority]), OUT ([Data-In  
432 Buffer], [Sense Data], [Sense Data Length], Status, [Status Qualifier]))

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

436 **5.5.1.5 SCSI Task Management Functions**

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

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

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

443 **5.6 UFS Application and Command Layer**

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

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

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

453 UFS SCSI command set includes:

454 1. SBC compliant commands [SBC]:

- 455 • FORMAT UNIT
- 456 • READ (6) and READ (10)
- 457 • READ CAPACITY (10)
- 458 • REQUEST SENSE
- 459 • SEND DIAGNOSTIC
- 460 • UNMAP
- 461 • WRITE (6) and WRITE (10)

462 2. SPC compliant commands [SPC]:

- 463 • INQUIRY
- 464 • REPORT LUNS
- 465 • READ BUFFER
- 466 • TEST UNIT READY
- 467 • WRITE BUFFER
- 468 • SECURITY PROTOCOL IN and SECURITY PROTOCOL OUT

469 3. SCSI operational commands for UFS applications and compatible with existing SCSI driver

- 470 • MODE SELECT (10) and MODE SENSE (10)
- 471 • PRE-FETCH (10)
- 472 • START STOP UNIT
- 473 • SYNCHRONIZE CACHE (10)
- 474 • VERIFY (10)

475 4. Value-added optional commands for UFS:

- 476 • READ (16), WRITE(16), PRE-FETCH (16), SYNCHRONIZE CACHE (16), and READ  
477 CAPACITY(16).

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

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

481 **5.7 Mechanical**

482 Packaging and requirements for UFS embedded device should adhere to the following guidelines if  
483 possible

- 484 • Reset and data transfer pins should be located in the second (PoP) or third row (MCP) in from the  
485 side of the package to prevent access.

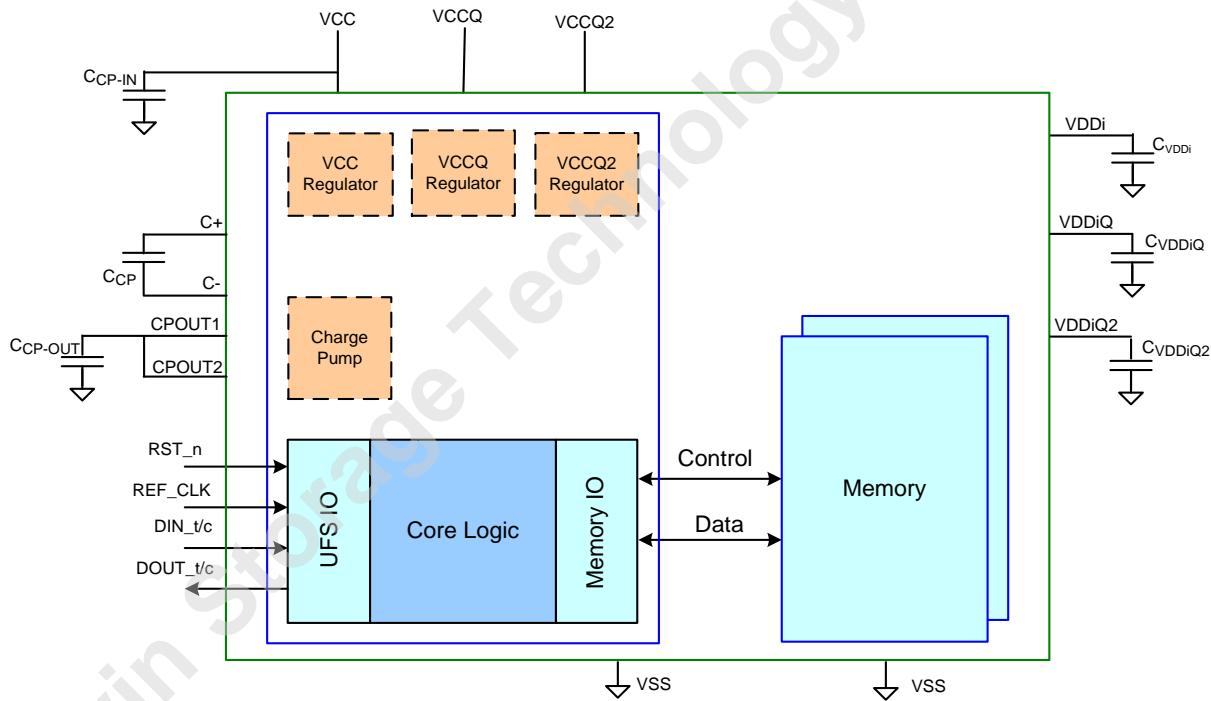
486

487

488 **6 UFS Electrical: Clock, Reset, Signals And Supplies**

489 **6.1 UFS Signals**

490 Figure 6.1 represents a conceptual drawing of UFS device. Utilization of internal regulators and  
491 connection of those to different parts of the sub-system may differ per implementation.



492  
493 NOTE 1 The memory core power supply may be connected to VCC power supply ball, or the VCC regulator  
494 output, while it is connected to the charge pump output if VCC = 1.8 V and the memory requires 2.5 V / 3.3 V core  
495 power supply.

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

497 NOTE 3 CCP-IN, CCP and CCP-OUT may be required only when internal charge pump is used.

498 **Figure 6.1 — UFS Device Block Diagram**

499

500      **6.1 UFS Signals (cont'd)**

501      **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 <sup>(1)</sup>	Input	Input terminal to provide bypass capacitor for VCCQ internal regulator
VDDiQ2 <sup>(1)</sup>	Input	Input terminal to provide bypass capacitor for VCCQ2 internal regulator
VDDi <sup>(1)</sup>	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 <sup>(2)</sup> DIN_c or DIN0_c <sup>(2)</sup>	Input	Downstream data lane 0. DIN_t is the positive node of the differential signal.
DIN1_t <sup>(2)</sup> , DIN1_c <sup>(2)</sup>	Input	Downstream data lane 1.
Differential output signals from the UFS device to the host		
DOUT_t or DOUT0_t <sup>(3)</sup> DOUT_c or DOUT0_c <sup>(3)</sup>	Output	Upstream data lane 0. DOUT_t is the positive node of the differential signal.
DOUT1_t <sup>(3)</sup> , DOUT1_c <sup>(3)</sup>	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.		

503    **6.2    Reset Signal**

504    To meet the requirements of the JEDEC Standard [JESD8-12A], the RST\_n signal voltages shall be  
505    within the specified ranges for VCCQ in Table 6.2.

506    **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	I <sub>lk</sub> g		10	μA	

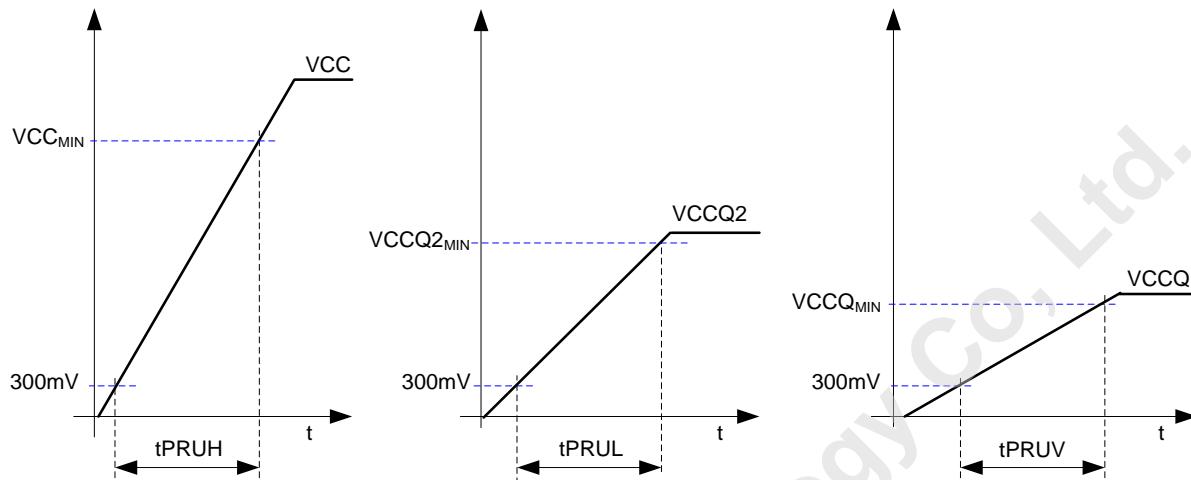
507    **6.3    Power Supplies**

508    **Table 6.3 — UFS Power Supply Parameters**

Parameter	Symbol	Min	Max	Unit	Notes
VCC DC operating range	VCC	2.7	3.6	V	3
		2.4	2.7	V	3
VCCQ DC operating range	VCCQ	1.14	1.26	V	1, 3
VCCQ2 DC operating range	VCCQ2	1.70	1.95	V	3
Supply Voltage power up timing for 3.3 V	tPRUH		35	ms	2
Supply Voltage power up timing for 2.5 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 <sub>VDDi</sub>	1		μF	
VCCQ internal regulator capacitor	C <sub>VDDiQ</sub>	1		μF	
VCCQ2 internal regulator capacitor	C <sub>VDDiQ2</sub>	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.					
NOTE 3 Depending on the vendor, valid power configuration may be defined in each UFS device vendor's data sheet. Refer to the vendor datasheet for the detail.					

510     **6.3**     Power Supplies (cont'd)

511 Figure 6.2 shows tPRUH, tPRUL and tPRUV timings for VCC = 3.3 V.



**Figure 6.2 — Supply voltage power up timings**

516    **6.4**    Reference Clock

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

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

528    6.4    Reference Clock (cont'd)

529

Table 6.4 — Reference Clock parameters

Parameter	Symbol	Nominal		Unit	Notes
Frequency	$f_{ref}$	19.2, 26, 38.4		MHz	1
Parameter	Symbol	Min	Max	Unit	Notes
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		$k\Omega$	7
	$CL_{RX}$		5	pF	

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

NOTE 2 Figure 6.4 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 6.4.

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 6.4.

NOTE 5 Integrated single side band phase noise from 50 KHz to 10 MHz. 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.

530

531    **6.4 Reference Clock (cont'd)**

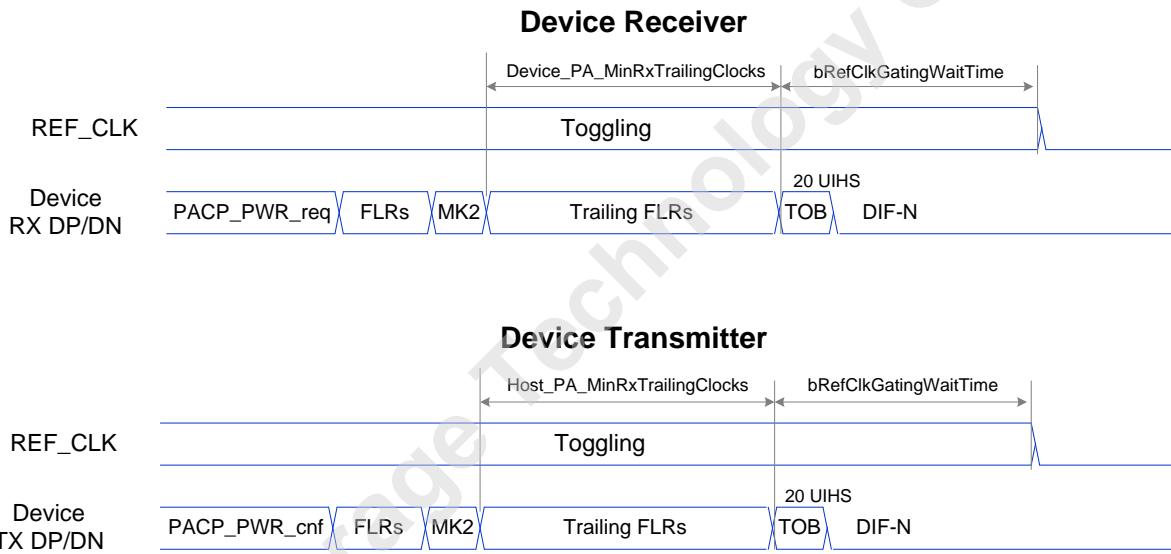
532    bRefClkFreq attribute indicates to the device the frequency of the REF\_CLK signal, and its default value  
 533    corresponds to 26 MHz.

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

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

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

540    If power mode change from HS-MODE to LS-MODE or HIBERN8 is initiated by UFS Host, then it shall  
 541    be ensured that at least the minimum time duration defined by bRefClkGatingWaitTime has elapsed  
 542    before turning off the reference clock, see **Figure 6.3**.



543    **Figure 6.3 — bRefClkGatingWaitTime**

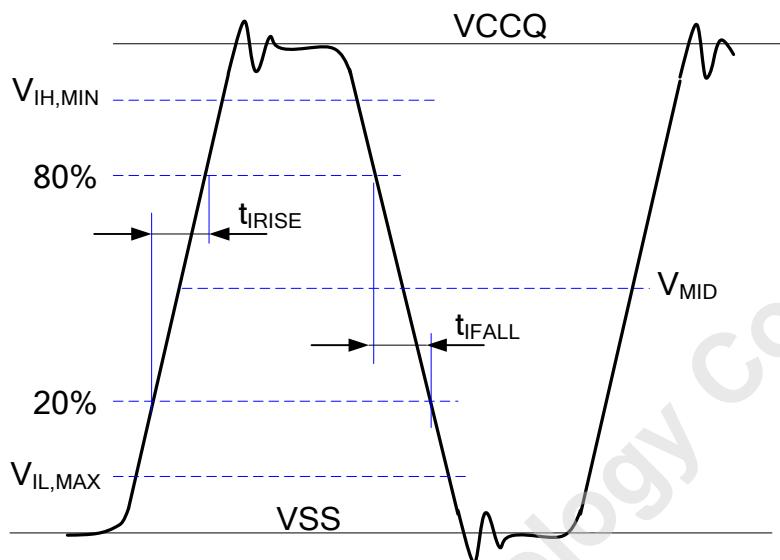
544    UFS Host may start a timer when DME\_POWERMODE.ind is received for HS-MODE to LS-MODE  
 545    transition or DME\_HIBERNATE\_ENTER.ind is received for HS-MODE to HIBERN8 transition. In  
 546    addition to bRefClkGatingWaitTime, Device PA\_MinRxTrailingClocks and Host  
 547    PA\_MinRxTrailingClocks should be considered to determine when the reference clock may be stopped.  
 548    In case of a transition from HS-MODE to HIBERN8 to reach the UFS-DeepSleep power mode, this  
 549    constraint for turning off the reference clock shall be applied as well.

550    Unipro layer defines re-initialization process using PA\_INIT mechanism. See [MIPI-Unipro] for details.  
 551    During this process both the sub-links may briefly enter LS-MODE before returning to HS-MODE. The  
 552    reference clock shall not be gated during the entire PA\_INIT procedure.

553    The reference clock shall be turned ON and stably running before initiation of the state transition to  
 554    STALL from a LS-MODE (LINE-CFG or SLEEP) or from the HIBERN8 state. Reference Clock shall  
 555    not be gated in HS modes.

557     **6.4**     Reference Clock (cont'd)

558 Figure 6.4 shows clock rise time and fall time measurements.



**Figure 6.4 — Clock input levels, rise time, and fall time**

## 561      6.4.1    HS Gear Rates

562 Table 6.5 defines the data rate values for the two rate series with respect REF\_CLK frequency value  
563 ( $f_{ref}$ ).

**Table 6.5 — HS-BURST Rates**

HS-GEAR	Rate A-series	Rate B-series		Rate A-series (from [MIPI-M-PHY])	Rate B-series <sup>(3)</sup> (from [MIPI-M-PHY])	Unit
	f <sub>ref</sub>	f <sub>ref</sub>		f <sub>ref</sub>	f <sub>ref</sub>	
	19.2 / 26 / 38.4	19.2 / 38.4	26	19.2 / 26 / 38.4		MHz
HS-GEAR1	1248 <sup>(2)</sup>	1459.2	1456.0	1248	1457.6	Mbps
HS-GEAR2	2496	2918.4	2912.0	2496	2915.2	Mbps
HS-GEAR3	4992	5836.8	5824.0	4992	5830.4	Mbps
HS-GEAR4	9984	11673.6	11648.0	9984	11660.8	Mbps

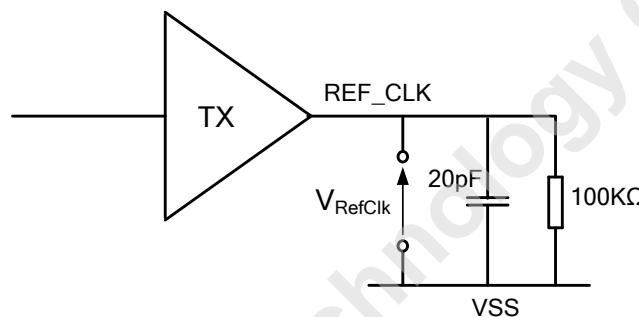
566      **6.4      Reference Clock (cont'd)**

567      **6.4.2    Host Controller requirements for reference clock generation**

568      **Table 6.6 — Host controller reference clock parameters**

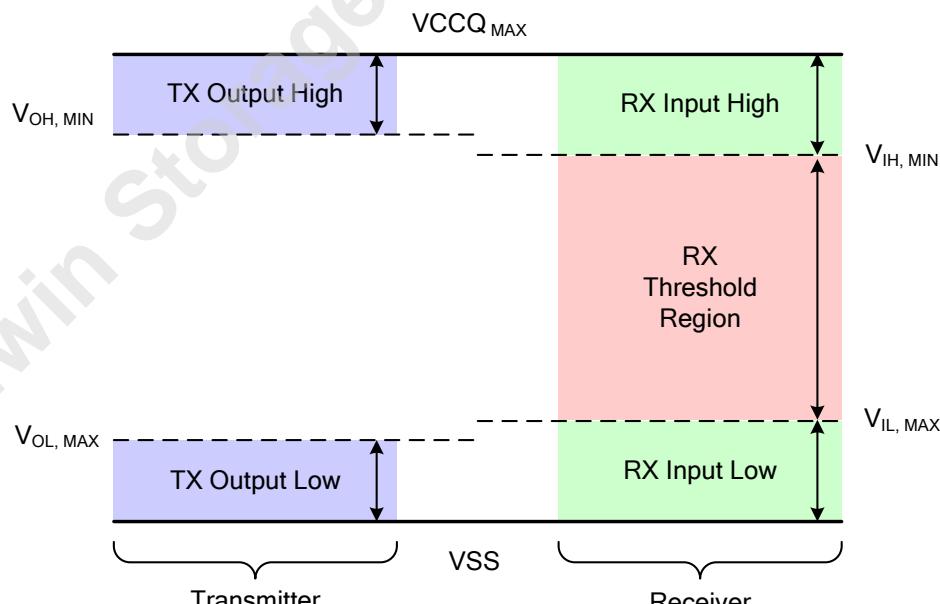
Parameter	Symbol	Min	Max	Unit	Notes
DC Output High Voltage	$V_{OH}$	0.75 * VCCQ		V	1, 2
DC Output Low Voltage	$V_{OL}$		0.25 * VCCQ	V	1, 2
Output Clock Rise Time	$t_{ORISE}$		2	ns	3
Output Clock Fall Time	$t_{OFALL}$		2	ns	3
Test Load Impedance	$RL_{Test}$	100		kΩ	1, 4, 5
	$CL_{Test}$	20		pF	1, 4, 5

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



**Figure 6.5 — Test Load Impedance**

NOTE 2 Figure 6.6 shows Output driver and Input receiver levels. REF\_CLK driver AC voltage (e.g., ring back) shall be kept inside Output Voltage limit.

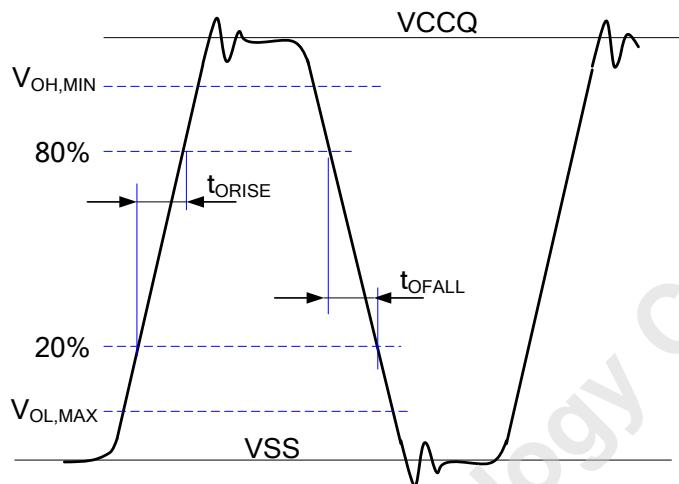


**Figure 6.6 — Output driver and Input receiver levels**

570 **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 6.7 — 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.

571 **6.5 External Charge Pump Capacitors (Optional)**

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

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

576 Figure 6.1 shows the electrical connections required in case of charge pump implementation that uses  
577 external capacitors.

578 Table 6.8 provides description of the capacitors to be used.

579 **Table 6.7 — 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

580 The charge pump capacitors are optional for UFS devices. Table 6.8 specifies name and description of the  
581 balls for connecting external charge pump capacitors.

582    **6.5 External Charge Pump Capacitors (Optional) (cont'd)**

583    **Table 6.8 — 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) <sup>1</sup>

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.

584    **6.6 Absolute Maximum DC Ratings and Operating Conditions**

585    Stresses greater than those listed in Table 6.9 may cause permanent damage to the device. This is a stress  
586    rating only, and functional operation of the device at these or any other conditions above those indicated  
587    in the operational sections of this standard is not implied. Exposure to absolute maximum rating  
588    conditions for extended periods may affect reliability.

589    **Table 6.9 — Absolute maximum DC ratings and Operating Conditions**

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 Standard	$T_{STG\_STD}$	- 40	85	°C	2, 5
Operating Temperature Standard	$T_{OPER\_STD}$	- 25	85	°C	3, 5
Storage Temperature Extended	$T_{STG\_EXT}$	- 40	105	°C	2, 4, 5
Operating Temperature Extended	$T_{OPER\_EXT}$	- 40	105	°C	3, 4, 5

NOTE 1 Voltage relative to VSS.

NOTE 2 Storage Temperature is the package case surface temperature of the UFS device when power is not supplied.

NOTE 3 Operating Temperature is the package case surface temperature of the UFS device when UFS device is operating.

NOTE 4 This is supported only when the device supports Extended Temperature which is indicated by bit[6] of bUFSFeaturesSupport as '1'.

NOTE 5 For device safety (e.g., keeping designed reliability characteristics of device), in case package case temperature exceeds defined temperature range, device may not guarantee proper operation.

---

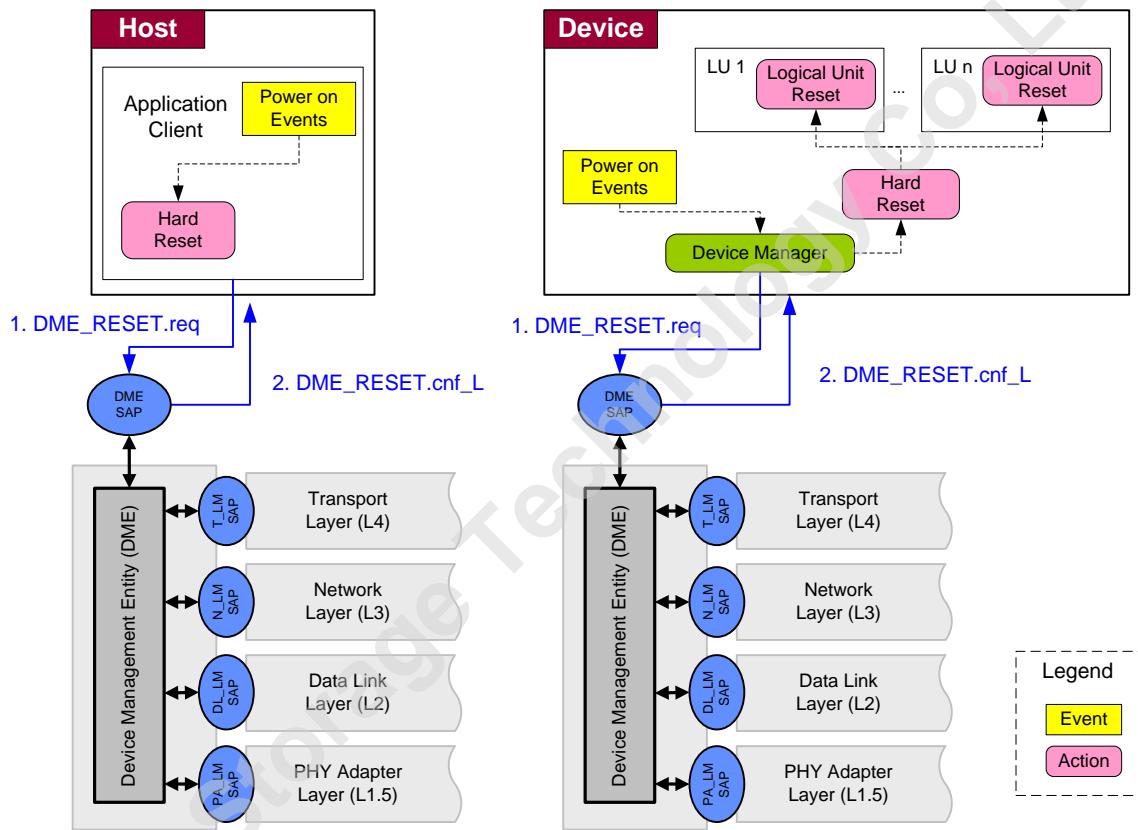
590 7 Reset, Power-Up And Power-Down

591 7.1 Reset

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

593 7.1.1 Power-on Reset

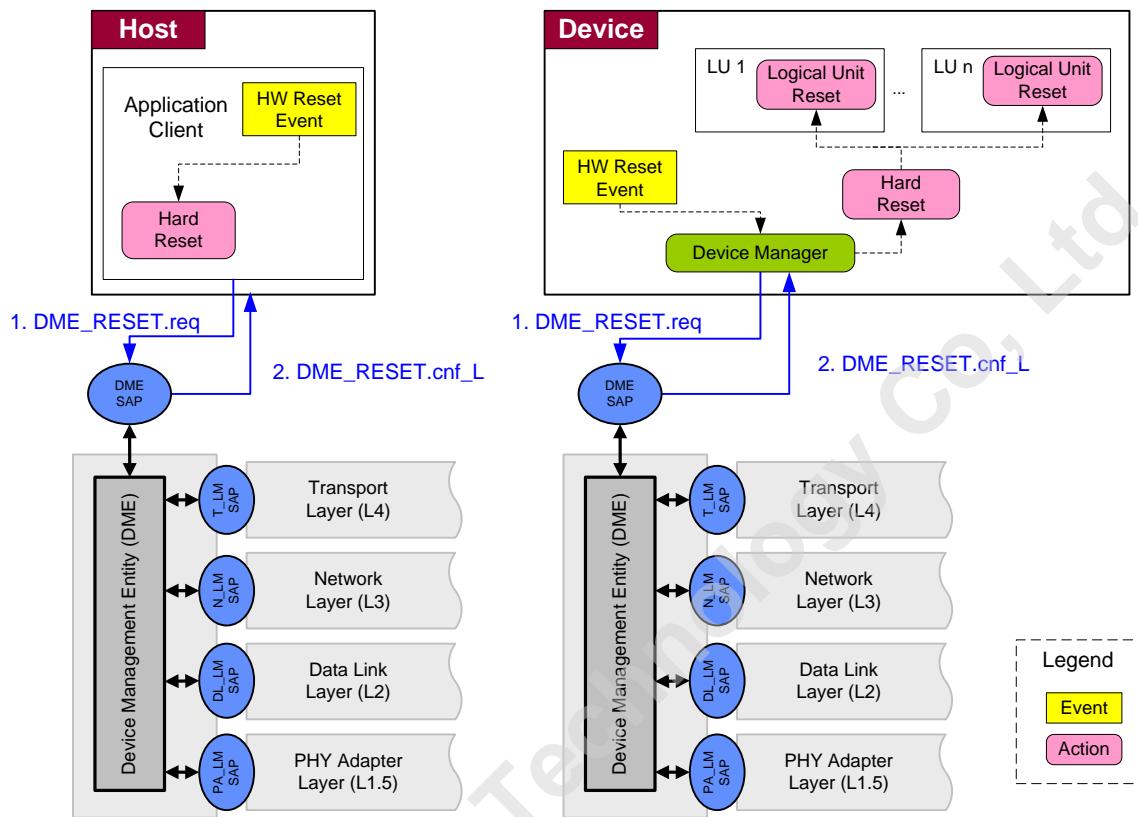
594 A power-on reset is obtained switching the VCCQ, VCCQ2 and VCC power supplies off and back on.  
595 The UFS device shall have its own power-on detection circuitry which puts the UFS device and all the  
596 different layers of it into a defined state after the power-on.



597  
598  
599 Figure 7.1 — Power-on Reset

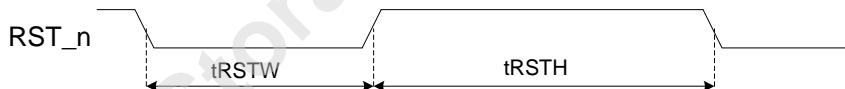
## 600 7.1.2 Hardware Reset

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



602  
603 **Figure 7.2 — Hardware Reset**

604 Figure 7.3 shows the hardware reset AC timings.



605  
606  
607  
608 **Figure 7.3 — Reset AC timings**

**Table 7.1 — Reset timing parameters**

Symbol	Comment	Min	Max	Unit
$t_{RSTW}$	RST_n Pulse Width	1		$\mu s$
$t_{RSTH}$	RST_n High Period (Interval)	1		$\mu s$
$t_{RSTF}$	RST_n filter	100		ns

609 The reset signal is active low. The UFS device shall not detect 100 ns or less of positive or negative  
610 RST\_n pulse. The UFS device shall detect more than or equal to 1us of positive or negative RST\_n pulse  
611 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.

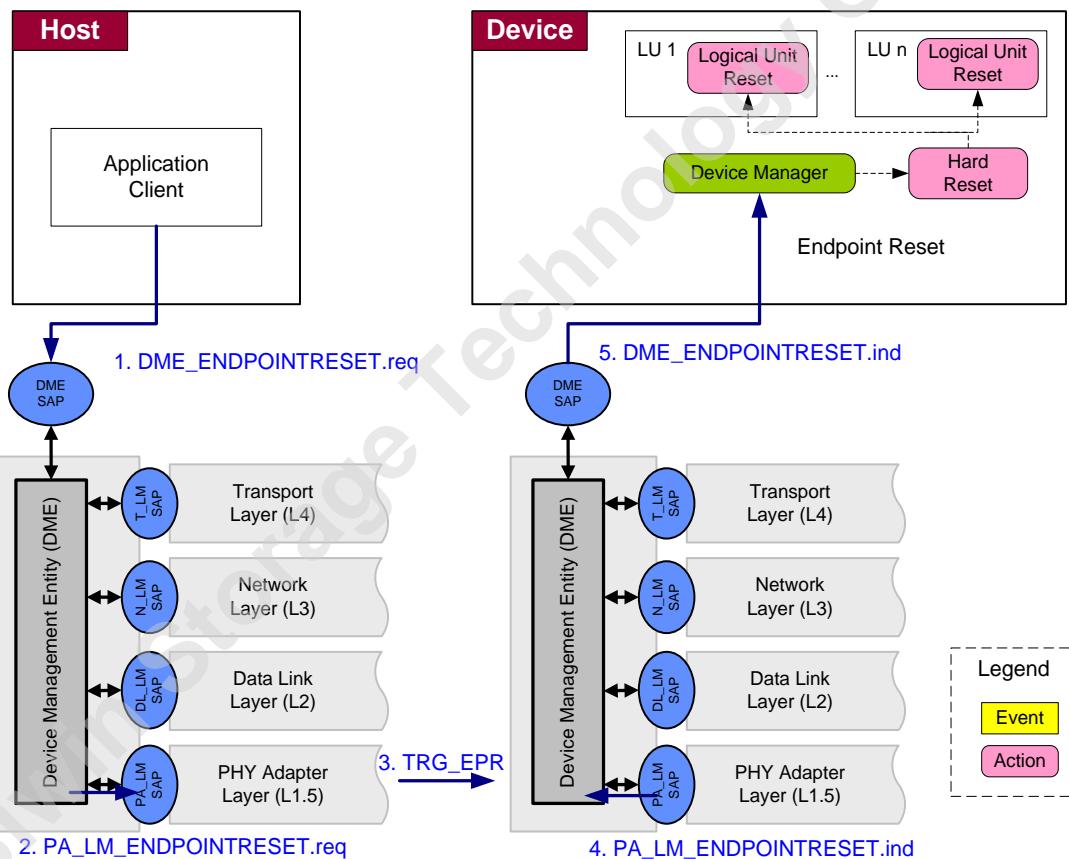


Figure 7.4 — EndPointReset

## 627 7.1.4 Logical Unit Reset

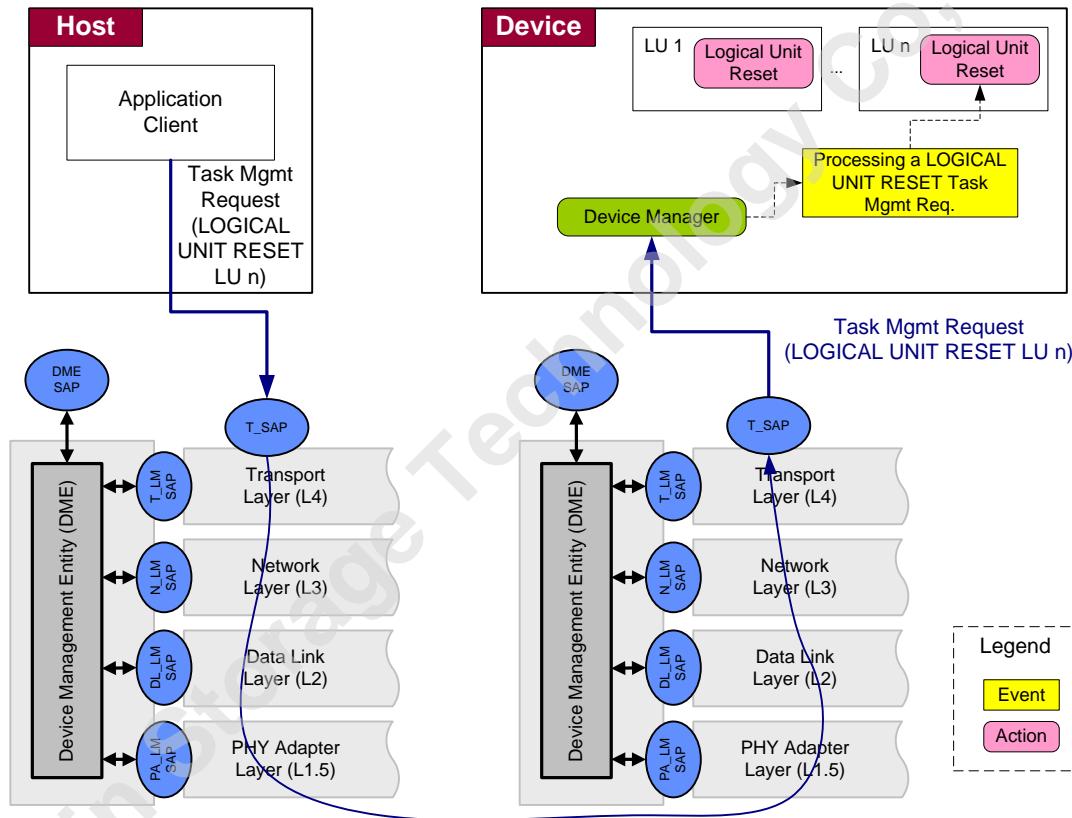
628 The Logical Unit Reset feature is defined in the SCSI Architectural Model [SAM]. This reset is triggered  
629 via the SCSI Task Management features described in 10.9.8.4.

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

631 Function Call from Host Application Client to Host UTP via UTP\_TM\_SAP: Task management  
632 LOGICAL UNIT RESET (IN ( I\_T\_L Nexus )).

633 LU Task manager shall receive the function call from device UTP via UTP\_TM\_SAP and executes the  
634 LU reset function.

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



637  
638 **Figure 7.5 — Logical Unit Reset**

## 639 7.1.5 Host UniPro Warm Reset

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

642 The Host System resets its own UniPro stack: the host's UniPro stack reset activity is signaled to the UFS  
643 Device's UniPro stack via the DME\_LINKLOST.ind message. In case the UFS device receives such  
644 DME\_LINKLOST.ind message from the host system, it shall start process of re-initializing its own  
645 UniPro stack. In addition all UFS device level activity shall be aborted, task queue lists in all logical units  
646 shall be cleared and UFS power mode shall return to UFS-Sleep power mode or Active power mode  
647 depending on bInitPowerMode.

648      **7.1.6 Summary of Resets and Device Behavior**

649      Table 7.2 and Table 7.3 summarize the different types of reset and the UFS device behavior related to  
650      them.

651      **Table 7.2 — Reset States**

Reset Type	Initiator Device	Current Power Mode	Power Mode after Reset		Boot Process <sup>(2)</sup>
			bInitPowerMode = 00h	bInitPowerMode = 01h	
Power-on	Host	Any	UFS-Sleep <sup>(1)</sup>	Active	Enabled
HW Reset	Host	Any	UFS-Sleep <sup>(1)</sup>	Active	Enabled
EndPointReset	Host	Any	UFS-Sleep <sup>(1)</sup>	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	UFS-Sleep <sup>(1)</sup>	Active	Enabled

NOTE 1 At the end of the device initialization, the power mode transitions from Active to Pre-Sleep and then UFS-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.

652

653

**Table 7.3 — UniPro Attributes, UFS Attributes and UFS Flags reset**

Reset Type	UniPro Stack and Attributes	Volatile and Set Only Attributes and Flags <sup>(1)</sup>	Power on reset Attributes and Flags <sup>(1)</sup>	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.

654

655 **7.2 Power up ramp**

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

657 • Ta is the point where VCCQ or VCCQ2 power supply first reaches 300 mV.

658 • After Ta is reached, VCCQ2 should be greater than VCCQ - 200 mV.

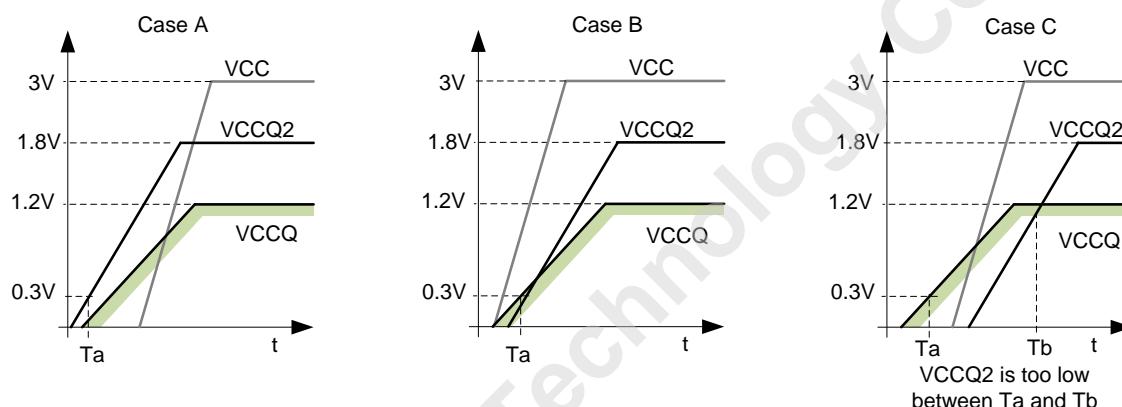
659 • VCC can be ramped up independently from VCCQ value or VCCQ2 value.

660 • While powering on the device,

661     ○ RST\_n signal should be kept low

662     ○ REF\_CLK signal should be between VSS and VCCQ.

663 Figure 7.6 shows three power up ramp examples: case A and case B meet the requirement, while case C  
664 violates it in the time interval from Ta to Tb (VCCQ2 is lower VCCQ - 200 mV).



665

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

667 **Figure 7.6 — Power up ramps**

668

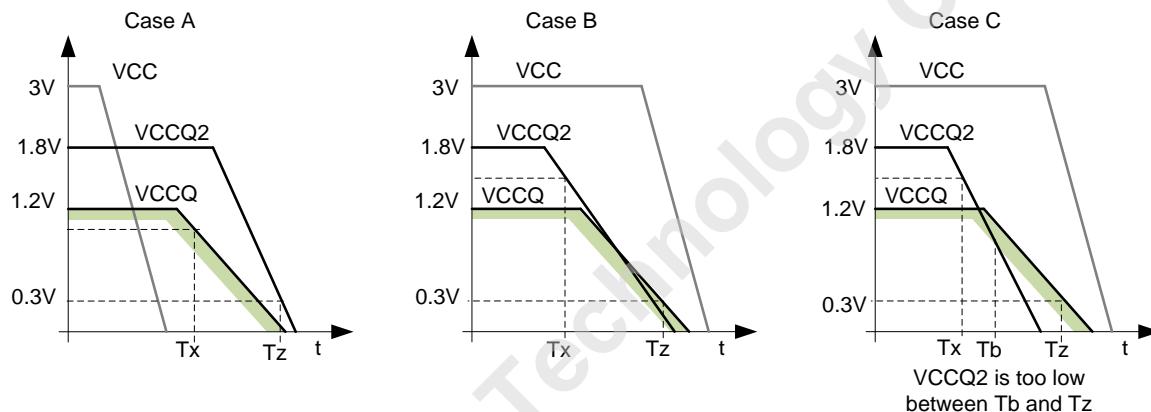
669

670 **7.3 Power off ramp**

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

- 672 • Tx is the point where VCCQ or VCCQ2 power supply decreases under its minimum operating  
673 condition value specified (see Table 6.3).
- 674 • Tz is the point where VCCQ and VCCQ2 power supplies are below 300 mV.
- 675 • VCCQ2 should be greater than VCCQ - 200 mV between Tx and Tz.
- 676 • VCC can be ramped down independently from VCCQ value or VCCQ2 value.
- 677 • While powering off the device, RST\_n signal and REF\_CLK signal should be between VSS and  
678 VCCQ.

679 Figure 7.7 shows three power down ramp examples: case A and case B meet the requirement, while case  
680 C violates it in the time interval from Tb to Tz.



681  
682

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

684 **Figure 7.7 — Power off ramps**

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

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

688 **7.4 UFS Device Power Modes and LU Power Condition**

689 **7.4.1 Device Power Modes**

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

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

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

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

712 **7.4.1.1 Active Power Mode**

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

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

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

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

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

731   **7.4.1.1 Active Power Mode (cont'd)**

732   Valid values for the bActiveICCLevel are from “00h” to “0Fh”, other values are reserved and should not  
733   be set. A request to set to bActiveICCLevel should be made only when there is no outstanding operation,  
734   i.e., queue of all logical units is empty. If a request to set to bActiveICCLevel is raised when any queue is  
735   not empty, then device may be terminated with Query Response field set to “General Failure”.

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

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

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

744   Active power mode may be entered from the Powered On power mode or the Pre-Active power mode  
745   after the completion of all setup necessary to handle commands.

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

747   All supported commands are available in Active Mode.

748   **7.4.1.2 Idle Power Mode**

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

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

754   **7.4.1.3 Pre-Active Power Mode**

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

758   Pre-Active power mode may be entered from Pre-Sleep, UFS-Sleep, Pre-PowerDown, or UFS-  
759   PowerDown power modes. The following power mode is the Active power mode.

760   While in Pre-Active power mode:

761   a) the Device well known logical unit may successfully complete only: START STOP UNIT command  
762   and REQUEST SENSE command; other commands may be terminated with CHECK CONDITION  
763   status, with the sense key set to NOT READY, with the additional sense code set to LOGICAL UNIT  
764   IS IN PROCESS OF BECOMING READY, see 0 for further details;

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

768

769 **7.4.1.4 UFS-Sleep Power Mode**

770 The UFS-Sleep power mode allows to reduce considerably the power consumption of the device.  
771 VCC power supply may be turned off in this power mode. The UFS-Sleep power mode is entered from  
772 Pre-Sleep power mode.

773 While in UFS-Sleep power mode:

- 774 a) the Device well known logical unit may successfully complete only: START STOP UNIT command  
775 and REQUEST SENSE command; other commands may be terminated with CHECK CONDITION  
776 status, with the sense key set to NOT READY and the additional sense code set to LOGICAL UNIT  
777 NOT READY, INITIALIZING COMMAND REQUIRED, see 0 for further details;
- 778 b) a REQUEST SENSE command shall be terminated with GOOD status and provide pollable sense  
779 data with the sense key set to NOT READY and the additional sense code set to LOGICAL UNIT  
780 NOT READY, INITIALIZING COMMAND REQUIRED.

781 In general, the M-PHY® interface may be in STALL, SLEEP or HIBERN8 state, but it is recommended  
782 for the host to put the link in HIBERN8 state to lower power consumption.

783 VCC power supply should be restored before issuing START STOP UNIT command to request transition  
784 to Active, UFS-DeepSleep, or UFS-PowerDown power mode.

785 **7.4.1.5 Pre-Sleep Power Mode**

786 The Pre-Sleep power mode is a transitional mode associated with UFS-Sleep entry. The power consumed  
787 shall be no more than that consumed in Active power mode. Pre-Sleep may be entered from Active  
788 power mode.

789 The device shall automatically advance to UFS-Sleep power mode once any outstanding operations and  
790 management activities have been completed.

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

793 While in Pre-Sleep power mode:

- 794 a) the Device well known logical unit may successfully complete only: START STOP UNIT command,  
795 REQUEST SENSE command and task management functions; other commands may be terminated  
796 with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, see 0 for further  
797 details;
- 798 b) a REQUEST SENSE command shall be terminated with GOOD status and provide pollable sense  
799 data with the sense key set to NO SENSE and the additional sense code set to LOGICAL UNIT  
800 TRANSITIONING TO ANOTHER POWER CONDITION.

802 **7.4.1.6 UFS-DeepSleep Power Mode**

803 The UFS-DeepSleep power mode is used to achieve the lowest power consumption of the device. VCC  
804 power supply may be turned off in this power mode.

805 The UFS-DeepSleep power mode is entered from Pre-DeepSleep power mode.

806 While in UFS-DeepSleep power mode, the Device does not respond to any host commands. The M-PHY®  
807 interface may be put in UNPOWERED state by the UFS device to minimize the power consumption.

808 Host is expected to request the transition to UFS-DeepSleep power mode only when there is no pending  
809 task request or task management request. No further commands are accepted in Pre-DeepSleep power  
810 mode and UFS-DeepSleep power mode. Also IMMED bit shall be set to zero for the START STOP  
811 UNIT command requesting transition to UFS-DeepSleep power mode. Device responds to START STOP  
812 UNIT command when the device is ready to transition from Pre-DeepSleep power mode to UFS-  
813 DeepSleep power mode. Transition to UFS-DeepSleep power mode occurs when the device link has  
814 entered HIBERN8 state.

815 The only way to exit from this power mode is using a hardware reset or a power cycle. Host is expected to  
816 wake up the device by using a hardware reset or a power cycle after receiving response to START STOP  
817 UNIT command because UFS device may be performing management activities in Pre-DeepSleep power  
818 mode. After a hardware reset or a power cycle, UFS power mode shall return to UFS-Sleep power mode  
819 or Active power mode depending on bInitPowerMode.

820 **7.4.1.7 Pre-DeepSleep Power Mode**

821 The Pre-DeepSleep power mode is a transitional mode associated with UFS-DeepSleep entry. The power  
822 consumed shall be no more than that consumed in Active power mode. Pre-DeepSleep may be entered  
823 from Active or UFS-Sleep power mode.

824 The device sends the response with GOOD status to START STOP UNIT command with the POWER  
825 CONDITION field set to 4h after any outstanding operations and management activities have been  
826 completed. Then the device waits for HIBERN8 state transition. The host is expected to put the link in  
827 HIBERN8 state after receiving the response to the START STOP UNIT command. The device shall  
828 transit to UFS-DeepSleep power mode after HIBERN8 state transition is completed.

829 While in Pre-DeepSleep power mode, the Device does not respond to any host commands.

830 **7.4.1.8 UFS-PowerDown Power Mode**

831 The UFS-PowerDown power mode should be used prior to completely powering off the UFS memory  
832 device. All volatile data may be lost, and VCC or all power supplies can be removed.

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

835 While in UFS-PowerDown power mode:

- 836 a) the Device well known logical unit may successfully complete only: START STOP UNIT command  
837 and REQUEST SENSE command; other commands may be terminated with CHECK CONDITION  
838 status, with the sense key set to NOT READY and the additional sense code set to LOGICAL UNIT  
839 NOT READY, INITIALIZING COMMAND REQUIRED, see 0 for further details;
- 840 b) a REQUEST SENSE command shall be terminated with GOOD status and provide pollable sense  
841 data with the sense key set to NOT READY, and the additional sense code set to LOGICAL UNIT  
842 NOT READY, INITIALIZING COMMAND REQUIRED.

843   **7.4.1.9 Pre-PowerDown Power Mode**

844   The Pre-PowerDown power mode is a transitional mode associated with UFS-PowerDown entry. The  
845   power consumed shall be no more than that consumed in Active power mode. Pre-PowerDown may be  
846   entered from Active or UFS-Sleep power mode.

847   The device shall automatically advance to UFS-PowerDown power mode once any outstanding  
848   operations and management activities have been completed.

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

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

852   While in Pre-PowerDown power mode:

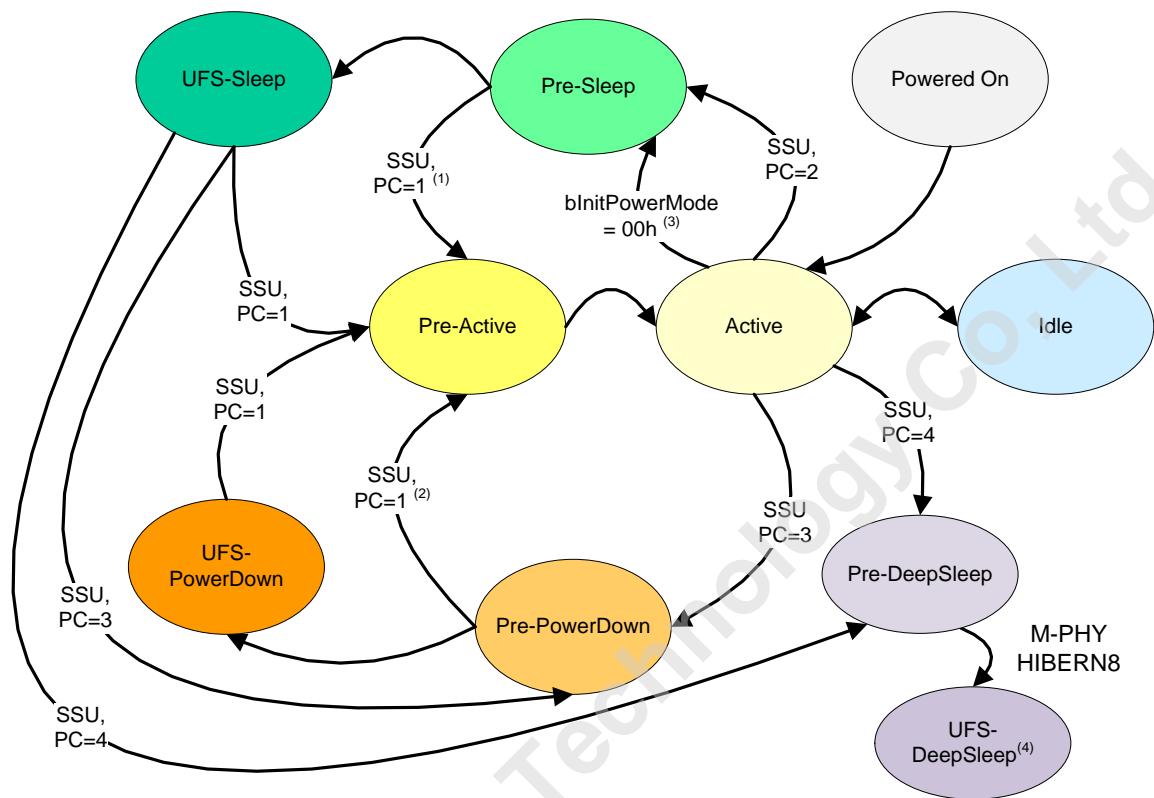
853   a) the Device well known logical unit may successfully complete only: START STOP UNIT command,  
854       REQUEST SENSE command and task management functions; other commands may be terminated  
855       with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, see 0 for further  
856       details;

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

860

861 **7.4.1.10 Power Mode State Machine**

862 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.

(4) The only way to exit from UFS-DeepSleep power mode is using a hardware reset or a power cycle.

863

864 **Figure 7.8 — Power Mode State Machine**

865

866 **7.4.1.10.1 Transitions from Powered On Power Mode**

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

869 **Transition from Powered\_On to Active**

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

871 **7.4.1.10.2 Transitions from Pre-Active Power Mode**

872 **Transition from Pre-Active to Active**

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

874 **7.4.1.10.3 Transitions from Active Power Mode**

875 **Transition from Active to Idle**

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

877 **Transition from Active to Pre-Sleep**

878 This transition shall occur

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

882 **Transition from Active to Pre-DeepSleep**

883 This transition shall occur if the device server processes a START STOP UNIT command with the  
884 POWER CONDITION field set to 4h.

885 **Transition from Active to Pre-PowerDown**

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

888 **7.4.1.10.4 Transitions from Idle Power Mode**

889 **Transition from Idle to Active**

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

891 **7.4.1.10.5 Transitions from Pre-Sleep Power Mode**

892 **Transition from Pre-Sleep to Pre-Active**

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

896 **Transition from Pre-Sleep to Sleep**

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

898 **7.4.1.10.6 Transitions from UFS-Sleep Power Mode**

899 **Transition from UFS-Sleep to Pre-Active**

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

902 **Transition from UFS-Sleep to Pre-DeepSleep**

903 This transition shall occur if the device server processes a START STOP UNIT command with the  
904 POWER CONDITION field set to 4h.

905 **Transition from UFS-Sleep to Pre-PowerDown**

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

908

909 **7.4.1.10.7 Transitions from Pre-DeepSleep Power Mode**

910 **Transition from Pre-DeepSleep to UFS-DeepSleep**

911 This transition shall occur if the device link has entered HIBERN8 state.

912 **7.4.1.10.8 Transitions from UFS-DeepSleep Power Mode**

913 **Transition from UFS-DeepSleep to Power On**

914 This transition shall occur if hardware reset or power cycle occurs.

915 **7.4.1.10.9 Transitions from Pre-PowerDown Power Mode**

916 **Transition from Pre-PowerDown to Pre-Active**

917 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.

920 **Transition from Pre-PowerDown to UFS-PowerDown**

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

923 **7.4.1.10.10 Transitions from UFS-PowerDown Power Mode**

924 **Transition from UFS-PowerDown to Pre-Active**

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

927 **7.4.1.10.11 SCSI command and UPIU transactions**

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

929 bCurrentPowerMode is the only attribute the device is required to return in any power mode, with exception that device is not required to return any attribute in UFS-DeepSleep and Pre-DeepSleep 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.

933 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.

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

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

940 A START STOP UNIT command with the IMMED bit set to zero causing a transition to UFS-DeepSleep power mode shall not complete with GOOD status until the device is ready to transition from Pre-DeepSleep power mode to UFS-DeepSleep power mode.

943 IMMED bit shall be set to zero for the START STOP UNIT command requesting transition to UFS-DeepSleep power mode, otherwise the command shall be terminated with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST.

946 Table 7.4 summarizes which SCSI commands and UPIU transactions are allowed for each power mode.

947

**Table 7.4 — Allowed SCSI commands and UPIU for each Power Mode**

<b>Power Mode</b>	<b>SCSI Commands</b>		<b>UPIU Transactions</b>
	<b>Device well known logical unit</b>	<b>Other logical units</b>	
Active	Any commands	Any commands	Any UPIU
Idle	Any commands	Any commands	Any UPIU
Pre-Active	START STOP UNIT, REQUEST SENSE	No command	COMMAND UPIU, RESPONSE UPIU, REJECT UPIU, DATA IN UPIU, QUERY REQUEST UPIU, QUERY RESPONSE UPIU
UFS-Sleep	START STOP UNIT, REQUEST SENSE	No command	COMMAND UPIU, RESPONSE UPIU, REJECT UPIU, DATA IN UPIU, QUERY REQUEST UPIU, QUERY RESPONSE UPIU
Pre-Sleep	START STOP UNIT, REQUEST SENSE	No command Task Managem. Fun.	Any UPIU
UFS-DeepSleep	None	None	None
Pre-DeepSleep	None	None	None
UFS-PowerDown	START STOP UNIT, REQUEST SENSE	No command	COMMAND UPIU, RESPONSE UPIU, REJECT UPIU, DATA IN UPIU, QUERY REQUEST UPIU, QUERY RESPONSE UPIU
Pre-PowerDown	START STOP UNIT, REQUEST SENSE	No command Task Managem. Fun.	Any UPIU

948  
949

950 **7.4.1.11 Responses to SCSI commands**

951 Table 7.5 defines the Device well known logical unit response to a START STOP UNIT command for a  
952 given power mode. It is assumed that the IMMED bit in START STOP UNIT commands is set to zero.

953 **Table 7.5 — Device Well Known Logical Unit Responses to SSU command**

<b>Current Power Mode</b>	<b>PC</b>	<b>STATUS</b>	<b>SENSE KEY</b>	<b>ASC, ASCQ</b>
Pre-Active	1h	GOOD <sup>(1)</sup>	-	-
	Others	CHECK CONDITION	NOT READY	LOGICAL UNIT NOT READY, START STOP UNIT COMMAND IN PROGRESS
Active	1h, 2h, 3h, 4h	GOOD <sup>(1)</sup>	-	-
	Others	CHECK CONDITION	ILLEGAL REQUEST	INVALID FIELD IN CDB
Pre-Sleep	2h	GOOD <sup>(1)</sup>	-	-
	Others	CHECK CONDITION	NOT READY	LOGICAL UNIT NOT READY, START STOP UNIT COMMAND IN PROGRESS
UFS-Sleep	1h, 2h, 3h, 4h	GOOD <sup>(1)</sup>	-	-
	Others	CHECK CONDITION	ILLEGAL REQUEST	INVALID FIELD IN CDB
Pre-DeepSleep	Device is not able to accept START STOP UNIT command in this power mode			
UFS-DeepSleep	Device is not able to accept START STOP UNIT command in this power mode			
Pre-PowerDown	3h	GOOD <sup>(1)</sup>	-	-
	Others	CHECK CONDITION	NOT READY	LOGICAL UNIT NOT READY, START STOP UNIT COMMAND IN PROGRESS
UFS-PowerDown	1h, 3h	GOOD <sup>(1)</sup>	-	-
	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.				

954 Table 7.6 summarizes the response that the Device well known logical unit may provide to a command  
955 other than START STOP UNIT for various device power modes.

956

957 **7.4.1.11 Responses to SCSI commands (cont'd)**

958 **Table 7.6 — Device Well Known Logical Unit Responses to commands other than SSU**

<b>Power Mode</b>	<b>Command</b>	<b>STATUS</b>	<b>SENSE KEY</b>	<b>ASC, ASCQ</b>
Pre-Active	REQUEST SENSE	GOOD <sup>(1)</sup>	-	-
	Others <sup>(1)</sup>	CHECK CONDITION	NOT READY	LOGICAL UNIT IS IN PROCESS OF BECOMING READY
Pre-Sleep, Pre-PowerDown.	REQUEST SENSE	GOOD <sup>(1)</sup>	-	-
	Others <sup>(1)</sup>	CHECK CONDITION	ILLEGAL REQUEST	-
UFS-Sleep, UFS-PowerDown	REQUEST SENSE	GOOD <sup>(1)</sup>	-	-
	Others <sup>(1)</sup>	CHECK CONDITION	NOT READY	LOGICAL UNIT NOT READY, INITIALIZING COMMAND REQUIRED
Pre-DeepSleep, UFS-DeepSleep	Device is not able to accept any command in this power mode			

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.

959 Table 7.7 defines the pollable sense data for various device power modes.

960 **Table 7.7 — Pollable Sense Data for each Power Modes**

<b>Power Mode</b>	<b>SENSE KEY</b>	<b>ASC, ASCQ</b>
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

961 **7.4.2 Power Management Command: START STOP UNIT**

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

965 When the START STOP UNIT command is sent to the UFS Device well-known logical unit (W-LUN =  
966 50h), it can be used to select the device power mode.

967 **Table 7.8 — START STOP UNIT command**

<b>Bit Byte</b>	<b>7</b>	<b>6</b>	<b>5</b>	<b>4</b>	<b>3</b>	<b>2</b>	<b>1</b>	<b>0</b>
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)							

968 **7.4.2 Power Management Command: START STOP UNIT (cont'd)**

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

971 **Table 7.9 — START STOP UNIT fields**

IMMED	No Flush	Power Condition	Start	LUN or WLUN	Action
				LUN Field in UPIU	
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 <sup>(1)</sup> 00h to N-1 <sup>(1)</sup>	Stop the designated LU.
-	-	0h	1	00h to N-1 <sup>(1)</sup> 00h to N-1 <sup>(1)</sup>	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
-	-	4h	0	50h D0h	Cause a transition to the Pre-DeepSleep power mode

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

973 **7.4.3 Power Mode Control**

974 Table 7.10 defines a series of attributes used to control the active current levels and power modes.

975 **Table 7.10 — Attribute for Power Mode Control**

Attribute Name	Size	Type	Description
bCurrentPowerMode	1 byte	Read only	<p>Current device Power Mode Status</p> <p>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</p>
bActiveICCLevel	1 byte	Read / Volatile	<p>bActiveICCLevel defines the maximum current consumption allowed during Active mode.</p> <p>00h: Lowest Active ICC level ... 0Fh: Highest Active ICC level Others: Reserved</p> <p>Valid range from 00h to 0Fh.</p>

976 Table 7.11 shows the Device Descriptor parameters that specify the power mode and the Active ICC level  
977 after power on or reset. See 14.1.4.3, Configuration Descriptor, for details about device configuration.

978 **Table 7.11 — Device Descriptor parameters**

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

979

#### 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.

Maximum peak current is defined as absolute maximum value not to be exceeded at all. The conditions under which wActiveICCLevelsVCC, wActiveICCLevelsVCCQ, and wActiveICCLevelsVCCQ2 are defined are:

- Maximum supported HS-GEAR mode with Rate B-series
- Maximum supported number of lanes enabled
- Worst case functional operation
- Worst case environmental parameters (temperature, ... )

Each parameter of wActiveICCLevelsVCC, wActiveICCLevelsVCCQ, or wActiveICCLevelsVCCQ2 may have its own operating condition to reach to its maximum value.

**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.

993

994

**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

995

996 **7.4.4 Logical Unit Power Condition**

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

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

1000 Transition from active power condition to stopped power condition shall occur if the device server  
1001 processes a START STOP UNIT command with the START bit set to zero and the POWER  
1002 CONDITION field set to 0h.

1003 Transition from stopped power condition to active power condition shall occur if the device server  
1004 processes a START STOP UNIT command with the START bit set to one and the POWER CONDITION  
1005 field set to 0h.

1006 START STOP UNIT command to change the logical unit power condition should be issued only if the  
1007 device is in Active power mode or Idle power mode.

1008 A request to move to the stopped power condition should be made only when the logical unit command  
1009 queue is empty.

1010 A transition in the device power mode state shall not change the logical unit power condition.

1011 Table 7.14 defines the logical unit responses to SCSI commands for various device power modes,  
1012 assuming that the logical unit is in active power condition. See 0 for details about Device well known  
1013 logical unit.

1014 **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	-
Pre-DeepSleep, UFS-DeepSleep	Device is not able to accept any command in this power mode			

1015 If the logical unit is in the stopped power condition, then the device server shall

- 1016 • provide pollable sense data with sense key set to NOT READY and the additional sense code set to  
1017 LOGICAL UNIT NOT READY, INITIALIZING COMMAND REQUIRED;
- 1018 • terminate each media access command or TEST UNIT READY command with CHECK  
1019 CONDITION status with the sense key set to NOT READY and the additional sense code set to  
1020 LOGICAL UNIT NOT READY, INITIALIZING COMMAND REQUIRED.

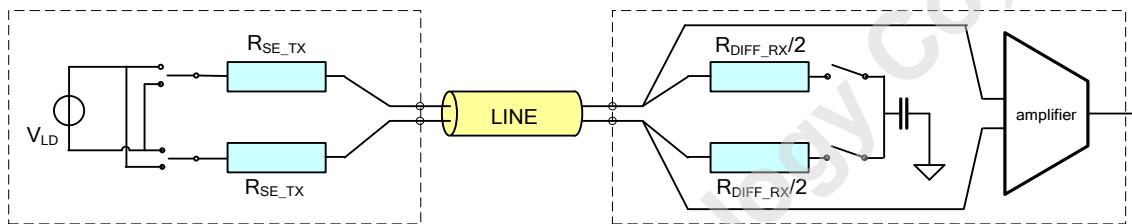
---

1021 **8 UFS UIC Layer: MIPI M-PHY**

1022 **8.1 Termination**

1023 The M-TX shall be terminated as defined in section “Termination Scheme” of the M-PHY specification  
1024 [MIPI-M-PHY].

1025 The M-RX shall include switchable differential termination. By default the M-RX termination shall be off  
1026 in PWM-BURST state and may be turned on setting the proper MIPI Attribute. The termination shall be  
1027 on by default in HS-BURST state as un-terminated HS-BURST is not supported. There shall be no  
1028 termination in SLEEP and STALL states. During DISABLE and HIBERNATE states M-TX drives High-  
1029 Z while M-RX terminates the lane by a 'Dif-Z keeper'. Dif-Z keeper means M-RX drives a weak  
1030 differential zero on the lane.



1031  
1032 **Figure 8.1 — Simplified example for I/O termination**

1033 M-RX of a SUBLINK in a LINK may have different termination settings.

1034 The supported termination settings are defined in the Capability Attributes in Table 8.1 and Table 8.2.  
1035 The termination is controlled via the Configuration Attributes. The receiver termination resistance is  
1036 defined in the M-PHY specification.

1037 Timings of the termination enable and disable are defined in the M-PHY specification.

1038 **8.2 Drive Levels**

1039 The M-PHY specification defines two drive amplitudes: the large amplitude (LA) and the small amplitude  
1040 (SA). The UFS interface utilizes the large amplitude (LA).

1041 Every M-TX in every LINK will start communication with LA after power up or reset.

1042 SUBLINKS in a LINK shall communicate with same amplitude.

1043 **8.3 PHY State machine**

1044 The UFS interface shall implement the Type I state machine.

1045 The M-PHY specification defines two different signaling schemes for low-speed mode (LS-MODE):  
1046 Non-Return-to-Zero (NRZ) and Pulse-Width-Modulation (PWM). The UFS interface shall utilize the  
1047 PWM signaling scheme in the LS-MODE as defined by the M-PHY specification for the State Machine  
1048 Type I [MIPI M-PHY].

1049 Support for LCC functionality is optional.

1050

1051 **8.4 HS Burst**

1052 A UFS device shall support the HS-GEAR1, HS-GEAR2, HS-GEAR3 and the HS-GEAR4. The  
1053 supported gears are indicated in the Capability Attribute Table 8.1 and Table 8.2.

1054 SUBLINKS in a LINK may communicate with different HS-GEAR or PWM-GEAR.

1055 **8.4.1 HS Prepare Length Control**

1056 The TX\_HS\_PREPARE\_LENGTH M-PHY configuration attribute defines the time to move from  
1057 STALL to HS-BURST. At reset, M-TX sets TX\_HS\_PREPARE\_LENGTH = 15.

1058 **8.4.2 HS Sync Length Control**

1059 The TX\_HS\_SYNC\_LENGTH M-PHY configuration attribute defines the number of synchronization  
1060 symbols before a HS Burst. In the UFS interface the synchronization sequence shall be generated by the  
1061 M-TX. Support for protocol controlled synchronization is optional. M-TX starts at reset with  
1062 TX\_HS\_SYNC\_LENGTH = 15, in COARSE type.

1063 **8.5 PWM Burst**

1064 A UFS device shall support the PWM-G1. Other PWM gears are optional. The supported PWM-GEARS  
1065 are indicated in the Capability Attributes Table 8.1 and Table 8.2.

1066 NOTE Even if the physical layer supports PWM-G0, this gear can not be used because it is not supported by  
1067 [MIPI-UniPro].

1068 The PWM-G1 shall be the active one by default after power up or reset.

1069 SUBLINKS in a LINK may communicate with different PWM-GEAR or HS-GEAR.

1070 **8.5.1 LS Prepare Length Control**

1071 The TX\_LS\_PREPARE\_LENGTH M-PHY configuration attribute defines the time to move from SLEEP  
1072 to PWM-BURST. At reset, M-TX sets TX\_LS\_PREPARE\_LENGTH = 10.

1073 **8.6 Adapt**

1074 The MIPI M-PHY version 4.1 supports a new Adapt sequence that is used to train the filter characteristics  
1075 of its equalizers in the M-RX module according to the channel. An UFS device and UFS host shall  
1076 implement the equalizer and the M-TX shall be able to provide the Adapt sequence to its M-RX  
1077 counterpart if needed.

1078 An UFS device shall be able to initiate an Adapt sequence, but it should start it only when it is requested  
1079 by the Host.

1080 The device's Transport Layer has the capability to initiate Adapt if there is a re-initialization request via  
1081 PA-INIT.

1082 **8.7 UFS PHY Attributes**

1083 The MIPI M-PHY includes several configurable attributes. There is range of values defined for the  
1084 attributes but it is left for the application to fix the actual required values inside the range. Following is  
1085 the list of such attributes. The UFS application specific requirement for the values can be found from  
1086 Table 8.1 and Table 8.2.

1087

1088 **8.7 UFS PHY Attributes (cont'd)**

1089

**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, HS_G1_TO_G3 = 4	4	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	≥ 1	3
TX_Amplitude_Capability	0x05	SA=1, LA=2, Both=3	≥ 2	4
TX_ExternalSYNC_Capability	0x06	0=False 1=True	0, 1	5
TX_HS_Underminated_LINE_Drive_Capability	0x07	0=No, 1=Yes	0	
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. Support for PWM gears greater than PWM-GEAR1 is optional in UFS.  
 NOTE 4 UFS device shall support large amplitude. Support for small amplitude is optional.  
 NOTE 5 Support for external SYNC pattern is optional and its definition is device specific.  
 NOTE 6 A UFS device shall support 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

1090 **8.7 UFS PHY Attributes (cont'd)**

1091

**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, HS_G1_TO_G3 = 4	4	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	$\geq 1$	3
RX_HS_Underminated_Capability	0x85	0=No, 1=Yes	0	
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 255	
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
RX_PHY_Editorial_Release_Capability		B[3:0] : Minor version number	0 to 9	7
RX_Hibern8Time_Capability	0x91	1 to 99	1 to 99	7
RX_PWM_G6_G7_SYNC_LENGTH_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] $\leq 15$	
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	$\leq 15$	
RX_HS_G3_PREPARE_LENGTH_Capability	0x97	B[3:0] = 0 to 15	$\leq 15$	
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	

Attribute Name	Attribute ID	Range	UFS Value	Notes
RX_Advanced_Min_ActivateTime_Capability	0x9A	B[3:0] = 1 to 14	B[3:0] = 1 to 14	
RX_HS_G4_SYNC_LENGTH_Capability	0x9B	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_G4_PREPARE_LENGTH_Capability	0x9C	B[3:0] = 0 to 15	$\leq 15$	
RX_HS_Equalizer_Setting_Capability	0x9D	B[0]=0: No equalization B[0]=1: Yes equalization	1	
RX_HS_ADAPT_LENGTH_FINE_Capability	0x9E	B[7] : ADAPT_type FINE=0, COARSE=1 B[6:0] : ADAPT_length 0 to 127 for FINE, 0 to 17 for COARSE	B[7] : 0,1 B[6:0] : 0 to 127 for FINE, 0 to 17 for COARSE	
RX_HS_ADAPT_LENGTH_COARSE_Capability	0x9F	B[7] : ADAPT_type FINE=0, COARSE=1 B[6:0] : ADAPT_length 0 to 127 for FINE, 0 to 17 for COARSE	0 to 17	
<p>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. Support for PWM gears greater than PWM-GEAR1 is optional in UFS.      NOTE 4 A UFS device shall support 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.</p>				

1092

## 1093 8.8 Electrical characteristics

### 1094 8.8.1 Transmitter Characteristics

1095 As defined in the M-PHY specification.

### 1096 8.8.2 Receiver Characteristics

1097 As defined in the M-PHY specification.

1098

---

1099 **9 UFS UIC Layer: MIPI UNIPRO**

---

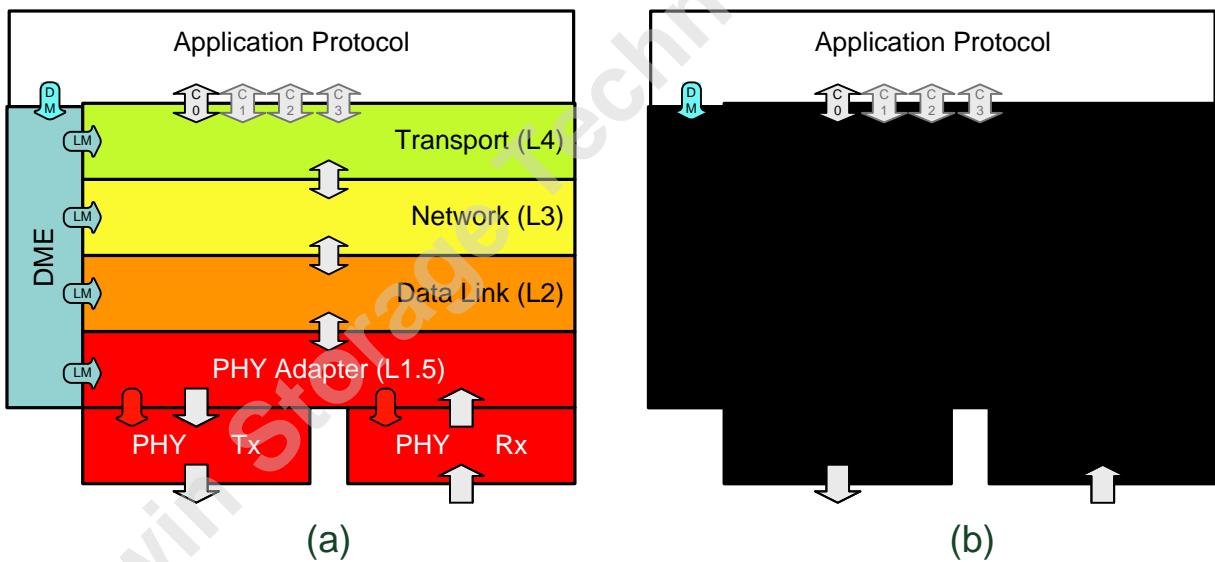
1100 **9.1 Overview**

1101 UFS builds on the MIPI Unified Protocol (UniPro) as its Interconnect (Service Delivery Subsystem) to  
1102 provide basic transfer capabilities to the UFS Transport Protocol (UTP) Layer. On the data plane UTP  
1103 and UniPro communicate via the Service Primitives of the UniPro Transport Layer CPorts  
1104 ( $T_{CO\_SAPs}$ ). Control plane interaction (e.g., discovery, enumeration and configuration of the Link)  
1105 between higher layer protocol functions of UFS and UniPro are accomplished using the Device  
1106 Management Entity Service Primitives as defined by the UniPro specification.

1107 **9.2 Architectural Model**

1108 UniPro is internally composed of several sub-layers which are all well defined by the MIPI UniPro  
1109 specification [MIPI-Unipro]. In the context of UFS the entire UniPro protocol stack shall be viewed as a  
1110 black box model (see Figure 9.1) to the greatest extent possible. The following sections therefore only:

- 1111 • Specify number and type of the required interfaces between UFS and UniPro  
1112 • Specify the mapping between UFS and UniPro addressing scheme  
1113 • Select optional features and definable attributes of the UniPro specification



1114  
1115 **Figure 9.1 — UniPro internal layering view (a) and UniPro Black Box view (b)**

1116

1117 **9.3 UniPro/UFS Transport Protocol Interface (Data Plane)**

1118 UniPro provides CPorts as conceptual interfaces to applications or protocol layers on top of UniPro.  
1119 CPorts can be viewed as instantiations of the T\_CO\_SAPs as specified in 8.8 of the UniPro specification.  
1120 The physical implementation of a T\_CO\_SAP was deliberately not defined in MIPI as implementers  
1121 should be free to choose, e.g., a SW implementations of higher UniPro layers, HW implementations based  
1122 on buffering per CPort or DMA channels per CPort, etc.

1123 A Service Access Point (SAP) provides Service Primitives (SP) which can be used by specifications of  
1124 applications or protocols as UFS on top of UniPro to define their interactions. For more information on  
1125 the concept of SAP/SP in protocol specifications please refer to Annex C of the UniPro specification.

1126 The T\_CO\_SAP provides the following core data transfer service primitives (see UniPro specification,  
1127 8.8.1):

- 1128 • T\_CO\_DATA.req( MessageFragment, EOM)

- 1129     ○ Issued by service user of UniPro to send a message (Fragment)

1130       NOTE Whenever a UFS layer requests the UIC layer to transfer data that UFS layer shall ensure that the  
1131       last fragment of said data will be transmitted with the EOM flag set. One way to ensure such a behavior is  
1132       for the UFS layer to invoke this UIC data transfer service primitive only once per atomic protocol data unit  
1133       (e.g., once per UFS Transport Layer ‘UPIU’) with the EOM flag set to ‘true’ always.

- 1134 • T\_CO\_DATA.cnf\_L( L4CPortresultCode )

- 1135     ○ Issued by UniPro to report the result of a Message (Fragment) transfer request

- 1136 • T\_CO\_DATA.ind( MessageFragment, EOM, SOM, MsgStatus )

- 1137     ○ Issued by UniPro to deliver a received Message (Fragment) towards the service user

- 1138     ○ EOM informs the service user that this is the last Message Fragment (EndOfMessage)

- 1139     ○ SOM informs the service user that this is the first Message Fragment (StartOfMessage)

- 1140 • T\_CO\_DATA.rsp\_L()

- 1141     ○ Issued by a service user of UniPro to report readiness to receive the next Message (Fragment)

1142 **9.3.1 Flow control**

1143 UFS will not make use of the End-to-End Flow Control feature of UniPro for data communication as the  
1144 UFS Transport Layer already avoids any overflow by a strict client-server communication model, tagged  
1145 command queues and Device side throttling of Data transfers. Therefore UFS will not use the  
1146 T\_CO\_FLOWCONTROL service primitive of UniPro and hence does not require its implementation.

1147 **9.3.2 Object sizes**

1148 A UniPro Message can be of any size and its content is not interpreted in any way by UniPro. Messages  
1149 can be delivered from/to UniPro as multiple Message Fragments.

1150 A Message Fragment is a portion of a Message that can be passed to, or received by, a CPort. Received  
1151 Fragments are not generally identical to transmitted Fragments. Message Fragments may or may not carry  
1152 an End-of-Message (EoM) flag.

1153 A Message Fragment shall have maximum of T\_MTU bytes to avoid further splitting in lower layers.

1155 **9.4 UniPro/UFS Control Interface (Control Plane)**

1156 UniPro provides access to its Device Management Entity (DME) via a Service Access Point (DME SAP)  
1157 with the following services exposed to UFS allowing control of properties and the behavior of UniPro:

1158 *DME Configuration Primitives*

- 1159 • DME\_GET / DME\_SET
  - 1160 ○ Provide read/write access to all UniPro and M-PHY attributes of the local UniPort
- 1161 • DME\_PEER\_GET (optional) / DME\_PEER\_SET (optional)
  - 1162 ○ Provide read/write access to all UniPro and M-PHY attributes of the peer UniPort

1163 NOTE The order in which attributes are set is in some cases relevant for UniPro's correct operation. Therefore  
1164 higher UFS layers shall preserve the ordering of DME Configuration Primitives invocations by UFS applications. If  
1165 internally generated by UFS itself, DME Configuration Primitives shall be issued correctly ordered as defined by the  
1166 UniPro specification.

1167 *DME Control Primitives*

- 1168 • DME\_POWERON (optional) / DME\_POWEROFF (optional)
    - 1169 ○ Allow to power up or power down all UniPro layers (L1.5 through L4)
  - 1170 • DME\_ENABLE
    - 1171 ○ Allow enabling of the entire local UniPro stack (UniPro L1.5 -L4)
  - 1172 • DME\_RESET
    - 1173 ○ Allows to reset the entire local UniPro stack (UniPro L1.5-L4)
  - 1174 • DME\_ENDPOINTRESET
    - 1175 ○ Allows sending an end-point reset request command to a link end point.
  - 1176 • DME\_LINKSTARTUP
    - 1177 ○ Allows locally to startup the Link and informs about remote link startup invocation
  - 1178 • DME\_HIBERNATE\_ENTER / DME\_HIBERNATE\_EXIT
    - 1179 ○ Allow to put the entire Link into HIBERNATE power mode and to wake the Link up
      - 1180 ▪ Affects the local and the peer UniPort (UniPro L1.5-L4 and M-PHY)
- 1181 NOTE After exit from Hibernate all UniPro Transport Layer attributes (including L4 T\_PeerDeviceID,  
1182 L4 T\_PeerCPortID, L4 T\_ConnectionState, etc.) will be reset to their reset values. All required attributes  
1183 must be restored properly on both ends before communication can resume.
- 1184 • DME\_POWERMODE
    - 1185 ○ Allows to change the power mode of one or both directions of the M-PHY Link
  - 1186 • DME\_TEST\_MODE (optional)
    - 1187 ○ Allows to set the peer UniPro Device on the Link in a specific test mode
  - 1188 • DME\_LINKLOST
    - 1189 ○ Indication of the UniPro stack towards higher layers that the Link has been lost
  - 1190 • DME\_ERROR
    - 1191 ○ Indication of the UniPro stack towards higher layers that an error condition has been  
1192 encountered in one of the UniPro Layers

1194 **9.5 UniPro/UFS Transport Protocol Address Mapping**

1195 UniPro has fundamentally two levels of addressing to control the exchange of information between  
1196 remote UniPro entities

1197 Network Layer (L3): Device ID, lowest level of addressability

- 1198   ○ Provided for future UniPro networks of devices. During connection establishment the side creating a  
1199   connection uses this value to select the physical entity on the remote end of the connection. It shall be  
1200   considered static for the lifetime of this connection.

1201 Transport Layer (L4): CPort ID, highest level of end-to-end addressability

- 1202   ○ During connection establishment the side creating a connection uses this value to select the logical  
1203   entity inside the targeted UniPro device on the remote end of the connection. It shall be considered  
1204   static for the lifetime of this connection.

1205 UFS adopts the addressing notation of the SCSI Architecture Model [SAM] based on Nexus definition.

1206 The Nexus (I\_T\_L\_Q) is composed of:

- 1207   • Initiator Port Identifier (I)  
1208   • Target Port Identifier (T)  
1209   • Logical Unit Number (L)  
1210   • Command Identifier (Q).

1211 An I\_T\_L\_Q Nexus uniquely defines a specific command slot (Q) inside a specific Logical Unit (L)  
1212 connected to a specific Device Target Port (T) accessed through a specific Host Initiator Port (I).

1213 UFS Interconnect Layer addresses (Device ID and CPort ID) are only related to the I\_T part of the Nexus.

1214 This standard only requires and uses a single UniPro CPort on the device side and on the host side.

1215 Mapping Rules

- 1216   • UFS Initiator Port Identifier (I) and UFS Target Port Identifier (T) shall be 16 bit wide each and  
1217     ○ UFS Initiator/Target Port Identifier shall contain the UniPro Network Layer Device ID of the  
1218       entity (host or device) containing said UFS Port  
1219       ■ The UniPro Network Layer Device ID reset value shall be 0 for the Host  
1220       ■ The UniPro Network Layer Device ID reset value shall be 1 for the Device  
1221     ○ UFS Initiator/Target Port Identifier contains the UniPro Transport Layer CPort ID which said  
1222       UFS Port uses to communicate to the remote entity  
1223       ■ The UniPro Transport Layer CPort ID reset value shall be 0 for the Host  
1224       ■ The UniPro Transport Layer CPort ID reset value shall be 0 for the Device  
1225   • UFS Initiator Port Identifier shall contain the Initiator ID (IID).

1226 Table 9.1 defines the Initiator Port Identifier (I) and Target Port Identifier (T) for UFS.

1228 **9.5 UniPro/UFS Transport Protocol Address Mapping (cont'd)**

1229 **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			

1230 The single UniPro connection between the UTP layer of a UFS host and the UTP layer of a UFS device  
1231 can be uniquely identified by the UFS I\_T Nexus above.

1232 NOTE The UFS I\_T Nexus elements (Device IDs and CPort IDs) can be modified by the Host after reset using the  
1233 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

1236 All attributes of the CPort on the Host side (including, e.g., "T\_ConnectionState") can be checked and  
1237 modified by the Host using the DME\_GET and DME\_SET primitives after reset.

1238 All attributes of the CPort on the Device side (including, e.g., the "T\_ConnectionState") can be checked  
1239 and modified by the Host using the DME\_PEER\_GET and DME\_PEER\_SET primitives after reset.

1240 **9.6 Options and Tunable Parameters of UniPro**

1241 MIPI UniPro has been designed as a versatile protocol specification and as such has several options and  
1242 parameters which an application like UFS should specify for its specialized UniPro usage scenario.  
1243 Annex E of the UniPro specification details all of the possible choices.

1244 The remaining sections of this chapter define the specific requirements towards these options and  
1245 parameters for this version of UFS standard. They apply to UniPro implementations for the UFS host side  
1246 as well as to UniPro implementations for the UFS device side if not explicitly stated otherwise.

1247 **9.6.1 UniPro PHY Adapter**

1248 For MIPI M-PHY related attribute values and implementation options as defined by UFS refer to 8.7,  
1249 UFS PHY Attributes.

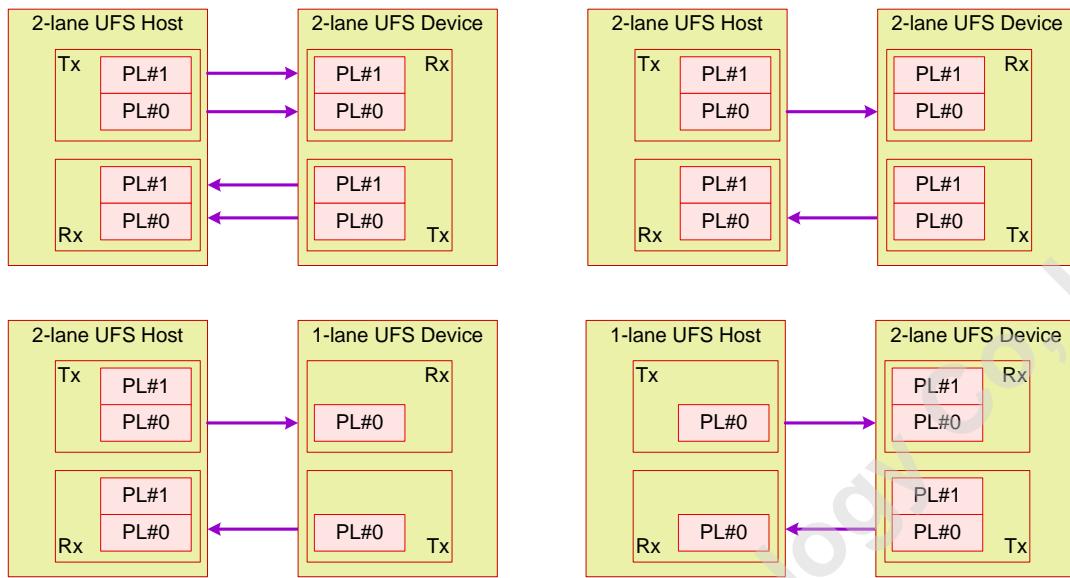
1250 In UFS system, host and device use the same reference clock, therefore skip symbol insertion  
1251 functionality is not used and its implementation is optional.

1252 UFS device shall support the following physical lane connections.

- One lane
  - Tx physical lane 0 connected to Rx physical lane 0
- Two lanes
  - Tx physical lane 0 connected to Rx physical lane 0
  - Tx physical lane 1 connected to Rx physical lane 1

1259 **9.6.1 UniPro PHY Adapter (cont'd)**

1260



1261 **Figure 9.2 — Physical lane connections**

1262 **9.6.2 UniPro Data Link Layer**

- 1263
- 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)

1264 **9.6.3 UniPro Network Layer**

- 1265
- Shall support transmission and reception of maximum sized L3 packets (N\_MTU)

1266

1272 **9.6.4 UniPro Transport Layer**

- 1273 • UFS Hosts and UFS Devices shall implement at least 1 CPort  
1274     NOTE This standard only requires and uses a single CPort on either side of the Link.  
1275 • UFS does not mandate any CPort arbitration scheme beyond the UniPro default if more than one  
1276     CPort is implemented  
1277 • Shall support the UniPro Test Feature  
1278 • UFS does not require the UniPro End-to-End Flow Control mechanism  
1279     ○ UFS will not use 'Controlled Segment Dropping' (CSD)  
1280         ■ Hence CSD shall be disabled  
1281 • UFS will not use "CPort Safety Valve" (CSV). Hence, CSV shall be disabled.  
1282 • Shall support transmission and reception of maximum sized L4 segments ( $T_{MTU}$ ).

1283 **9.6.5 UniPro Device Management Entity Transport Layer**

- 1284 DME service primitives provide the means to  
1285 • retrieve or set attributes,  
1286 • control the reset and run mode of the entire UniPro protocol stack.

1287 UFS Hosts and UFS Devices shall implement the following DME service primitives:

- 1288 • DME\_GET, DME\_SET,  
1289 • DME\_ENABLE,  
1290 • DME\_RESET, DME\_ENDPOINTRESET,  
1291 • DME\_LINKSTARTUP, DME\_LINKLOST,  
1292 • DME\_HIBERNATE\_ENTER, DME\_HIBERNATE\_EXIT,  
1293 • DME\_POWERMODE,  
1294 • DME\_ERROR.

1295 **UFS Hosts**

- 1296 • shall implement DME\_PEER\_GET primitive and DME\_PEER\_SET primitive, which are optional in  
1297 [MIPI-UniPro].

1298 **UFS Devices**

- 1299 • shall not use DME\_SET primitive to modify the local PA\_PWRMode attribute,  
1300 • shall use DME\_RESET only in the following cases: at power-on or hardware reset, or after a  
1301     DME\_LINKLOST.ind,  
1302 • shall not use the following primitives  
1303     ○ DME\_PEER\_GET.req, DME\_PEER\_SET.req,  
1304     ○ DME\_POWERON.req, DME\_POWEROFF.req,  
1305     ○ DME\_ENDPOINTRESET.req,  
1306     ○ DME\_HIBERNATE\_ENTER.req, DME\_HIBERNATE\_EXIT.req,  
1307     ○ DME\_POWERMODE.req, DME\_TEST\_MODE.req.

1308 **9.6.6 UniPro Attributes**

1309 To optimize the UFS Boot procedure the UFS UIC implementation shall use the default reset values for  
1310 all UniPro Attributes as defined by the MIPI UniPro specification. As an exception to this, the reset  
1311 values of Network Layer Attributes and specific Attributes for *CPort 0* shall reflect the settings which  
1312 have been defined in the sections above and therefore shall contain the values as depicted in Table 9.2.

1313  
1314

**Table 9.2 — 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		

1315

---

1316    **10    UFS Transport Protocol (UTP) Layer**

---

1317    **10.1    Overview**

1318    The SCSI Architecture Model [SAM] is used as the general architectural model for UTP, and the SAM  
1319    Task Management functions for task management. A task is generally a SCSI command or service  
1320    request. While the model uses the SCSI command set as the command set, it is not necessary to use SCSI  
1321    commands exclusively.

1322    The SAM architecture is a client-server model or more commonly a request-response architecture. Clients  
1323    are called Initiator devices and servers are called Target devices. Initiator devices and Target devices are  
1324    mapped into UFS physical network devices. An Initiator device issues commands or service requests to a  
1325    Target device that will perform the service requested. A Target device is a UFS device. A UFS device  
1326    will contain one or more Logical Units. A Logical Unit is an independent processing entity within the  
1327    device.

1328    A client request is directed to a single Logical Unit within a device. A Logical Unit will receive and  
1329    process the client command or request. Each Logical Unit has an address within the Target device called a  
1330    Logical Unit Number (LUN).

1331    Communication between the Initiator device and Target device is divided into a series of messages. These  
1332    messages are formatted into UFS Protocol Information Units (UPIU) as defined within this standard.  
1333    There are a number of different UPIU types defined. All UPIU structures contain a common header area  
1334    at the beginning of the data structure (lowest address). The remaining fields of the structure vary  
1335    according to the type of UPIU.

1336    A Task is a command or sequence of actions that perform a requested service. A Logical Unit contains a  
1337    task queue that will support the processing of one or more Tasks. The Task queue is managed by the  
1338    Logical Unit. A unique Task Tag is generated by the Initiator device when building the Task. This Task  
1339    Tag is used by the Target device and the Initiator device to distinguish between multiple Tasks. All  
1340    transactions and sequences associated with a particular Task will contain that Task Tag in the transaction  
1341    associated data structures.

1342    Command structures consist of Command Descriptor Blocks (CDB) that contain a command opcode and  
1343    related parameters, flags and attributes. The description of the CDB content and structure are defined in  
1344    detail in [SAM], [SBC] and [SPC] INCITS T10 draft standards.

1345    A command transaction consists of a Command, an optional Data Phase, and a Status Phase. These  
1346    transactions are represented in the form of UPIU structures. The Command Phase delivers the command  
1347    information and supporting parameters from the Initiator device to the Target device. If a Data Phase is  
1348    required, the direction of data flow is relative to the Initiator device. A data WRITE travels from Initiator  
1349    device to Target device. A data READ travels from Target device to Initiator device. At the completion of  
1350    the command, the Target device will deliver a response to the Initiator device during the Status Phase.  
1351    The response will contain the status and a UFS response status indicating successful completion or failure  
1352    of the command. If an error is indicated the response will contain additional detailed UFS error  
1353    information.

1354

1355 **10.2 UTP and UniPro Specific Overview**

1356 UTP will deliver commands, data and responses as standard message packets (T\_SDUs) over the UniPro  
1357 network.

1358 The UFS transactions will be grouped into data structures called UFS Protocol Information Units (UPIU).

1359 There are UPIU's defined for UFS SCSI commands, responses, data in and data out, task management,  
1360 utility functions, vendor functions, transaction synchronization and control. The list is extensible for  
1361 future additions.

1362 For enumeration and configuration, UFS supports a system of Descriptors, Attributes and Flags that  
1363 define and control the specifics of the device, including operating characteristics, interfaces, number of  
1364 logical units, operating speeds, power profiles, etc. The system is a hierarchical tree of related elements. It  
1365 is open to be expanded.

1366 **10.2.1 Phases**

1367 The SCSI based Command protocol requires that the UPIU packets follow the transitions required to  
1368 execute a command. Briefly, a command execution requires the sending of a COMMAND UPIU, zero or  
1369 more DATA IN UPIU or DATA OUT UPIU packets and terminates with a RESPONSE UPIU that  
1370 contains the status.

1371 **10.2.2 Data Pacing**

1372 A device may have limited memory resources for buffering or limited processing throughput. During a  
1373 Command that requires a large Data Out transaction the Target device can pace the Data Out phase by  
1374 sending a READY TO TRANSFER UPIU when it is ready for the next DATA OUT UPIU. In addition,  
1375 the READY TO TRANSFER UPIU contains an embedded transfer context that is used to initiate a DMA  
1376 transfer on a per packet basis at the host.

1377 During the Data In phase, no READY TO TRANSFER UPIU is required as the host has the ability to  
1378 specify the size of the Data In transfer and thereby is able to allocate in advance the appropriate memory  
1379 resources for the incoming data. A device issued DATA IN UPIU packet also contains an embedded  
1380 DMA context that can be used to initiate a DMA transfer on a per packet basis.

1381 **10.2.3 UniPro**

1382 In keeping with the requirements of the UniPro Protocol the UFS Initiator device and Target device will  
1383 divide its transactions into UniPro messages that will contain UPIU's. UniPro messages can handle  
1384 T\_SDUs of theoretically unlimited size. UFS will impose a practicable limit on the maximum  
1385 T\_SDUs message size. The limit is 65600 bytes which includes UPIU header, optional extended headers  
1386 area and data segment. The minimum message size is determined by the basic header format, which is 32  
1387 bytes. There is a possibility that in the future this value will increase to allow a larger data segment area.

1388

### 1389 **10.3 UFS Transport Protocol Transactions Overview**

1390 UFS transactions consist of packets called UFS Protocol Information Units (UPIU) that travel between  
1391 devices on the UniPro bus. A transaction begins between an Initiator device and a Target device in the  
1392 form of a Request-Response operation. The Initiator device starts the sequence of transactions by sending  
1393 a request to a Target device and logical unit. The Target device will then respond with a series of  
1394 transactions that eventually end in a response transaction.

1395 All UFS UPIU's consist of a single basic header segment, transaction specific fields, possibly one or  
1396 more extended header segments and zero or more data segments.

1397 A basic header segment has a fixed length of 12 bytes. The minimum UPIU size is 32 bytes which  
1398 includes a basic header segment and transaction specific fields.

1399 The maximum UPIU size is defined as being 65600 bytes.

1400 The UPIU format is flexible enough to be easily extended to support future transactions and larger data  
1401 segments and will allow the application of this protocol to network protocols other than UniPro.

### 1402 **10.4 Service Delivery Subsystem**

1403 The Service Delivery Subsystem is an I/O system that transmits service requests and responses between  
1404 Initiator devices and the Target device connected via a physical or logical bus. The UFS UTP attempts to  
1405 define a protocol that is independent of the Service Delivery Subsystem. This will allow for the easy  
1406 porting of UTP to different Service Delivery Subsystems.

1407 Currently, UFS is using the MIPI UniPro bus and the MIPI M-PHY® as the Service Delivery Subsystem.  
1408 For convenience and to aid in better understanding, portions of this standard directly reference UniPro  
1409 and M-PHY®. Regardless of these references, the UTP protocol is independent of the Service Delivery  
1410 Subsystem and should be able to port to other I/O systems.

1411 UPIU structures will be handed off to MIPI UniPro as UniPro Service Data Units (T\_SDUs). Currently,  
1412 the UniPro T\_SDU requires no additional headers or trailer wrapped around the UPIU structure. This  
1413 means that the T\_SDU size will be exactly the UPIU size. The minimum size T\_SDU will be 32 bytes.  
1414 The maximum T\_SDU size will be 65600 bytes.

### 1415 **10.5 UPIU Transactions**

1416 Every UPIU data structure contains a Transaction Code. This code defines the content and implied  
1417 function or use of the UPIU data structure. Table 10.1 lists currently defined transaction codes.

1418 **Table 10.1 — UPIU Transaction Codes**

<b>Initiator To Target</b>	<b>Transaction Code</b>	<b>Target to Initiator</b>	<b>Transaction Code</b>
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.

1419 **10.5 UPIU Transactions (cont'd)**

1420 **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.

1421 UFS devices are able to process only either a NOP OUT or a QUERY REQUEST at any point of time.

1422

1423 **10.6 General UFS Protocol Information Unit Format**

1424 Table 10.3 represents the general structure of a UPIU. All UPIU's will contain a fixed size and location  
1425 basic header and additional fields as required to support the transaction type.

1426 **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 Initiator ID	5 Command Set Type	6 Query Function / Task Manag. Function	7 Response
8 Total EHS Length	9 Device Information	10 (MSB)	11 (LSB) Data Segment Length
12	13	14	15
16			19
20	Transaction Specific Fields		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.			

1427

### 1428 10.6.1 Overview

1429 UPIU total size will vary depending upon the UPIU transaction type but all UPIU sizes will be an integer  
1430 multiple of 32-bits, meaning they will be addressed on a 4-byte boundary. If the aggregation of data and  
1431 header segments does not end on a 32-bit boundary then additional padding will be added to round up the  
1432 UPIU to the next 32-bit, 4-byte address boundary.

1433 The UPIU size can be fixed or variable depending upon the Transaction Type field and extension flags.  
1434 Some Transaction Types will have different lengths for the same code others will always be a fixed size.  
1435 In addition, any UPIU can be extended if necessary to include extra header and data segments. The  
1436 general format allows for extension and has flags and size fields defined within the structure to indicate to  
1437 the processing entity where the extension areas are located within the structure and their size (not  
1438 including padding) and in some cases the type of extension data.

### 1439 10.6.2 Basic Header Format

1440 This is the format of the basic header contained within every UPIU structure. This data packet will be sent  
1441 between Initiator devices and Target devices and will be part of a larger function specific UPIU. There is  
1442 enough information in this header to allow the Initiator device or the Target device to track the destination  
1443 and the source, the function request, if additional data and parameters are required and whether they are  
1444 included in this UPIU or will follow in subsequent UPIU's.

1445 The smallest sized UPIU is currently defined to have 32 bytes. The 32 bytes area will contain the basic  
1446 header plus additional fields. This means that the smallest datum sent over the Service Delivery  
1447 Subsystem will be 32 bytes.

1448 **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	

1449 The basic header formats are defined as follows:

#### 1450 a) Transaction Type

1451 The Transaction Type indicates the type of request or response contained within the data structure.  
1452 The Transaction Type contains the HD bit, the DD bit and the Transaction Code, see Table 10.5.

1453 **Table 10.5 — Transaction Type Format**

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

#### 1454 b) HD

1455 The HD bit when set to '1' specifies that an end-to-end CRC of all Header Segments is included  
1456 within the UPIU. The CRC fields include all fields within the header area. The CRC is placed at the  
1457 32-bit word location following the header.

1458 End-to-end CRC is not supported in this version of the standard, therefore HD shall be '0'.

1459

1460 **10.6.2 Basic Header Format (cont'd)**

1461 **c) DD**

1462 The DD bit when set to '1' specifies that an end-to-end CRC of the Data Segment is included with the  
1463 UPIU. The 32-bit CRC is calculated over all the fields within the Data Segment. The 32-bit CRC  
1464 word is placed at the end of the Data Segment. This will be the last word location of the UPIU.

1465 End-to-end CRC is not supported in this version of the standard, therefore DD shall be '0'.

1466 **d) Transaction Code**

1467 The Transaction Code indicates the operation that is represented within the data fields of the UPIU  
1468 and the number and location of the defined fields within the UPIU (see Table 10.1).

1469 **e) Flags**

1470 The content of the Flags field vary with the Transaction Type opcode<sup>(1)</sup>.

1471 **Table 10.6 — UPIU Flags**

UPIU Type	Operational Flags				Rsvd	CP <sup>(2)</sup>	Task Attribute	
	Bit 7	Bit 6	Bit 5	Bit 4			Bit 1	Bit 0
NOP Out	-	-	-	-	-	-	-	-
NOP In	-	-	-	-	-	-	-	-
Command	-	R	W	-	-	CP <sup>(2)</sup>	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

1472 **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

**1475 10.6.2 Basic Header Format (cont'd)**

**1476 f) Response**

1477 If a response is required from a Target device, this field indicates whether the requested function  
1478 succeeded or failed. This field is reserved in UPIU transactions from Initiator device to Target device.

**1479 Table 10.8 — UTP Response Values**

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

**1480 g) Status**

1481 This field contains the SCSI status (as defined in [SAM]) if the transaction is a RESPONSE UPIU for  
1482 a COMMAND UPIU with Command Set Type = 00h (SCSI Command). Otherwise it contains an  
1483 opcode specific status or it is reserved.

**1484 h) Reserved**

1485 All fields marked as reserved shall contain a value of zero.

**1486 i) LUN**

1487 This field contains the Logical Unit Number to which a request is targeted. Target devices will  
1488 contain at a minimum one logical unit numbered unit 0. This field is generated by the Initiator device  
1489 and maintained by the Target device and Initiator device for all UPIU transactions relating to a single  
1490 request or task.

**1491 j) Task Tag**

1492 The Task Tag is generated by the Initiator device when creating a task request. This field will be  
1493 maintained by the Initiator device and Target device for all UPIU transactions relating to a single  
1494 task. The Initiator device will contain a register or variable that represents the Task Tag value. The  
1495 Initiator device will generate unique Task Tag by incrementing the internal variable when creating a  
1496 new task request. When a task request is made up of or generates a series of UPIU transactions, all  
1497 UPIU will contain the same value in the Task Tag field.

1498 In particular, the same Task Tag value shall be maintained for the UPIU grouped in each row of  
1499 Table 10.9.

**1500 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

1502 **10.6.2 Basic Header Format (cont'd)**

1503 **k) Initiator ID (IID)**

1504 The Initiator ID field is 4 bits wide, encoded in bits [7:4] of byte 4. This field indicates the identity of  
1505 the Initiator device who created the task request.

1506 The Initiator ID shall be set to zero if there is only one Initiator device.

1507 UFS devices shall support all sixteen Initiator ID values. The Initiator ID shall be encoded in this  
1508 field by the Host when creating a request. This field is maintained by the Initiator device and Target  
1509 device for all UPIU transactions relating to the same task.

1510 All requests from the same Initiator device have the same IID value. Details about Initiator ID  
1511 assignment are available in UFS HCI standard specification.

1512 **l) Command Set Type**

1513 Command set type field is 4 bits wide, encoded in bits [3:0] of byte 4. This field indicates the  
1514 command set type the Command and RESPONSE UPIU is associated with. This field is defined for  
1515 the COMMAND UPIU and the RESPONSE UPIU. This field is reserved in all other UPIU's. This  
1516 field shall be used to indicate the type of command that is in the CDB field. The currently supported  
1517 command types are listed in Table 10.10.

1518 **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 This standard does not define any UFS specific command, therefore the value 1h is reserved for future use.

1519 **m) Query Function, Task Manag. Function**

1520 This field is used in QUERY REQUEST and QUERY RESPONSE UPIU's to define the Query  
1521 function, and in TASK MANAGEMENT REQUEST UPIU to define the task management function.

1522 **n) Device Information**

1523 This field provides device level information required by specific UFS functionality in all RESPONSE  
1524 UPIU.

1525 **o) Total Extra Header Segment Length**

1526 This field represents the size in 32-bit units (DWORDS) of all Extra Header Segments contained  
1527 within the UPIU. This field is used if additional header segments are needed. The length of each Extra  
1528 Header Segment shall be a multiple of four bytes. The value in this field is the number of total  
1529 number of bytes in all EHS divided by four.

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

1530 The maximum size of all EHS fields combined is 1024 bytes. A value of zero in this field indicates  
1531 that there are no EHS contained within the UPIU. Extra Header Segments are not used in this  
1532 standard, therefore the value of this field shall be set zero.

1533 **10.6.2 Basic Header Format (cont'd)**

1534 **p) Data Segment Length**

1535 The Data Segment Length field contains the number of valid bytes within the Data Segment of the  
1536 UPIU. When the number of bytes within the Data Segment is not a multiple of four then the last 32-  
1537 bit field will be padded with zeros to terminate on the next nearest 32-bit boundary. The number of  
1538 32-bit units (DWORDS) that make up the Data Segment is calculated as follows:

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

1539 Since the Data Segment Length field size is two bytes, the data segment can contain a maximum of  
1540 65535 valid bytes. A value of zero in this field indicates that there is no Data Segment within the  
1541 UPIU.

1542 **q) Transaction Specific Fields**

1543 Additional fields as required by certain Transaction Codes are located within this area. For UTP, this  
1544 area starts at byte address 12 within the UPIU and terminates on a 32 byte boundary at byte address  
1545 31. Since all UPIU contain a 12 byte Basic Header this leaves 20 bytes remaining for this area.

1546 **r) Extra Header Segments**

1547 The Extra Header Segments exist if the Total EHS Length field contains a non-zero value. For UTP,  
1548 this area will start at byte address 32 within the UPIU. The UPIU may contain zero or more EHS. The  
1549 length of each Extra Header Segment shall be a multiple of four bytes. This version of standard does  
1550 not use EHS.

1551 **s) Data Segment**

1552 The Data Segment field starts on the next 32-bit (DWORD) boundary after the EHS area within the  
1553 UPIU. For UTP, there are no EHS areas used meaning that the Data Segment will begin at byte  
1554 address 32 (byte address 36 if E2ECRC is enabled) within the UPIU. The Data Segment will be a  
1555 multiple of 32-bits, thereby making the UPIU packet size a multiple of 4 bytes. The Data Segment  
1556 Length field can contain a value that is not a multiple of 4 bytes but the Data Segment area will be  
1557 padded with zeros to fill to the next nearest 32-bit (DWORD) boundary. The Data Segment Length  
1558 field indicates the number of valid bytes within the Data Segment.

1559

1560 **10.7 UFS Protocol Information Units**

1561 This section provides the details of each UFS Protocol Information Unit.

1562 **10.7.1 COMMAND UPIU**

1563 The COMMAND UPIU contains the basic UPIU header plus additional information needed to specify a  
1564 command. The Initiator device will generate this UPIU and send it to a Target device to request a SCSI  
1565 command service to be performed by the Target.

1566 **Table 10.11 — COMMAND UPIU**

COMMAND UPIU			
0 xx00 0001b	1 Flags	2 LUN	3 Task Tag
4 IID	5 Command Set Type	6 Reserved	7 Reserved
8 Total EHS Length (00h)	9 Reserved	10 (MSB) Data Segment Length (0000h)	11 (LSB)
12 (MSB)	13 Expected Data Transfer Length	14	15 (LSB)
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)			

1567

1568 **10.7.1.1 Basic Header**

1569 The first 12 bytes of the COMMAND UPIU contain the Basic Header as described in 10.6.2, Basic  
1570 Header Format. Specific details are as follows:

1571 **a) Transaction Type**

1572 A type code value of xx00 0001b indicates a COMMAND UPIU.

1573 **b) Flags**

1574 Table 10.12 describes the flags used in COMMAND UPIU.

1575 **Table 10.12 — Flags definition for COMMAND UPIU**

Flag	Description																		
Flags.R	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.																		
Flags.W	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.																		
Flags.ATTR	<p>The .ATTR field contains the task attribute value as defined by [SAM].</p> <table border="1" style="margin-left: auto; margin-right: auto;"> <thead> <tr> <th colspan="3">ATTR Definition</th> </tr> <tr> <th>Task Attribute</th> <th>Bit 1</th> <th>Bit 0</th> </tr> </thead> <tbody> <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> </tbody> </table> <p>The relative order of execution between Head of Queue commands is left for implementation.</p>	ATTR Definition			Task Attribute	Bit 1	Bit 0	Simple	0	0	Ordered	0	1	Head of Queue	1	0	ACA (Not Used)	1	1
ATTR Definition																			
Task Attribute	Bit 1	Bit 0																	
Simple	0	0																	
Ordered	0	1																	
Head of Queue	1	0																	
ACA (Not Used)	1	1																	
Flags.CP	<p>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.</p> <p>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.</p> <p>A task manager may use command priority to determine an ordering to process commands with the Simple task attribute within the task set.</p> <p>A value of "1" indicates high priority. A value of "0" indicates no priority.</p>																		
NOTE The bit assignment of the Flags field is shown in Table 10.6.																			

1577

1578 **10.7.1.1 Basic header (cont'd)**

1579 **c) Data Segment Length**

1580 The Data Segment Length field shall contain zero as there is no Data Segment in this UPIU.

1581 **d) Expected Data Transfer Length**

1582 The Expected Data Transfer Length field contains a value that represents the number of bytes to be  
1583 transferred that are required to complete the SCSI command request as indicated in the CDB (e.g.,  
1584 TRANSFER LENGTH, ALLOCATION LENGTH, PARAMETER LIST LENGTH, etc.). Data may  
1585 be transferred from the Initiator device to the Target device or from the Target device to the Initiator  
1586 device. This field is valid only if one of the Flags.W or Flags.R bits are set to '1'.

1587 For a data transfer from the Initiator device to the Target device, the .W flag shall be set to '1' and the  
1588 .R flag shall be set to '0'. The value in the Expected Data Transfer Length field represents the number  
1589 of bytes that the Initiator device expects to send to the Target device.

1590 For a data transfer from the Target device to the Initiator device, the .R flag shall be set to '1' and the  
1591 .W flag shall be set to '0'. The value in the Expected Data Transfer Length field represents the number  
1592 of bytes that the Initiator device expects to receive from the Target device.

1593 When the COMMAND UPIU encodes a SCSI WRITE or SCSI READ command (specifically  
1594 WRITE (6), READ (6), WRITE (10), READ (10), WRITE (16), or READ (16)), the value of this  
1595 field shall be the product of the Logical Block Size (bLogicalBlockSize) and the TRANSFER  
1596 LENGTH field of the CDB.

1597 This model requires that the Initiator device will allocate sufficient buffer space to receive the full  
1598 size of the data requested by a command that requires a Data In operation. That size measured in  
1599 bytes shall be the value in Expected Data Transfer Length field. This requirement is important in  
1600 order to realize the full throughput of the Data In phase without the use of additional handshaking  
1601 UPIU's.

1602 The Initiator device may request a Data Out size larger than the size of the receive buffer in the  
1603 Target device. In this case, the Target device will pace the DATA OUT UPIU's by sending READY  
1604 TO TRANSFER UPIU's as needed. The Initiator device will not send a DATA OUT UPIU before it  
1605 receives a READY TO TRANSFER UPIU.

1606 **e) CDB**

1607 The CDB fields contain the Command Descriptor Block. This area is an array of 16 bytes that will  
1608 contain a standard Command Descriptor Block as defined by one of the supported UFS Command Set  
1609 Types. For SCSI commands, specifications such as [SPC] can be referenced. Up to a 16 byte CDB  
1610 can be utilized. The CDB size is implicitly indicated by the group bits of the operation code field in  
1611 CDB[0] for SCSI, which is the SCSI command operation code. If the CDB size is lower than 16 bytes  
1612 the unused COMMAND UPIU bytes are defined as reserved. For other commands, the CDB size is  
1613 dependent upon the command opcode.

1614 **f) Initiator ID (IID)**

1615 The Initiator ID field (bits [7:4] of byte 4) indicates the identity of the Initiator device who created the  
1616 task request. See Initiator ID description in 10.6.2, Basic Header Format, for details.

1617 **g) Command Set Type**

1618 The Command Set Type field will specify an enumerated value that indicates which particular  
1619 command set is used to define the command bytes in the CDB fields. See Command Set Type  
1620 description for details.

### 1621 10.7.2 RESPONSE UPIU

1622 The RESPONSE UPIU contains the basic UPIU header plus additional information indicating the  
1623 command and device level status resulting from the successful or failed execution of a command. The  
1624 Target will generate this UPIU and send it to the Initiator device after it has completed the requested task.

1625 Before terminating a command which requires Data-Out data transfer and before sending the RESPONSE  
1626 UPIU, the Target device shall wait until it receives all DATA OUT UPIUs related to any outstanding  
1627 READY TO TRANSFER UPIUs. Also, the Target device should stop sending READY TO TRANSFER  
1628 UPIUs for the command which requires Data-Out data transfer and to be terminated.

1629 **Table 10.13 — RESPONSE UPIU**

RESPONSE UPIU			
0 xx10 0001b	1 Flags	2 LUN	3 Task Tag
4 IID	5 Command Set Type	6 Reserved	7 Response
8 Total EHS Length (00h)	9 Device Information	10 (MSB)	11 (LSB)
12 (MSB)	13 Residual Transfer Count	14	15 (LSB)
16	17 Reserved	18	19
20	21 Reserved	22	23
24	25 Reserved	26	27
28	29 Reserved	30	31
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.			

1631 **10.7.2.1 Basic Header**

1632 The first 12 bytes of the RESPONSE UPIU contain the Basic Header as described in 10.6.2, Basic Header  
1633 Format. Specific details are as follows:

1634 **a) Transaction Type**

1635 A type code value of xx10 0001b indicates a RESPONSE UPIU.

1636 **b) Flags**

1637 Table 10.14 describes the flags used in RESPONSE UPIU.

1638 **Table 10.14 — Flags definition for RESPONSE UPIU**

Flag	Description
Flags.O	<p>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.</p> <p>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.</p> <p>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 "i) Residual Transfer Count" for further explanation.</p>
Flags.U	<p>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.</p> <p>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.</p> <p>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 "i) Residual Transfer Count" for further explanation.</p>
Flags.D	<p>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.</p>

NOTE The bit assignment of the Flags field is shown in Table 10.6.

1639 **c) Initiator ID (IID)**

1640 The Initiator ID field (bits [7:4] of byte 4) indicates the identity of the Initiator device who created the  
1641 task request. See Initiator ID description in section 10.6.2, Basic Header Format, for details.

1642 **d) Command Set Type**

1643 The Command Set Type field will specify an enumerated value that indicates which particular  
1644 command set is used to define the command bytes in the CDB fields. See 10.6.2, Basic Header  
1645 Format, for details.

1646 **e) Response**

1647 The Response field will contain the UFS response that indicates the UFS defined overall success or  
1648 failure of the series of Command, Data and RESPONSE UPIU's that make up the execution of a task.  
1649 See 10.6.2, Basic Header Format, for details.

1651 **10.7.2.1 Basic Header (cont'd)**

1652 **f) Status**

1653 The Status field contains the command set specific status for a specific command issued by the  
1654 Initiator device. The Status field is command set specific. The Command Set Type field will indicate  
1655 with which command set the status is associated. Specific command sets may or may not define  
1656 detailed extended status indicated as Sense Data. If the command requires extended status, that  
1657 information will be stored in the Sense Data field.

1658 **1) SCSI Command Set Status**

1659 When the Command Set Type field indicates SCSI Command Set the Status field will contain the  
1660 standard SPC defined SCSI status value. Possible values are listed in the Table 10.15. See the  
1661 [SPC] or [SAM] for detailed definition of the status conditions.

1662 A GOOD status indicates successful SCSI completion and therefore no Sense Data will be  
1663 returned.

1664 A status of CHECK CONDITION requires that the Data Segment contain Sense Data for the  
1665 failed command.

1666 Other status values may or may not return Sense Data. In this case a non-zero value in the Data  
1667 Segment Length field indicates that this UPIU contains Sense Data in the Data Segment area.

1668 'M' indicates mandatory implementation of this field and the value specified if fixed. 'O'  
1669 indicates that the support of this field is optional; if it is not supported then a value of zero should  
1670 be inserted in the field otherwise the value will be indicated as described. n/a indicates "not  
1671 applicable" to UFS.

1672 **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.

1674 **10.7.2.1 Basic Header (cont'd)**

1675 **g) Device Information**

1676 The Device Information field provides information at device level not necessarily related with the  
1677 logical unit executing the command.

1678 In general, the information is about events that have an evolution much slower than the regular  
1679 commands, and for which the host response latency is not critical. The use of this field avoids the  
1680 execution of a continuous polling on some UFS attributes.

1681 Bit 0 of the Device Information field is defined, and bit 1 is reserved for HPB (Host Performance  
1682 Booster) Extension Standard. 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
bit 1	Reserved	Reserved for HPB (Host Performance Booster) Extension Standard
Others	Reserved	

1683 The exception sources include: background operations, dynamic capacity, system data pool, etc. See  
1684 13.4.11, Exception Events Mechanism, for details.

1685 **h) Data Segment Length**

1686 The Data Segment Length field will contain the number of valid bytes in the Data Segment.

1687 In the RESPONSE UPIU the Data Segment will contain the Sense Data bytes and the Sense Data  
1688 Length field.

1689 When this field contains zero it indicates that there is no Data Segment area in the UPIU and  
1690 therefore no Sense Data is returned.

1691 This version of the standard, when the Command Set Type field indicates SCSI Command Set, the  
1692 number of Sense Data bytes is 18, therefore this field will contain a value of 20 (18 bytes of Sense  
1693 Data + 2 bytes for Sense Data Length = 20 bytes).

1694 As stated previously, the Data Segment field size is located on a 32-bit (DWORD) boundary. The  
1695 Data Segment Length field indicates the number of "valid" bytes in the Data Segment area and  
1696 therefore its value may not be an integer multiple of four.

1697 **i) Residual Transfer Count**

1698 This field is valid only if one of the Flags.U or Flags.O fields are set to '1', otherwise this field will  
1699 contain zero.

1700 When the Flags.O field is set to '1' then this field indicates the number of bytes that were not  
1701 transferred from/to the Initiator device because the Expected Data Transfer Length field contained a  
1702 value that was lower than the Target device expected to transfer. In other words, the Target device has  
1703 more bytes to receive/send to complete the request but the Initiator device is not expecting more than  
1704 the amount indicated in the Expected Data Transfer Length. For example, the Initiator device may  
1705 intentionally request less bytes than it knows the Target device has available to transfer, because it  
1706 only needs the first N bytes.

1708 **10.7.2.1 Basic Header (cont'd)**

1709 **i) Residual Transfer Count (cont'd)**

1710 When the Flags.U field is set to '1' then this field indicates the number of bytes that were not  
 1711 transferred from/to the Initiator because the Expected Data Transfer Length field contained a value  
 1712 that was higher than the available data bytes. In other words, the Target device has less bytes to  
 1713 receive/send than the Initiator is requesting to transfer. For example, the Initiator device may  
 1714 intentionally request more bytes than the Target device has to transfer when it does not know how  
 1715 many bytes the Target device actually has and it asks for the max or more than possible.

1716 **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

1717 **j) Sense Data Fields**

1718 The Sense Data fields will contain additional information on error condition.

1719 For SCSI command they will provide a copy of first 18 sense data bytes as defined for the fixed  
 1720 format sense data, which corresponds to Response Code value of 70h. See the following subsection  
 1721 for further details.

1722 A successfully executed command will not normally need to return Sense Data, therefore in this case  
 1723 the Data Segment may be empty and the Data Segment Length may have a zero value.

1724 The Sense Data fields will be padded with zeros to place the data on the next nearest 32-bit boundary  
 1725 if the length of valid Sense Data fields plus two is not a multiple of 32-bit.

1726 **k) SCSI Sense Data Fields**

1727 The Sense Data fields will contain standard 18 byte SPC defined sense data when using format for a  
 1728 Response Code value of 70h. See [SPC] for further information.

1729 Sense Data consists of three levels of error codes, each in increasing detail. The purpose is to provide  
 1730 the application client a means to determine the cause of an error or exceptional condition at various  
 1731 levels of detail. The Sense Key provides a general category of what error or exceptional condition  
 1732 occurred and has caused the current command from successfully completing. Further and finer error  
 1733 detail is provided in the Additional Sense Code field (ASC). The Additional Sense Code Qualifier  
 1734 (ASCQ) field refines the error information even further. It is required to implement the Sense Key  
 1735 value when indicating an error or exceptional condition. It is not required to implement the ASC or  
 1736 ASCQ values not described in this document; a value of zero can be placed in these fields if the  
 1737 implementation does not require more refined error detail.

1738 All SCSI commands that terminate in error or exceptional condition will automatically return Sense  
 1739 Data in the RESPONSE UPIU, relieving the host from issuing a subsequent REQUEST SENSE  
 1740 command to retrieve the additional sense error information.

1741

1742 **10.7.2.1 Basic Header (cont'd)**

1743 **I) Sense Data Length**

1744 The Sense Data Length field indicates the number of valid Sense Data bytes that follow. The Sense  
1745 Data Length plus two may be less than the number of bytes contained in the Data Segment area, if  
1746 padding bytes have been added to reach 32-bit boundary.

1747 A successfully executed command will not normally need to return Sense Data, therefore in this case  
1748 the Data Segment area may be empty and the Data Segment Length may have a zero value.

1749 A command that terminated in error or an exception may or may not return Sense Data. If the Sense  
1750 Data Length indicates a value of zero, then that error condition did not return Sense Data. A zero  
1751 value in the Data Segment Length also indicates that no Sense Data was returned. Otherwise, the  
1752 Sense Data Length will contain a value that indicates the number of additional bytes of Sense Data  
1753 information.

1754 **m) SCSI Sense Data Length**

1755 The Sense Data Length field shall indicate a value of 18 when using the SCSI Command Set.

1756 **n) Sense Data Format**

1757 Table 10.17 describes the sense data structure that gives detailed error information about the  
1758 previously executed SCSI command. Eighteen bytes are returned and the Additional Sense Length  
1759 field is set to a value of ten.

1760 'M' indicates mandatory implementation of this field and the value specified if fixed. 'O' indicates  
1761 that the support of this field is optional; if it is not supported then a value of zero should be inserted in  
1762 the field otherwise the value will be indicated as described.

1763

1764 **10.7.2.1 Basic Header (cont'd)**

1765

**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.	M
16:17	7:0	SENSE KEY SPECIFIC	Default value = 00 00 00h	

1766

1767 **10.7.2.1 Basic Header (cont'd)**

1768 The SENSE KEY is used for normal error handling during operation.

1769 The ADDITIONAL SENSE CODE (ASC) and the ADDITIONAL SENSE CODE QUALIFIER (ASCQ) are  
1770 mainly used for detailed diagnostic and logging (post-mortem) information. If the device server does  
1771 not have further information related to the error or exception condition, these fields shall be set to  
1772 zero. Generally, except for a certain few, they're not mandatory and they may be set to zero, which  
1773 means no additional information provided. See [SPC] for a list of additional sense codes and  
1774 additional sense code qualifiers.

1775 o) **Sense Key**

1776 The Sense Key value provides a means to categorize errors and exceptional conditions. The Sense  
1777 Key indicates a particular type of error. The Additional Sense Code and Additional Sense Code  
1778 Qualifier can be used to further detail and describe the condition that the Sense Key indicates. Sense  
1779 Keys are specific to the action performed by a particular command.

1780

1781 **10.7.2.1 Basic Header (cont'd)**

1782

**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 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.	

1783

### 1784 10.7.3 DATA OUT UPIU

1785 The DATA OUT UPIU contains the basic UPIU header plus additional information needed to manage the  
1786 data out transfer. The data transfer flows from Initiator device to Target device (write). The DATA OUT  
1787 UPIU will usually contain a data segment. It is possible to have a null DATA OUT UPIU: the Data  
1788 Segment is empty and Data Segment Length value is zero.

1789 The DATA OUT UPIU is sent in response to READY TO TRANSFER UPIU generated by a Target  
1790 device, according to the rules detailed in 10.7.13, Data out transfer rules.

1791 **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 Data Buffer Offset	14	15 (LSB)
16 (MSB)	17 Data Transfer Count	18	19 (LSB)
20	21	22 Reserved	23
24	25	26 Reserved	27
28	29	30 Reserved	31
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			

1792

1793 **10.7.3.1 Basic Header**

1794 The first 12 bytes of the DATA OUT UPIU contain the Basic Header as described in 10.6.2, Basic  
1795 Header Format. Specific details are as follows:

1796 **a) Transaction Type**

1797 A type code value of 02h indicates a DATA OUT UPIU.

1798 **b) Initiator ID (IID)**

1799 The Initiator ID field (bits [7:4] of byte 4) indicates the identity of the Initiator device who created the  
1800 task request. See Initiator ID description in section 10.6.2, Basic Header Format, for details.

1801 **c) Data Segment Length**

1802 The Data Segment Length shall indicate the number of valid bytes within the Data Segment area, and  
1803 it shall not include the number of padding bytes (if present).

1804 **d) Data Buffer Offset**

1805 The Data Buffer Offset field contains the offset of this UPIU data payload within the complete data  
1806 transfer area. The sum of the Data Buffer Offset and the Data Segment Length shall not exceed the  
1807 Expected Data Transfer Length that was indicated in the COMMAND UPIU.

1808 This field permits out of order sequencing of the DATA OUT UPIU packets. Therefore the order of  
1809 the DATA OUT UPIU packets do not have to be sequential.

1810 NOTE Out of order sequencing will only occur if a UFS device supports it (bDataOrdering = 01h) and if this  
1811 feature is enabled (bOutOfOrderDataEn = 01h).

1812 When the DATA OUT UPIU is a part of a SCSI WRITE transaction [i.e., a transaction which started  
1813 with a WRITE (6), a WRITE (10), or a WRITE (16) command], the value of this field shall be equal  
1814 to an integer multiple of the Logical Block Size (bLogicalBlockSize).

1815 **e) Data Transfer Count**

1816 This field indicates the number of bytes that the Initiator device is transferring to the Target device in  
1817 this UPIU. This value is the number of bytes that are contained within the Data Segment of the UPIU.  
1818 The maximum number of bytes that can be transferred within a single DATA OUT UPIU packet is  
1819 65535 bytes. Therefore, multiple DATA OUT UPIU packets will need to be issued by the Initiator  
1820 device if the Expected Data Transfer Length of the original command requires more than 65535 bytes  
1821 to be transferred.

1822 When the DATA OUT UPIU is a part of a SCSI WRITE transaction [i.e., a transaction which started  
1823 with a WRITE (6), a WRITE (10), or a WRITE (16) command], the value of this field shall be equal  
1824 to an integer multiple of the Logical Block Size (bLogicalBlockSize).

1825 This field and the Data Segment Length field of the UPIU shall contain the same value. This field is  
1826 intended to be used along with the Data Buffer Offset field as part of a DMA context.

1828 **10.7.3.1 Basic Header (cont'd)**

1829 **f) Data Segment**

1830 This is the Data Segment area that contains the data payload.

1831 The maximum data payload size that can be transferred within a single DATA OUT UPIU packet is  
1832 65535 bytes.

1833 The Data Segment area always starts on a 32-bit (DWORD) boundary. The Data Segment area shall  
1834 be entirely filled with data payload to a 32-bit (DWORD) boundary unless the UPIU is the one that  
1835 transmits the last data portion. In this case, if necessary, the Data Segment area shall be padded out to  
1836 the next nearest 32-bit boundary.

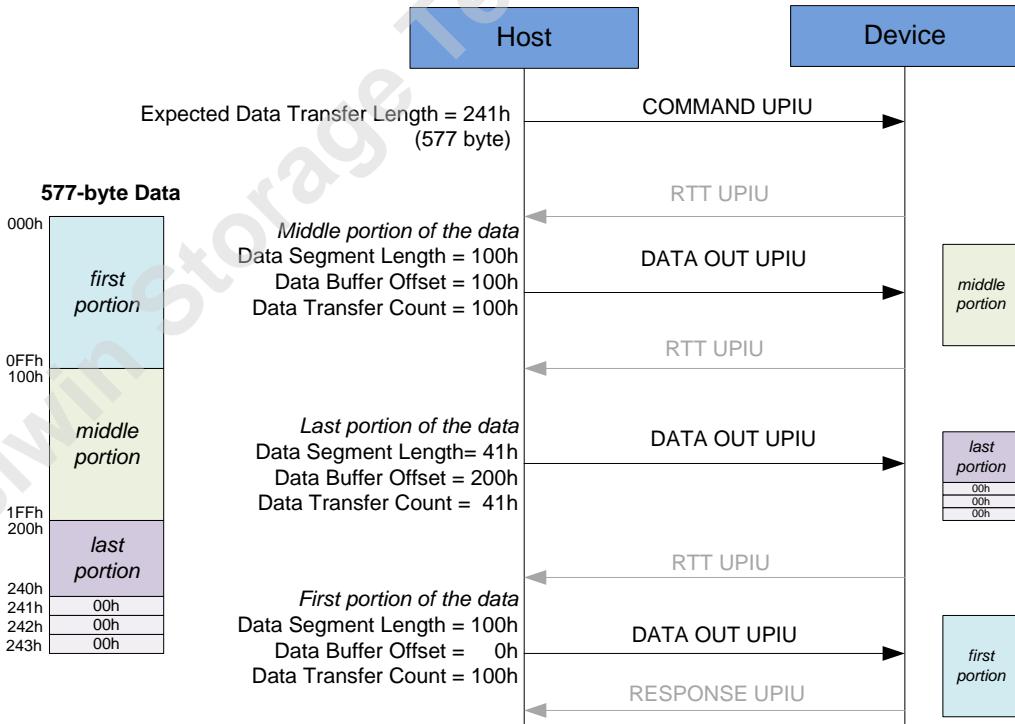
1837 When the DATA OUT UPIU is a part of a SCSI WRITE transaction [i.e., a transaction which started  
1838 with a WRITE (6), a WRITE (10), or a WRITE (16) command], the Data Segment area shall contain  
1839 an integer number of logical blocks.

1840 NOTE For out of order DATA OUT UPIUs, the last data portion may not be transmitted by the final UPIU.

1841 **g) Data out transfer example**

1842 Figure 10.1 shows an example of data transfer from the Initiator device to the Target device. In  
1843 particular, the command processing requires the transfer of 577-byte data. The data transfer is done  
1844 out of order: at the beginning the middle portion of the data, then the last portion and finally the first  
1845 portion.

1846 NOTE The second DATA OUT UPIU delivers the last portion of the data. The Data Segment in this UPIU  
1847 has 65 bytes of valid data and three pad bytes. The Data Segment in the other UPIUs is fully filled (no pad  
1848 bytes).



1849  
1850 **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 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 Data Buffer Offset	14	15 (LSB)
16 (MSB)	17 Data Transfer Count	18	19 (LSB)
20	21 Reserved	22	23
24	25 Reserved	26	27
28	29 Reserved	30	31
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.			

1857

1858 **10.7.4 DATA IN UPIU (cont'd)**

1859 **10.7.4.1 Basic Header**

1860 The first 12 bytes of the DATA IN UPIU contain the Basic Header as described in 10.6.2, Basic Header  
1861 Format. Specific details are as follows:

1862 **a) Transaction Type**

1863 A type code value of xx10 0010b indicates a DATA IN UPIU.

1864 **b) Initiator ID (IID)**

1865 The Initiator ID field (bits [7:4] of byte 4) indicates the identity of the Initiator device who created the  
1866 task request. See Initiator ID description in 10.6.2, Basic Header Format, for details.

1867 **c) Data Segment Length**

1868 The Data Segment Length shall indicate the number of valid bytes within the Data Segment area, and  
1869 it shall not include the number of padding bytes (if present).

1870 **d) Data Buffer Offset**

1871 The Data Buffer Offset field contains the offset of this UPIU data payload within the complete data  
1872 transfer area. The sum of the Data Buffer Offset and the Data Segment Length shall not exceed the  
1873 Expected Data Transfer Length that was indicated in the COMMAND UPIU.

1874 This field permits out of order sequencing of the DATA IN UPIU packets. Therefore the order of the  
1875 SCSI In UPIU packets do not have to be sequential.

1876 NOTE Out of order sequencing will only occur if a UFS device supports it (bDataOrdering = 01h) and if this  
1877 feature is enabled (bOutOfOrderDataEn = 01h).

1878 When the DATA IN UPIU is a part of a SCSI READ transaction [i.e., a transaction which started  
1879 with a READ (6), a READ (10), or a READ (16) command], the value of this field shall be equal to  
1880 an integer multiple of the Logical Block Size (bLogicalBlockSize).

1881 **e) Data Transfer Count**

1882 This field indicates the number of bytes that the Target device has placed in the UPIU Data Segment,  
1883 for transfer back to the Initiator device. This value is the number of valid bytes that are contained  
1884 within the Data Segment of this UPIU. The maximum number of bytes that can be transferred within  
1885 a single DATA IN UPIU packet is 65535 bytes.

1886 When the DATA IN UPIU is a part of a SCSI READ transaction [i.e., a transaction which started  
1887 with a READ (6), a READ (10), or a READ (16) command], the value of this field shall be equal to  
1888 an integer multiple of the Logical Block Size (bLogicalBlockSize).

1889 This field and the Data Segment Length field of the UPIU shall contain the same value.

1890

### 1891 10.7.4.1 Basic Header (cont'd)

#### 1892 f) Data Segment

1893 This is the Data Segment area that contains the data payload.

1894 The maximum data payload size that can be transferred within a single DATA IN UPIU packet is  
1895 65535 bytes.

1896 The Data Segment area always starts on a 32-bit (DWORD) boundary. The Data Segment area shall  
1897 be entirely filled with data payload to a 32-bit (DWORD) boundary unless the UPIU is the one that  
1898 transmits the last data portion. In this case, if necessary, the Data Segment area shall be padded out to  
1899 the next nearest 32-bit boundary.

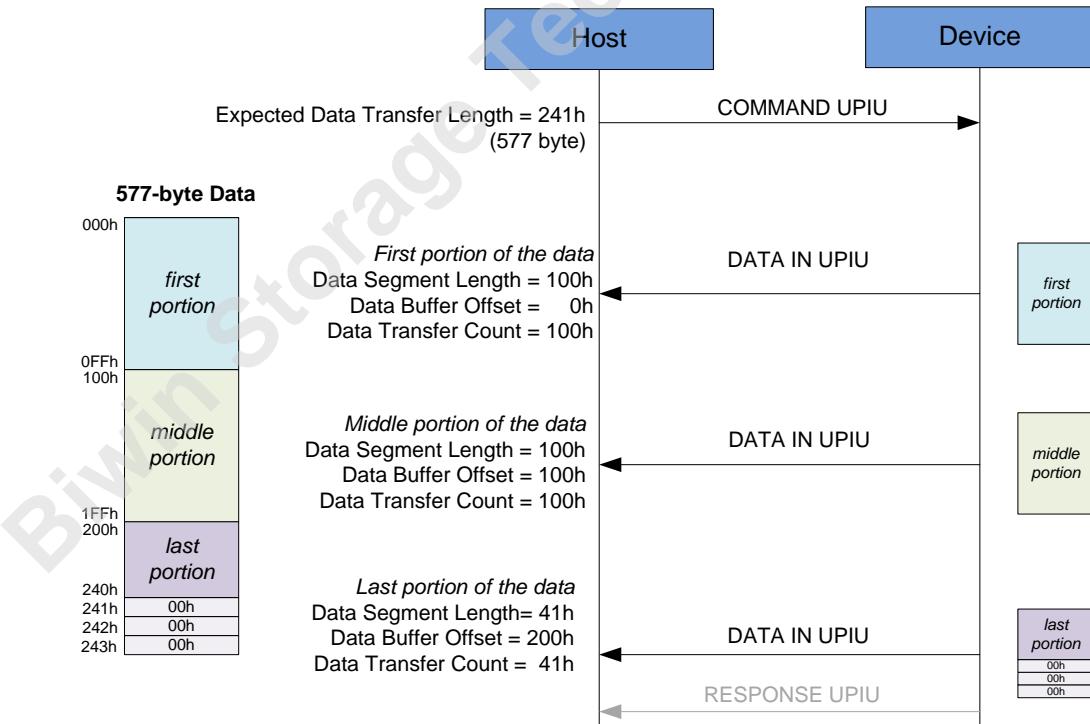
1900 When the DATA IN UPIU is a part of a SCSI READ transaction [i.e., a transaction which started  
1901 with a READ (6), a READ (10), or a READ (16) command], the Data Segment area shall contain an  
1902 integer number of logical blocks.

1903 NOTE For out of order DATA IN UPIUs, the final data portion may not be transmitted by the last UPIU.

#### 1904 g) Data in transfer example

1905 Figure 10.2 shows an example of data transfer from the Target device to the Initiator device. In  
1906 particular, during the command processing 577-byte data is sent to the Initiator device. The data  
1907 transfer is done in sequence: at the beginning the first portion, then the middle portion and finally the  
1908 last portion.

1909 NOTE The last DATA IN UPIU delivers the last portion of the data. The Data Segment in this UPIU has 65  
1910 bytes of valid data and three pad bytes. The Data Segment in the other UPIUs is fully filled (no pad bytes).



1911  
1912 **Figure 10.2 — Data in transfer example**

1913 **10.7.5 READY TO TRANSFER UPIU**

1914 The READY TO TRANSFER UPIU is issued by the Target device when it is ready to receive data blocks  
1915 when processing a SCSI command that requires a data out transfer (e.g., a write command). The Target  
1916 device may request the data in sequence or out of order by setting the appropriate fields within the UPIU.

1917 The Initiator device responds to a READY TO TRANSFER UPIU packet with one DATA OUT UPIU  
1918 packet. The Target device may send one or more READY TO TRANSFER UPIU to satisfy the Expected  
1919 Data Transfer Length that was indicated within the associated COMMAND UPIU. The maximum number  
1920 of bytes that can be requested with a single READY TO TRANSFER UPIU shall not be greater than the  
1921 value indicated by bMaxDataOutSize attribute.

1922 See 10.7.13, Data out transfer rules, for further details about Initiator device to Target device data  
1923 transfer.

1924 **Table 10-21 — READY TO TRANSFER UPIU**

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

1925

1926 **10.7.5 READY TO TRANSFER UPIU (cont'd)**

1927 **10.7.5.1 Basic Header**

1928 The first 12 bytes of the READY TO TRANSFER UPIU contain the Basic Header as described in 10.6.2,  
1929 Basic Header Format. Specific details are as follows:

1930 **a) Transaction Type**

1931 A type code value of xx11 0001b indicates a READY TO TRANSFER UPIU.

1932 **b) Initiator ID (IID)**

1933 The Initiator ID field (bits [7:4] of byte 4) indicates the identity of the Initiator device who created the  
1934 task request. See Initiator ID description in 10.6.2, Basic Header Format, for details.

1935 **c) Data Segment Length**

1936 The Data Segment Length field shall contain zero as there is no Data Segment in this UPIU.

1937 **d) Data Buffer Offset**

1938 The Data Buffer Offset field indicates to the Initiator device the location of the beginning of the  
1939 segment of data to send. The Target device may request the Initiator device to transfer the data in a  
1940 number of UPIUs, not necessarily in sequential order. The sum of the Data Buffer Offset and the Data  
1941 Transfer Count should not exceed the Expected Data Transfer Length that was indicated in the  
1942 COMMAND UPIU.

1943 The Data Buffer Offset shall be an integer multiple of four.

1944 When the RTT UPIU is a part of a SCSI WRITE transaction [i.e., a transaction which started with a  
1945 WRITE (6), a WRITE (10), or a WRITE (16) command], the value of this field shall be equal to an  
1946 integer multiple of the Logical Block Size (bLogicalBlockSize).

1947 **e) Data Transfer Count**

1948 This field indicates the number of bytes the Target device is requesting.

1949 The Data Transfer Count field shall be always an integer multiple of four bytes except for the  
1950 READY TO TRANSFER UPIU which requests the final portion of data in the transfer.

1951 When the RTT UPIU is a part of a SCSI WRITE transaction [i.e., a transaction which started with a  
1952 WRITE (6), a WRITE (10), or a WRITE (16) command], the value of this field shall be equal to an  
1953 integer multiple of the Logical Block Size (bLogicalBlockSize).

1954 The maximum number of bytes that can be requested in a single READY TO TRANSFER UPIU  
1955 shall not be greater than the value indicated by bMaxDataOutSize attribute.

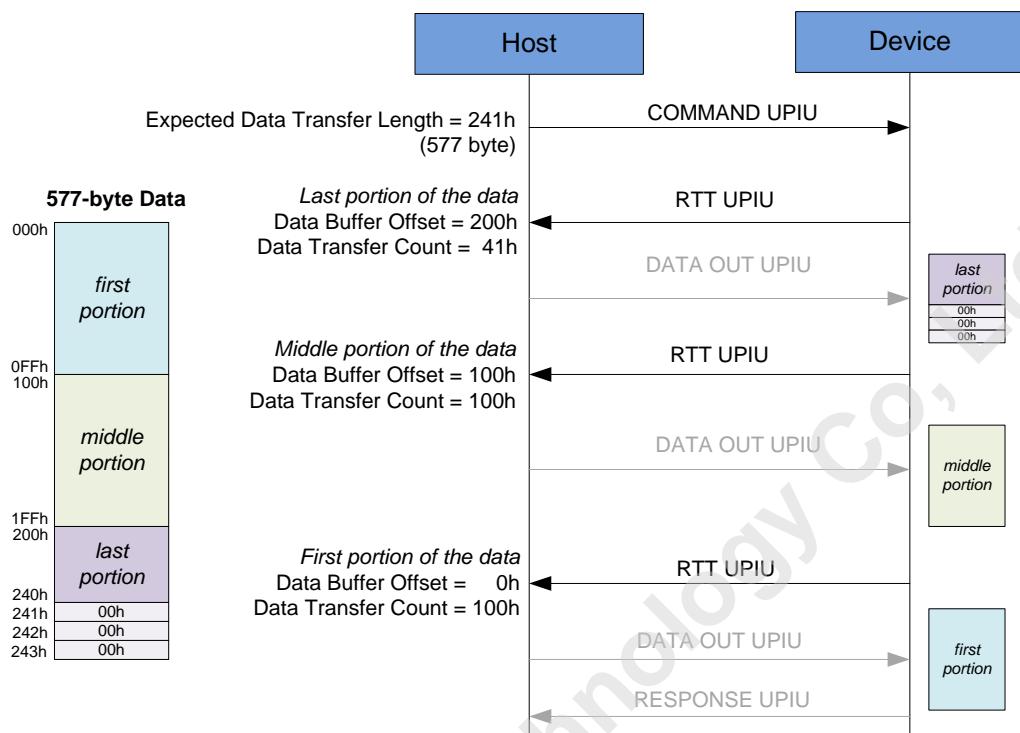
1956 **f) READY TO TRANSFER UPIU sequence example**

1957 Figure 10.3 shows an example of READY TO TRANSFER UPIU sequence. In particular, during the  
1958 command processing the Target device requests the Initiator device to send a total amount of 577-  
1959 byte data. The data transfer is done out of order (reverse): at the beginning the last portion, then the  
1960 middle portion and finally the first portion.

1961 NOTE The first READY TO TRANSFER UPIU requests to send the last portion of the data.

1963

### 10.7.5.1 Basic Header (cont'd)



1964

1965  
1966

Figure 10.3 — READY TO TRANSFER UPIU sequence example

### 1967 10.7.6 TASK MANAGEMENT REQUEST UPIU

1968 The TASK MANAGEMENT REQUEST UPIU is used by the Initiator device to manage the execution of  
1969 one or more tasks within the Target device. The Task Management Request function closely follows the  
1970 SCSI Architecture Model [SAM].

1971 Task Management functions in UFS device requires the LU task manager to have the capability to process  
1972 at least one Task Management Request. If more than one Task Management Request is accepted by a task  
1973 manager in the device, the task manager may execute the requests in any order. The task manager may  
1974 reject a Task Management Request with Task Management Function Failed service response when the  
1975 number of outstanding Task Management Requests submitted by the Host exceeds the capability of the  
1976 Task Manager.

1977 **Table 10.22 — Task Management Request UPIU**

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

1978

## 1979 **10.7.6 TASK MANAGEMENT REQUEST UPIU (cont'd)**

### 1980 **10.7.6.1 Basic Header**

1981 The first 12 bytes of the TASK MANAGEMENT REQUEST UPIU contain the Basic Header as  
1982 described in 10.6.2 Basic Header Format. Specific details are as follows:

#### 1983 **a) Transaction Type**

1984 A type code value of xx00 0100b indicates a TASK MANAGEMENT REQUEST UPIU.

#### 1985 **b) Initiator ID (IID)**

1986 The Initiator ID field (bits [7:4] of byte 4) indicates the identity of the Initiator device who created the  
1987 task request. See Initiator ID description in 10.6.2, Basic Header Format, for details.

#### 1988 **c) Task Management Function**

1989 Table 10.23 defines UFS Task Management Functions based on [SAM].

1990 **Table 10.23 — Task Management Function values**

Function	Value	Description
Abort Task	01h	Abort specific task in queue in a specific LU. Identify by IID, LUN and Task Tag
Abort Task Set	02h	Abort the task queue list in a specific LU. Identify by IID and LUN.
Clear Task Set <sup>(1)</sup>	04h	Clear the task queue list in specific LU. Identify by LUN.
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 IID, 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 IID and LUN. If there is one or 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.

1991 NOTE 1 If TST = 1 (i.e., per I\_T nexus), it aborts all commands in the task set, identified by IID and LUN.  
1992 Since this standard supports TST = 0, all commands in the task set from all initiators are aborted.

#### 1991 **d) Task Management Input Parameters**

1992 **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

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

NOTE 2 Input Parameter 3 and IID field in the basic header shall indicate the same value.

1993 **10.7.7 TASK MANAGEMENT RESPONSE UPIU**

1994 The TASK MANAGEMENT RESPONSE UPIU is sent by the Target device in response to a Task  
1995 Management Request from the Initiator device. The Task Management Response function closely follows  
1996 the SCSI Architecture Model [SAM].

1997 If the Target device is processing a task which requires Data-Out data transfer, and it receives a task  
1998 management request to abort that command, then Target device should stop sending READY TO  
1999 TRANSFER UPIUs for the command requested to abort. Target device shall wait until it receives all  
2000 DATA OUT UPIUs related to any outstanding READY TO TRANSFER UPIUs before sending the  
2001 TASK MANAGEMENT RESPONSE UPIU.

2002 Task management functions that may cause a task abort are: Abort Task, Abort Task Set, Clear Task Set  
2003 and Logical Unit Reset.

2004 **Table 10.25 — Task Management Response UPIU**

TASK MANAGEMENT RESPONSE UPIU			
0 xx10 0100b	1 Flags	2 LUN	3 Task Tag
4 IID	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)			

2005

2006 **10.7.7 TASK MANAGEMENT RESPONSE UPIU (cont'd)**

2007 **10.7.7.1 Basic Header**

2008 The first 12 bytes of the TASK MANAGEMENT RESPONSE UPIU contain the Basic Header as  
2009 described in 10.6.2, Basic Header Format. Specific details are as follows:

2010 **a) Transaction Type**

2011 A type code value of xx10 0100b indicates a TASK MANAGEMENT RESPONSE UPIU.

2012 **b) Initiator ID (IID)**

2013 The Initiator ID field (bits [7:4] of byte 4) indicates the identity of the Initiator device who created the  
2014 task request. See Initiator ID description in 10.6.2, Basic Header Format, for details.

2015 **c) Response**

2016 The Response field contains the UFS response that indicates the success or failure of the Task  
2017 Management Request. See 10.6.2 for details.

2018 **d) Task Management Output Parameters**

2019 **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

2020 **e) Task Management Service Response**

2021 **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 <sup>(1)</sup>	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.	

### 2023 10.7.8 QUERY REQUEST UPIU

2024 The QUERY REQUEST UPIU is used to transfer data between the Initiator device and Target device that  
2025 is outside domain of standard user data transfers for command read and write.

2026 The QUERY REQUEST UPIU can be used to read and write parametric data to or from the Target  
2027 device. It can be used to get information for configuration or enumeration, to set or clear bus or overall  
2028 device conditions, to set or reset global flag values, parameters or attributes, to set or get power or bus or  
2029 network information or to get or set descriptors, to get serial numbers or GUID's (globally unique  
2030 identifiers), etc.

2031 The Target device will send a QUERY RESPONSE UPIU in response to a QUERY REQUEST UPIU.  
2032 After sending a QUERY REQUEST UPIU the Initiator device shall not send a new QUERY REQUEST  
2033 UPIU until it receives the QUERY RESPONSE UPIU for the pending request. If the Target device  
2034 receives a QUERY REQUEST UPIU while it is still processing a previous QUERY REQUEST UPIU, it  
2035 shall ignore the latest request.

2036 The QUERY REQUEST UPIU follows the general UPIU format with a field defined for query function.  
2037 The Transaction Specific Fields are defined specifically for each type of operation.

2038 The Data Segment Area is optional depending upon the query function value. The Data Segment Length  
2039 field will be set to zero if there is no data segment in the packet.

2040 **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 Transaction Specific Fields	14	15
16	17 Transaction Specific Fields	18	19
20	21 Transaction Specific Fields	22	23
24	25 Transaction Specific Fields	26	27
28	29 Reserved	30	31
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)			

2042 **10.7.8 QUERY REQUEST UPIU (cont'd)**

2043 **10.7.8.1 Overview**

2044 Queries are used to read and write data structures between the host and the device. This data is outside the  
2045 scope of normal device reads or writes; data that would be considered system data, configuration data,  
2046 production information, descriptors, special parameters and flags and other.

2047 For UFS the query function will generally be used to read or write Descriptors, Attributes and Flags.  
2048 There are also a range of Vendor Specific operations that can be used to transfer vendor specific data  
2049 between host and device.

2050 All these items reside within the device memory and used by the device to control or define its operation.

2051 The following is a short overview of the most common data structures that are transferred using the Query  
2052 Request function. Please see the related section for more detail on these data structures:

2053 **a) Descriptors**

2054 A Descriptor is a block or page of parameters that describe something about a Device. For example,  
2055 there are Device Descriptors, Configuration Descriptors, Unit Descriptors, etc.

2056 **b) Attributes**

2057 An Attribute is a single parameter that represents a specific range of numeric values that can be set or  
2058 read. This value could be a byte or word or floating point number. For example, baud rate or block  
2059 size would be an attribute. Attribute size can be from 1-bit to 32-bit. Attributes of the same type can  
2060 be organized in arrays, each element of them identified by an index.

2061 **c) Flags**

2062 A Flag is a single Boolean value that represents a TRUE or FALSE, '0' or '1', ON or OFF type of  
2063 value. A Flag can be cleared or reset, set, toggled or read. Flags are useful to enable or disable certain  
2064 functions or modes or states within the device.

2065

2066 **10.7.8 QUERY REQUEST UPIU (cont'd)**

2067 **10.7.8.2 Query Function**

2068 The Query Function field holds the requested query type describing the query function to perform.  
2069 Common query functions are listed in Table 10.29. Currently, there are two general query functions  
2070 defined: Read Request and Write Request. Additional Transaction Specific Fields will be used to specify  
2071 further information needed for the transaction. These fields can describe the specific operation to perform,  
2072 the target data or information to access, the amount of data to transfer and additional parameters and data.

2073 **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

2074 **a) Standard Read Request**

2075 The Standard Read Request function type is used to read requested information from a Target device.  
2076 The Target device will return the requested information to the Initiator device within a QUERY  
2077 RESPONSE UPIU packet.

2078 **b) Standard Write Request**

2079 The Standard Write Request function type is used to write information and data to a Target device.  
2080 The information and data to write to the Target device will be included within the Data Segment field  
2081 of the QUERY REQUEST UPIU packet.

2082

**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 OSF[6]	22	23 (LSB)
24 (MSB)	25 OSF[7]	26	27 (LSB)

**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.

2097 **10.7.8 QUERY REQUEST UPIU (cont'd)**

2098 **10.7.8.4 Read Descriptor Opcode**

2099 The READ DESCRIPTOR OPCODE is used to retrieve a UFS Descriptor from the Target device. A  
2100 descriptor can be a fixed or variable length. There are up to 256 possible descriptor types. The OSF fields  
2101 are used to select a particular descriptor and to read a number of descriptor bytes. The OSF fields are  
2102 defined in Table 10.32.

2103 **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)	19 (LSB) LENGTH
20	21	22	23 Reserved
24	25	26	27 Reserved

2104 **a) DESCRIPTOR IDN**

2105 The Descriptor IDN field contains a value that indicates the particular type of descriptor to retrieve.  
2106 For example, it could indicate a Device Descriptor or Unit Descriptor or String Descriptor. Some  
2107 descriptor types are unique and can be fully identified by the Descriptor Type value. Other descriptors  
2108 can exist in multiple forms, such as String Descriptors, and they are furthered identified with  
2109 subsequent fields.

2110 **b) INDEX**

2111 The Index value is used to further identify a particular descriptor. For example, there may be multiple  
2112 String Descriptors defined. In the case of multiple descriptors the INDEX field is used to select a  
2113 particular one. Multiple descriptors are indexed starting from 0 through 255. The actual index value  
2114 for a particular descriptor will be provided by other means, usually contained within a field of some  
2115 other related descriptor.

2116 **c) SELECTOR**

2117 The SELECTOR field may be needed to further identify a particular descriptor.

2118 **d) LENGTH**

2119 The LENGTH field is used to indicate the number of bytes to read of the descriptor. These bytes will  
2120 be returned in a QUERY RESPONSE UPIU packet. This is the requested length to read, which may  
2121 be less than, or equal to, or greater than the number of bytes within the actual descriptor. If less than,  
2122 or equal to the actual descriptor size, the number of bytes specified will be returned. If the LENGTH  
2123 is greater than the descriptor size, the response will provide the exact descriptor size in the LENGTH  
2124 field of the QUERY RESPONSE UPIU .

2125 **e) Data Segment**

2126 The Data Segment area is empty.

2127 **10.7.8 QUERY REQUEST UPIU (cont'd)**

2128 **10.7.8.5 Write Descriptor Opcode**

2129 The WRITE DESCRIPTOR OPCODE is used to write a UFS Descriptor and it is sent from the host to the  
2130 device. A descriptor can be a fixed or variable length. There are up to 256 possible descriptor types. The  
2131 OSF fields are used to select a particular descriptor. The OSF fields are defined as listed in Table 10.33.

2132 **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)	19 (LSB) LENGTH
20	21	22 Reserved	23
24	25	26 Reserved	27

2133 **a) DESCRIPTOR IDN**

2134 The Descriptor IDN field contains a value that identifies a particular of descriptor to write. For  
2135 example, it could indicate a Device Descriptor or Unit Descriptor or String Descriptor. Some  
2136 descriptor types are unique and can be fully identified by the Descriptor IDN value. Other descriptors  
2137 can exist in multiple forms, such as STRING DESCRIPTORS, and they are furthered identified with  
2138 subsequent fields.

2139 **b) INDEX**

2140 The Index value is used to further identify a particular descriptor. For example, there may be multiple  
2141 String Descriptors defined. In the case of multiple descriptors the INDEX field is used to select a  
2142 particular one. Multiple descriptors are indexed starting from 0 through 255. The actual index value  
2143 for a particular descriptor will be provided by other means, usually contained within a field of some  
2144 other related descriptor.

2145 **c) SELECTOR**

2146 The SELECTOR field may be needed to further identify a particular descriptor.

2147 **d) LENGTH**

2148 The LENGTH field is used to indicate the number of descriptor bytes to write. Only the entire  
2149 descriptor may be written; there is no partial write or update possible. These bytes will be contained  
2150 within the DATA SEGMENT area of the QUERY REQUEST UPIU packet. The DATA SEGMENT  
2151 LENGTH field of the UPIU shall also be set to this same value. If LENGTH is not equal to the  
2152 descriptor size the operation will fail: the descriptor is not updated and the Query Response field of  
2153 the QUERY RESPONSE UPIU is set FAILURE.

2154 **e) Data Segment**

2155 The Data Segment area contains the data to be written.

2157 **10.7.8 QUERY REQUEST UPIU (cont'd)**

2158 **10.7.8.6 Read Attribute Opcode**

2159 The READ ATTRIBUTE OPCODE is used to retrieve a UFS Attribute from the Target device. Attribute  
2160 size can be from 1-bit to 32-bit. There are up to 256 possible Attributes, identified by an identification  
2161 number, IDN, which ranges from 0 to 255. The OSF fields for this opcode are listed in Table 10.34.

2162 **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 Reserved	23	
24	25	26 Reserved	27	

2163 **a) ATTRIBUTE IDN**

2164 The ATTRIBUTE IDN contains a value that identifies a particular Attribute to retrieve from the  
2165 Target device.

2166 **b) INDEX**

2167 For attributes that are organized in array, the index value is used to identify the particular element.  
2168 For example, the LUN is used as index to select the particular element of attributes that have logical  
2169 unit specific values.

2170 The range for the index is defined for each attribute, and it can be from 0 through 255. Index shall be  
2171 set to zero for attributes composed by single element.

2172 **c) SELECTOR**

2173 The SELECTOR field may be needed to further identify a particular element of an attribute. Selector  
2174 field shall be set to zero for attributes that do not require it.

2175 **d) Data Segment**

2176 The Data Segment area is empty.

2179 **10.7.8 QUERY REQUEST UPIU (cont'd)**

2180 **10.7.8.7 Write Attribute Opcode**

2181 The **WRITE ATTRIBUTE OPCODE** is used to write a UFS Attribute to the Target device. Attribute  
2182 size can be from 1-bit to 32-bit. There are up to 256 possible Attributes, identified by an identification  
2183 number, IDN, which ranges from 0 to 255. The OSF fields for this opcode are listed in Table 10.35.

2184 **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

2185 **a) ATTRIBUTE IDN**

2186 The ATTRIBUTE IDN contains a value that identifies a particular Attribute to write in the Target  
2187 device.

2188 **b) INDEX**

2189 For attributes that are organized in array, the index value is used to identify the particular element.  
2190 For example, the LUN is used as index to select the particular element of attributes that have logical  
2191 unit specific values.

2192 The range for the index is defined for each attribute, and it can be from 0 through 255. Index shall be  
2193 set to zero for attributes composed by single element.

2194 **c) SELECTOR**

2195 The SELECTOR field may be needed to further identify a particular element of an attribute. Selector  
2196 field shall be set to zero for attributes that do not require it.

2197 **d) VALUE [31:0]**

2198 The 32-bit VALUE field contains the data value of the Attribute. The VALUE is a right justified, big  
2199 Endian value. Unused upper bits shall be set to zero.

2200 **e) Data Segment**

2201 The Data Segment area is empty.

2202 **10.7.8 QUERY REQUEST UPIU (cont'd)**

2203 **10.7.8.8 Read Flag Opcode**

2204 The READ FLAG OPCODE is used to retrieve a UFS Flag value from the Target device. A Flag is a  
2205 fixed size single byte value that represents a Boolean value. There can be defined up to 256 possible Flag  
2206 values. A Flag is identified by its FLAG IDN, an identification number that ranges in value from 0 to 255.  
2207 The OSF fields for this opcode are listed in Table 10.36.

2208 The FLAG data, either one (01h) or zero (00h), is returned within the Transaction Specific Fields area of  
2209 a QUERY RESPONSE UPIU packet.

2210 **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 Reserved	23
24	25	26 Reserved	27

2211 **a) FLAG IDN**

2212 The FLAG IDN field contains a value that identifies a particular Flag to retrieve from the Target  
2213 device.

2214 **b) INDEX**

2215 The index field may be needed to identify a particular element of a flag.

2216 **c) SELECTOR**

2217 The selector field may be needed to further identify a particular element of a flag. Selector field is not  
2218 used in this version of the standard and its value shall be zero.

2219 **d) Operation**

2220 The Boolean value of the addressed flag is returned in a QUERY RESPONSE UPIU.

2221 **e) Data Segment**

2222 The Data Segment area is empty.

2223 **10.7.8 QUERY REQUEST UPIU (cont'd)**

2224 **10.7.8.9 Set Flag**

2225 **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 Reserved	23
24	25	26	27 Reserved

2226

2227 **a) FLAG IDN**

2228 The FLAG IDN field contains a value that identifies a particular Flag to set in the Target device.

2229 **b) INDEX**

2230 The index field may be needed to identify a particular element of a flag.

2231 **c) SELECTOR**

2232 The selector field may be needed to further identify a particular element of a flag. Selector field is not  
2233 used in this version of the standard and its value shall be zero.

2234 **d) Operation**

2235 The Boolean value of the addressed flag is set to TRUE or one.

2236 **e) Data Segment**

2237 The Data Segment area is empty.

2238 **10.7.8 QUERY REQUEST UPIU (cont'd)**

2239 **10.7.8.10 Clear Flag**

2240 **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	

2241 **a) FLAG IDN**

2242 The FLAG IDN field contains a value that identifies a particular Flag to clear in Target device.

2243 **b) INDEX**

2244 The index field may be needed to identify a particular element of a flag.

2245 **c) SELECTOR**

2246 The selector field may be needed to further identify a particular element of a flag. Selector field is not  
2247 used in this version of the standard and its value shall be zero.

2248 **d) Operation**

2249 The Boolean value of the addressed flag is cleared to FALSE or zero.

2250 **e) Data Segment**

2251 The Data Segment area is empty.

2252 **10.7.8 QUERY REQUEST UPIU (cont'd)**

2253 **10.7.8.11 Toggle Flag**

2254 **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 Reserved	23
24	25	26	27 Reserved

2255 **a) FLAG IDN**

2256 The FLAG IDN field contains a value that identifies a particular Flag to toggle in the Target device.

2257 **b) INDEX**

2258 The index field may be needed to identify a particular element of a flag.

2259 **c) SELECTOR**

2260 The selector field may be needed to further identify a particular element of a flag. Selector field is not  
2261 used in this version of the standard and its value shall be zero.

2262 **d) Operation**

2263 The Boolean value of the addressed flag is set to the negated current value.

2264 **e) Data Segment**

2265 The Data Segment area is empty.

2266 **10.7.8 QUERY REQUEST UPIU (cont'd)**

2267 **10.7.8.12 NOP**

2268 Table 10.40 defines NOP OPCODE for QUERY REQUEST UPIU.

2269 **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 Reserved	22	23
24	25	26 Reserved	27

2270

2271 **a) Data Segment**

2272 The Data Segment area is empty.

2273 **10.7.9 QUERY RESPONSE UPIU**

2274 The QUERY RESPONSE UPIU is used to transfer data between the Target device and Initiator device in  
2275 response to a QUERY REQUEST UPIU.

2276 The QUERY RESPONSE UPIU is used to return parametric data to the requesting Initiator device in case  
2277 of read descriptor/attribute(flag query request, or to provide response to write descriptor/attribute query  
2278 request or set/clear/toggle flag query request.

2279 **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)			

2280 The QUERY RESPONSE UPIU follows the general UPIU format a field defined for query function.

2281 The transaction specific fields are defined specifically for each type of operation.

2282 The Data Segment Area is optional depending upon the Query Function value. The Data Segment Length  
2283 field will be set to zero if there is no data segment in the packet.

2284

2285 **10.7.9 QUERY RESPONSE UPIU (cont'd)**

2286 **10.7.9.1 Overview**

2287 The Query Response function will respond to the query function that was sent from the Initiator device in  
2288 the QUERY REQUEST UPIU. The Query Response may or may not return data depending upon the  
2289 function. If data needs to be returned it will be returned in the Data Segment of the UPIU or one of the  
2290 Transaction Specific Fields.

2291 **10.7.9.2 Query Function**

2292 The Query Function field will contain the original query function value that was received in the  
2293 corresponding QUERY REQUEST UPIU.

2294 **10.7.9.3 Query Response**

2295 The Query Response field indicates the completion code of the action taken in response to the QUERY  
2296 REQUEST UPIU. Possible values are listed in Table 10.42.

2297 NOTE In case of unsuccessful operation, the Target device may either set Query Response field to FFh, or  
2298 optionally provide more detailed information about the failure using one of the other values.

2299 **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 <sup>(1)</sup>
F9h	Invalid LENGTH
FAh	Invalid value <sup>(2)</sup>
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.

2301 **10.7.9 QUERY RESPONSE UPIU (cont'd)**

2302 **10.7.9.4 Device Information**

2303 See Device Information field definition in RESPONSE UPIU, 10.7.2.1 Basic Header.

2304 **10.7.9.5 Transaction Specific Fields**

2305 The transaction specific fields are defined specifically for each type of operation. For the STANDARD  
2306 READ REQUEST and STANDARD WRITE REQUEST, the operation specific fields are defined as in  
2307 Table 10.43.

2308 **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)	19 (LSB)	OSF[5]
20 (MSB)	21	22 OSF[6]	23	(LSB)
24 (MSB)	25	26 OSF[7]	27	(LSB)

2309 **a) OPCODE**

2310 The opcode indicates the operation to perform. Possible opcode values are listed in Table 10.31.

2311 If in a QUERY REQUEST UPIU, the Query Function field is set to STANDARD READ REQUEST  
2312 and the OPCODE field is set to WRITE DESCRIPTOR, WRITE ATTRIBUTE, SET FLAG, CLEAR  
2313 FLAG, or TOGGLE FLAG; then the query request shall fail and the Query Response field shall be  
2314 set to either “Invalid OPCODE” or “General failure”.

2315 If in a QUERY REQUEST UPIU, the Query Function field is set to STANDARD WRITE  
2316 REQUEST and the OPCODE field is set to READ DESCRIPTOR, READ ATTRIBUTE, or READ  
2317 FLAG; then the query request shall fail and the Query Response field shall be set to either “Invalid  
2318 OPCODE” or “General failure”.

2319 **b) OSF**

2320 The OSF field is an Opcode Specific Field. The OSF fields will be defined for each specific  
2321 OPCODE.

2323 **10.7.9 QUERY RESPONSE UPIU (cont'd)**

2324 **10.7.9.6 Read Descriptor Opcode**

2325 The READ DESCRIPTOR OPCODE is used to retrieve a UFS DESCRIPTOR from the Target device. A  
2326 descriptor can be a fixed or variable length. There are up to 256 possible descriptor types. The OSF fields  
2327 are used to select a particular descriptor and to read a number of descriptor bytes. The OSF fields are  
2328 defined in Table 10.44.

2329 The READ DESCRIPTOR OPCODE is returned in response to a QUERY REQUEST UPIU containing  
2330 the same value in the OPCODE field.

2331 **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)	19 (LSB)	LENGTH
20	21	22	23	
		Reserved		
24	25	26	27	Reserved

2332 **a) DESCRIPTOR IDN**

2333 The DESCRIPTOR IDN field contains the same DESCRIPTOR IDN value sent from the  
2334 corresponding QUERY REQUEST UPIU.

2335 **b) INDEX**

2336 The Index field value returned is the same INDEX value of the corresponding QUERY REQUEST  
2337 UPIU.

2338 **c) SELECTOR**

2339 The SELECTOR field returned is the same SELECTOR value of the corresponding QUERY  
2340 REQUEST UPIU.

2341 **d) LENGTH**

2342 The LENGTH field is used to indicate the number of bytes returned in response to the corresponding  
2343 QUERY REQUEST UPIU. This value could be less than the requested size, if the size of the data  
2344 item is smaller than the size requested in the corresponding QUERY REQUEST UPIU.

2345 **e) Data Segment**

2346 The Data Segment area contains the descriptor data.

2347

2348 **10.7.9 QUERY RESPONSE UPIU (cont'd)**

2349 **10.7.9.7 Write Descriptor Opcode**

2350 The WRITE DESCRIPTOR OPCODE is used to respond to a write descriptor query request. A descriptor  
2351 can be a fixed or variable length. There are up to 256 possible descriptor types. The OSF fields are used to  
2352 select a particular descriptor. The OSF fields are defined in Table 10.45.

2353 **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)	19 (LSB)	LENGTH
20	21	22 Reserved	23	
24	25	26	27	Reserved

2354

2355 **a) DESCRIPTOR IDN**

2356 The Descriptor IDN field contains the same DESCRIPTOR IDN value sent from the corresponding  
2357 QUERY REQUEST UPIU.

2358 **b) INDEX**

2359 The Index field value returned is the same INDEX value of the corresponding QUERY REQUEST  
2360 UPIU.

2361 **c) SELECTOR**

2362 The SELECTOR field returned is the same SELECTOR value of the corresponding QUERY  
2363 REQUEST UPIU.

2364 **d) LENGTH**

2365 The LENGTH field is used to indicate the number of descriptor bytes written. Only the entire  
2366 descriptor may be written; there is no partial write or update possible.

2367 **e) Data Segment**

2368 The Data Segment area is empty.

2369

2370 **10.7.9 QUERY RESPONSE UPIU (cont'd)**

2371 **10.7.9.8 Read Attribute Opcode**

2372 The READ ATTRIBUTE OPCODE is used to retrieve a UFS attribute from the Target device. Attribute  
2373 size can be from 1-bit to 32-bit. There are up to 256 possible attributes, identified by an identification  
2374 number, IDN, which ranges from 0 to 255.

2375 The response to a READ ATTRIBUTE request will be returned in a QUERY RESPONSE UPIU. A  
2376 success or failure code for the entire operation will be contained within the RESPONSE field.

2377 The attribute data will be returned within the transaction specific fields. The Transaction Specific fields  
2378 are formatted as indicated in Table 10.46. The first two 32-bit words of those fields will echo the first two  
2379 32-bit words of the transaction specific fields of the QUERY REQUEST UPIU. The third word will  
2380 contain the Attribute data.

2381 **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

2382 **a) ATTRIBUTE IDN**

2383 The ATTRIBUTE IDN field contains the same ATTRIBUTE IDN value sent from the corresponding  
2384 QUERY REQUEST UPIU.

2385 **b) INDEX**

2386 The Index field value returned is the same INDEX value of the corresponding QUERY REQUEST UPIU.

2387 **c) SELECTOR**

2388 The SELECTOR field returned is the same SELECTOR value of the corresponding QUERY REQUEST  
2389 UPIU.

2390 **d) VALUE [31:0]**

2391 The 32-bit VALUE field contains the data value of the ATTRIBUTE. The VALUE is a right justified, big  
2392 Endian value. Unused upper bits shall be set to zero.

2393 **e) Data Segment**

2394 The Data Segment area is empty.

2395 **10.7.9 QUERY RESPONSE UPIU (cont'd)**

2396 **10.7.9.9 Write Attribute Opcode**

2397 The WRITE ATTRIBUTE OPCODE is used to respond to a write attribute query request. Attribute size  
2398 can be from 1-bit to 32-bit. There are up to 256 possible attributes, identified by an identification number,  
2399 IDN, which ranges from 0 to 255. The OSF fields for this opcode are listed in Table 10.47

2400 **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 Reserved	27	

2401 **a) ATTRIBUTE IDN**

2402 The ATTRIBUTE IDN field contains the same ATTRIBUTE IDN value sent from the corresponding  
2403 QUERY REQUEST UPIU.

2404 **b) INDEX**

2405 The Index field value returned is the same INDEX value of the corresponding QUERY REQUEST  
2406 UPIU.

2407 **c) SELECTOR**

2408 The SELECTOR field returned is the same SELECTOR value of the corresponding QUERY  
2409 REQUEST UPIU.

2410 **d) VALUE [31:0]**

2411 This field contains the same data provided in write attribute query request.

2412 **e) Data Segment**

2413 The Data Segment area is empty.

2415 **10.7.9 QUERY RESPONSE UPIU (cont'd)**

2416 **10.7.9.10 Read Flag Opcode**

2417 The READ FLAG OPCODE is used to retrieve a UFS FLAG value from the Target device. A FLAG is a  
2418 fixed size single byte value that represents a Boolean value. There can be defined up to 256 possible  
2419 FLAG values. A FLAG is identified by its FLAG IDN, an identification number that ranges in value from  
2420 0 to 255. The FLAG data, either '1' or '0', is returned within the Transaction Specific Fields area of a  
2421 QUERY RESPONSE UPIU packet.

2422 The response to a READ ATTRIBUTE request will be returned in a QUERY RESPONSE UPIU. A  
2423 success or failure code for the entire operation will be contained within the RESPONSE field.

2424 The attribute data will be returned within the transaction specific fields. The Transaction Specific fields  
2425 are formatted as indicated in Table 10.48. The first two 32-bit words of those fields will echo the first two  
2426 32-bit words of the transaction specific fields of the QUERY REQUEST UPIU. The third word will  
2427 contain the FLAG data, a '0' or '1' value.

2428 **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

2429 **a) FLAG IDN**

2430 The FLAG IDN field contains the same FLAG IDN value sent from the corresponding QUERY  
2431 REQUEST UPIU

2432 **b) INDEX**

2433 The Index field value returned is the same INDEX value of the corresponding QUERY REQUEST UPIU.

2434 **c) SELECTOR**

2435 The SELECTOR field returned is the same SELECTOR value of the corresponding QUERY REQUEST  
2436 UPIU

2437 **d) FLAG VALUE**

2438 The FLAG VALUE field contains the FLAG data: 00h or 01h.

2439 **e) Data Segment**

2440 The Data Segment area is empty.

2441 **10.7.9 QUERY RESPONSE UPIU (cont'd)**

2442 **10.7.9.11 Set Flag**

2443 **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

2444 **a) FLAG IDN**

2445 The FLAG IDN field contains the same FLAG IDN value sent from the corresponding QUERY  
2446 REQUEST UPIU

2447 **b) INDEX**

2448 The INDEX field value returned is the same INDEX value of the corresponding QUERY REQUEST  
2449 UPIU.

2450 **c) SELECTOR**

2451 The SELECTOR field returned is the same SELECTOR value of the corresponding QUERY  
2452 REQUEST UPIU.

2453 **d) FLAG VALUE**

2454 The FLAG VALUE field contains the FLAG data: 00h or 01h.

2455 This field is valid only if the Query Response field indicates that the operation has been successfully  
2456 completed ("Success").

2457 **e) Data Segment**

2458 The Data Segment area is empty.

2459

2460 **10.7.9 QUERY RESPONSE UPIU (cont'd)**

2461 **10.7.9.12 Clear Flag**

2462 **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

2463 **a) FLAG IDN**

2464 The FLAG IDN field contains the same FLAG IDN value sent from the corresponding QUERY  
2465 REQUEST UPIU

2466 **b) INDEX**

2467 The Index field value returned is the same INDEX value of the corresponding QUERY REQUEST  
2468 UPIU.

2469 **c) SELECTOR**

2470 The SELECTOR field returned is the same SELECTOR value of the corresponding QUERY  
2471 REQUEST UPIU.

2472 **d) FLAG VALUE**

2473 The FLAG VALUE field contains the FLAG data: 00h or 01h.

2474 This field is valid only if the Query Response field indicates that the operation has been successfully  
2475 completed ("Success").

2476 **e) Data Segment**

2477 The Data Segment area is empty.

2478 **10.7.9 QUERY RESPONSE UPIU (cont'd)**

2479 **10.7.9.13 Toggle Flag**

2480 **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	25	26	27	Reserved

2481 **a) FLAG IDN**

2482 The FLAG IDN field contains the same FLAG IDN value sent from the corresponding QUERY  
2483 REQUEST UPIU.

2484 **b) INDEX**

2485 The Index field value returned is the same INDEX value of the corresponding QUERY REQUEST  
2486 UPIU.

2487 **c) SELECTOR**

2488 The SELECTOR field returned is the same SELECTOR value of the corresponding QUERY  
2489 REQUEST UPIU

2490 **d) FLAG VALUE**

2491 The FLAG VALUE field contains the FLAG data: 00h or 01h.

2492 This field is valid only if the Query Response field indicates that the operation has been successfully  
2493 completed ("Success").

2494 **e) Data Segment**

2495 The Data Segment area is empty.

2496

2497 **10.7.9 QUERY RESPONSE UPIU (cont'd)**

2498 **10.7.9.14 NOP**

2499 Table 10.52 defines NOP OPCODE for QUERY RESPONSE UPIU.

2500 **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 Reserved	23
24	25	26	27 Reserved

2501

2502 **a) Data Segment**

2503 The Data Segment area is empty.

2504 **10.7.10 REJECT UPIU**

2505 All UPIU packets include the basic header segment and some transaction specific fields. In addition to  
2506 them, UPIU packets may have: Data Segment, Extra Header Segment, Header E2ECRC, Data E2ECRC.

2507 The purpose of the REJECT UPIU is to simplify the software development and the system debug.

2508 **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) Data Segment Length (0000h)	11 (LSB)
12 Basic Header Status	13 Reserved	14 E2E Status	15 Reserved
16	17 Reserved	18 Reserved	19
20	21 Reserved	22 Reserved	23
24	25 Reserved	26 Reserved	27
28	29 Reserved	30 Reserved	31
Header E2ECRC (omit if HD=0)			

2509 The device shall send a REJECT UPIU if it receives a UPIU with an invalid Transaction Type.

2510 The Transaction Type is defined in 10.6.2a, Basic Header Format, and it is composed by the following  
2511 fields: HD bit, DD bit and the Transaction Code.

2512 Since this version of the standard does not support end-to-end CRC for header and data segments, a  
2513 Transaction Type value is valid if;

- 2514 • HD bit and DD bit are set to zero.  
2515 • the Transaction Code identifies one of the defined UPIU transactions from the Initiator device to  
2516 Target device, see Table 10.1, UPIU Transaction Codes, (reserved values excluded).

2517 The device shall not respond with a REJECT UPIU in the following cases:

- 2518 • Incorrect LUN field or Command Set Type field in a COMMAND UPIU: the device shall send a  
2519 RESPONSE UPIU. In particular, in case of an incorrect Command Set Type field value, the Data  
2520 Segment Area of the RESPONSE UPIU shall be empty (Data Segment Length shall be equal to  
2521 zero).  
2522 • Incorrect LUN field or Task Management Function field in TASK MANAGEMENT REQUEST  
2523 UPIU: the device shall send a TASK MANAGEMENT RESPONSE UPIU.  
2524 • Incorrect Query Function field in QUERY REQUEST UPIU: the device shall send a QUERY  
2525 RESPONSE UPIU

2526

2527 **10.7.10.1 Basic Header**

2528 The first 12 bytes of the Reject UPIU contain the Basic Header as described in 10.6.2, Basic Header  
2529 Format. Specific details are as follows:

2530 **a) Transaction Type**

2531 A type code value of xx11 1111b indicates a Reject UPIU.

2532 **b) Flags**

2533 The Flags field value shall be equal to zero.

2534 **c) LUN**

2535 The LUN shall be equal to the LUN value of the rejected UPIU.

2536 **d) Task Tag**

2537 The Task Tag shall be equal to the Task Tag value of the rejected UPIU.

2538 **e) Initiator ID (IID)**

2539 The IID shall be equal to the IID value of the rejected UPIU.

2540 **f) Response**

2541 The Response field shall be set to 01h (Target Failure) indicating that the Target device was not able  
2542 to execute the requested operation.

2543 **g) Data Segment Length**

2544 The Data Segment Length field shall contain zero as there is no Data Segment in this UPIU.

2545 **h) Basic Header Status**

2546 The Basic Header Status field provides information about error detected in the UPIU received by the  
2547 Initiator device.

2548 Table 10.54 defines the possible values for the Basic Header Status field.

2549 **Table 10.54 — Basic Header Status Description**

Value	Name
00h	Reserved
01h	Invalid Transaction Type
02h to FFh	Reserved

2550 **i) E2E Status**

2551 The E2E Status field provides the result of the end-to-end CRC of the rejected UPIU for both Header  
2552 and Data. E2E Status is reserved if end-to-end CRC is not supported.

2553

**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

2554 **10.7.11 NOP OUT UPIU**

2555 The Initiator device may use NOP OUT UPIU to check the connection to a device. The Target device will  
2556 respond to a NOP OUT UPIU sending a NOP IN UPIU back to the Initiator device.

2557

**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)

2558

2559    **10.7.11.1 Basic Header**

2560    The first 12 bytes of the NOP OUT UPIU contain the Basic Header  
2561    Format. Specific details are as follows:

2562    **a) Task Tag**

2563    Task Tag normally is related to I\_T\_L\_Q nexus addressing of SCSI while here it is used in a pure  
2564    UTP (device level) context.

2565    **b) Transaction Type**

2566    A type code value of xx00 00000b indicates a NOP OUT UPIU.

2567    **c) Flags**

2568    The Flags field value shall be equal to zero.

2569    **d) Data Segment Length**

2570    The Data Segment Length field shall contain zero as there is no Data Segment in this UPIU.

2571

2572 **10.7.12 NOP IN UPIU**

2573 NOP IN UPIU is the response from the Target device to a NOP OUT UPIU sent by the Initiator device.

2574

**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)

2575

2576 **10.7.12.1 Basic Header**

2577 The first 12 bytes of the NOP IN UPIU contain the Basic Header as described in 10.6.2, Basic Header  
2578 Format. Specific details are as follows:

2579 **a) Transaction Type**

2580 A type code value of xx10 00000b indicates a NOP IN UPIU.

2581 **b) Flags**

2582 The Flags field value shall be equal to zero.

2583 **c) Task Tag**

2584 The Task Tag shall be equal to the Task Tag value of the corresponding NOP OUT UPIU.

2585 **d) Response**

2586 The Response field shall be set to 00h (Target Success) indicating that the Target device was able to  
2587 respond to the NOP OUT UPIU.

2588 **e) Data Segment Length**

2589 The Data Segment Length field shall contain zero as there is no Data Segment in this UPIU.

2590

### 2591 10.7.13 Data out transfer rules

2592 Data out transfer rules related with RTT have been defined for the following reasons:

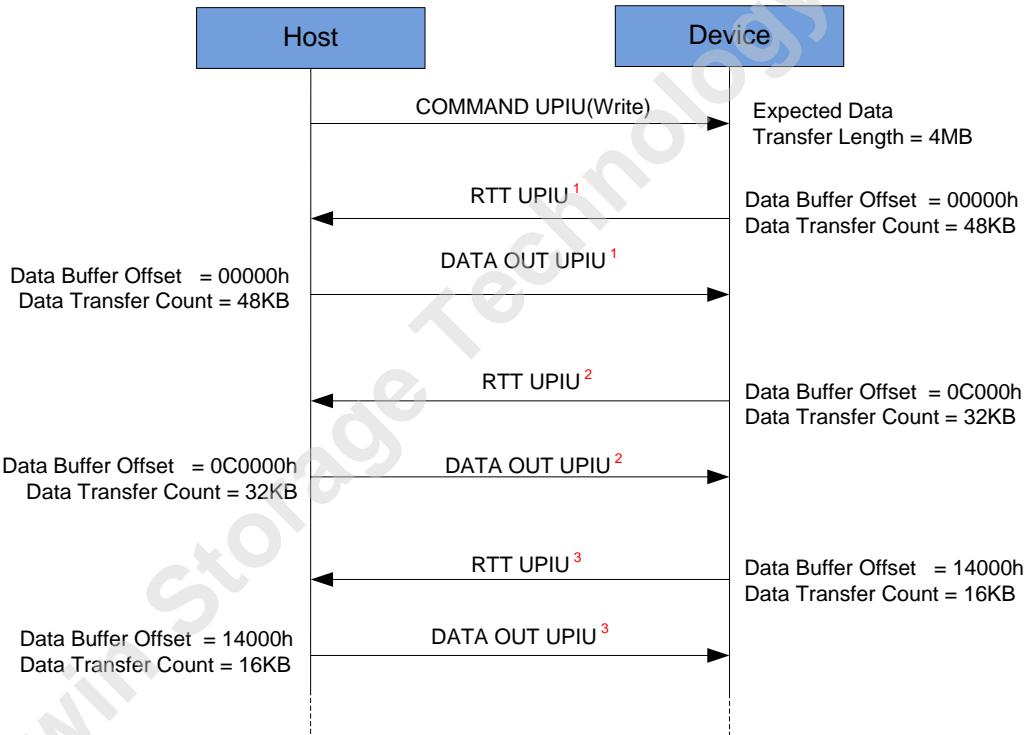
- 2593 • To have the same implementation of UFS data transfer mechanisms for data out transactions across  
2594 various host and device vendors.
- 2595 • To limit the number of outstanding RTTs sent by the device based on host capability.

2596 RTT requests related to several commands and from any logical units may be sent in any order.

2597 Examples of commands with data out transfer are: MODE SELECT (10), WRITE (6), WRITE (10),  
2598 WRITE (16), FORMAT UNIT, SECURITY PROTOCOL OUT, etc.

2599 The following rules are applicable for device in generating the RTT and the host in handling the RTT:

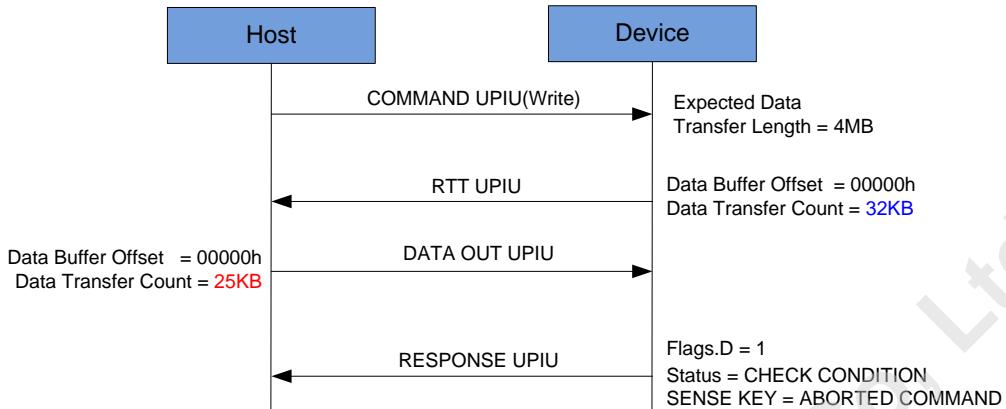
- 2600 • Rule 1 - The host shall send only one DATA OUT UPIU for each RTT received from the device.
- 2601     ○ Figure 10.4 shows an example of UTP traffic related to a write command processing: the host  
2602       sends one and only one DATA OUT UPIU for each READY TO TRANSFER UPIU sent by the  
2603       device.



2604 2605 **Figure 10.4 — Example for data out transfer rule 1**

- 2606     ○ If the Data Buffer Offset field value or the Data Transfer Count field value in the DATA OUT  
2607       UPIU does not match the corresponding parameters in the RTT request, the device shall terminate  
2608       the command by sending RESPONSE UPIU with UTP Data Out Mismatch Error flag (Flags.D)  
2609       set and Status = CHECK CONDITION with SENSE KEY = ABORTED COMMAND. In this  
2610       case, the device shall wait until it receives all DATA OUT UPIUs related to any outstanding  
2611       READY TO TRANSFER UPIUs before sending the Response UPIU with UTP Data Out  
2612       Mismatch Error flag (Flags.D) set and Status = CHECK CONDITION with SENSE KEY =  
2613       ABORTED COMMAND. Figure 10.5 describes a scenario, where the Data Transfer Count value  
2614       does not match.

2615    **10.7.13 Data out transfer rules (cont'd)**



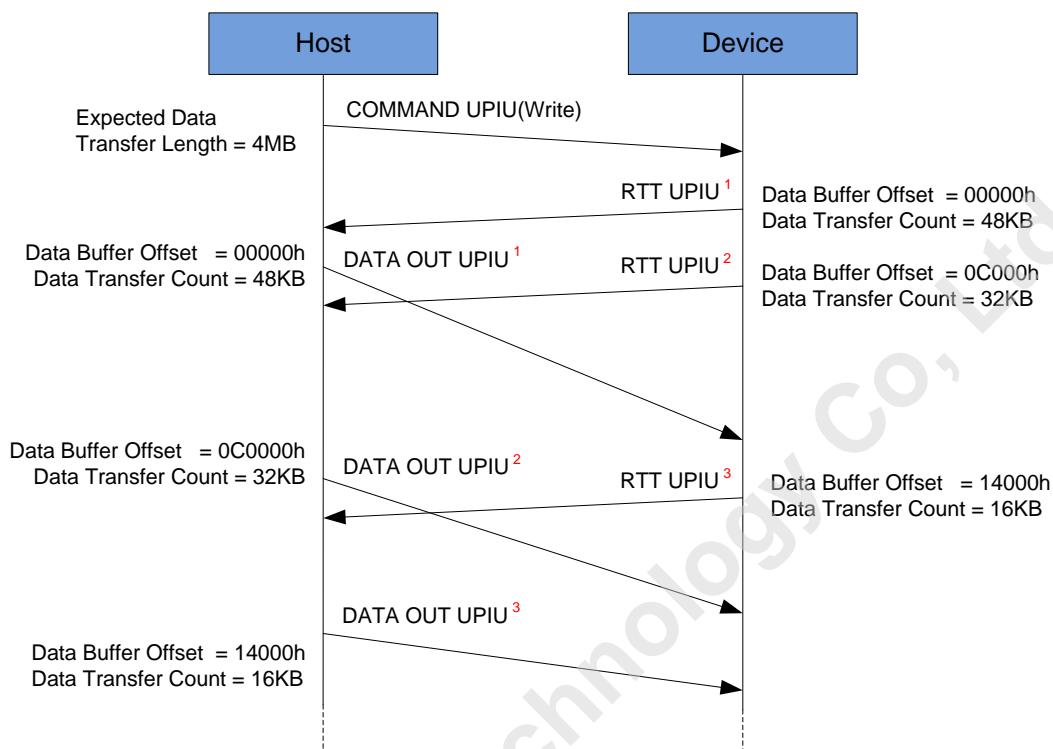
2616

2617    **Figure 10.5 — Example for Data Transfer Count mismatch**

- 2618    • Rule 2 – Device shall not have outstanding RTTs more than specified by host
- 2619        o bDeviceRTTCap read-only parameter in Device Descriptor defines the maximum number of  
2620        outstanding RTTs which can be supported by device. bMaxNumOfRTT read-write attribute limits  
2621        the number of outstanding RTTs generated by the device. bMaxNumOfRTT is equal to two after  
2622        device manufacturing, and it may be changed by the host according to its capability to increase  
2623        performance. In particular, bMaxNumOfRTT value shall not be set to a value greater than  
2624        bDeviceRTTCap value, and it may be set only when the command queues of all logical units are  
2625        empty. If the host attempts to write a value higher than what indicated by bDeviceRTTCap, the  
2626        value shall not be changed and the QUERY RESPONSE UPIU shall have Query Response field  
2627        set to "Invalid VALUE". If the host attempts to write bMaxNumOfRTT when there is at least one  
2628        logical unit with command queue not empty, the operation shall fail, and Query Response field in  
2629        the QUERY RESPONSE UPIU shall be set to FFh ("General failure").
- 2630        o Figure 10.6 shows an example of UTP traffic related to a write command processing assuming  
2631        bMaxNumOfRTT = 2. The Target device sends RTT UPIU<sup>1</sup> and RTT UPIU<sup>2</sup>, and the Initiator  
2632        device starts to provide data related to first request. There are two outstanding RTTs (RTT UPIU<sup>1</sup>  
2633        and RTT UPIU<sup>2</sup>) therefore the Target device cannot send additional RTT UPIUs. The Target  
2634        device sends the third data request (RTT UPIU<sup>3</sup>) only after receiving DATA OUT UPIU<sup>1</sup>.

2635

2636 **10.7.13 Data out transfer rules (cont'd)**



2637

2638

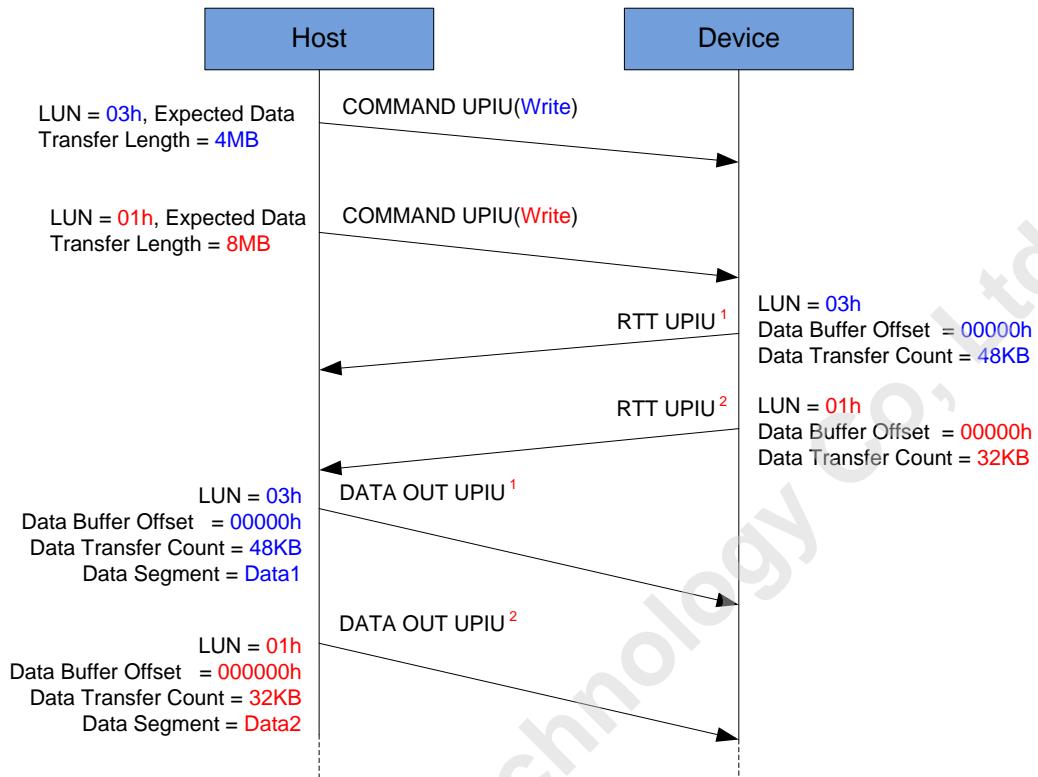
**Figure 10.6 — Example for data out transfer rule 2**

2639

- 2640 • Rule 3 – Across all logical units, DATA OUT UPIUs shall be sent in the same order of RTT UPIUs:  
2641 data for RTT<sup>N</sup> shall be transferred before transferring data for RTT<sup>N+1</sup>.  
2642     ○ For example, if the host first receives RTT UPIU<sup>1</sup> and then RTT UPIU<sup>2</sup> from the device, it shall  
2643       send data related to the first request (Data1) prior to data related to the second request (Data2).

2644

2645 10.7.13 Data out transfer rules (cont'd)



2646

2647

**Figure 10.7 — Example for data out transfer rule 3**

2648

- It is recommended for device to determine the sequence of RTTs based on logical unit priority, command priority, internal optimization, etc.

2650

#### 2651 10.7.13.1 Implementation

2652 The following parameters are defined to implement data out transfer rules.

2653

**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

2654

## 2655 10.8 Logical Units

2656 This section gives more details on the definition of logical unit in the UFS Standard.

### 2657 10.8.1 UFS SCSI Domain

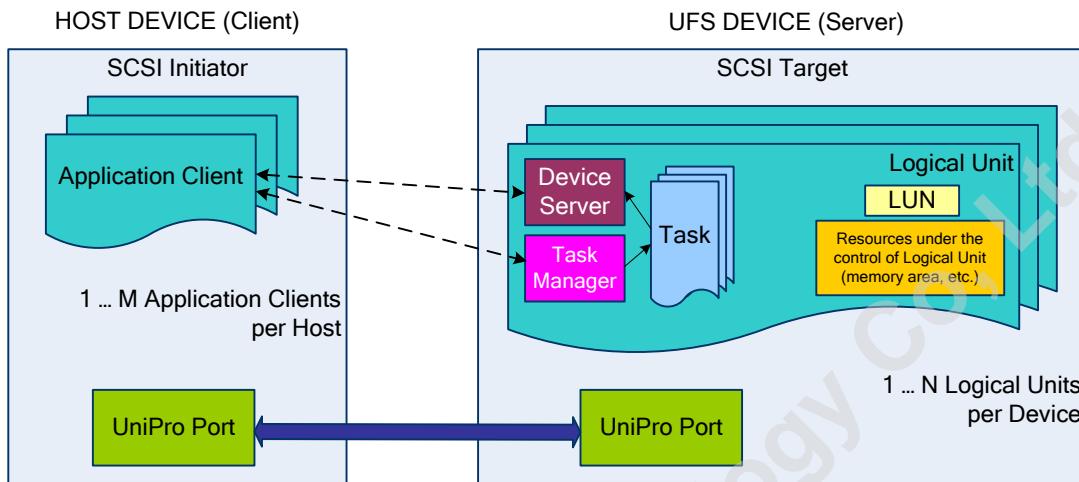


Figure 10.8 — UFS SCSI domain

### 2660 10.8.2 UFS Logical Unit Definition

2661 A logical unit (LU) is an externally addressable, independent, processing entity that processes SCSI tasks  
2662 (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

2667 Commands addressed to logical unit ‘i’ are handled by logical unit ‘i’ exclusively, not visible, handled or  
2668 processed by logical unit ‘j’

2669 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.)

2674

2675 **10.8.3 Well Known Logical Unit Definition**

2676 Well known logical units, as defined by SCSI, support very specific types of commands, usually only four  
2677 or five commands such as REPORT LUNS command to allow an application client to issue requests to  
2678 receive specific information usually relating to the entire device.

2679 In this standard, additional well known logical units are defined for specific UFS functions, including  
2680 Boot and RPMB. Each well known logical unit has a well known logical unit number (W-LUN).

2681 **10.8.4 Logical Unit Addressing**

2682 The 8-bit LUN field in UPIU is used to provide either LUN or W-LUN. In particular, the most significant  
2683 bit of this field (WLUN\_ID) shall be set according to the logical unit type as follows:

- 2684 • WLUN\_ID = 0b for logical unit,  
2685 • WLUN\_ID = 1b for well known logical unit.

2686 The remaining 7 bits of the LUN field (UNIT\_NUMBER\_ID) shall be set to either the LUN value or the  
2687 W-LUN value, depending on the logical unit type.

LUN Field in UPIU							
7	6	5	4	3	2	1	0
WLUN_ID	UNIT_NUMBER_ID						

WLUN\_ID specifies whether a logical unit or a well known logical unit is being addressed:  
- a value of zero indicates a logical unit is being addressed,  
- a value of one indicates a well known logical unit is being addressed.

UNIT\_NUMBER\_ID specifies the unit number (LUN or W-LUN)

2699 **Figure 10.9 — Logical Unit Addressing**

2700 Therefore, the encoding of the LUN field in UPIU supports up to 128 LUN's and up to 128 W-LUN's  
2701 ( $0 \leq \text{UNIT\_NUMBER\_ID} \leq 127$ ) .

2702

### 2703 **10.8.5 Well Known Logical Unit Defined in UFS**

2704 The following well known logical units are defined in this standard for SCSI and UFS specific functions:  
2705 REPORT LUNS, UFS Device, Boot, RPMB.

2706 The REPORT LUNS well known logical unit is defined in [SPC] and provides the logical unit inventory.  
2707 The UFS Device well known logical unit provides UFS device level interaction (i.e., Power mode control,  
2708 Wipe Device). The Boot well known logical unit is a virtual reference to the actual logical unit containing  
2709 boot code, as designated by the host. The Boot well known logical unit is read at the system startup to  
2710 access the boot code. The RPMB well known logical unit supports the RPMB function with its own  
2711 independent processes and memory space as dictated by the RPMB security definition.

2712 Each well known logical unit shall only process the commands listed in Table 10.59. If one of the four  
2713 well known logical units receives a command that is not listed in Table 10.59, then the command shall be  
2714 terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the  
2715 additional sense code set to INVALID COMMAND OPERATION CODE.

2716

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

2717 NOTE If bBootEnable field in the Device Descriptor is set to zero or if the Boot well known logical is not mapped  
2718 to an enabled logical unit (see bLUEnable, bBootLunID and bBootLunEn), then the Boot well known logical unit  
2719 shall terminate TEST UNIT READY, READ (6), READ (10), and READ (16) commands with CHECK  
2720 CONDITION status.

2721 **10.8.6 Translation of 8-bit UFS LUN to 64-bit SCSI LUN Address**

2722 The SCSI Architecture Model describes a 64-bit LUN addressing scheme. The value of C1h in the first 8  
2723 bits of the 64-bit address indicates a well known LUN address in the SAM SCSI format. Examples of  
2724 translation of the 8-bit LUN field value in UPIU to 64-bit SCSI address are shown in Table 10.60.

2725

2726 **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 <u>01</u> 00 00 00 00 00 00h
Logical Unit 6	06h	06h	00 <u>06</u> 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 <u>30</u> 00 00 00 00 00 00h
RPMB	C4h	44h	C1 <u>44</u> 00 00 00 00 00 00h

2727

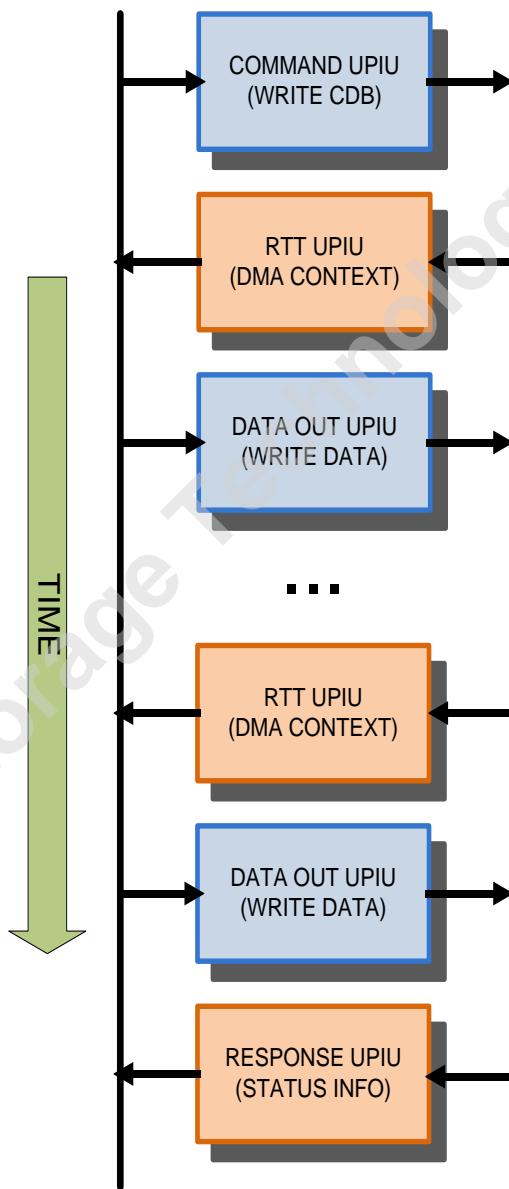
2728 **10.8.7 SCSI Write Command**

2729 The execution of a SCSI write command is composed by the following subsequent phases: command,  
2730 data, status. In the command phase a write command is sent to the device using COMMAND UPIU.

2731 During the subsequent phase data is delivered to the UFS device using DATA OUT UPIU: the UFS  
2732 device paces the data delivery by sending a READY TO TRANSFER UPIU when it is ready for the next  
2733 DATA OUT UPIU. (The UFS device may send new READY TO TRANSFER UPIU's before it receives  
2734 data for previous request.)

2735 The write command terminates with a RESPONSE UPIU that contains the status.

2736 Figure 10.10 shows an example of UPIU sequence for SCSI write command.



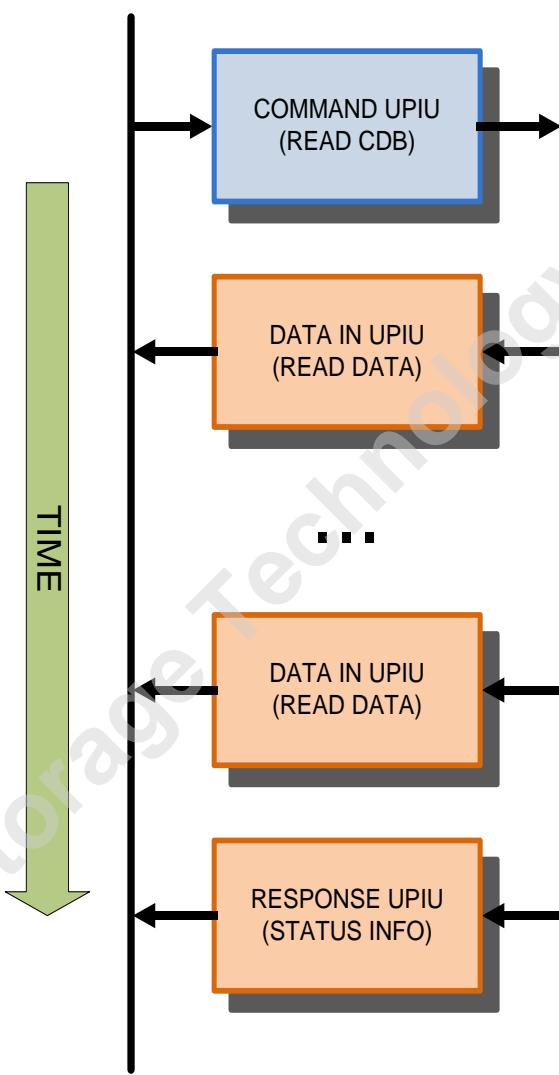
2737  
2738

**Figure 10.10 — SCSI Write**

2739 **10.8.8 SCSI Read Command**

2740 The execution of a SCSI read command is composed by the following subsequent phases: command,  
2741 data, status. In the command phase a read command is sent to the device using COMMAND UPIU.  
2742 During the subsequent phase the UFS device delivers data to the host using DATA IN UPIU. The read  
2743 command terminates with a RESPONSE UPIU that contains the status.

2744 Figure 10.11 shows an example of UPIU sequence for SCSI read command.



2745 **Figure 10.11 — SCSI Read**  
2746  
2747

2748 **10.8.9 Unit Attention Condition**

2749 Unit Attention Condition (UAC) is a condition which needs to be serviced before the logical unit can  
2750 process commands.

2751 Table 10.61 shows all events which shall establish UAC. Power-on, HW Reset, EndPointReset, and Host  
2752 UniProWarm Reset shall establish UAC on all LUs including all Well-known LUs (REPORT LUNS,  
2753 UFS Device, RPMB and BOOT). LU Reset shall establish UAC only for an addressed LU.  
2754

2755 **Table 10.61 — Events for UAC establishment**

Event	Value
Power-on	All LUs including all well-known LUs
HW Reset	
EndPointReset	
Host UniPro Warm Reset	
LU Reset	Addressed LU

2756  
2757 UAC on each LU shall be cleared before it can be accessed successfully (See Table 10.62 for exceptions).  
2758 Except for some specific commands listed in Table 10.62, if a command is sent to a LU with a pending  
2759 UAC, the command will fail. The device server shall respond with CHECK CONDITION status, Sense  
2760 Key set to UNIT ATTENTION. Afterwards the device server shall clear UAC on the LU. Table 10.62  
2761 shows some exceptional commands on UAC behavior.  
2762

2763 **Table 10.62 — Commands for exceptional behavior on UAC**

Command	Status on pending UAC		UAC after command response
REQUEST SENSE	GOOD <sup>(1)</sup>	-	Clear
INQUIRY	GOOD	-	Not clear
REPORT LUNS	GOOD	-	Not clear
Others	CHECK CONDITION	UNIT ATTENTION	Clear

NOTE 1 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 (See 11.3.17.4 Request Sense Status Response)

2765 **10.9 Application Layer and Device Manager Transport Protocol Services**

2766 **10.9.1 UFS Initiator Port and Target Port Attributes**

2767

2768 **Table 10.63 — 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

2769

### 2770 10.9.2 Execute Command procedure call transport protocol services

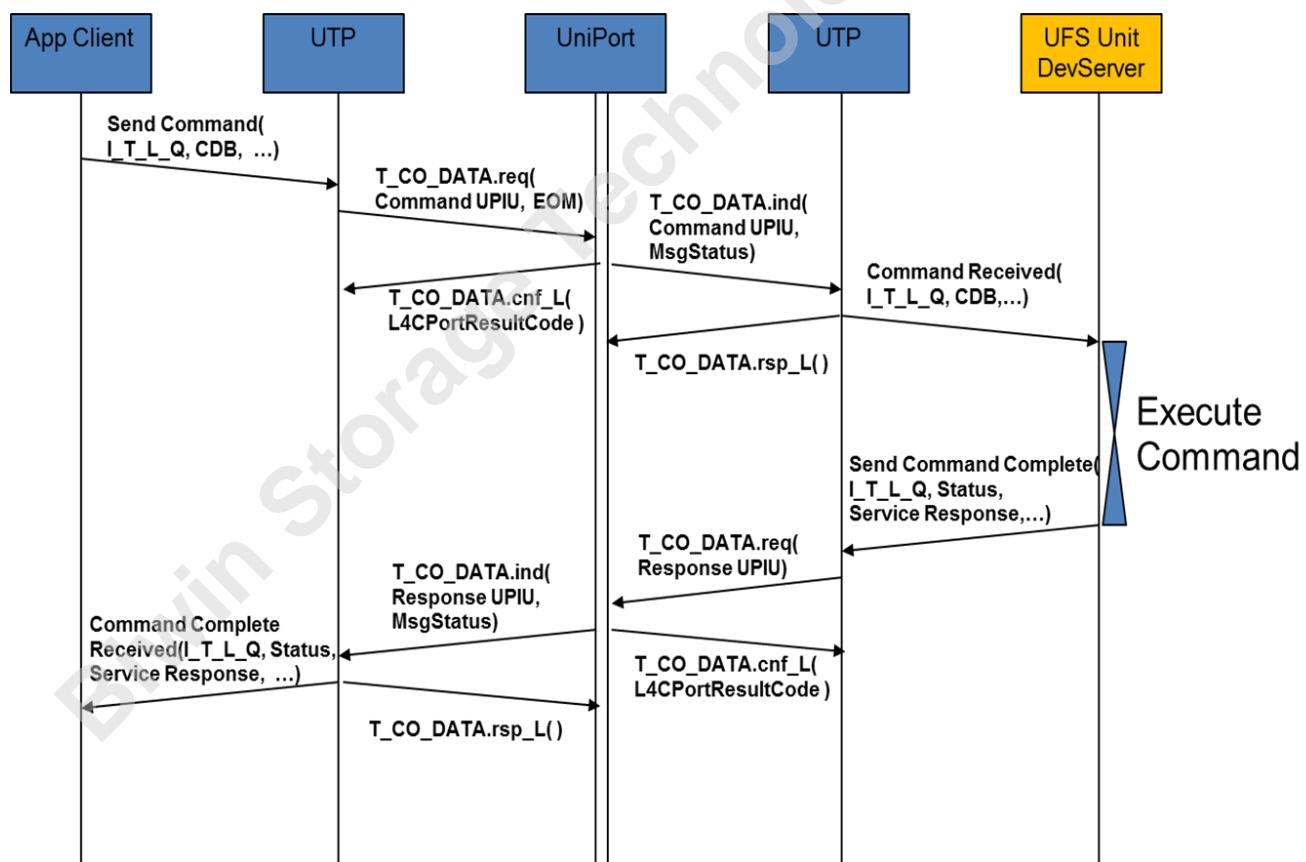
2771 All SCSI transport protocol standards shall define the SCSI transport protocol specific requirements for  
2772 implementing the Send SCSI Command request, the SCSI Command Received indication, the Send  
2773 Command Complete response, and the Command Complete Received confirmation SCSI transport  
2774 protocol services.

2775 All SCSI initiator devices shall implement the Send SCSI Command request and the Command Complete  
2776 Received confirmation SCSI transport protocol services as defined in the applicable SCSI transport  
2777 protocol standards. All SCSI target devices shall implement the SCSI Command Received indication and  
2778 the Send Command Complete response SCSI transport protocol services as defined in the applicable  
2779 SCSI transport protocol standards.

2780

Initiator device	Target device
Send SCSI Command request	SCSI Command Received indication
Command Complete Received confirmation	Send Command Complete response

2781  
2782



2783  
2784  
2785

Figure 10.12 — Command without Data Phase

2786 **10.9.3 SCSI Command transport protocol service**

2787 An application client uses the Send Command transport protocol service request to request a UFS Initiator  
2788 port transmit a COMMAND UPIU via UniPort

- 2789 • Send SCSI Command (IN (I\_T\_L\_Q Nexus, CDB, Task Attribute, [Data-in Buffer Size], [Data-out  
2790 Buffer], [Data-out Buffer Size], [CRN], [Command Priority], [First Burst Enabled]))

2791 **Table 10.64 — 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.

2792

2793 **10.9.4 SCSI Command Received transport protocol**

2794 A UFS target port uses the SCSI Command Received transport protocol service indication to notify a task  
2795 manager that it has received a COMMAND UPIU.

- 2796 • SCSI Command Received (IN (I\_T\_L\_Q Nexus, CDB, Task Attribute, [CRN], [Command Priority], [First  
2797 Burst Enabled]))

2798 **Table 10.65 — 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.

2799

2800 **10.9.5 Send Command Complete transport protocol service**

2801 A device server uses the Send Command Complete transport protocol service response to request that a  
2802 UFS target port transmit a RESPONSE UPIU.

- 2803 • Send Command Complete (IN (I\_T\_L\_Q Nexus, [Sense Data], [Sense Data Length], Status, [Status  
2804 Qualifier], Service Response))

2805 A device server shall only call Send Command Complete () after receiving SCSI Command Received ().

2806 A device server shall not call Send Command Complete () for a given I\_T\_L\_Q nexus until:

- 2807 a) all its outstanding Receive Data-Out () calls for that I\_T\_L\_Q nexus have been responded to with  
2808     • Data-Out Received (); and  
2809 b) all its outstanding Send Data-In () calls for that I\_T\_L\_Q nexus have been responded to with Data-In  
2810 Delivered ().

2811 **Table 10.66 — 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

2812

- 2813    **10.9.6 Command Complete Received transport protocol service**
- 2814    A UFS initiator port uses the Command Complete Received transport protocol service confirmation to  
2815    notify an application client that it has received a response for its COMMAND UPIU.
- 2816    • Command Complete Received (IN (I\_T\_L\_Q Nexus, [Data-in Buffer], [Sense Data], [Sense Data Length],  
2817    Status, [Status Qualifier], Service Response))

2818    **Table 10.67 — 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

2819

2820

### 2821 **10.9.7 Data transfer SCSI transport protocol services**

2822 The data transfer services provide mechanisms for moving data to and from the SCSI initiator port while  
2823 processing commands. All SCSI transport protocol standards shall define the protocols required to  
2824 implement these services.

2825 The application client's Data-In Buffer and/or Data-Out Buffer each appears to the device server as a  
2826 single, logically contiguous block of memory large enough to hold all the data required by the command.

2827

#### 2828 **10.9.7.1 Send Data-In transport protocol service**

2829 A device server uses the Send Data-In transport protocol service request to request that a UFS target port  
2830 sends data.

- 2831 • Send Data-In (IN (I\_T\_L\_Q Nexus, Device Server Buffer, Application Client Buffer Offset, Request Byte  
2832 Count))

2833 A device server shall only call Send Data-In () during a read operation.

2834 A device server shall not call Send Data-In () for a given I\_T\_L\_Q nexus after it has called Send  
2835 Command Complete () for that I\_T\_L\_Q nexus (e.g., a RESPONSE UPIU with for that I\_T\_L\_Q nexus)  
2836 or called Task Management Function Executed for a task management function that terminates that task  
2837 (e.g., an ABORT TASK).

2838 **Table 10.68 — 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

2839

#### 2840 **10.9.7.2 Data-In Delivered transport protocol service**

2841 This confirmation notifies the device server that the specified data was successfully delivered to the  
2842 application client buffer, or that a UniPro delivery subsystem error occurred while attempting to deliver  
2843 the data.

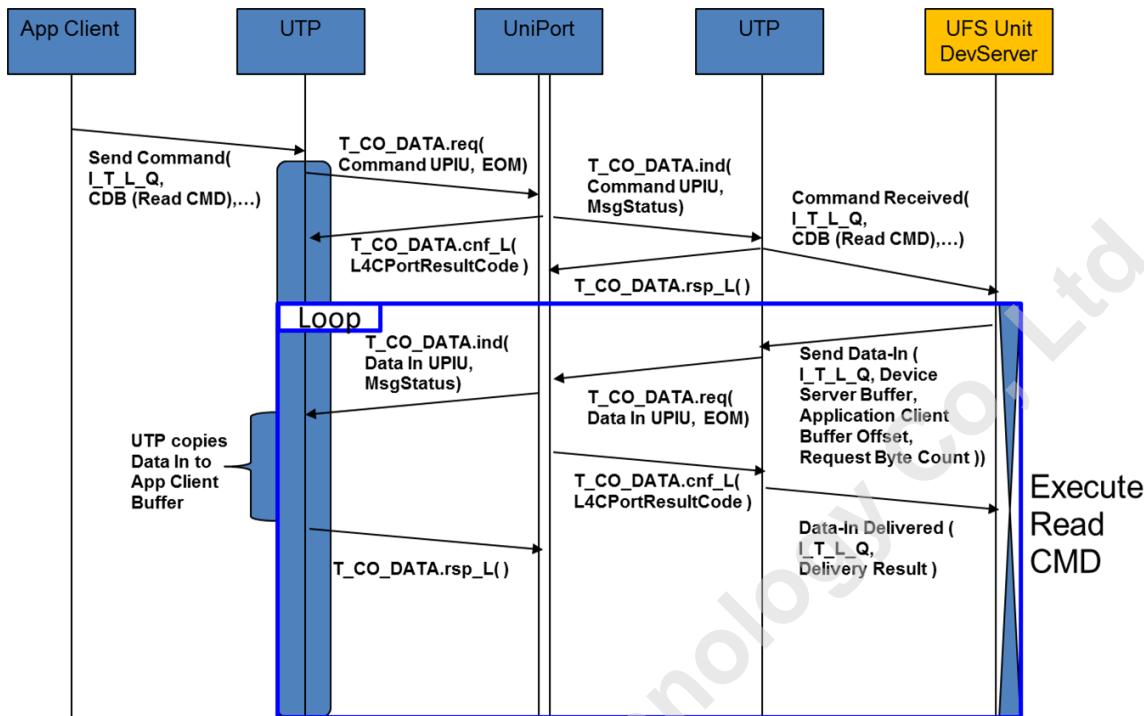
- 2844 • Data-In Delivered (IN (I\_T\_L\_Q Nexus, Delivery Result))

2845 **Table 10.69 — Data-In Delivered transport protocol service**

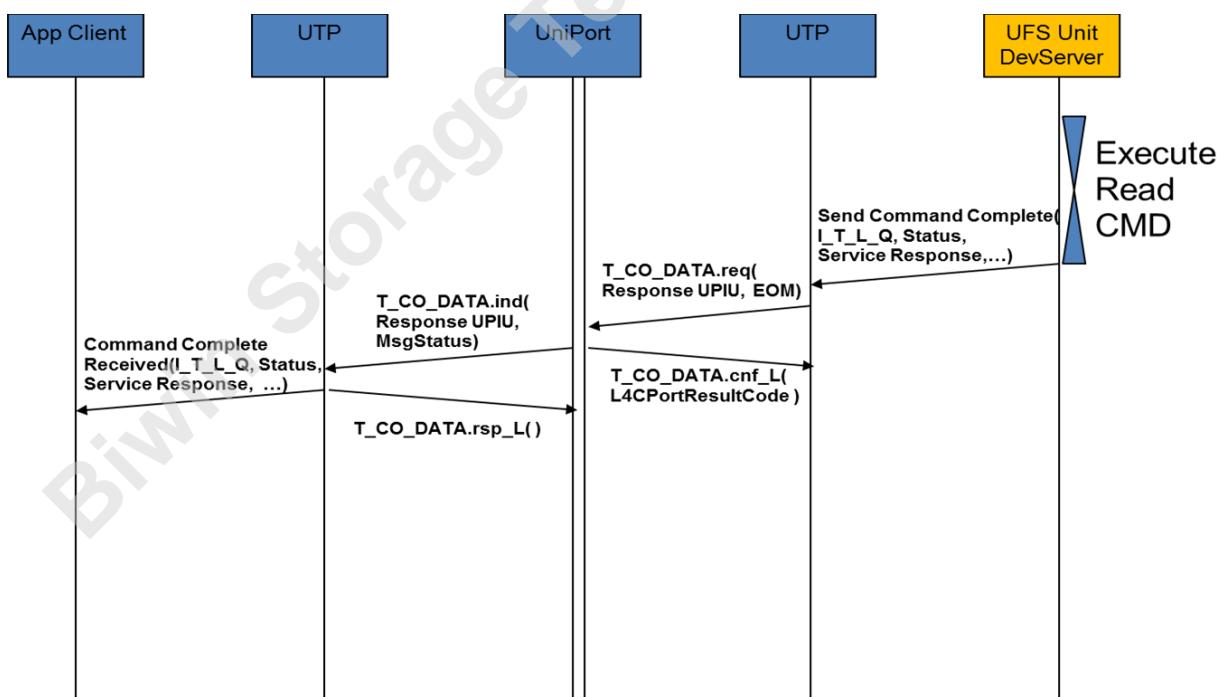
Argument	Implementation
I_T_L_Q Nexus	I_T_L_Q of the corresponding COMMAND UPIU
Delivery Result	DELIVERY SUCCESSFUL: The data was delivered successfully. DELIVERY FAILURE: A UTP service delivery error occurred while attempting to deliver the data.

2846

2847 10.9.7.2 Data-In Delivered transport protocol service (cont'd)



2848  
2849  
2850 Figure 10.13 — Command + Read Data Phase 1/2



2851  
2852  
2853 Figure 10.14 — Command + Read Data Phase 2/2

2854 **10.9.7.3 Receive Data-Out transport protocol service**

2855 A device server uses the Receive Data-Out transport protocol service request to request that a UFS target  
2856 port receives data.

- 2857 • Receive Data-Out (IN (I\_T\_L\_Q Nexus, Application Client Buffer Offset, Request Byte Count, Device  
2858 Server Buffer))

2859 A device server shall only call Receive Data-Out () during a write operation.

2860 A device server shall not call Receive Data-Out () for a given I\_T\_L\_Q nexus after a Send Command  
2861 Complete () has been called for that I\_T\_L\_Q nexus or after a Task Management Function Executed ()  
2862 has been called for a task management function that terminates that command (e.g., an ABORT TASK).

2863 **Table 10.70 — 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

2864 **10.9.7.4 Data-Out Received transport protocol service**

2865 A UFS target port uses the Data-Out Received transport protocol service indication to notify a device  
2866 server that it has received data.

- 2867 • Data-out Received (IN (I\_T\_L\_Q Nexus, Delivery Result))

2868 **Table 10.71 — 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.

2869

2870 10.9.7.4 Data-Out Received transport protocol service (cont'd)

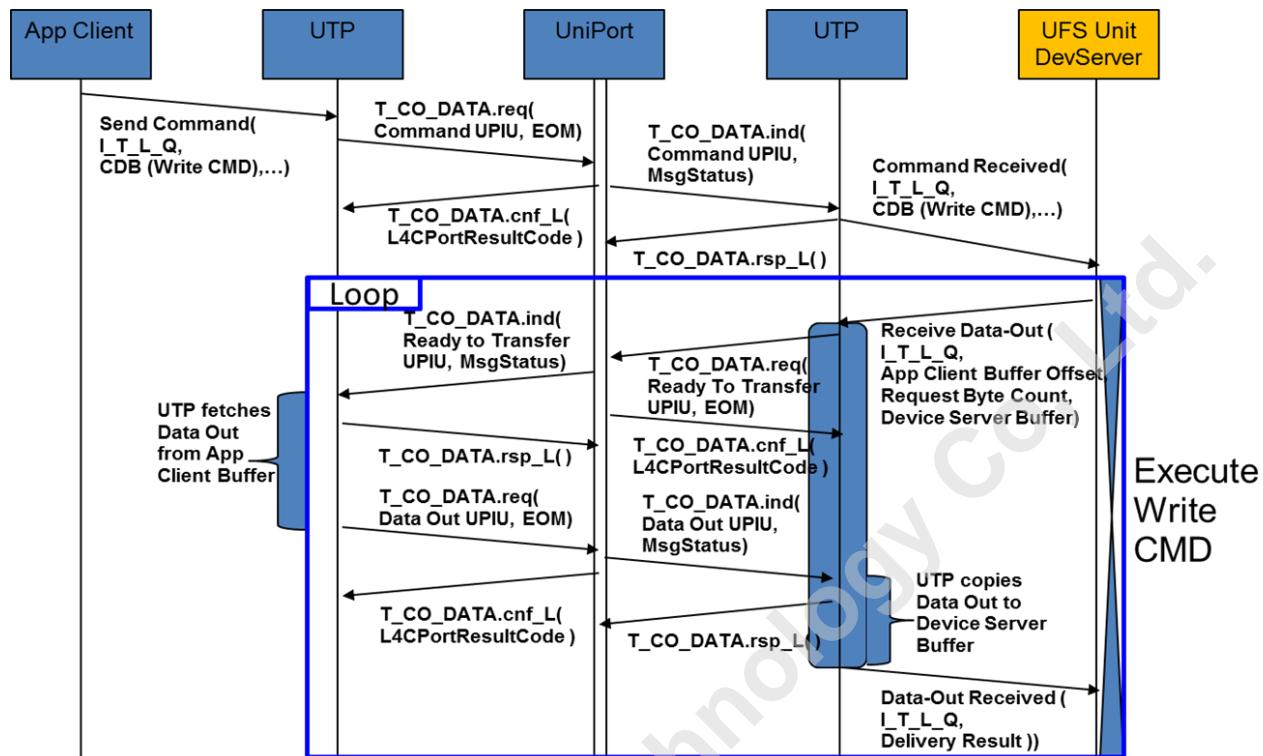


Figure 10.15 — Command + Write Data Phase 1/2

2871  
2872  
2873

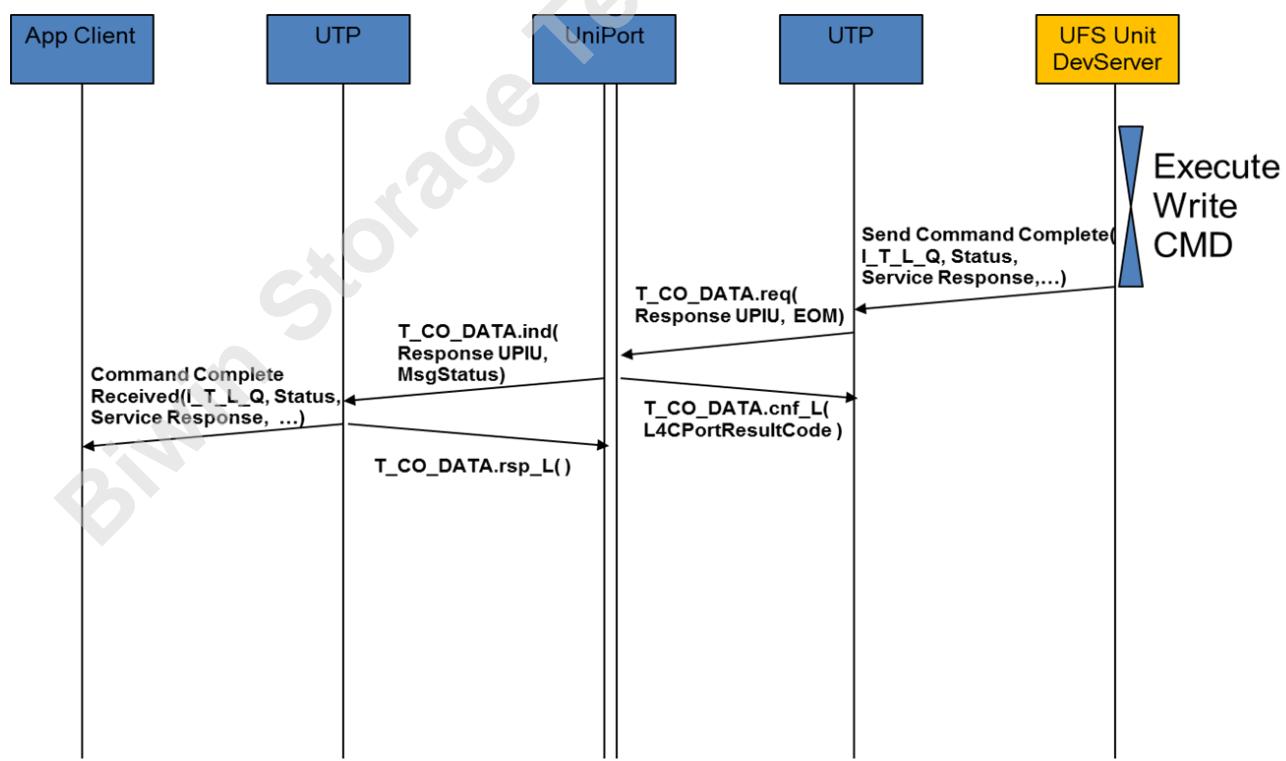


Figure 10.16 — Command + Write Data Phase 2/2

2874  
2875  
2876

2877 **10.9.8 Task Management Function procedure calls**

2878 An application client requests the processing of a task management function by invoking the SCSI  
2879 transport protocol services:

- 2880 • Service Response = Function name (IN (Nexus), OUT ([Additional Response Information]))

2881 **Table 10.72 — 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

2882 One of the following SCSI transport protocol specific service responses shall be returned:

2883 **Table 10.73 — SCSI transport protocol service responses**

<b>FUNCTION COMPLETE</b>	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.
<b>FUNCTION SUCCEEDED</b>	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)
<b>FUNCTION REJECTED</b>	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.
<b>INCORRECT LOGICAL UNIT NUMBER</b>	A task router response indicating that the function requested processing for an incorrect logical unit number.
<b>SERVICE DELIVERY OR TARGET FAILURE</b>	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.

2884

2885 **10.9.8.1 ABORT TASK**

2886 This function shall be supported by all logical units.

2887 The task manager shall abort the specified command, if it exists. Previously established conditions,  
2888 including mode parameters, and reservations shall not be changed by the ABORT TASK function.

2889 A response of FUNCTION COMPLETE shall indicate that the command was aborted or was not in the  
2890 task set. In either case, the SCSI target device shall guarantee that no further requests or responses are  
2891 sent from the command.

2892 If the processing of the task that is requested to be aborted requires Data-Out data transfer, then Target  
2893 device shall wait until it receives all DATA OUT UPIUs related to any outstanding READY TO  
2894 TRANSFER UPIUs before sending the task management response.

2895 All SCSI transport protocol standards shall support the ABORT TASK task management function.

2896 Service Response = ABORT TASK (IN ( I\_T\_L\_Q Nexus ))

2897 **10.9.8.2 ABORT TASK SET**

2898 This function shall be supported by all logical units.

2899 The task manager shall abort all commands in the task set that were received on the specified I\_T\_L  
2900 nexus. Commands received on other I\_T nexuses or in other task sets shall not be aborted. This task  
2901 management function performed is equivalent to a series of ABORT TASK requests.

2902 All pending status and sense data for the commands that were aborted shall be cleared. Other previously  
2903 established conditions, including mode parameters, and reservations shall not be changed by the ABORT  
2904 TASK SET function.

2905 If the processing of one or more tasks in the task set require Data-Out data transfer, then Target device  
2906 shall wait until it receives all DATA OUT UPIUs related to any outstanding READY TO TRANSFER  
2907 UPIUs before sending the task management response.

2908 All SCSI transport protocol standards shall support the ABORT TASK SET task management function.

2909 Service Response = ABORT TASK SET (IN ( I\_T\_L Nexus ))

2910 **10.9.8.3 CLEAR TASK SET**

2911 This function shall be supported by all logical units.

2912 The task manager shall abort all commands in the task set.

2913 If the processing of one or more tasks in the task set require Data-Out data transfer, then Target device  
2914 shall wait until it receives all DATA OUT UPIUs related to any outstanding READY TO TRANSFER  
2915 UPIUs before sending the task management response.

2916 All pending status and sense data for the task set shall be cleared. Other previously established conditions,  
2917 including mode parameters, and reservations shall not be changed by the CLEAR TASK SET function.

2918 All SCSI transport protocol standards shall support the CLEAR TASK SET task management function.

2919 Service Response = CLEAR TASK SET (IN ( I\_T\_L Nexus ))

2921 **10.9.8.4 LOGICAL UNIT RESET**

2922 This function shall be supported by all logical units.

2923 Before returning a FUNCTION COMPLETE response, the logical unit shall perform the logical unit reset  
2924 functions.

2925 If the processing of one or more tasks in the logical unit requires Data-Out data transfer, then Target  
2926 device shall wait until it receives all DATA OUT UPIUs related to any outstanding READY TO  
2927 TRANSFER UPIUs before sending the task management response.

2928 All SCSI transport protocol standards shall support the LOGICAL UNIT RESET task management  
2929 function.

2930 Service Response = LOGICAL UNIT RESET (IN ( I\_T\_L Nexus ))

2931 **10.9.8.5 QUERY TASK**

2932 UFS transport protocols shall support QUERY TASK.

2933 The task manager in the specified logical unit shall:

2934 1) If the specified command is present in the task set, then return a service response set to FUNCTION  
2935 SUCCEEDED; or

2936 2) If the specified command is not present in the task set, then return a service response set to  
2937 FUNCTION COMPLETE.

2938 Service Response = QUERY TASK (IN ( I\_T\_L\_Q Nexus ))

2939 **10.9.8.6 QUERY TASK SET**

2940 UFS transport protocols shall support QUERY TASK SET.

2941 The task manager in the specified logical unit shall:

2942 1) If there is any command present in the task set specified I\_T\_L nexus, then return a service response  
2943 set to FUNCTION SUCCEEDED; or

2944 2) If there is no command present in the task set specified I\_T nexus, then return a service response set  
2945 to FUNCTION COMPLETE.

2946 Service Response = QUERY TASK SET (IN ( I\_T\_L Nexus ))

2947 **10.9.8.7 Task Management SCSI Transport Protocol Services**

2948 UFS standard shall define the SCSI transport protocol specific requirements for implementing the Send  
2949 Task Management Request request, the Task Management Request Received indication, the Task  
2950 Management Function Executed response, and the Received Task Management Function Executed  
2951 confirmation SCSI transport protocol services.

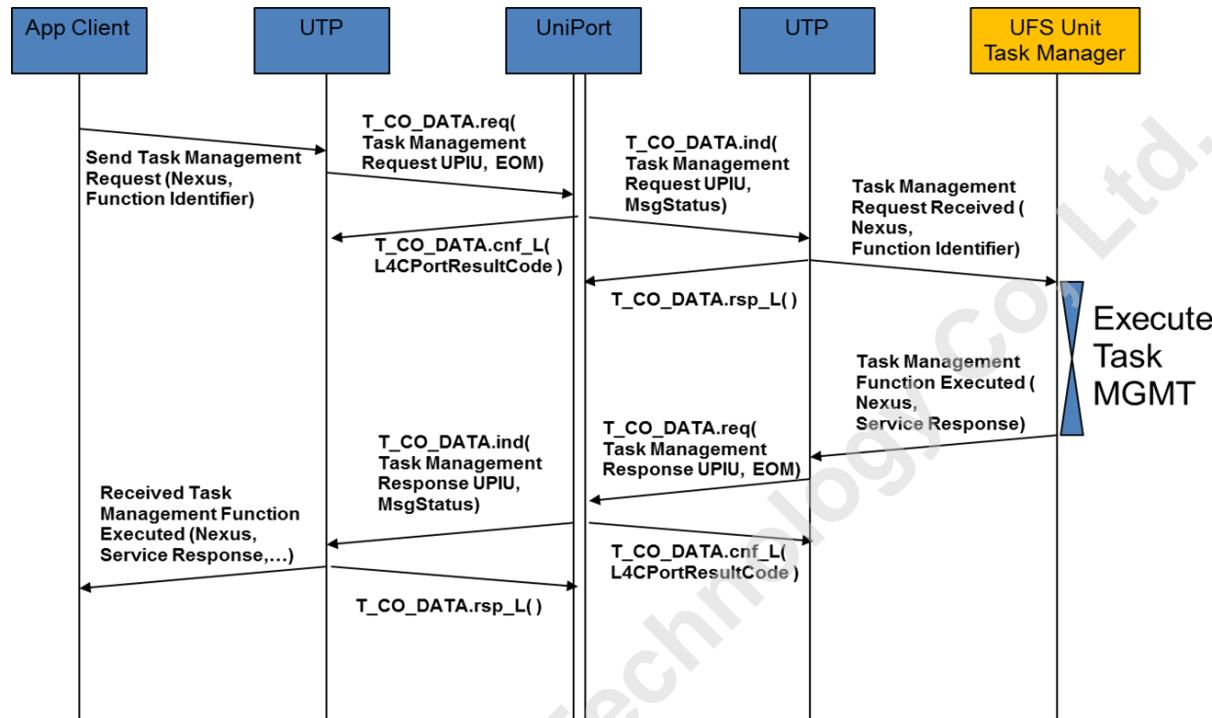
2952 A SCSI transport protocol standard may specify different implementation requirements for the Send Task  
2953 Management Request request SCSI transport protocol service for different values of the Function  
2954 Identifier argument.

2955 All SCSI initiator devices shall implement the Send Task Management Request request and the Received  
2956 Task Management Function Executed confirmation SCSI transport protocol services as defined in the  
2957 applicable SCSI transport protocol standards.

2958

2959 **10.9.8.7 Task Management SCSI Transport Protocol Services (cont'd)**

2960 All SCSI target devices shall implement the Task Management Request Received indication and the Task  
2961 Management Function Executed response SCSI transport protocol services as defined in the applicable  
2962 SCSI transport protocol standards.



2963  
2964 **Figure 10.17 — Task Management Function**  
2965

2966 **10.9.8.8 Send Task Management Request SCSI transport protocol service request**

2967 An application client uses the Send Task Management Request SCSI transport protocol service request to  
2968 request that a SCSI initiator port send a task management function.

2969 Send Task Management Request SCSI transport protocol service request:

2970 Send Task Management Request (IN ( Nexus, Function Identifier ))

2971 **Table 10.74 — 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.

2972 **10.9.8.9 Task Management Request Received SCSI transport protocol service indication**

2973 A task router uses the Task Management Request Received SCSI transport protocol service indication to  
2974 notify a task manager that it has received a task management function.

2975 Task Management Request Received SCSI transport protocol service indication:

2976 Task Management Request Received (IN ( Nexus, Function Identifier ))

2977 **Table 10.75 — 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.

2978 **10.9.8.10 Task Management Function Executed SCSI transport protocol service response**

2979 A task manager uses the Task Management Function Executed SCSI transport protocol service response  
2980 to request that a SCSI target port transmit task management function executed information.

2981 Task Management Function Executed SCSI transport protocol service response:

2982 Task Management Function Executed (IN (Nexus, Service Response, [Additional Response  
2983 Information]))

2984 **Table 10.76 — 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

2985 **10.9.8.11 Received Task Management Function Executed SCSI transport protocol service**  
2986 **confirmation**

2987 A SCSI initiator port uses the Received Task Management Function Executed SCSI transport protocol  
2988 service confirmation to notify an application client that it has received task management function  
2989 executed information.

2990 Received Task Management Function Executed SCSI transport protocol service confirmation:

2991 Received Task Management Function Executed (IN (Nexus, Service Response, [Additional Response  
2992 Information]))

2993 **Table 10.77 — Received Task Management Function Executed SCSI transport protocol service**  
2994 **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

## 2996 10.9.9 Query Function transport protocol services

2997 UFS defines Query Function to get/set UFS-specific device-level registers and parameters (not part of  
2998 SCSI definition).

### 2999 10.9.9.1 Send Query Request UFS transport protocol service

3000 An application client uses the Send Query Request UFS transport protocol service request to request that  
3001 a UFS initiator port send a Query Request function.

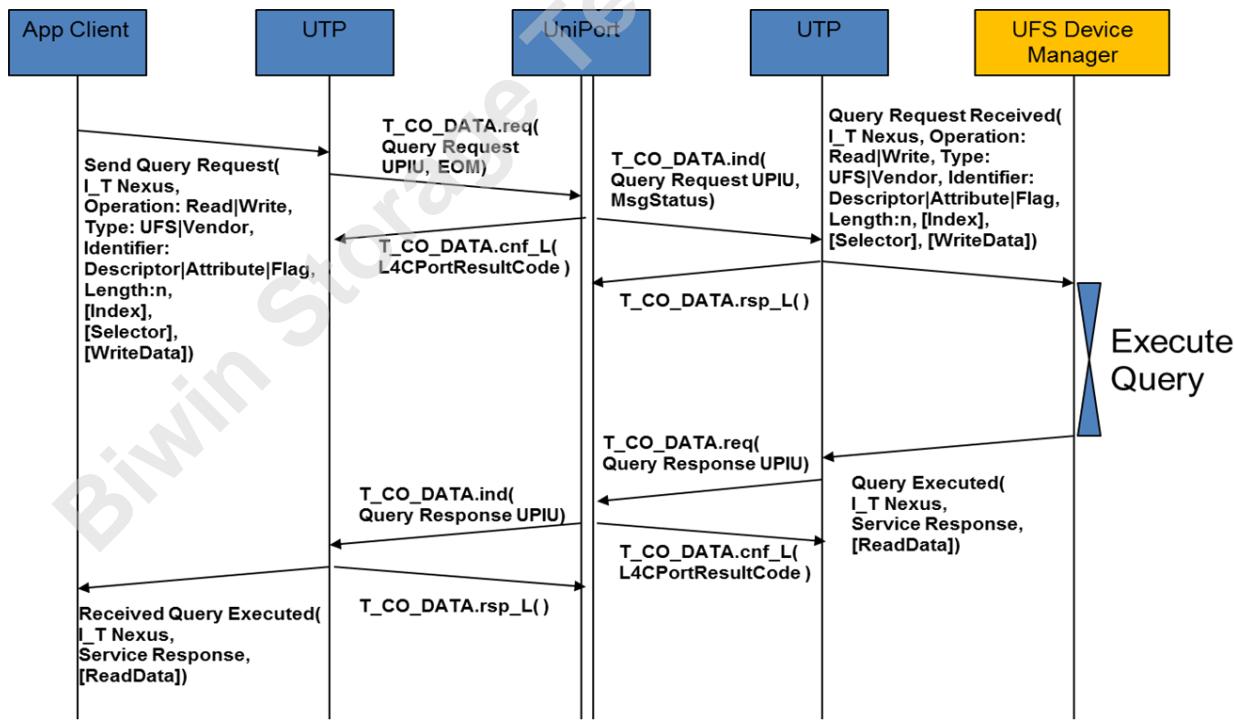
3002 Send Query Request UFS transport protocol service request:

3003 Send Query Request(I\_T Nexus, Operation: Read|Write, Type: UFS|Vendor, Identifier:  
3004 Descriptor|Attribute|Flag, Length: n, [Index], [Selector], [WriteData])

3005 **Table 10.78 — 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

3006



3007  
3008  
3009 **Figure 10.18 — UFS Query Function**

3010 **10.9.9.2 Query Request Received UFS transport protocol service indication**

3011 A UFS target port uses the Query Request Received UFS transport protocol service indication to notify a  
3012 UFS device manager that it has received a query function.

3013 Query Request Received UFS transport protocol service indication:

3014 Query Request Received(I\_T Nexus, Operation: Read|Write, Type: UFS|Vendor,  
3015 Identifier:Descriptor|Attribute|Flag,Length:n, [Index], [Selector], [WriteData])

3016 **Table 10.79 — 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

3017 **10.9.9.3 Query Function Executed UFS transport protocol service response**

3018 A device manager uses the Query Function Executed UFS transport protocol service response to request  
3019 that a UFS target port transmit query function executed information.

3020 Query Function Executed UFS transport protocol service response:

3021 Query Executed(I\_T Nexus, Service Response,[ReadData])

3022 **Table 10.80 — 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

3023

3024 **10.9.9.4 Received Query Function Executed UFS transport protocol service confirmation**

3025 A UFS initiator port uses the Received Query Function Executed UFS transport protocol service  
3026 confirmation to notify an application client that it has received task management function executed  
3027 information.

3028 Received Query Function Executed UFS transport protocol service confirmation:

3029 Received Query Executed(I\_T Nexus, Service Response,[ReadData])

3030 **Table 10.81 — 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

3031

---

3032    **11    UFS Application (UAP) Layer – SCSI Commands**

---

3033    **11.1    Universal Flash Storage Command Layer (UCL) Introduction**

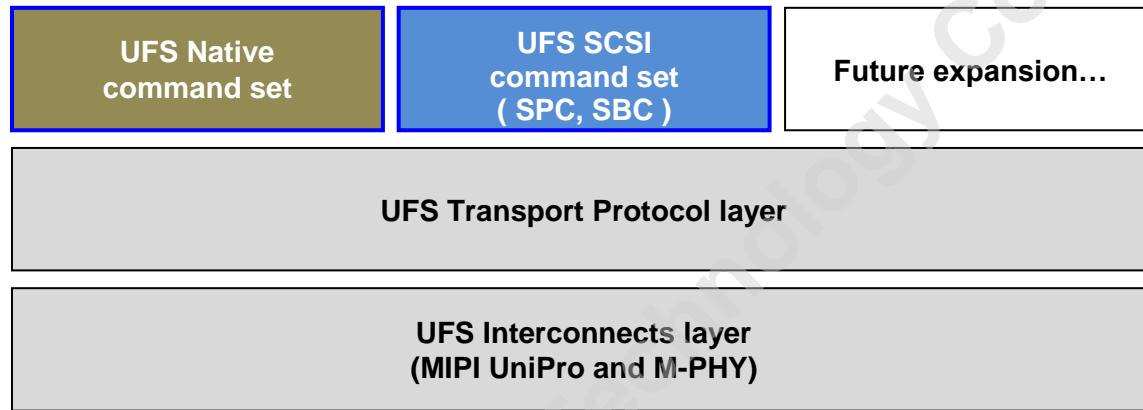
3034    This chapter defines the mandatory commands set supported by the UFS device.

3035    Commands may belong to the UFS Native command set or to the UFS SCSI command set.

3036    This version of the standard does not define UFS native commands. These command set may be defined in  
3037    the future to support specific flash storage or UFS native basic needs.

3038    The UFS SCSI command set (USC) consists of a selection of commands from SCSI Primary Commands  
3039    [SPC], and SCSI Block Commands [SBC]. Both command types share similar command descriptor block  
3040    (CDB) format.

3041



3042    **Figure 11.1 — UFS Command Layer**

3043    **11.1.1    The Command Descriptor Block (CDB)**

3044    SCSI commands are communicated by sending the SCSI Command Descriptor Block (CDB) to the  
3045    device. There are only fixed length CDB format for UFS, unlike SCSI which has additional Variable  
3046    Length CDB format.

3047    All UFS CDBs shall have an OPERATION CODE field as their first byte and these values shall be  
3048    defined by each SCSI and UFS. Detail SCSI CDB usages and structure are defined in [SPC], section 4.3,  
3049    The Command Descriptor Block (CDB).

3050    The General Common CDB fields are defined in [SPC], section 4.3.5, Common CDB fields.

3051    **Operation code**

3052    The first byte of a SCSI and USC CDB shall contain an operation code identifying the operation being  
3053    requested by the CDB.

3054    The OPERATION CODE of the CDB contains a GROUP CODE field and COMMAND CODE field.  
3055    The GROUP CODE field provides eight groups of command codes and the COMMAND CODE provides  
3056    thirty-two command codes in each group, see [SPC] for further details.

3057    **11.2    Universal Flash Storage Native Commands (UNC)**

3058    The UFS Native commands are not defined in this version of the standard, they may be defined in future  
3059    versions if needed.

3061 **11.3 Universal Flash Storage SCSI Commands**

3062 The Basic Universal Flash Storage (UFS) SCSI commands are compatible with SCSI Primary Commands  
3063 - 4 [SPC] and SCSI Block Commands - 3 [SBC].

3064 If enabled (bLUEnable = 01h), each logical unit shall support the commands defined as mandatory in  
3065 Table 11.1.

3066

**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	M
READ CAPACITY (10)	25h	M
READ CAPACITY (16)	9Eh	M
REPORT LUNS	A0h	M
REQUEST SENSE	03h	M
SECURITY PROTOCOL IN <sup>(1)</sup>	A2h	M
SECURITY PROTOCOL OUT <sup>(1)</sup>	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		
NOTE 1 SECURITY PROTOCOL IN command and SECURITY PROTOCOL OUT command are supported by the RPMB well known logical unit.		

3067

3068 **11.3.1 General information about SCSI commands in UFS**

3069 The remaining part of this section describes the SCSI commands used in UFS devices. A dedicated  
3070 paragraph for each command provides: CDB table, brief command description, relevant command fields,  
3071 details about mandatory and optional features, and some other fundamental information.

3072 Fields that are not supported by UFS should be set to zero, and are documented using the notation  
3073 “= 00h” (e.g., RDPROTECT = 000b). The device may ignore values in fields that are not supported by  
3074 UFS.

3075 NOTE The values enclosed in parenthesis are defined in SCSI standards and are not UFS specific  
3076 (e.g., OPERATION CODE (12h)).

3077 In the following some information that apply to all SCSI commands used in UFS.

3078 **CONTROL**

3079 The CONTROL byte is present in several CDB and it is defined in [SAM]. The CONTROL byte is not  
3080 used in this standard: the CONTROL byte should be set to zero and it shall be ignored by UFS device. No  
3081 vendor specific interpretation and Normal ACA are assumed.

3082 **Auto contingent allegiance (ACA)**

3083 Establishing an ACA condition the application client may request that the device server alter command  
3084 processing when a command terminates with a CHECK CONDITION status.

3085 UFS device does not support ACA.

3086 **11.3.2 INQUIRY Command**

3087 The INQUIRY command (see Table 11.2) is a request for information regarding the logical units and  
3088 UFS target device be sent to the application client. Refer to [SPC] for more details regarding the  
3089 INQUIRY command.

3090 **Table 11.2 — INQUIRY command**

Bit Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (12h)							
1	Reserved						Obsolete	EVPD
2	PAGE CODE							
3	(MSB) ALLOCATION LENGTH*							
4								
5	CONTROL = 00h							

\* Allocation Length = Number of response bytes to return

3091 **11.3.2.1 VITAL PRODUCT DATA**

3092 When EVPD = 1, the device server shall return the vital product data specified by the PAGE CODE field  
3093 as defined in [SPC]. Support for all vital product data except Mode Page Policy VPD is optional for UFS.  
3094 Mode Page Policy VPD shall be supported by UFS device to provide information about Mode pages  
3095 which are applicable to Device level or logical unit level. See [SPC] for data format definition of Mode  
3096 Page Policy VPD page.

### 3097 11.3.2.2 STANDARD INQUIRY DATA

3098 When EVPD = 0 and Page Code = 0, the Standard INQUIRY DATA is responded to INQUIRY  
3099 command. The standard INQUIRY data format is shown on Table 11.3, INQUIRY data shall contain at  
3100 least 36 bytes. Table 11.4 defines the INQUIRY response data for UFS.

3101 The INQUIRY command requests that information regarding the logical unit and SCSI target device be  
3102 sent to the Application Client.

3103 The INQUIRY command may be used by an Application Client after a hard reset or power on condition  
3104 to determine information about the device for system configuration. If a INQUIRY command is received  
3105 with a pending UNIT ATTENTION condition (i.e., before the device server reports CHECK  
3106 CONDITION status), the device server shall perform the INQUIRY command.

3107 INQUIRY information is returned in standard INQUIRY response data structure (see Table 11.3).

- 3108 • Client requests number of bytes to return
- 3109 • First 36 bytes are defined for UFS as standard
- 3110 • Requesting zero byte is valid
- 3111 • Requesting 36 bytes will result in the device returning the complete record
- 3112 • Requesting more bytes than defined will result in truncation to max number device has defined

3113 The Device Server should process the INQUIRY command even when an error occurs that prohibits  
3114 normal command completion

- 3115 • When in UNIT ATTENTION
- 3116 • During other conditions that may affect medium access

3117 The Command CDB shall be sent in a single COMMAND UPIU

3118 **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)							

### 3119    11.3.2.3 Inquiry Command Data Response

- 3120    • Data returned from an INQUIRY command will be transferred to the Application Client in a single  
3121    DATA IN UPIU
- 3122    • The Device Server will transfer the Response Data in the Data Segment area of a DATA IN UPIU
- 3123    • An Allocation Length of zero specifies that no data shall be transferred. This condition shall not be  
3124    considered as an error, and DATA IN UPIU shall not be generated.
- 3125    • No DATA IN UPIU will be transferred if an error occurs
- 3126    • For Standard INQUIRY Data,
  - 3127      ○ the Device Server shall return a number of bytes equal to the minimum between 36 and the value  
3128      specified in the Allocation Length.
  - 3129      ○ Standard INQUIRY Response Data is shown in Table 11.4.

### 3130    11.3.2.4 Inquiry Response Data

3131    **Table 11.4 — Standard INQUIRY Response Data**

<b>Byte</b>	<b>Bit</b>	<b>Value</b>	<b>Description</b>
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.			

3132    The 4-byte PRODUCT REVISION LEVEL in the Inquiry Response Data shall identify the firmware  
3133    version of the UFS device and shall be uniquely encoded for any firmware modification implemented by  
3134    the UFS device vendor.

### 3135 11.3.2.5 Inquiry Command Status Response

- 3136 • STATUS response will be sent in a single RESPONSE UPIU
- 3137 • If the requested data is successfully transferred, the INQUIRY command will terminate with a
- 3138 STATUS response of GOOD
- 3139 • If the unit is not ready to accept a new command (e.g., still processing previous command) a
- 3140 STATUS response of BUSY will be returned
- 3141 • When the INQUIRY command fails a STATUS response of CHECK CONDITION will be returned
- 3142 along with an appropriate SENSE KEY, such as
  - 3143 ○ ILLEGAL REQUEST (range or CDB errors)
  - 3144 ○ HARDWARE ERROR (hardware failure)
- 3145 • Will not fail due to a pending UNIT ATTENTION condition

### 3146 11.3.3 MODE SELECT (10) Command

3147 MODE SELECT command provides a means for the application client to specify medium, logical unit, or

3148 peripheral device parameters to the device server.

- 3149 • Parameters are managed by means of parameter pages called mode pages
  - 3150 ○ UFS devices shall support the following mode pages
    - 3151 ■ CONTROL, CACHING, READ-WRITE ERROR RECOVERY
  - 3152 ○ UFS devices may support vendor specific mode pages
  - 3153 ○ See 11.4 for further details.
- 3154 • Writes parameters to one or more mode pages in a list
  - 3155 ○ The Application Client can specify a single, multiple or all supported pages in a single command
- 3156 • Complementary command to the MODE SENSE command

3157 The Command CDB shall be sent in a single COMMAND UPIU

3158 **Table 11.5 — MODE SELECT (10) command**

Bit Byte	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												

3159

3160 **11.3.3.1 Mode Select Command Parameters**

3161 **Table 11.6 — Mode Select Command Parameters**

Byte	Bit	Description
1	4:4	<b>PF:</b> 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	<b>SP:</b> 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	<b>PARAMETER LIST LENGTH:</b> 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.

3162 **11.3.3.2 Mode Select Command Data Transfer**

3163 The Device Server requests to transfer the mode parameter list from the Application Client data-out  
3164 buffer by issuing one or more READY TO TRANSFER UPIU's (RTT).

3165 The mode parameter list is delivered in one or more segments sending DATA OUT UPIU packets, as  
3166 indicated in the RTT requests,

3167 Zero or an incomplete number of segments may be requested, if an error occurs before the entire data  
3168 transfer is complete.

3169 Mode parameters are changed as specified in the received mode parameter list if the command completes  
3170 successfully.

3171 See 11.4.1.2, Mode parameter list format, for details about the mode parameter list.

3172

3173   **11.3.3.3 Mode Select Command Status Response**

- 3174   • STATUS response will be sent in a single RESPONSE UPIU
- 3175   • If the requested data is successfully transferred and written, the MODE SELECT command will
- 3176    terminate with a STATUS response of GOOD
- 3177   • If the unit is not ready to accept a new command (e.g., still processing previous command) a
- 3178    STATUS response of BUSY will be returned
- 3179   • Failure can occur for numerous reasons. When the MODE SELECT command fails a STATUS
- 3180    response of CHECK CONDITION will be returned along with an appropriate SENSE KEY, such as
  - 3181      ○ ILLEGAL REQUEST (CDB or parameter errors)
  - 3182      ○ MEDIUM ERROR (medium failure, ECC, etc.)
  - 3183      ○ HARDWARE ERROR (hardware failure)
  - 3184      ○ UNIT ATTENTION (reset, power-on, etc.)

3185   **11.3.4 MODE SENSE (10) Command**

3186   The MODE SENSE command provides a means for a Device Server to report parameters to an

3187   Application Client

- 3188   • Parameters are managed by means of parameter pages called Mode Pages
  - 3189      ○ UFS devices shall support the following mode pages
    - 3190       ■ CONTROL, READ-WRITE ERROR RECOVERY, CACHING
  - 3191      ○ UFS devices may support vendor specific mode pages
  - 3192      ○ See 11.4 for further details
- 3193   • Reads parameter pages in a list
  - 3194      ○ The Application Client may request any one or all of the supported pages from the Device
  - 3195      Server
    - 3196       ○ If all pages requested, they will be returned in ascending page order
- 3197   • Mode Sense returns DEVICE-SPECIFIC PARAMETER in header
  - 3198       ○ See paragraph 11.4.1.3, Mode Parameter Header for details.
- 3199   • Complementary command to the MODE SELECT command

3200   The Command CDB shall be sent in a single COMMAND UPIU

3201

3202 **11.3.4 MODE SENSE (10) Command (cont'd)**

3203 **Table 11.7 — MODE SENSE (10) command**

Byte	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														
5	Reserved													
6														
7	(MSB)	ALLOCATION LENGTH			(LSB)									
8														
9	CONTROL = 00h													

3204

3205 **11.3.4.1 Mode Sense Command Parameters**

3206

3207 **Table 11.8 — Mode Sense Command Parameters**

Byte	Bit	Description
2	7:6	<b>PC:</b> 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	<b>PAGE CODE:</b> Specifies which mode page to return. The Page and Subpage code specify the page to return.
3	7:0	<b>SUBPAGE CODE:</b> Specifies which subpage mode page to return
7:8	7:0	<b>ALLOCATION LENGTH:</b> Specifies the maximum number of bytes the Device Server will transfer to the Application Client

3208

3209 **11.3.4.2 Page Control Function**

3210 **Table 11.9 — Page Control Function**

<b>Page Control (PC)</b>	<b>Description</b>	<b>Use</b>
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.

3211 **11.3.4.3 Mode Sense Command Data Transfer**

3212 The Device Server will transfer up to Allocation Length number of data bytes of Mode Parameter Data to  
3213 the Application Client.

- 3214 • Less than Allocation Length will be transferred if Device Server contains less bytes

3215 The Device Server will transfer the data as indicated by the Mode Parameter Layout

- 3216 • Header (8 bytes)

- 3217 • Block Descriptor (0 bytes for UFS)

- 3218 • Page Data (N bytes, dependent up page requested)

3219 Data will be transferred from the Device Server to the Application Client via a series of DATA IN  
3220 UPIU's

- 3221 • The data transferred from the Device Server will be contained within the Data Segment of the DATA  
3222 IN UPIU

3223 Zero or an incomplete number of DATA IN UPIU's will be transferred if an error occurs before the entire  
3224 data transfer is complete.

3225 **11.3.4.4 Mode Sense Command Status Response**

- 3226 • STATUS response will be sent in a single RESPONSE UPIU
- 3227 • If the requested data is successfully transferred and written, the MODE SENSE command will  
3228 terminate with a STATUS response of GOOD
- 3229 • If the unit is not ready to accept a new command (e.g., still processing previous command) a  
3230 STATUS response of BUSY will be returned
- 3231 • Failure occurs when requesting a page or subpage that is not supported. A STATUS response of  
3232 CHECK CONDITION will be returned with a SENSE KEY indicating ILLEGAL REQUEST
- 3233 • Failure can occur for numerous reasons. When the MODE SENSE command fails a STATUS  
3234 response of CHECK CONDITION will be returned along with an appropriate SENSE KEY, such as
  - 3235 ○ ILLEGAL REQUEST (CDB errors)
  - 3236 ○ HARDWARE ERROR (hardware failure)
  - 3237 ○ UNIT ATTENTION (reset, power-on, etc.)

3238 **11.3.5 READ (6) Command**

3239 The READ (6) command (see Table 11.10) requests that the Device Server read from the medium the  
3240 specified number of logical block(s) and transfer them to the Application Client.

3241 The Command CDB shall be sent in a single COMMAND UPIU.

3242 **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													

3243 **11.3.5.1 Read (6) Command Parameters**

- 3244 • LOGICAL BLOCK ADDRESS: Address of first block  
3245 • TRANSFER LENGTH: Number of contiguous logical blocks of data that shall be read and  
3246 transferred. A transfer length of zero specifies that 256 logical blocks shall be read. Any other value  
3247 specifies the number of logical blocks that shall be read.

3248 **11.3.5.2 Read (6) Command Data Transfer**

- 3249 • The Device Server will read the specified logical block(s) from the medium and transfer them to the  
3250 Application Client in a series of DATA IN UPIU's  
3251 • The data segment of each DATA IN UPIU shall contain an integer number of logical blocks  
3252 • Zero or an incomplete number of DATA IN UPIU's could be transferred if a read error occurs before  
3253 the entire data transfer is complete

3254 **11.3.5.3 Read (6) Command Status Response**

- 3255 • Status response will be sent in a single RESPONSE UPIU  
3256 • If all requested data is successfully read and transferred, the READ command will terminate with a  
3257 STATUS response of GOOD  
3258 • If the unit is not ready to accept a new command (e.g., still processing previous command) a  
3259 STATUS response of BUSY will be returned  
3260 • Failure can occur for numerous reasons. When the READ command fails a STATUS response of  
3261 CHECK CONDITION will be returned along with an appropriate SENSE KEY, such as  
3262     ○ ILLEGAL REQUEST (range or CDB errors)  
3263     ○ MEDIUM ERROR (medium failure, ECC, etc.)  
3264     ○ HARDWARE ERROR (hardware failure)  
3265     ○ UNIT ATTENTION (reset, power-on, etc.)  
3266     ○ etc.

3267 **11.3.6 READ (10) Command**

3268 The READ (10) command (see Table 11.11) requests that the Device Server read from the medium the  
3269 specified number of logical block(s) and transfer them to the Application Client.

3270 The Command CDB shall be sent in a single COMMAND UPIU

3271 **Table 11.11 — READ (10) command**

Bit Byte	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															
9	(LSB)															
	CONTROL = 00h															

3272

- 3273 • The RDPROTECT field is set to zero for UFS.

3274

### 3275 11.3.6.1 Read (10) Command Parameters

- 3276 • DPO = Disable Page Out
  - 3277     “0” = specifies that the retention priority shall be determined by the RETENTION PRIORITY fields
  - 3278     in the Caching mode page
  - 3279     “1” = specifies that the device server shall assign the logical blocks accessed by this command the
  - 3280     lowest retention priority for being fetched into or retained by the cache. A DPO bit set to one
  - 3281     overrides any retention priority specified in the Caching mode page.
- 3282 • FUA: Force Unit Access
  - 3283     ‘0’ = The Device Server may read the logical blocks from the cache and/or the medium.
  - 3284     ‘1’ = The Device Server shall read the logical blocks from the medium. If a cache contains a more
  - 3285     recent version of a logical block, then the device server shall write the logical block to the medium
  - 3286     before reading it.
- 3287 • FUA\_NV is defined per SBC. Since non-volatile cache support is not currently defined in this
- 3288     standard, the FUA\_NV parameter value in the CDB is ignored by the UFS Device Server.
- 3289 • LOGICAL BLOCK ADDRESS: Address of first block
- 3290 • TRANSFER LENGTH: Number of contiguous logical blocks of data that shall be read and
- 3291     transferred. A transfer length of zero specifies that no logical blocks will be read. This condition shall
- 3292     not be considered an error.
- 3293 • 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

3294 In case the GROUP NUMBER is set to a reserved value, the operation shall fail and a status  
3295 response of CHECK CONDITION will be returned along with the sense key set to ILLEGAL  
3296 REQUEST.

### 3297 11.3.6.2 Read (10) Command Data Transfer

- 3298 • The Device Server will read the specified logical block(s) from the medium and transfer them to the
- 3299     Application Client in a series of DATA IN UPIU's
- 3300 • The data segment of each DATA IN UPIU shall contain an integer number of logical blocks
- 3301 • Zero or an incomplete number of DATA IN UPIU's could be transferred if a read error occurs before
- 3302     the entire data transfer is complete.

### 3303 11.3.6.3 Read (10) Command Status Response

- 3304 • Status response will be sent in a single RESPONSE UPIU
- 3305 • If all requested data is successfully read and transferred, the READ command will terminate with a
- 3306     STATUS response of GOOD
- 3307 • If the unit is not ready to accept a new command (e.g., still processing previous command) a
- 3308     STATUS response of BUSY will be returned
- 3309 • Failure can occur for numerous reasons. When the READ command fails a STATUS response of
- 3310     CHECK CONDITION will be returned along with an appropriate SENSE KEY, such as
  - 3311         ○ ILLEGAL REQUEST (range or CDB errors)
  - 3312         ○ MEDIUM ERROR (medium failure, ECC, etc.)
  - 3313         ○ HARDWARE ERROR (hardware failure)
  - 3314         ○ UNIT ATTENTION (reset, power-on, etc.)
  - 3315         ○ etc.

3316 **11.3.7 READ (16) Command**

3317 The READ (16) command (see Table 11.12) requests that the Device Server read from the medium the  
3318 specified number of logical block(s) and transfer them to the Application Client

3319 The Command CDB shall be sent in a single COMMAND UPIU

3320 **Table 11.12 — READ (16) command**

Byte	7	6	5	4	3	2	1	0						
0	OPERATION CODE (88h)													
1	RDPROTCT = 000b		DPO	FUA	Reserved	FUA_NV = 0b	Reserved							
2	(MSB)													
....	LOGICAL BLOCK ADDRESS													
9														
10	(MSB)													
....	TRANSFER LENGTH													
13														
14	Reserved	Reserved	GROUP NUMBER											
15	CONTROL = 00h													

- 3321 • The RDPROTCT field is set to zero for UFS.

3322

3323 **11.3.7.1 Read (16) Command Parameters**

- 3324 • **DPO** = Disable Page Out  
3325     “0” = specifies that the retention priority shall be determined by the RETENTION PRIORITY fields  
3326     in the Caching mode page  
3327     “1” = specifies that the device server shall assign the logical blocks accessed by this command the  
3328     lowest retention priority for being fetched into or retained by the cache. A DPO bit set to one  
3329     overrides any retention priority specified in the Caching mode page.
- 3330 • **FUA**: Force Unit Access  
3331     ‘0’ = The Device Server may read the logical blocks from the cache and/or the medium. ‘1’ = The  
3332     Device Server shall read the logical blocks from the medium. If a cache contains a more recent  
3333     version of a logical block, then the device server shall write the logical block to the medium before  
3334     reading it.
- 3335 • **FUA\_NV** is defined per SBC. Since non-volatile cache support is not currently defined in this  
3336     standard, the FUA\_NV parameter value in the CDB is ignored by the UFS Device Server.
- 3337 • **LOGICAL BLOCK ADDRESS**: Address of first block
- 3338 • **TRANSFER LENGTH**: Number of contiguous logical blocks of data that shall be read and  
3339     transferred. A transfer length of zero specifies that no logical blocks will be read. This condition shall  
3340     not be considered an error.
- 3341 • **GROUP NUMBER**: See Read (10) Command.

3342 **11.3.7.2 Read (16) Command Data Transfer**

- 3343 • The Device Server will read the specified logical block(s) from the medium and transfer them to the  
3344     Application Client in a series of DATA IN UPIU's
- 3345 • The data segment of each DATA IN UPIU shall contain an integer number of logical blocks
- 3346 • Zero or an incomplete number of DATA IN UPIU's could be transferred if a read error occurs before  
3347     the entire data transfer is complete

3348 **11.3.7.3 Read (16) Command Status Response**

- 3349 • Status response will be sent in a single RESPONSE UPIU
- 3350 • If all requested data is successfully read and transferred, the READ command will terminate with a  
3351     STATUS response of GOOD
- 3352 • If the unit is not ready to accept a new command (e.g., still processing previous command) a  
3353     STATUS response of BUSY will be returned
- 3354 • Failure can occur for numerous reasons. When the READ command fails a STATUS response of  
3355     CHECK CONDITION will be returned along with an appropriate SENSE KEY, such as
- 3356     ○ ILLEGAL REQUEST (range or CDB errors)
- 3357     ○ MEDIUM ERROR (medium failure, ECC, etc.)
- 3358     ○ HARDWARE ERROR (hardware failure)
- 3359     ○ UNIT ATTENTION (reset, power-on, etc.)
- 3360     ○ etc.

3361 **11.3.8 READ CAPACITY (10) Command**

3362 The READ CAPACITY (10) command provides a means for the application client to discover the logical  
3363 unit capacity. Refer to [SBC] for more details regarding the READ CAPACITY (10) command.

- 3364 • The READ CAPACITY (10) command requests that the device server transfer 8 bytes of parameter  
3365 data describing the capacity and medium format of the direct-access block device

3366 The Command CDB shall be sent in a single COMMAND UPIU

3367 **Table 11.13 — READ CAPACITY (10) command**

Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (25h)							
1	Reserved							Obsolete = 0b
2	(MSB) LOGICAL BLOCK ADDRESS							
5	(LSB)							
6	Reserved							
7	Reserved							
8	Reserved							PMI = 0b
9	CONTROL = 00h							

- 3368 • PMI = 0 for UFS

- 3369 • Logical Block Address = 0 for UFS

3370

### 3371    11.3.8.1 Read Capacity (10) Data Response

- 3372    • Data returned from a READ CAPACITY (10) command will be transferred to the Application Client  
3373    in a single DATA IN UPIU
- 3374    • The Device Server will transfer 8 bytes of Capacity Data in the Data Segment area of a DATA IN  
3375    UPIU
- 3376    • Data will be returned in the indicated Read Capacity Parameter format described in the following.
- 3377    • No DATA IN UPIU will be transferred if an error occurs

### 3378    11.3.8.2 Read Capacity (10) Parameter Data

3379    **Table 11.14 — Read Capacity (10) Parameter Data**

Bit Byte	7	6	5	4	3	2	1	0
0	(MSB)							
1								
2								
3								(LSB)
4	(MSB)							
5								
6								
7								(LSB)

- 3380    • RETURNED LOGICAL BLOCK ADDRESS: last addressable block on medium under control of  
3381    logical unit (LU)
  - 3382    ○ If the number of logical blocks exceeds the maximum value that is able to be specified in the  
3383    RETURNED LOGICAL BLOCK ADDRESS field, then the device server shall set the  
3384    RETURNED LOGICAL BLOCK ADDRESS field to FFFF FFFFh. The application client should  
3385    then issue a READ CAPACITY (16) command to request that the device server transfer the  
3386    READ CAPACITY (16) parameter data to the data-in buffer.
- 3387    • LOGICAL BLOCK LENGTH IN BYTES: size of block in bytes
  - 3388    ○ For UFS minimum size shall be 4096 bytes

### 3390 11.3.8.3 Read Capacity (10) Status Response

- 3391 • STATUS response will be sent in a single RESPONSE UPIU
- 3392 • If the requested data is successfully transferred, the READ CAPACITY command will terminate
- 3393 with a STATUS response of GOOD
- 3394 • If the unit is not ready to accept a new command (e.g., still processing previous command) a
- 3395 STATUS response of BUSY will be returned
- 3396 • Failure can occur for a number of reasons. When the READ CAPACITY command fails a STATUS
- 3397 response of CHECK CONDITION will be returned along with an appropriate SENSE KEY, such as
- 3398     ○ ILLEGAL REQUEST (range or CDB errors)
- 3399     ○ HARDWARE ERROR (hardware failure)
- 3400     ○ UNIT ATTENTION (reset, power-on, etc.)
- 3401     ○ etc.

### 3402 11.3.9 READ CAPACITY (16) Command

3403 The READ CAPACITY (16) command provides a means for the application client to discover the logical  
3404 unit capacity. Refer to [SBC] for more details regarding the READ CAPACITY (16) command.

- 3405 • The READ CAPACITY (16) command requests that the device server transfer 32 bytes of parameter  
3406 data describing the capacity and medium format of the direct-access block device

3407 The Command CDB shall be sent in a single COMMAND UPIU.

3408 **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													
10	(MSB)												
....	ALLOCATION LENGTH												
13													
14	Reserved							PMI = 0b					
15	CONTROL = 00h												

- 3409 • PMI = 0 for UFS
- 3410 • Logical Block Address = 0 for UFS
- 3411 • Allocation Length = the maximum number of bytes that the application client has allocated for the
- 3412 returned parameter data.
- 3413

3414    **11.3.9.1 Read Capacity (16) Data Response**

- 3415    • Data returned from a READ CAPACITY (16) command will be transferred to the Application Client  
3416    in a single DATA IN UPIU
- 3417    • The Device Server will transfer 32 bytes of Capacity Data in the Data Segment area of a DATA IN  
3418    UPIU
- 3419    • Data will be returned in the indicated Read Capacity Parameter format described in the following.
- 3420    • No DATA IN UPIU will be transferred if an error occurs
- 3421

3422 **11.3.9.2 Read Capacity (16) Parameter Data**

3423 **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								

- 3424 • RETURNED LOGICAL BLOCK ADDRESS: last addressable block on medium under control of logical unit (LU)
- 3425
- 3426 • LOGICAL BLOCK LENGTH IN BYTES: size of block in bytes
  - 3427 ○ For UFS minimum size shall be 4096 bytes
- 3428 • P\_TYPE, PROT\_EN are set to ‘0’ for UFS
- 3429 • The P\_I\_EXPONENT field can be ignored for PROT\_EN = ‘0’
- 3430 • Logical Blocks per Physical Block Exponent (vendor-specific information)
  - 3431 ○ 0: one or more physical blocks per logical block
  - 3432 ○ n>0:  $2^n$  logical blocks per physical block
- 3433 • 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.
- 3434
- 3435 • 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)
- 3436
- 3437
- 3438
- 3439
- 3440
- 3441

3442   **11.3.9.3 Read Capacity (16) Status Response**

- 3443   • STATUS response will be sent in a single RESPONSE UPIU
- 3444   • If the requested data is successfully transferred, the READ CAPACITY command will terminate
- 3445   with a STATUS response of GOOD
- 3446   • If the unit is not ready to accept a new command (e.g., still processing previous command) a
- 3447   STATUS response of BUSY will be returned
- 3448   • Failure can occur for a number of reasons. When the READ CAPACITY command fails a STATUS
- 3449   response of CHECK CONDITION will be returned along with an appropriate SENSE KEY, such as
- 3450    ○ ILLEGAL REQUEST (range or CDB errors)
- 3451    ○ HARDWARE ERROR (hardware failure)
- 3452    ○ UNIT ATTENTION (reset, power-on, etc.)
- 3453    ○ etc.
- 3454

3455 **11.3.10 START STOP UNIT Command**

3456 The START STOP UNIT command requests that the device server change the power condition of the  
3457 logical unit or load or eject the medium.

- Enable or disable the direct-access block device for medium access operations by controlling power

3459 The Command CDB shall be sent in a single COMMAND UPIU.

3460 **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							

3461 IMMED = 0 : Return STATUS (RESPONSE UPIU) after operation is completed

3462 IMMED = 1 : Return STATUS after CDB is decoded

3463 **11.3.10.1 START STOP UNIT Parameters**

3464 Power Condition Modifier shall be set to '0' (reserved) in UFS spec.

3465 Use of the other parameters is defined in the Power Mode section.

3466 **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

3469 **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

3483 **11.3.11 TEST UNIT READY Command**

3484 The TEST UNIT READY command provides a means to check if the logical unit is ready. This is not a  
3485 request for a self-test.

- 3486 • If the logical unit is able to accept an appropriate medium-access command without returning  
3487 CHECK CONDITION status, this command shall return a GOOD status.  
3488 • If the logical unit is unable to become operational or is in a state such that an Application Client  
3489 action (e.g., START UNIT command) is required to make the logical unit ready, the command shall  
3490 be terminated with CHECK CONDITION status, with the sense key set to NOT READY (02h).

3491 The Command CDB shall be sent in a single COMMAND UPIU

3492 **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							

3493 **11.3.11.1 Test Unit Ready Data Response**

- 3494 • The TEST UNIT READY command does not have a data response  
3495 • No DATA IN or DATA OUT UPIU's are transferred

3496 **11.3.11.2 Test Unit Ready Status Response**

- 3497 • STATUS response will be sent in a single RESPONSE UPIU.  
3498 • If the command succeeds without error, the TEST UNIT READY command will terminate with a  
3499 STATUS response of GOOD  
3500 • If the unit is not ready to accept a new command (e.g., still processing previous command) a  
3501 STATUS response of BUSY will be returned  
3502 • Failure can occur for numerous reasons. When the TEST UNIT READY command fails a STATUS  
3503 response of CHECK CONDITION will be returned along with an appropriate SENSE KEY, such as  
3504     ○ ILLEGAL REQUEST (CDB errors)  
3505     ○ HARDWARE ERROR (hardware failure)  
3506     ○ UNIT ATTENTION (reset, power-on, etc.)  
3507     ○ etc.

3508 **11.3.12 REPORT LUNS Command**

3509 The REPORT LUNS command requests that the peripheral device logical unit inventory be sent to the  
3510 Application Client.

- 3511 • The logical unit inventory is a list that shall include the logical unit numbers of all logical units  
3512 accessible to a UFS Application Client
- 3513 • If a REPORT LUNS command is received with a pending unit attention condition (i.e., before the  
3514 device server reports CHECK CONDITION status), the device server shall perform the REPORT  
3515 LUNS command

3516 The Command CDB shall be sent in a single COMMAND UPIU

3517 **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													

3518

3519 **11.3.12.1 Report LUNS Command Parameters**

3520 **Table 11.20 — Report LUNS Command Parameters**

Byte	Bit	Description
2	7:0	<b>SELECT REPORT:</b> Specifies the type of logical unit addresses that shall be reported. The report types available are listed in the Table 11.21. UFS will support all report types.
6:9	7:0	<b>ALLOCATION LENGTH:</b> Specifies the maximum number of bytes of buffer space that the Application Client has allocated for data reception.

3521

3522 **11.3.12.2 Report LUNS Command Select Report Field Values**

3523 **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

3524 NOTE The well known logical units are not included in the list when the SELECT REPORT field is set to zero.

3525 **11.3.12.3 Report LUNS Data Response**

- 3526 • Data returned from a REPORT LUNS command will be transferred to the Application Client in a one  
3527 or more DATA IN UPIU's
- 3528 • Most likely one DATA IN UPIU
- 3529 • The Device Server will transfer less than or equal to Allocation Length data bytes to the Application  
3530 Client.
- 3531 • Less if Device Server has less total data than requested
- 3532 • Data will be returned in the indicated Parameter Data Format described in the following.
- 3533 • Each reportable logical unit will produce 8 bytes of data.

3534

3535 **11.3.12.4 Report LUNS Parameter Data Format**

3536 **Table 11.22 — Report LUNS Parameter Data Format**

Byte	Description
0:3	<b>LUN LIST LENGTH:</b> 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	<b>RESERVED:</b> 0000000h
8:15	<b>FIRST LUN RECORD:</b> 8 byte record that contains the first LUN
...	... next LUN record ...
N-7:N	<b>LAST LUN RECORD:</b> 8 byte record that contains the last LUN

- 3537 • Total List Length = LUN LIST LENGTH + 8 (i.e., 8\*number of LUN's + 8)

- 3538 • Each LUN record is 8 bytes in length

- 3539 • UFS uses two formats:

- 3540     ○ The single level LUN structure using peripheral device addressing method  
3541     ○ The well known logical unit extended addressing format

3542 **11.3.12.5 Report LUNS LUN Addressing Formats**

3543 **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								

- 3544 • Format used for standard Logical Unit addressing

- 3545 • LUN = Logical Unit Number

- 3546     ○ For UFS: 00h <= LUN <= 7Fh

3547     NOTE The expected value is the SCSI LUN and not the LUN field in UPIU.

3548 **11.3.12.5 Report LUNS LUN Addressing Formats (cont'd)**

3549 **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	NULL (0000h)									
4	NULL (0000h)									
5	NULL (0000h)									
6	NULL (0000h)									
7	NULL (0000h)									

- 3550 • Format used for well known logical unit addressing

- 3551 • W-LUN = Well Known Logical Unit Number

- 3552 ○ For UFS: 00h <= W-LUN <= 7Fh

3553 NOTE The expected value is the SCSI LUN and not the LUN field in UPIU.

3554

3555 **11.3.12.6 Report LUNS Status Response**

- 3556 • Status response will be sent in a single RESPONSE UPIU

- 3557 • If all requested data is successfully read and transferred, the REPORT LUNS command will terminate  
3558 with a STATUS response of GOOD

- 3559 • The REPORT LUNS command will succeed when a pending UNIT ATTENTION condition exists

- 3560 • Failure can occur for very few reasons, mainly for illegal values in the CDB. When the REPORT  
3561 LUNS command fails a STATUS response of CHECK CONDITION will be returned along with an  
3562 appropriate SENSE KEY, such as

- 3563 ○ ILLEGAL REQUEST (CDB errors)

3564

3565 **11.3.12.7 UFS LUN Format**

3566 **Table 11.25 — Format of LUN field in UPIU**

7	6	5	4	3	2	1	0
WLUN_ID	UNIT_NUMBER_ID						

3567

3568 The UFS 8-bit LUN field in UPIU supports two types of LUN addressing:

- 3569 • If WLUN\_ID bit = ‘0’ then the UNIT\_NUMBER\_ID field addresses a standard logical unit (LUN)  
3570 • If WLUN\_ID bit = ‘1’ then the UNIT\_NUMBER\_ID field addresses a well known logical unit (W-LUN)

3572 Up to 128 LUN’s and up to 128 W-LUN’s

- 3573 •  $0 \leqslant \text{UNIT\_NUMBER\_ID} \leqslant 127$

3574 The following table defines the logical unit number for UFS well known logical units (WLUN\_ID bit set to ‘1’)

3576 **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

3577

3578 **11.3.13 VERIFY (10) Command**

3579 The VERIFY command requests that the UFS device verify that the specified logical block(s) and range  
3580 on the medium can be accessed.

- 3581 • Logical units that contain cache shall write referenced cached logical blocks to the medium for the  
3582 logical unit before verification

3583 The Command CDB shall be sent in a single COMMAND UPIU.

3584 **Table 11.27 — VERIFY (10) command**

Bit Byte	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															
4	LOGICAL BLOCK ADDRESS														
5															
6	Reserved			GROUP NUMBER = 00000b											
7	(MSB)														
8	VERIFICATION LENGTH														
9	(LSB)														
	CONTROL = 00h														

3586

3587 UFS device is required to support only the value zero for the byte check (BYTCHK) bit. Therefore the  
3588 BYTCHK bit should be set to zero, and the device shall perform a medium verification with no data  
3589 comparison and not transfer any data from the data-out buffer for any mapped LBA specified by the  
3590 command.

3591

3592    **11.3.13.1 Verify Command Parameters**

3593    **Table 11.28 — Verify Command Parameters**

Byte	Bit	Description
2:5	7:0	<b>LOGICAL BLOCK ADDRESS:</b> Address of first block
7:8	7:0	<b>VERIFICATION LENGTH:</b> 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.

3594    **11.3.13.2 Verify Command Data Transfer**

3595    The VERIFY command does not have a data response

- 3596    • No DATA IN or DATA OUT UPIU's are transferred

3597    **11.3.13.3 Verify Command Status Response**

- 3598    • STATUS response will be sent in a single RESPONSE UPIU

3599    • If all requested data is successfully verified against the medium, the VERIFY command will  
3600    terminate with a STATUS response of GOOD

3601    • If the unit is not ready to accept a new command (e.g., still processing previous command) a  
3602    STATUS response of BUSY will be returned

3603    • Other failures can occur for numerous reasons. When the VERIFY command fails a STATUS  
3604    response of CHECK CONDITION will be returned along with an appropriate SENSE KEY, such as:

3605        o ILLEGAL REQUEST (range or CDB errors)

3606        o MEDIUM ERROR (medium failure, ECC, etc.)

3607        o HARDWARE ERROR (hardware failure)

3608        o UNIT ATTENTION (reset, power-on, etc.)

3609        o etc.

3610 **11.3.14 WRITE (6) Command**

3611 The WRITE (6) UFS command requests that the Device Server transfer the specified number of logical  
3612 blocks(s) from the Application Client and write them to the medium.

3613 The Command CDB shall be sent in a single COMMAND UPIU.

3614 **Table 11-29 — WRITE (6) command**

Bit Byte	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											

3615

3616 **11.3.14.1 Write (6) Command Parameters**

- 3617 • LOGICAL BLOCK ADDRESS: Address of first block
- 3618 • TRANSFER LENGTH: Number of contiguous logical blocks of data that shall be transferred and  
3619 written. A transfer length of zero specifies that 256 logical blocks shall be written. Any other value  
3620 specifies the number of logical blocks that shall be written.

3621

3622 **11.3.14.2 Write (6) Command Data Transfer**

3623 The Device Server requests to transfer the specified logical block(s) from the Application Client data-out  
3624 buffer by issuing one or more READY TO TRANSFER UPIU's (RTT).

3625 The data is delivered in one or more segments sending DATA OUT UPIU packets, as indicated in the  
3626 RTT requests. The data contained in DATA OUT UPIU is written.

3627 The number of bytes requested and the Data Buffer Offset field in each RTT shall both be integer  
3628 multiples of the Logical Block Size (bLogicalBlockSize).

3629 The data segment of each DATA OUT UPIU shall contain an integer number of logical blocks.

3630 Zero or an incomplete number of segments may be requested, if an error occurs before the entire data  
3631 transfer is complete.

3632

3633   **11.3.14.3 Write (6) Command Status Response**

- 3634   • STATUS response will be sent in a single RESPONSE UPIU
- 3635   • If all requested data is successfully transferred and written, the WRITE command will terminate with  
3636    a STATUS response of GOOD
- 3637   • If the logical blocks are transferred directly to a cache then the Device Server may complete the  
3638    command with a GOOD status prior to writing the logical blocks to the medium
- 3639   • If the unit is not ready to accept a new command (e.g., still processing previous command) a  
3640    STATUS response of BUSY will be returned
- 3641   • Failure can occur for numerous reasons. When the WRITE command fails a STATUS response of  
3642    CHECK CONDITION will be returned along with an appropriate SENSE KEY, such as
- 3643       ○ ILLEGAL REQUEST (range or CDB errors)
- 3644       ○ MEDIUM ERROR (medium failure, ECC, etc.)
- 3645       ○ HARDWARE ERROR (hardware failure)
- 3646       ○ UNIT ATTENTION (reset, power-on, etc.)
- 3647       ○ DATA PROTECT (permanent, power-on, secure write protect, etc.)
- 3648       ○ etc.
- 3649

3650 **11.3.15 WRITE (10) Command**

3651 The WRITE (10) UFS command requests that the Device Server transfer the specified number of logical  
3652 blocks(s) from the Application Client and write them to the medium.

3653 The Command CDB shall be sent in a single COMMAND UPIU.

3654 **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)		TRANSFER LENGTH							
8										
9	CONTROL = 00h									

3655

- 3656 • The WRPROTECT field is set to zero for UFS.

3657

3658    **11.3.15.1 Write (10) Command Parameters**

- 3659    • **DPO:** Disable Page Out  
3660       “0” = specifies that the retention priority shall be determined by the RETENTION PRIORITY fields  
3661       in the Caching mode page  
3662       “1” = specifies that the device server shall assign the logical blocks accessed by this command the  
3663       lowest retention priority for being fetched into or retained by the cache. A DPO bit set to one  
3664       overrides any retention priority specified in the Caching mode page.
- 3665    • **FUA:** Force Unit Access  
3666       ○ ‘0’ = The Device Server shall write the logical blocks to the cache and/or the medium.  
3667       ○ ‘1’ = The Device Server shall write the logical blocks to the medium, and shall not complete the  
3668       command with GOOD status until all the logical blocks have been written on the medium without  
3669       error.
- 3670    • FUA\_NV is defined per SBC. Since non-volatile cache support is not currently defined in this  
3671    standard, the FUA\_NV parameter value in the CDB is ignored by the UFS Device Server.
- 3672    • **LOGICAL BLOCK ADDRESS:** Address of first block
- 3673    • **TRANSFER LENGTH:** Number of contiguous logical blocks of data that shall be transferred and  
3674    written. A transfer length of zero specifies that no logical blocks will be written. This condition shall  
3675    not be considered an error.
- 3676    • **GROUP NUMBER:** Notifies the Target device that the data has System Data characteristics or linked  
3677    to a ContextID:  
3678

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

3679  
3680    In case the GROUP NUMBER is set to a reserved value, then the operation shall fail and a status  
3681    response of CHECK CONDITION will be returned along the sense key set to ILLEGAL REQUEST.  
3682

3683 **11.3.15.2 Write(10) Command Data Transfer**

3684 The Device Server requests to transfer the specified logical block(s) from the Application Client data-out  
3685 buffer by issuing a series of READY TO TRANSFER UPIU's (RTT).

3686 The data is delivered in one or more segments sending DATA OUT UPIU packets, as indicated in the  
3687 RTT requests. The data contained in DATA OUT UPIU is written.

3688 The number of bytes requested and the Data Buffer Offset field in each RTT shall both be integer  
3689 multiples of the Logical Block Size (bLogicalBlockSize).

3690 The data segment of each DATA OUT UPIU shall contain an integer number of logical blocks.

3691 Zero or an incomplete number of segments may be requested, if an error occurs before the entire data  
3692 transfer is complete.

3693

3694 **11.3.15.3 Write (10) Command Status Response**

3695 • STATUS response will be sent in a single RESPONSE UPIU.

3696 • If all requested data is successfully transferred and written, the WRITE command will terminate with  
3697 a STATUS response of GOOD.

3698 • If the logical blocks are transferred directly to a cache then the Device Server may complete the  
3699 command with a GOOD status prior to writing the logical blocks to the medium.

3700 • If the unit is not ready to accept a new command (e.g., still processing previous command) a  
3701 STATUS response of BUSY will be returned.

3702 • Failure can occur for numerous reasons. When the WRITE command fails a STATUS response of  
3703 CHECK CONDITION will be returned along with an appropriate SENSE KEY, such as

3704     ○ ILLEGAL REQUEST (range or CDB errors)

3705     ○ MEDIUM ERROR (medium failure, ECC, etc.)

3706     ○ HARDWARE ERROR (hardware failure)

3707     ○ UNIT ATTENTION (reset, power-on, etc.)

3708     ○ DATA PROTECT (permanent, power-on, secure write protect, etc.)

3709     ○ etc.

3710

3711 **11.3.16 WRITE (16) Command**

3712 The WRITE (16) UFS command requests that the Device Server transfer the specified number of logical  
3713 blocks(s) from the Application Client and write them to the medium.

3714 The Command CDB shall be sent in a single COMMAND UPIU.

3715 **Table 11.31 — WRITE (16) command**

Bit Byte	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													
....	(LSB)															
13																
14	Reserved <sup>(1)</sup>	Reserved	GROUP NUMBER													
15	CONTROL = 00h															
NOTE 1 Bit 7 of byte 14 shall be ignored.																

3716

- 3717 • The WDPROTECT field is set to zero for UFS.

3718

3719 **11.3.16.1 Write (16) Command Parameters**

- 3720 • **DPO:** Disable Page Out  
3721     “0” = specifies that the retention priority shall be determined by the RETENTION PRIORITY fields  
3722     in the Caching mode page  
3723     “1” = specifies that the device server shall assign the logical blocks accessed by this command the  
3724     lowest retention priority for being fetched into or retained by the cache. A DPO bit set to one  
3725     overrides any retention priority specified in the Caching mode page.
- 3726 • **FUA:** Force Unit Access  
3727     ‘0’ = The Device Server shall write the logical blocks to the cache and/or the medium.  
3728     ‘1’ = The Device Server shall write the logical blocks to the medium, and shall not complete the  
3729     command with GOOD status until all the logical blocks have been written on the medium without  
3730     error.
- 3731 • FUA\_NV is defined per SBC. Since non-volatile cache support is not currently defined in this  
3732     standard, the FUA\_NV parameter value in the CDB is ignored by the UFS Device Server.
- 3733 • **LOGICAL BLOCK ADDRESS:** Address of first block
- 3734 • **TRANSFER LENGTH:** Number of contiguous logical blocks of data that shall be transferred and  
3735     written. A transfer length of zero specifies that no logical blocks will be written. This condition shall  
3736     not be considered an error.
- 3737 • GROUP NUMBER: See Write (10) Command.

3738

3739 **11.3.16.2 Write (16) Command Data Transfer**

3740 The Device Server requests to transfer the specified logical block(s) from the Application Client data-out  
3741 buffer by issuing a series of READY TO TRANSFER UPIU's (RTT).

3742 The data is delivered in one or more segments sending DATA OUT UPIU packets, as indicated in the  
3743 RTT requests. The data contained in DATA OUT UPIU is written.

3744 The number of bytes requested and the Data Buffer Offset field in each RTT shall both be integer  
3745 multiples of the Logical Block Size (bLogicalBlockSize).

3746 The data segment of each DATA OUT UPIU shall contain an integer number of logical blocks.

3747 Zero or an incomplete number of segments may be requested, if an error occurs before the entire data  
3748 transfer is complete.

3749

3750   **11.3.16.3 Write (16) Command Status Response**

- 3751   • STATUS response will be sent in a single RESPONSE UPIU
- 3752   • If all requested data is successfully transferred and written, the WRITE command will terminate with  
3753    a STATUS response of GOOD.
- 3754   • If the logical blocks are transferred directly to a cache then the Device Server may complete the  
3755    command with a GOOD status prior to writing the logical blocks to the medium.
- 3756   • If the unit is not ready to accept a new command (e.g., still processing previous command) a  
3757    STATUS response of BUSY will be returned.
- 3758   • Failure can occur for numerous reasons. When the WRITE command fails a STATUS response of  
3759    CHECK CONDITION will be returned along with an appropriate SENSE KEY, such as
- 3760     ○ ILLEGAL REQUEST (range or CDB errors)
- 3761     ○ MEDIUM ERROR (medium failure, ECC, etc.)
- 3762     ○ HARDWARE ERROR (hardware failure)
- 3763     ○ UNIT ATTENTION (reset, power-on, etc.)
- 3764     ○ DATA PROTECT (permanent, power-on, secure write protect, etc.)
- 3765     ○ etc.

3766 **11.3.17 REQUEST SENSE Command**

3767 The REQUEST SENSE Command requests that the Device Server transfer parameter data containing  
3768 sense data information to the Application Client.

- 3769 • Sense Data describes error or exception condition and/or current operational status of device
  - 3770 ○ i.e., get the device "status"
- 3771 • UFS devices will return a fixed format data record of 18 bytes of sense data as described in Table  
3772 10.17.
- 3773 • Three tiered error code for detailed status
  - 3774 ○ Sense Key: main indicator
  - 3775 ○ ASC: Additional Sense Code
  - 3776 ○ ASCQ: Additional Sense Code Qualifier
- 3777 • If a REQUEST SENSE command is received with a pending UNIT ATTENTION condition (i.e., before  
3778 the device server reports CHECK CONDITION status), the device server shall perform the REQUEST  
3779 SENSE command and clear the UNIT ATTENTION condition.

3780 The Command CDB shall be sent in a single COMMAND UPIU

3781 **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								
4	ALLOCATION LENGTH							
5	CONTROL = 00h							

3782 UFS devices are not required to support descriptor format sense data.

3783 **11.3.17.1 Request Sense Data Response**

- 3784 • Data returned from a REQUEST SENSE command will be transferred to the Application Client in a  
3785 single DATA IN UPIU.
- 3786 • The Device Server will transfer up to 18 bytes of Response Data in the Data Segment area of a DATA  
3787 IN UPIU.
  - 3788 ○ Return 18 bytes if Allocation Length in CDB  $\geq$  18.
  - 3789 ○ Return Allocation Length bytes if Allocation Length in CDB  $<$  18.
  - 3790 ○ An Allocation Length of zero specifies that no data shall be transferred. This condition shall not  
3791 be considered as an error, and DATA IN UPIU shall not be generated.
- 3792 • Data will be returned in the indicated Sense Data Format described in 11.3.17.2.

3793   **11.3.17.2 Sense Data**

3794   See Table 10.17.

3795

3796   **11.3.17.3 Sense Key**

3797   See Table 10.18.

3798

3799   **11.3.17.4 Request Sense Status Response**

3800   • STATUS response will be sent in a single RESPONSE UPIU.

3801   • If the requested data is successfully transferred, the REQUEST SENSE command will terminate with  
3802   a STATUS response of GOOD.

3803   • If the unit is not ready to accept a new command (e.g., still processing previous command) a  
3804   STATUS response of BUSY will be returned.

3805   • Failure is very rare. When the REQUEST SENSE command fails a STATUS response of CHECK  
3806   CONDITION will be returned along with an appropriate SENSE KEY, such as

3807    ◦ ILLEGAL REQUEST (range or CDB errors).

3808   • Will not fail due to a pending UNIT ATTENTION condition.

3809   • If the REQUEST SENSE command was received with a pending unit attention condition, the returned  
3810   sense data will indicate the cause of the unit attention condition, and the unit attention condition  
3811   within the device server will be cleared.

3812   • If a REQUEST SENSE command is terminated with CHECK CONDITION status, then the device  
3813   server shall not clear the pending unit attention condition.

3814

3815 **11.3.18 FORMAT UNIT Command**

3816 The FORMAT UNIT command requests that the Device Server format the medium into Application  
3817 Client accessible logical blocks as specified in the parameter lists. The Device Server may also certify the  
3818 medium and create control structures for the management of the medium and defects. The degree that the  
3819 medium is altered by the command is vendor specific.

3820 A FORMAT UNIT command sent to the Device well known logical unit requests the device format all  
3821 enabled logical units except the RPMB well known logical unit (see in 12.2.3.4, Wipe Device).

3822 If the medium is write-protected, then the command shall be terminated with CHECK CONDITION  
3823 status with the sense key set to DATA PROTECT.

3824 Following a successful format operation all LBAs:

3825 a) shall be mapped on a fully provisioned logical unit (bProvisioningType set to 00h)

3826 b) shall be unmapped on a thin provisioned logical unit (bProvisioningType set to 02h or 03h)

3827 For a LBA in a formatted logical unit specified by a read operation, the device server shall send user data  
3828 with all bits set to zero to the data in buffer.

3829 The Command CDB shall be sent in a single COMMAND UPIU.

3830 **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										

3831

3832 **11.3.18.1 Format Unit Command Parameters**

3833 **Table 11.34 — Format Unit Command Parameters**

<b>Byte</b>	<b>Bit</b>	<b>Description</b>
1	7:6	<b>FMTINFO:</b> Specifies the FORMAT PROTECTION INFORMATION as detailed in [SBC].
1	5	<b>LONGLIST:</b> 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	<b>FMTDATA:</b> If set to '1' specifies that parameter list data shall be transferred from the data-out buffer.
1	3	<b>CMLPLST:</b> 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	<b>DEFECT LIST FORMAT:</b> 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	<b>VENDOR SPECIFIC:</b> Vendor specified field.

3834 **11.3.18.2 Format Unit Command Data Transfer**

- 3835 • If needed, the Device Server requests to transfer the FORMAT UNIT parameter list from the  
3836 Application Client data-out buffer by issuing a series of READY TO TRANSFER UPIU's (RTT).  
3837 • The FORMAT UNIT parameter list is delivered in one or more segments sending DATA OUT UPIU  
3838 packets, as indicated in the RTT requests.  
3839 • Zero or an incomplete number of segments may be requested if an error occurs before the entire data  
3840 transfer is complete.

3841 **11.3.18.3 Format Unit Command Status Response**

- 3842 • STATUS response will be sent in a single RESPONSE UPIU.  
3843 • If the requested format of the medium is performed successfully then the command will terminate  
3844 with a STATUS response of GOOD.  
3845 • If the unit is not ready to accept a new command (e.g., still processing previous command) a  
3846 STATUS response of BUSY will be returned.  
3847 • Other failures can occur for numerous reasons. When the FORMAT UNIT command fails, a  
3848 STATUS response of CHECK CONDITION will be returned along with an appropriate SENSE  
3849 KEY, such as:  
3850     ○ ILLEGAL REQUEST (range or CDB errors)  
3851     ○ MEDIUM ERROR (medium failure, ECC, etc.)  
3852     ○ HARDWARE ERROR (hardware failure)  
3853     ○ UNIT ATTENTION (reset, power-on, etc.)  
3854     ○ DATA PROTECT (permanent, power-on, secure write protect, etc.)  
3855     ○ etc.

3856 **11.3.19 PRE-FETCH (10) Command**

3857 The PRE-FETCH (10) command shall require the device server to transfer the logical blocks for the  
3858 mapped LBAs specified by the command from the medium to the volatile cache (if such cache exists) and  
3859 for any unmapped LBAs specified by the command to update the volatile cache to prevent retrieval of  
3860 stale data as defined in [SBC].

3861 The Command CDB shall be sent in a single COMMAND UPIU.

3862 **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													
4	LOGICAL BLOCK ADDRESS												
5													
6	Reserved			GROUP NUMBER = 00000b									
7	(MSB)												
8	PREFETCH LENGTH												
9	(LSB)												
	CONTROL= 00h												

3863

3864 **11.3.19.1 PRE-FETCH (10) Command Parameters**

3865 **Table 11.36 — PRE-FETCH Command Parameters**

<b>Byte</b>	<b>Bit</b>	<b>Description</b>
1	1	<b>IMMED:</b> 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	<b>LOGICAL BLOCK ADDRESS:</b> 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	<b>GROUP NUMBER:</b> 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	<b>PREFETCH LENGTH:</b> 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	<b>CONTROL:</b> 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.

3866

3867 **11.3.19.2 PRE-FETCH Command Data Transfer**

3868 The PRE-FETCH command does not have a data transfer phase

- 3869 • No DATA IN or DATA OUT UPIU's are transferred

3870 **11.3.19.3 PRE-FETCH Command Status Response**

- 3871 • STATUS response will be sent in a single RESPONSE UPIU

3872 • If the command is performed successfully then it will terminate with a STATUS response of GOOD

3873 • If the unit is not ready to accept a new command (e.g., still processing previous command) a  
3874 STATUS response of BUSY will be returned

3875 • Other failures can occur for numerous reasons. When the PRE-FETCH command fails, a STATUS  
3876 response of CHECK CONDITION will be returned along with an appropriate SENSE KEY, such as:

3877 ○ ILLEGAL REQUEST (range or CDB errors)

3878 ○ MEDIUM ERROR (medium failure, ECC, etc.)

3879 ○ HARDWARE ERROR (hardware failure)

3880 ○ UNIT ATTENTION (reset, power-on, etc.)

3881 ○ etc.

3882

3883 **11.3.20 PRE-FETCH (16) Command**

3884 See PRE-FETCH (10) command for details.

3885 **Table 11.37 — PRE-FETCH (16) command**

Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (90h)							
1	Reserved						IMMED	Reserved
2	(MSB)							
....	LOGICAL BLOCK ADDRESS							
9								
10	(MSB)							
....	PREFETCH LENGTH							
13								
14	Reserved			GROUP NUMBER = 00000b				
15	CONTROL= 00h							

3886

3887 **11.3.21 SECURITY PROTOCOL IN Command**

3888 The SECURITY PROTOCOL IN command is used to retrieve security protocol information or the results  
3889 of one or more SECURITY PROTOCOL OUT commands. See [SPC] for details.

3890 **Table 11.38 — SECURITY PROTOCOL IN command**

Bit Byte	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															
10	Reserved														
11	CONTROL = 00h														

3891 **11.3.21.1 SECURITY PROTOCOL IN Command Parameter**

- The SECURITY PROTOCOL field specifies which security protocol is being used.  
3892 UFS devices shall support the following value:
  - ECh: JEDEC UFS application3893 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  
3897 512 bytes.

3898 **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  
3900 IN UPIU's.

3901 **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  
3905 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.

3910 **11.3.22 SECURITY PROTOCOL OUT Command**

3911 The SECURITY PROTOCOL OUT command is used to send data to the logical unit. The data sent  
3912 specifies one or more operations to be performed by the logical unit. The format and function of the  
3913 operations depends on the contents of the SECURITY PROTOCOL field. See [SPC] for details.

3914 **Table 11.39 — SECURITY PROTOCOL OUT command**

Bit Byte	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															
10	Reserved														
11	CONTROL = 00h														

3915 **11.3.22.1 SECURITY PROTOCOL OUT Command Parameter**

- 3916 • The SECURITY PROTOCOL field specifies which security protocol is being used.  
3917 UFS devices shall support the following value:
- 3918     ○ ECh: JEDEC UFS application
- 3919     Support of other SECURITY PROTOCOL values is device specific.
- 3920 • A INC\_512 bit set to one specifies that the TRANSFER LENGTH is expressed in increments of 512  
3921 bytes.

3922 **11.3.22.2 SECURITY PROTOCOL OUT Command Data Transfer**

3923 The Device Server requests to transfer data from the Application Client data-out buffer by issuing a series  
3924 of READY TO TRANSFER UPIU's (RTT).

3925 The data is delivered in one or more segments sending DATA OUT UPIU packets, as indicated in the  
3926 RTT requests.

3927 Zero or an incomplete number of segments may be requested, if an error occurs before the entire data  
3928 transfer is complete.

3929

3930   **11.3.22.3 SECURITY PROTOCOL OUT Command Status Response**

- 3931   • Status response shall be sent in a single RESPONSE UPIU
- 3932   • If the command is successfully executed, it shall terminate with a STATUS response of GOOD
- 3933   • Failure can occur for numerous reasons. When the command fails a STATUS response of CHECK
- 3934    CONDITION will be returned along with an appropriate SENSE KEY, such as
- 3935      ○ ILLEGAL REQUEST (range or CDB errors)
- 3936      ○ HARDWARE ERROR (hardware failure)
- 3937      ○ UNIT ATTENTION (reset, power-on, etc.)
- 3938      ○ etc.

3939 **11.3.23 SEND DIAGNOSTIC Command**

3940 The SEND DIAGNOSTIC command requests the Device Server to perform diagnostic operations on the  
3941 SCSI target device, on the logical unit or on both. Logical units shall implement, at a minimum, the  
3942 default self-test feature (i.e., the SELFTEST bit equal to one and a parameter list length of zero).

3943 The Command CDB shall be sent in a single COMMAND UPIU.

3944 **Table 11.40 — SEND DIAGNOSTIC command**

Byte	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										

3945 **11.3.23.1 Send Diagnostic Parameters**

3946 **Table 11.41 — Send Diagnostic Parameters**

Byte	Bit	Description
1	7:5	<b>SELF-TEST CODE:</b> Specifies the self-test code as defined in SPC4.
1	4	<b>PF:</b> Specifies the format of any parameter list sent by the Application Client
1	2	<b>SELFTEST:</b> Specifies the device server shall perform a self test.
1	1	<b>DEVOFFL:</b> 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	<b>UNITOFFL:</b> 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	<b>PARAMETER LIST LENGTH:</b> 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.

3947 **11.3.23.2 Send Diagnostic Command Data Transfer**

3948 The SEND DIAGNOSTIC command will transfer out the number of bytes specified by the Parameter List  
3949 Length. If that value is zero then no data out transfer will occur. The Device Server will request the  
3950 transfer of the specified bytes from the Application Client by issuing a series READY TO TRANSFER  
3951 UPIU (RTT). RTT will be followed by DATA OUT UPIU containing the number of bytes to transfer.

3952 **11.3.23.3 Send Diagnostic Command Status Response**

- 3953 • STATUS response will be sent in a single RESPONSE UPIU.
- 3954 • If the requested diagnostics are performed successfully then the command will terminate with a  
3955 STATUS response of GOOD.
- 3956 • If the unit is not ready to accept a new command (e.g., still processing previous command) a  
3957 STATUS response of BUSY will be returned.
- 3958 • Other failures can occur for numerous reasons. When the SEND DIAGNOSTIC command fails, a  
3959 STATUS response of CHECK CONDITION will be returned along with an appropriate SENSE  
3960 KEY, such as
- 3961     ○ ILLEGAL REQUEST (range or CDB errors)
  - 3962     ○ MEDIUM ERROR (medium failure, ECC, etc.)
  - 3963     ○ HARDWARE ERROR (hardware failure)
  - 3964     ○ UNIT ATTENTION (reset, power-on, etc.)
  - 3965     ○ etc.

3966

3967 **11.3.24 SYNCHRONIZE CACHE (10) Command**

3968 The SYNCHRONIZE CACHE (10) command requests that the device server ensure that the specified  
3969 logical blocks have their most recent data values recorded on the medium.

3970 **Table 11.42 — SYNCHRONIZE CACHE (10) command**

Bit Byte	7	6	5	4	3	2	1	0							
0	OPERATION CODE (35h)														
1	Reserved				SYNC_NV	IMMED	Obsolete =0b								
2	(MSB)														
3															
4	LOGICAL BLOCK ADDRESS														
5															
6	Reserved			GROUP NUMBER = 00000b											
7	(MSB)														
8	NUMBER OF LOGICAL BLOCKS														
9	(LSB)														
	CONTROL = 00h														

3971

3972 **11.3.24.1 Synchronize Cache Command Parameters**

3973 **Table 11.43 — Synchronize Cache Command Parameters**

<b>Byte</b>	<b>Bit</b>	<b>Description</b>
1	2	<b>SYNC_NV:</b> 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	<b>IMMED:</b> 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	<b>LOGICAL BLOCK ADDRESS:</b> 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	<b>GROUP NUMBER:</b> 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	<b>NUMBER OF LOGICAL BLOCKS:</b> 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	<b>CONTROL:</b> 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.

3974

3975 **11.3.24.2 Synchronize Cache Command Data Transfer**

3976 The SYNCHRONIZE CACHE command does not have a data transfer phase.

- 3977 • No DATA IN or DATA OUT UPIU's are transferred.

3978 **11.3.24.3 Synchronize Cache Command Status Response**

- 3979 • STATUS response will be sent in a single RESPONSE UPIU.

3980 • If the command is performed successfully then it will terminate with a STATUS response of GOOD.

3981 • If the unit is not ready to accept a new command (e.g., still processing previous command) a  
3982 STATUS response of BUSY will be returned.

3983 • Other failures can occur for numerous reasons. When the SYNCHRONIZE CACHE command fails, a  
3984 STATUS response of CHECK CONDITION will be returned along with an appropriate SENSE  
3985 KEY, such as

- 3986     ○ ILLEGAL REQUEST (range or CDB errors)
- 3987     ○ MEDIUM ERROR (medium failure, ECC, etc.)
- 3988     ○ HARDWARE ERROR (hardware failure)
- 3989     ○ UNIT ATTENTION (reset, power-on, etc.)
- 3990     ○ etc.

3991

3992 **11.3.25 SYNCHRONIZE CACHE (16) Command**

3993 See SYNCHRONIZE CACHE (10) command for details.

3994 **Table 11.44 — SYNCHRONIZE CACHE (16) Command Descriptor Block**

Bit Byte	7	6	5	4	3	2	1	0		
0	OPERATION CODE (91h)									
1	Reserved				SYNC_NV	IMMED	Reserved			
2	(MSB)									
...	LOGICAL BLOCK ADDRESS									
9										
10	(MSB)									
...	NUMBER OF LOGICAL BLOCKS									
13										
14	Reserved			GROUP NUMBER = 00000b						
15	CONTROL = 00h									

3995

3996 **11.3.26 UNMAP Command**

3997 The UNMAP command shall require the device server to cause one or more LBAs to be unmapped (de-  
3998 allocated).

3999 UFS defines that a logical unit shall be either Full Provisioning or Thin Provisioning as described in SCSI  
4000 SBC. To use UNMAP command, SCSI SBC requires that a logical unit to be thin-provisioned and  
4001 support logical block provisioning management. UNMAP command is not supported in a full-provisioned  
4002 logical unit.

4003 In UFS, a thin provisioned logical unit shall have sufficient physical memory resources to support the  
4004 logical block address space when the device is configured by the user. Mapped State and De-Allocated  
4005 State are mandatory in a UFS thin provisioned logical unit.

4006 **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													
4	Reserved												
5													
6	Reserved			GROUP NUMBER = 00000b									
7	(MSB)												
8	PARAMETER LIST LENGTH												
9													
	CONTROL = 00h												

- 4007 • GROUP NUMBER = ‘0’  
4008 • ANCHOR = 0 for UFS. If ANCHOR = 1, the device server shall terminate the command with CHECK  
4009 CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to  
4010 INVALID FIELD IN CDB.  
4011 • The PARAMETER LIST LENGTH field specifies the length in bytes of the UNMAP parameter data that  
4012 are sent from the application client to the device server.

4013

4014

### **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.

4019

**Table 11.46 — UNMAP parameter list**

4020  
4021  
4022  
4023  
4024  
4025  
4026  
4027  
4028

- 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**

- 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.

4057    **11.3.26.3 UNMAP parameter list transfer**

- 4058    • The Device Server requests to transfer the UNMAP parameter list from the Application Client data-  
4059    out buffer by issuing a series of READY TO TRANSFER UPIU's (RTT).  
4060    • The UNMAP parameter list is delivered in one or more segments sending DATA OUT UPIU packets,  
4061    as indicated in the RTT requests.  
4062    • Zero or an incomplete number of segments may be requested, if an error occurs before the entire data  
4063    transfer is complete.

4064    **11.3.26.4 UNMAP Command Status Response**

- 4065    • STATUS response will be sent in a single RESPONSE UPIU  
4066    • If the command is performed successfully then it will terminate with a STATUS response of GOOD.  
4067    • If the unit is not ready to accept a new command (e.g., still processing previous command) a  
4068    STATUS response of BUSY will be returned.  
4069    • Failure can occur for numerous reasons. When the UNMAP command fails a STATUS response of  
4070    CHECK CONDITION will be returned along with an appropriate SENSE KEY, such as  
4071       o ILLEGAL REQUEST (range or CDB errors)  
4072       o MEDIUM ERROR (medium failure, ECC, etc.)  
4073       o HARDWARE ERROR (hardware failure)  
4074       o UNIT ATTENTION (reset, power-on, etc.)  
4075       o DATA PROTECT (permanent, power-on, secure write protect, etc.)  
4076       o etc.

4077

4078 **11.3.27 READ BUFFER Command**

4079 The READ BUFFER command is used in conjunction with the WRITE BUFFER command for

- 4080 • testing logical unit buffer memory  
4081 • testing the integrity of the service delivery subsystem  
4082 • Downloading microcode  
4083 • Retrieving error history and statistics

4084 The READ BUFFER command transfers a specified number of data bytes from a specified offset within a  
4085 specified buffer in the Device Server to a buffer in the Application Client.

4086 The Command CDB shall be sent in a single COMMAND UPIU.

4087 **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													
6	(MSB)												
7	ALLOCATION LENGTH												
8													
9	CONTROL = 00h												

4088

4089 **11.3.27.1 Read Buffer Command MODE Field Values**

4090 **Table 11.49 — Read Buffer Command Mode Field Values**

MODE	DESCRIPTION
00h	Not used in UFS
01h	Vendor Specific
02h	Data
03h-1Bh	Not used in UFS
1Ch	Error history
1Dh-1Fh	Reserved

- 4091 • The device shall support the MODE value of 02h, indicating Data Mode. The definition and structure  
4092 of the data being transferred in Data Mode is device specific.  
4093 • The device shall support the MODE value of 1Ch, indicating Error History Mode. The format of the  
4094 error history is device specific.

4095 **11.3.27.2 Data Mode (MODE = 02h)**

- 4096 • The BUFFER ID field specifies a buffer within the logical unit from which data shall be transferred.  
4097 Buffer ID 0 shall be supported. If more than one buffer is supported, then additional buffer ID codes  
4098 shall be assigned contiguously, beginning with one.  
4099 • The BUFFER OFFSET field contains the byte offset within the specified buffer from which data shall  
4100 be transferred.  
4101 • The Device Server will read up to Allocation Length number of data bytes from the specified Buffer  
4102 Offset within a buffer specified by the Buffer ID in the Device Server and transfer them to a buffer in  
4103 the Application Client  
4104     ○ Less than Allocation Length will be transferred if Device Server contains less bytes  
4105 • Data will be transferred from the Device Server to the Application Client via a series of DATA IN  
4106 UPIU's  
4107     ○ The data transferred from the Device Server will be contained within the Data Segment of the  
4108 DATA IN UPIU  
4109 • Zero or an incomplete number of DATA IN UPIU's will be transferred if an error occurs before the  
4110 entire data transfer is complete

4112 **11.3.27.3 Error History Mode (MODE = 1Ch)**

- 4113 • The BUFFER ID field specifies the action that the device server shall perform, and the parameter  
4114 data, if any, that the device server shall return.

4115 **Table 11.50 — Buffer ID Field for Error History Mode**

CODE	DESCRIPTION	BUFFER OFFSET
00h	Return error history directory <sup>1</sup>	Zero <sup>2</sup>
01h – 03h	Not used in UFS	
04h – 0Fh	Reserved	
10h – EFh	Return error history from corresponding error history data buffer ID	Zero <sup>2</sup> to Maximum <sup>3</sup>
F0h – FDh	Reserved	
FEh – FFh	Not used in UFS	

NOTE 1 A error history snapshot is never created in this standard.  
 NOTE 2 Zero is 000000h for the READ BUFFER (10) command.  
 NOTE 3 Maximum is FFFFFFFh for the READ BUFFER (10) command.

- 4116 • In UFS standard, error history I\_T nexus is always established and valid.
- 4117 • In UFS standard, there is no error history snapshot exist. The returned error history may be  
4118 real time contents or may be the contents captured at a vendor specific point in time.
- 4119 • The BUFFER OFFSET field specifies the byte offset from the start of the buffer specified by  
4120 the BUFFER ID field from which the device server shall return data.
- 4121 • The Device Server will read up to Allocation Length number of data bytes from the specified  
4122 Buffer Offset within a buffer specified by the Buffer ID in the Device Server and transfer  
4123 them to a buffer in the Application Client
  - 4124 ○ Less than Allocation Length will be transferred if Device Server contains less bytes
- 4125 • Data will be transferred from the Device Server to the Application Client via a series of  
4126 DATA IN UPIU's
  - 4127 ○ The data transferred from the Device Server will be contained within the Data Segment  
4128 of the DATA IN UPIU
- 4129 • Zero or an incomplete number of DATA IN UPIU's will be transferred if an error occurs  
4130 before the entire data transfer is complete
- 4131 • See [SPC] for further details.

4132  
4133

4134 **11.3.27.3.1 Error History Directory**

4135 The error history directory is defined in Table 11.51, and the error history directory entry is defined in  
4136 Table 11.53.

4137

**Table 11.51 — Error history directory**

Bit Byte	7	6	5	4	3	2	1	0
0	(MSB)							
...								T10 VENDOR IDENTIFICATION
7								(LSB)
8								VERSION
9		Reserved		EHS_RETRIEVED = 00b		EHS_SOURCE = 00b		CLR_SUP = 0b
10	(MSB)							
...								Reserved
29								(LSB)
30	(MSB)			DIRECTORY LENGTH (n-31)				
31								(LSB)
				Error history directory list				
32								
...				Error history directory entry [first]				
39								
...				...				
n-7								
...				Error history directory entry [last]				
n								

- 4138 • The T10 VENDOR IDENTIFICATION field contains eight bytes of left-aligned ASCII data as  
4139 defined in [SPC].
- 4140 • The VERSION field indicates the version and format of the vendor specific error history. The  
4141 VERSION field is assigned by the vendor indicated in the T10 VENDOR IDENTIFICATION field.
- 4142 • The DIRECTORY LENGTH field indicates the number of error history directory list bytes available  
4143 to be transferred. This value shall not be altered even if the allocation length is not sufficient to  
4144 transfer the entire error history directory list.
- 4145 • The error history source (EHS\_SOURCE) field indicates the source of the error history snapshot.

4146 **11.3.27.3.1 Error History Directory (cont'd)**

4147 **Table 11.52 — EHS\_SOURCE field**

Code	Description
00b	The error history snapshot was created by the device server and was not created due to processing a READ BUFFER command.
01b	Not used in UFS
10b	Not used in UFS
11b	Reserved

- 4148 • The error history directory list contains an error history directory entry for each supported buffer ID in the  
4149 range of 10h to EFh. The first entry shall be for buffer ID 10h and the entries shall be in order of ascending  
4150 buffer IDs. The supported buffer IDs are not required to be contiguous. There shall not be any entries for  
4151 buffer IDs greater than or equal to F0h.

4152 **Table 11.53 — Error history directory entry**

Bit Byte	7	6	5	4	3	2	1	0
0	SUPPORTED BUFFER ID							
1								
2	Reserved							
3								
4	(MSB)							
...	MAXIMUM AVAILABLE LENGTH							
7	(LSB)							

- 4153 • The SUPPORTED BUFFER ID field indicates the error history buffer ID associated with this entry.  
4154 • The MAXIMUM AVAILABLE LENGTH field indicates the maximum number of data bytes contained in  
4155 the buffer indicated by the SUPPORTED BUFFER ID field. The actual number of bytes available for  
4156 transfer may be smaller.

4157

4158   **11.3.27.3.2 Retrieving error history with the READ BUFFER command**

4159   In UFS standard, Error History Snapshot is not supported. The returned error history may be real-time  
4160   contents or may be the contents captured at a vendor specific point in time.

4161   Error history retrieve operation uses the following mechanism.

- 4162   1) Device failure is detected. Host may need to reset UFS device and re-establish the link before retrieving  
4163   the error history if the device is not responding after the failure.
- 4164   2) Host retrieves the error history directory by sending a READ BUFFER command by specifying: MODE =  
4165   1Ch, BUFFER ID = 00h, BUFFER OFFSET = zero, and ALLOCATION LENGTH set to at least 2088  
4166   (i.e., large enough to transfer the complete error history directory).
- 4167   3) Host retrieves the error history. For each buffer ID indicated in the error history directory in the range of  
4168   10h to EFh, the host sends one or more READ BUFFER commands with BUFFER ID set to the error  
4169   history data buffer ID. During the error history retrieval, it is recommended that host does not send any  
4170   normal request other than READ BUFFER command because the other normal request could change the  
4171   error history information which is being retrieved.

4172   **11.3.27.4 Read Buffer Command Status Response**

- 4173   • Status response will be sent in a single RESPONSE UPIU
- 4174   • If all requested data is successfully read and transferred, the READ BUFFER command will  
4175   terminate with a STATUS response of GOOD
- 4176   • If the unit is not ready to accept a new command (e.g., still processing previous command) a  
4177   STATUS response of BUSY will be returned
- 4178   • Failure can occur for numerous reasons. When the READ BUFFER command fails a STATUS  
4179   response of CHECK CONDITION will be returned along with an appropriate SENSE KEY, see  
4180   [SPC] for further details.
  - 4181   ○ ILLEGAL REQUEST (range or CDB errors)
  - 4182   ○ MEDIUM ERROR (medium failure, ECC, etc.)
  - 4183   ○ HARDWARE ERROR (hardware failure)
  - 4184   ○ UNIT ATTENTION (reset, power-on, etc.)
  - 4185   ○ etc.

4187 **11.3.28 WRITE BUFFER Command**

4188 The WRITE BUFFER command is used in conjunction with the READ BUFFER command for

- 4189 • testing logical unit buffer memory  
4190 • testing the integrity of the service delivery subsystem  
4191 • Field Firmware Update  
4192 • Retrieving error history and statistics

4193 The WRITE BUFFER command transfers a specified number of data bytes from a buffer in the  
4194 Application Client to a specified buffer in the Device Server at a specified buffer offset

4195 The Command CDB shall be sent in a single COMMAND UPIU

4196 **Table 11.54 — 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													
6	(MSB)												
7	PARAMETER LIST LENGTH												
8													
9	CONTROL = 00h												

4197

4198 **11.3.28.1 Write Buffer Command Parameters**

4199 **Table 11.55 — Write Buffer Command Parameters**

Byte	Bit	Description
1	4:0	<b>MODE:</b> Specifies the function of this command. See Table 11.56 for more detail.
2	7:0	<b>BUFFER ID:</b> 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	<b>BUFFER OFFSET:</b> Specifies the byte offset within the specified buffer from which data shall be transferred.
6:8	7:0	<b>PARAMETER LIST LENGTH:</b> Specifies the maximum number of bytes the Application Client buffer will transfer to the Device Server.

4200 **11.3.28.2 Write Buffer Command Mode Field Values**

4201 **Table 11.56 — 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

- 4202 • The device shall support the MODE value of 02h, indicating Data Mode. The BUFFER ID field specifies a  
4203 buffer to which data shall be transferred. The BUFFER OFFSET field specifies the location to which the  
4204 data is written.  
4205 • The definition and structure of the data being transferred in Data Mode is device specific.  
4206 • UFS device shall support a MODE value of 0Eh for microcode download as defined in section Field  
4207 Firmware Update.

4208

4209 **11.3.28.2.1 Field Firmware Update**

4210 UFS Field Firmware Update (FFU) is based on microcode download definition in [SPC].

4211 [SPC] describes multiple operation modes for microcode download which are selected using the MODE  
4212 field in the WRITE BUFFER command. UFS supports only the MODE field value 0Eh: “Download  
4213 microcode with offsets, save, and defer active”.

4214 The deferred microcode shall be activated and no longer considered deferred when a power on or a hard  
4215 reset occurs. Note that in UFS, START STOP UNIT command, FORMAT UNIT command or WRITE  
4216 BUFFER command (MODE=0Fh) will not activate the microcode.

4217 UFS FFU uses the following mechanism:

- 4218 1) Host delivers the microcode using one or more WRITE BUFFER commands through any logical unit  
4219 which supports the WRITE BUFFER command. The host specifies: MODE = 0Eh, BUFFER  
4220 OFFSET, which should be aligned to 4 Kbyte, BUFFER ID = 00h, and the PARAMETER LIST  
4221 LENGTH field indicating the number of bytes to be transferred.  
4222 All WRITE BUFFER commands should be sent to the same logical unit with task attribute set to  
4223 simple or ordered. In the sequence of WRITE BUFFER commands used to deliver the microcode, the  
4224 BUFFER OFFSET values should be in increasing order and it should start from zero.
- 4225 2) bFFUTimeout indicates the maximum time in which the device may handle the WRITE BUFFER  
4226 command. Within this time access to the device is limited or not possible.
- 4227 3) Following a successful delivery of the microcode, the host activates the new firmware using a  
4228 hardware reset or a power cycle. The UFS device shall use new firmware upon hard reset or power  
4229 up. Host should be aware that the first initialization flow after a successful delivery of the microcode  
4230 may be longer than usual.
- 4231 4) After device initialization, the host should read bDeviceFFUStatus attribute and verify that the new  
4232 firmware was updated successfully.

4233 Other modes of WRITE BUFFER command are not supported by the UFS device for FFU process.

4234 **11.3.28.3 Write Buffer Command Data Transfer**

4235 The Device Server requests to transfer the buffer data from the Application Client data-out buffer by  
4236 issuing a series of READY TO TRANSFER UPIU's (RTT).

4237 The buffer data is delivered in one or more segments sending DATA OUT UPIU packets, as indicated in  
4238 the RTT requests.

4239 Zero or an incomplete number of segments may be requested, if an error occurs before the entire data  
4240 transfer is complete.

4241 The received data is written to the location specified by the BUFFER OFFSET field within the buffer  
4242 specified by the BUFFER ID field.

4243

4244      **11.3.28.4 Write Buffer Command Status Response**

- 4245      • STATUS response will be sent in a single RESPONSE UPIU  
4246      • If the requested data is successfully transferred and written, the WRITE BUFFER command will terminate  
4247      with a STATUS response of GOOD  
4248      • If the unit is not ready to accept a new command (e.g., still processing previous command) a STATUS  
4249      response of BUSY will be returned  
4250      • Failure can occur for numerous reasons. When the WRITE BUFFER command fails a STATUS response  
4251      of CHECK CONDITION will be returned along with an appropriate SENSE KEY, such as:  
4252        o ILLEGAL REQUEST (range or CDB errors)  
4253        o MEDIUM ERROR (medium failure, ECC, etc.)  
4254        o HARDWARE ERROR (hardware failure)  
4255        o UNIT ATTENTION (reset, power-on, etc.)  
4256        o etc.

4257

## 4258 11.4 Mode Pages

4259 This section describes the mode pages used with MODE SELECT command and MODE SENSE  
4260 command. Subpages are identical to mode pages except that they include a SUBPAGE CODE field that  
4261 further differentiates the mode page contents.

### 4262 11.4.1 Mode Page Overview

#### 4263 11.4.1.1 Mode Page/Subpage Codes

- 4264 • Mode pages and subpages are selected by Page Code field and the Subpage Code field
- 4265 • UFS devices are not required to support subpages (subpage = 0)

4266 **Table 11.57 — Mode page code usage**

<b>Page Code</b>	<b>Subpage Code</b>	<b>Description</b>	<b>Page Format</b>
00h	Vendor specific	Vendor specific page	Vendor specific format
(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

4267

4268 **11.4.1.2 Mode parameter list format**

4269 General format for reading or writing mode pages.

4270 UFS will not implement Block Descriptor field

4271 **Table 11.58 — 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							

4272 **11.4.1.3 Mode Parameter Header**

4273 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.

4275 **Table 11.59 — UFS Mode parameter header (10)**

Bit Byte	7	6	5	4	3	2	1	0
0	(MSB) MODE DATA LENGTH							
1								
2	MEDIUM TYPE = 00h							
3	DEVICE SPECIFIC PARAMETER							
4	WP Reserved = 00b DPOFUA Reserved = 0000b							LONGLBA = 0b
5	Reserved = 00h							
6	(MSB) BLOCK DESCRIPTOR LENGTH = 0000h							
7								

4276 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.

4280

4281 **11.4.1.4 Mode Parameter Header Detail**

4282

**Table 11.60 — Mode Parameter Header Detail**

Byte	Bit	Description
0:1	7:0	<b>MODE DATA LENGTH:</b> 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	<b>MEDIUM TYPE:</b> Indicates the medium type of the device. For UFS this value shall be set to 00h, indicating Data Medium.
3	7:0	<b>DEVICE SPECIFIC PARAMETER:</b> 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 <sup>(1)</sup> . 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	<b>BLOCK DESCRIPTOR LENGTH:</b> 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.

4283 NOTE 1 The WP bit shall be set to one when the logical unit is write-protected by any method, including any one of  
4284 the following:

- 4285 • the software write protect (SWP) bit in the Control mode page is set to one
- 4286 • the logical unit is configured as permanently write protected and fPermanentWPEn = 1
- 4287 • the logical unit is configured as power on write protected and fPowerOnWPEn = 1
- 4288 • the device is write-protected by vendor-specific electrical or mechanical mechanism.

4289

4290 **11.4.1.5 Page\_0 mode page format**

4291 **Table 11.61 — 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									

4292

4293

**Table 11.62 — Page 0 Format parameters**

Byte	Bit	Description
0	7:7	<b>PS:</b> 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	<b>SPF:</b> 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	<b>PAGE CODE:</b> Indicates the format and parameters for particular mode page.
1	7:0	<b>PAGE LENGTH:</b> Indicates the size in bytes of the following mode page parameters.
2:N	7:0	<b>MODE PARAMETERS:</b> The contents of the indicated mode page.

4294

4295 **11.4.1.6 Sub\_page mode page format**

4296 **Table 11.63 — 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)							(LSB)
3									
4		Mode Parameters							
n									

4297

4298

**Table 11.64 — Subpage Format parameters**

Byte	Bit	Description
0	7:7	<b>PS:</b> 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	<b>SPF:</b> 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	<b>PAGE CODE:</b> Specifies the mode page.
1	7:0	<b>SUBPAGE CODE:</b> Specifies the subpage of a particular mode page.
2:3	7:0	<b>PAGE LENGTH:</b> Indicates the size in bytes of the following mode page parameters.
4:N	7:0	<b>MODE PARAMETERS:</b> The contents of the indicated mode page.

4299

4300 **11.4.2 UFS Supported Pages**

4301 Table 11.65 shows the mode pages supported by UFS device. This standard does not define any  
4302 additional subpages.

4303 **Table 11.65 — 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

4304  
4305 If the device has more than one logical unit, host should read Mode Page Policy VPD in order to know  
4306 whether the logical unit maintains its own copy of the mode page and subpage or all logical units share  
4307 the mode page and subpage.

4308 **11.4.2.1 Control Mode Page**

4309 The Control mode page provides controls over SCSI features that are applicable to all device types (e.g.,  
4310 task set management and error logging).

4311 Table 11.66 defines the Control mode page default value (PC = 10b).

4312 **Table 11.66 — Control Mode Page default value**

Bit Byte	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.								

4313 The following Control mode page field shall be changeable: SWP. The following Control mode page  
4314 fields are not changeable: TST and BUSY TIMEOUT PERIOD. Other fields may or may not be  
4315 changeable, refer to the vendor datasheet for details.

4316

4317    **11.4.2.1.1      Control Mode Page Parameters**

4318    **Table 11.67 — Control Mode Page Parameters**

<b>Byte</b>	<b>Bit</b>	<b>Description</b>
1	7:5	<b>TST:</b> Indicates Task Set Type. 000b indicates the logical unit maintains one task set for all I_T nexuses. Others: reserved.
4	3:3	<b>SWP:</b> 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	<b>BUSY TIMEOUT PERIOD:</b> 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.		

4319

#### 4320 **11.4.2.2 Read-Write Error Recovery Mode Page**

4321 The Read-Write Error Recovery mode page specifies the error recovery parameters the device server shall  
4322 use during any command that performs a read or write operation to the medium (e.g., READ command,  
4323 WRITE command, or VERIFY command).

4324 Table 11.68 defines the Read-Write Error Recovery mode page default value (PC = 10b).

4325 **Table 11.68 — Read-Write Error Recovery Mode Page default value**

<b>Bit Byte</b>	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.															

4326  
4327 This standard does not define which Read-Write Error Recovery mode page fields are changeable, refer to  
4328 vendor datasheet for details.

4329

4330 **11.4.2.2.1 Read-Write Error Recovery Parameters**

4331 **Table 11.69 — Read-Write Error Recovery Parameters**

Byte	Bit	Description
3	7:0	<b>READ RETRY COUNT:</b> 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	<b>WRITE RETRY COUNT:</b> 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	<b>RECOVERY TIME LIMIT:</b> 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.

4332

4333 **11.4.2.3 Caching Mode Page**

4334 The Caching mode page defines the parameters that affect the use of the cache. A UFS device shall  
4335 implement support for following parameters.

4336 Table 11.70 defines the Caching mode page default value (PC = 10b).

4337 **Table 11.70 — Caching Mode Page default value**

Bit Byte	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					(LSB)			
5							(LSB)			
6	(MSB)	MINIMUM PRE-FETCH = 0000h					(LSB)			
7							(LSB)			
8	(MSB)	MAXIMUM PRE-FETCH = 0000h					(LSB)			
9							(LSB)			
10	(MSB)	MAXIMUM PRE-FETCH CEILING = 0000h					(LSB)			
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					(LSB)			
15							(LSB)			
16		Reserved = 00h								
17										
18		Obsolete = 000000h								
19										

4338 The following Caching mode page fields shall be changeable: WCE and RCD. Other fields may or may  
4339 not be changeable, refer to the vendor datasheet for details.

4340

4341 **11.4.2.3.1 Caching Mode Page Parameters**

4342 **Table 11.71 — Caching Mode Page Parameters**

Byte	Bit	Description
2	2:2	<b>WCE:</b> 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	<b>RCD:</b> 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.		

4343

4344 **11.5 Vital product data parameters**

4345 **11.5.1 Overview**

4346 The vital product data (VPD) pages are returned by an INQUIRY command with the EVPD bit set to one  
4347 and contain vendor specific product information about a logical unit and SCSI target device.

4348 A UFS device shall support the following VPD pages:

- 4349 • Supported VPD Pages  
4350 • Mode Page Policy;

4351 Support for other VPD pages is optional.

4352 **11.5.2 VPD page format**

4353 Table 11.72 shows the VPD page structure.

4354 **Table 11.72 — VPD page format**

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								
4	(MSB) VPD parameters							
n								

4355

4356 The PERIPHERAL QUALIFIER field and the PERIPHERAL DEVICE TYPE field are the same as  
4357 defined for standard INQUIRY data (see 11.3.2.2).

4358 The PAGE CODE field identifies the VPD page and contains the same value as in the PAGE CODE field  
4359 in the INQUIRY CDB (see 11.3.2).

4360 The PAGE LENGTH field indicates the length in bytes of the VPD parameters that follow this field.

4361 See [SPC] for further details.

4362

4363 **11.5.3 Supported VPD Pages VPD page**

4364 The Supported VDP Pages VPD page contains a list of the VPD page codes supported by the logical unit  
4365 (see Table 11.73).

4366  
4367

**Table 11.73—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								

4368  
4369 The supported VPD page list shall contain a list of all VPD page codes implemented by the logical unit in  
4370 ascending order beginning with page code 00h.

#### 4371 11.5.4 Mode Page Policy VPD page

4372 The Mode Page Policy VPD page (see Table 11.74) indicates which mode page policy is in effect for  
4373 each mode page supported by the logical unit.

4374

**Table 11.74 — 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										

4375 Each mode page policy descriptor (see Table 11.75) contains information describing the mode page  
4376 policy for one or more mode pages or subpages.

4377

**Table 11.75 — 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							

4378 The POLICY PAGE CODE field and POLICY SUBPAGE CODE field indicate the mode page and  
4379 subpage to which the descriptor applies. See [SPC] for further details.

4380

4381 **11.5.4 Mode Page Policy VPD page (cont'd)**

4382 If more than one logical unit are configured in the device, a multiple logical units share (MLUS) bit set to  
4383 one indicates the mode page and subpage identified by the POLICY PAGE CODE field and POLICY  
4384 SUBPAGE CODE field is shared by more than one logical unit.

4385 A MLUS bit set to zero indicates the logical unit maintains its own copy of the mode page and subpage  
4386 identified by the POLICY PAGE CODE field and POLICY SUBPAGE CODE field.

4387 Table 11.76 describes the mode page policies.

4388 **Table 11.76 — MODE PAGE POLICY field**

Code	Description
00b	Shared
01b	Per target port
10b	Obsolete
11b	Per I_T nexus

4389 NOTE This standard defines only one target port and one initiator port.

4390 MODE PAGE POLICY field shall be set to zero (Shared).

4391 See [SPC] for further details about Mode Page Policy VPD page.

4392

---

4393    **12    UFS Security**

4394    This section summarizes UFS device security features and the implementation details. These features  
4395    include: Secure mode operation, data and register protection, RPMB and reset.

4396    **12.1    UFS Security Feature Support Requirements**

4397    The security features defined in this standard are mandatory for all devices.

4398    The following security features are defined: replay protected memory block (RPMB), secure mode and  
4399    different types of logical unit write protection.

4400    **12.2    Secure Mode**

4401    **12.2.1    Description**

4402    UFS devices will be used to store user's personal and/or corporate data information. The UFS device  
4403    provides a way to remove the data permanently from the device when requested, ensuring that it cannot  
4404    be retrieved using reverse engineering on the memory device.

4405    The UFS device shall support a secure and insecure mode of operation. In the secure mode all operations  
4406    that result in the removal or retiring of information on the device will purge this information in a secure  
4407    manner, as outlined in 12.2.2.1, Secure Removal.

4408    The secure mode is applied at the logical unit level, so different logical unit may have different secure  
4409    modes.

4410

4411 **12.2.2 Requirements**

4412 **12.2.2.1 Secure Removal**

4413 The way in which data is removed securely from the device is dependent on the type of memory  
4414 technology that is used to implement the UFS device. Three common methods that apply to most memory  
4415 types implemented at the time of this spec are:

- 4416 1) The device controller shall issue an erase operation to the addressed location.
- 4417 2) The device controller shall overwrite the addressed locations with a single character and erase the  
4418 device.
- 4419 3) The device controller shall overwrite the addressed locations with a character, its complement, then a  
4420 random character

4421 UFS devices shall support at least one secure removal method.

4422 **12.2.2.2 Erase Operation**

4423 Erase is an operation that moves data from the mapped address space to the unmapped address space.  
4424 Logical blocks where erase was applied will be set to the erased value of zero. This operation places no  
4425 requirement on what the device is required to do with the data in the unmapped address space. After an  
4426 erase is executed, software on the host should not be able to retrieve the erased logical block data.

4427 The minimum data range that an erase operates on is the logical block.

4428 **12.2.2.3 Discard Operation**

4429 Discard is a non-secure variant of the erase functionality. The distinction between discard and erase is the  
4430 device behavior where the device is not required to guarantee that host would not retrieve the original  
4431 data from one or more LBA's that were marked for discard when a read operation is directed to the  
4432 LBA's.

4433 **12.2.2.4 Purge Operation**

4434 The Purge operation shall be performed on physical blocks that are not being used to store logical block  
4435 data (e.g., physical blocks previously used to store logical block data). When the operation is executed it  
4436 results in removing all the data from such physical blocks. Note that data of LBA that were discarded  
4437 (bProvisioningType=02h) may not be removed. This is done in accordance with the  
4438 bSecureRemovalType parameter value of the Device Descriptor. This mode allows the host system to  
4439 protect against die level attacks.

4440

4441 **12.2.3 Implementation**

4442 **12.2.3.1 Erase**

4443 The erase functionality is implemented using the UNMAP command and it is enabled if the  
4444 bProvisioningType parameter in the Unit Descriptor is set to 03h (TPRZ = 1).

4445 The device behavior shall comply with the UNMAP definition in [SBC] when the TPRZ bit in the READ  
4446 CAPACITY(16) parameter data is set to one.

4447 As defined in [SBC],

- The UNMAP command causes a mapped LBA to transition from mapped state to deallocated state if  
4449 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  
4451 deallocated LBA shall return zero.
- The device server may maintain a deallocated LBA in deallocated state until a write operation  
4453 specifying that LBA is completed without error.
- Or, the device server may transition a deallocated LBA from deallocated state to mapped state at any  
4455 time (autonomous state transition). For UFS, if TPRZ bit is set to one and an autonomous transition to  
4456 the mapped state occurs, the LBA shall be mapped to a physical block(s) containing data with all bits  
4457 set to zero.

4458 LBA's to be erased may be aligned to multiples of the dEraseBlockSize parameter value, where it is  
4459 possible, to minimize performance impact. dEraseBlockSize is a parameter included in the Unit  
4460 Descriptor.

4461 **12.2.3.2 Discard**

4462 The discard functionality is implemented using the UNMAP command and it is enabled if the  
4463 bProvisioningType parameter in the Unit Descriptor is set to 02h (TPRZ = 0). The device behavior shall  
4464 comply with the UNMAP definition in [SBC] when the TPRZ bit in the READ CAPACITY(16)  
4465 parameter data is set to zero.

4466 As defined in [SBC],

- The UNMAP command causes a mapped LBA to transition from mapped state to deallocated state if  
4468 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  
4470 deallocated LBA may return any data.
- The device server may maintain a deallocated LBA in deallocated state until a write operation  
4472 specifying that LBA is completed without error.
- Or, the device server may transition a deallocated LBA from deallocated state to mapped state at any  
4474 time (autonomous state transition). For UFS, if TPRZ bit is set to zero and an autonomous transition  
4475 to the mapped state occurs, the LBA shall be mapped to a physical block(s) containing any data  
4476 including the original data before UNMAP operation.

4477 LBA's to be discarded may align to multiples of the dEraseBlockSize where possible to minimize  
4478 performance impact. dEraseBlockSize is a parameter included in the Unit Descriptor.

4479

4480 **12.2.3.3 Purge operation**

4481 The purge operation is implemented via Query Functions with Attributes and Flags. In particular, the  
4482 fPurgeEnable flag allows to enable or disable the execution of a purge operation, and the bPurgeStatus  
4483 attribute provides information about the operation status.

- 4484 • fPurgeEnable flag
  - 4485 ○ Write only volatile flag, set to zero after power on or reset.
  - 4486 ○ Purge operation is enabled when this flag is equal to one, otherwise it is disabled.
  - 4487 ○ This flag can only be set when the command queue of all logical units are empty.
  - 4488 ○ This flag is automatically cleared by the UFS device when the operation completes or an error  
4489 condition occurs.
  - 4490 ○ This flag can be cleared by the host to interrupt an ongoing purge operation.
- 4491 • bPurgeStatus attribute
  - 4492 ○ Read only attribute.
  - 4493 ○ This attribute can be set to one of the following values:
    - 4494 - 00h: Idle (purge operation disabled).
    - 4495 - 01h: Purge operation in progress.
    - 4496 - 02h: Purge operation stopped prematurely by the host.
    - 4497 - 03h: Purge operation completed successfully.
    - 4498 - 04h: Purge operation failed due to logical unit queue not empty
    - 4499 - 05h: Purge operation general failure.
  - 4500 Other values are reserved and shall not be set.
  - 4501 ○ bPurgeStatus is set to 00h (Idle) after power on or reset.
  - 4502 ○ When the host enables the purge operation setting fPurgeEnable flag to one, and if all logical unit  
4503 command queue are empty, the bPurgeStatus will be set to 01h to indicate that the purge  
4504 operation is in progress. The bPurgeStatus shall be set to 03h if the operation is completed  
4505 successfully, or to 05h if a failure occurred.
  - 4506 ○ The host should send a query request to set fPurgeEnable flag to one only if command queues are  
4507 empty. A query request to set fPurgeEnable flag which is processed when device command  
4508 queues are not empty may fail. If the request fails, Query Response field in the QUERY  
4509 RESPONSE UPIP shall be set to FFh (“General Failure”), the purge operation shall not start, and  
4510 the bPurgeStatus shall be set to 04h.
  - 4511 ○ If an ongoing purge operation is interrupted by the host setting the fPurgeEnable flag to zero, the  
4512 bPurgeStatus shall be set to 02h.
  - 4513 ○ When the bPurgeStatus is equal to the values 02h, 03h, 04h or 05h, the bPurgeStatus shall be  
4514 automatically cleared to 00h (Idle) the first time that it is read. The bPurgeStatus values of 00h  
4515 and 01h shall not be modified as a result of a read.

#### 4517 12.2.3.3 Purge operation (cont'd)

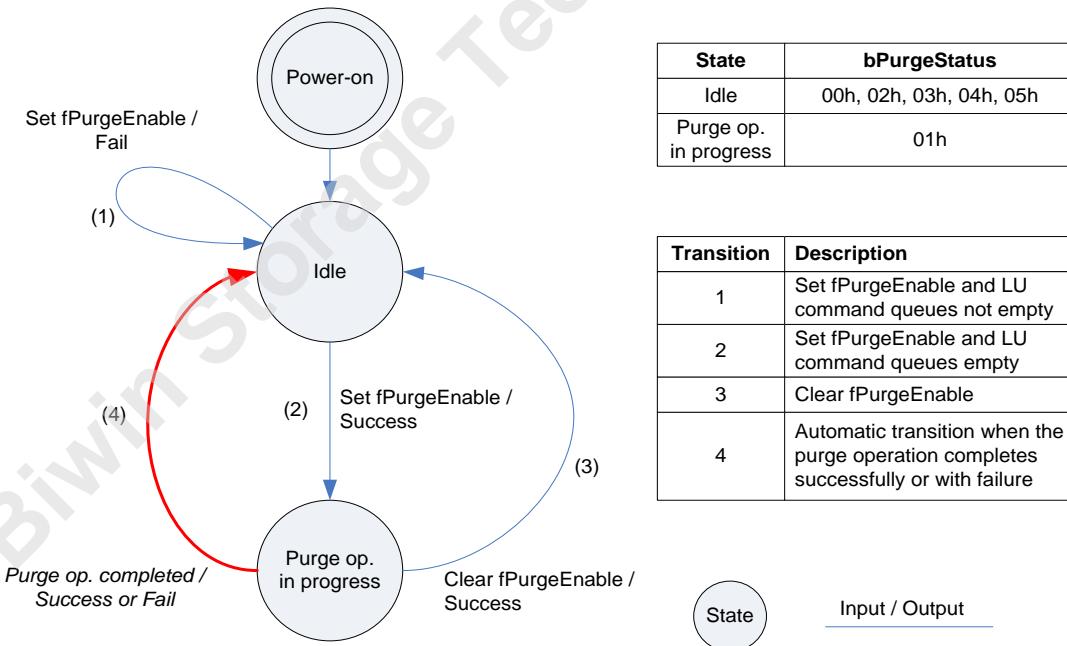
- 4518 If a purge operation is in progress ( $bPurgeStatus = 01h$ ) commands sent to any logical units or to the  
4519 RPMB well known logical unit will fail. The device shall return the sense key "NOT READY" to  
4520 show that the command failed because a purge operation was in progress. Descriptors, attributes and  
4521 flags may be read when a purge operation is in progress, while only  $fPurgeEnable$  flag may be  
4522 written. A query request to write descriptors, attributes or flags (except  $fPurgeEnable$ ) shall be  
4523 terminated with Query Response field set to "General Failure".
- 4524 If the host needs to execute a command urgently when a purge operation is in progress, it may  
4525 interrupt the purge operation. In particular, before issuing any command, the host sets  $fPurgeEnable$   
4526 flag to zero, waits until the device interrupts the operation, and then sets the  $bPurgeStatus$  attribute to  
4527 02h (purge operation stopped prematurely).
- 4528 If a power failure occurs the  $fPurgeEnable$  flag and  $bPurgeStatus$  attribute shall be reset to zero. In  
4529 this case the device will not indicate that operation failed.

4530 Figure 12.1 shows the Purge operation state machine. There are two states: "Idle" and "Purge Op. in  
4531 progress".

4532 After power on, the purge operation state is "Idle", and the purge operation is disabled.

4533 To enable the execution of a purge operation, the host sets  $fPurgeEnable$  flag to one sending a QUERY  
4534 REQUEST UPIU. If the setting is executed successfully, the state will transition to "Purge Op. in  
4535 progress" and the purge operation will start ( $bPurgeStatus = 01h$ ). If there is a least one logical unit with  
4536 command queue not empty, the setting of  $fPurgeEnable$  flag shall fail, the purge operation shall not start,  
4537 the state shall remain "Idle", and  $bPurgeStatus$  shall be set to 04h.

4538 When the purge operation is completed, the state will transition automatically to "Idle", and  $bPurgeStatus$   
4539 shall be set to 03h if the operation is completed successfully, or 05h in case of failure.



4540 NOTE 1 On each transition the input event (triggering the state transition) and the output of the transition itself are mentioned.  
4541

4542 **Figure 12.1 — Purge operation state machine**

4543 The host may interrupt an ongoing purge operation clearing the  $fPurgeEnable$  flag, when the operation  
4544 has been interrupted the state will transition to "Idle" and  $bPurgeStatus$  will be set to 02h.

4545 **12.2.3.4 Wipe Device**

4546 The wipe device operation is fulfilled issuing the FORMAT UNIT command to all enabled logical units.  
4547 If the logical unit is write protected using one of the methods described in 12.3, Device Data Protection,  
4548 or if the SWP bit in Control Mode Page is one, then the FORMAT UNIT command shall fail and the  
4549 content of the medium shall not be altered.

4550 A FORMAT UNIT command sent to the Device well known logical unit requests the device format all  
4551 enabled logical units except the RPMB well known logical unit. If any logical unit is write protected  
4552 when the FORMAT UNIT command is issued to Device well known logical unit, the FORMAT UNIT  
4553 command shall fail and the content of the medium shall not be altered.

4554 The fields of the FORMAT UNIT command should be set as described in the following:

- 4555 • The Format data (FMTDATA) bit shall be set to zero to specify that no parameter list will be provided.  
4556 • The DEFECT LIST FORMAT shall be set to 000b.  
4557 • The format protection information (FMTPIINFO) shall be set to 00b.  
4558 • The vendor specific byte shall be set to 00h.

4559 The UFS device shall ignore CMPLST and LONGLIST bits since FMTDATA is set to zero.

4560

4561 **12.2.3.5 bProvisioningType Parameter**

4562 Logical units can be configured in secure mode using bProvisioningType parameter of the Unit  
4563 Descriptor. This parameter allows to enable thin provisioning and define TPRZ bit value in the READ  
4564 CAPACITY parameter data.

4565 The secure mode is enabled if thin provisioning is enabled and TPRZ bit is equal to one. In this mode all  
4566 operations shall be performed using the mode defined by bSecureRemovalType parameter in the Device  
4567 Descriptor. Only one type of removal type can be defined for an entire device.

4568 bProvisioningType parameter can be set to the following values:

- 4569 • 00h: to disable thin provisioning  
4570 • 02h: to enable thin provisioning and set TPRZ to zero  
4571 • 03h: to enable thin provisioning and set TPRZ to one

4572 TPRZ bit value of zero indicates the device is in normal mode.

4573 As all other Unit Descriptor configurable parameters, the bProvisioningType value is set writing the  
4574 Configuration Descriptor. See 14.1.4.3, Configuration Descriptor, for details.

4575

4576

4577 **12.2.3.6 bSecureRemovalType Parameter**

4578 The bSecureRemovalType parameter within the Device Descriptor defines how information is removed  
4579 from the physical memory during a Purge operation. This parameter may be set during system integration  
4580 writing the Configuration Descriptors. bSecureRemovalType values are defined as follows:

- 4581 • Value of '03h' will result in the information being removed using a vendor defined mechanism.
- 4582 • Value of '02h' will result in all information being removed by overwriting the addressed locations  
4583 with a character, its complement, then a random character.
- 4584 • Value of '01h' will result in all information being removed by overwriting the addressed locations  
4585 with a single character followed by an erase.
- 4586 • Value of '00h' will result in all information being removed by an erase of the physical memory  
4587 (default).
- 4588 • Other values are reserved for future use and shall not be set.

4589 Device manufacturers are only required to support the mechanism required by their memory array  
4590 technology.

4591 For additional information please refer to the <http://www.killdisk.com/dod.htm> or to the following  
4592 documents for more details:

- 4593 ○ DoD 5220.22-M (<http://www.dtic.mil/whs/directives/corres/pdf/522022m.pdf>) and
- 4594 ○ NIST SP 800-88 ([http://csrc.nist.gov/publications/nistpubs/800-88/NISTSP800-88\\_rev1.pdf](http://csrc.nist.gov/publications/nistpubs/800-88/NISTSP800-88_rev1.pdf))

4595

4596 **12.3 Device Data Protection**

4597 **12.3.1 Description and Requirements**

4598 UFS device data content can be protected at the logical unit level. The following protection modes shall  
4599 be available:

- 4600 • Permanent write protection (permanent: once enabled it cannot be reversed)
- 4601 • Power on write protection (write protection can be cleared with a power cycle or a hardware reset  
4602 event)
- 4603 • Secure write protection (write protection can only be configured and enabled/disabled using secure  
4604 authenticated methods)

4605 These modes of write protections are not implemented in the RPMB well known logical unit.

4606 There shall also be a method to read the protection mode that is currently enabled for a logical unit.

4607 **12.3.2 Implementation**

4608 The protection mode can be defined at logical unit level configuring the bLUWriteProtect parameter of  
4609 the Unit Descriptor. The write protection modes are encoded as in the following:

- 4610 00h: Logical unit not write protected (or secure write protected if one or more secure write protect  
4611 entry is set)
- 4612 01h: Logical unit power on write protected
- 4613 02h: Logical unit permanently write protected

4614 A power on write protected logical unit (bLUWriteProtect = 01h) can be written only if fPowerOnWPEn  
4615 flag is equal to zero. The fPowerOnWPEn flag is set to zero after a power cycle or hardware reset event,  
4616 once it is set to one it cannot be toggled or cleared by the host.

4617 The fPermanentWPEn flag shall be set to one to enable the write protection of all permanently write  
4618 protected logical unit (bLUWriteProtect = 02h); logical unit can be written if fPermanentWPEn flag is  
4619 equal to 0b. The fPermanentWPEn flag is write once: it cannot be toggled or cleared once it is set. The  
4620 fPermanentWPEn flag shall be zero after device manufacturing.

4621 LBA areas within logical units may be write protected using the secure write protection mode.

4622 Secure write protect areas are configured setting the Secure Write Protect Configuration Block, and they  
4623 may only be created in logical units configured as “not write protected” (bLUWriteProtect=00h).

4624 One logical unit may have up to four secure write protect areas. However the total number of secure write  
4625 protect areas in a device shall not be more than bNumSecureWPArea.

4626 A secure write protect area can be written only if write protection is disabled in the related Secure Write  
4627 Protect Entry (WPF flag = 0b). See 12.4.3.1 for more details.

4628 It is recommended to set fPowerOnWPEn flag, fPermanentWPEn flag or WPF flag when all command  
4629 queues are empty, and wait device query response before enqueueing write command.

4630 If an LBA is write protected, then otherwise valid commands that request unmap, format or alteration of  
4631 the medium related to that LBA shall be rejected with CHECK CONDITION status with the sense key set  
4632 to DATA PROTECT.

4633 **12.4 RPMB**

4634 **12.4.1 Introduction**

4635 A signed access to a Replay Protected Memory Block is provided. This function provides means for the  
4636 system to store data to the specific memory area in an authenticated and replay protected manner. This is  
4637 provided by first programming authentication key information to the UFS device memory (shared secret).

4638 As the system cannot be authenticated yet in this phase the authentication key programming have to take  
4639 in a secure environment like in an OEM production. Further on the authentication key is utilized to sign  
4640 the read and write accesses made to the replay protected memory area with a Message Authentication  
4641 Code (MAC).

4642 Usage of random number generation and count register are providing additional protection against replay  
4643 of messages where messages could be recorded and played back later by an attacker.

4644 **12.4.2 RPMB Well Known Logical Unit Description**

4645 The RPMB is contained in a unique well known logical unit whose size is defined in the RPMB Unit  
4646 Descriptor. RPMB well known logical unit size shall be a multiple of 128 Kbytes, therefore its minimum  
4647 size is 128 Kbytes. The contents of the RPMB well known logical unit can only be read or written via  
4648 successfully authenticated read and write accesses. The data may be overwritten by the host but can never  
4649 be erased.

4650 All accesses to the RPMB will reference the specific RPMB well known logical unit number (W-LUN).

4651 RPMB well known logical unit may be configured into multiple RPMB regions where each RPMB region  
4652 has its own dedicated authentication key, write counter, result register, and logical address which starts  
4653 from zero. Refer to 13.2.3 to see how to configure RPMB well known logical unit into multiple RPMB  
4654 regions.

4655 Each RPMB region can process a single RPMB authenticated operation at any given point in time where  
4656 a single authenticated RPMB operation corresponds to whole Figure 12.2 Authentication Key Programming  
4657 Flow, Figure 12.3 Read Counter Value Flow, or Figure 12.6 Authenticated Secure Write Protect  
4658 Configuration Block Read Flow, and so on as listed in Table 12.5 Request Messge Types. For example,  
4659 the Initiator 3 may start an authenticated operation in RPMB Region 0 and Initiator 4 in RPMB Region 1.  
4660 The RPMB Region 0 should be accessed by a new authenticated operation request after completing the  
4661 authenticated operation started by Initiator 3; and RPMB Region 1 after completing the request from the  
4662 Initiator 4.

4663

4664

4665 **12.4.3 Requirements**

4666 **12.4.3.1 RPMB Resources**

- 4667 • Authentication Key
  - 4668 ○ Type: Write once, not erasable or readable
  - 4669 ○ Size: 32 bytes
  - 4670 ○ Description: Authentication key register which is used to authenticate accesses when MAC is calculated. Each RPMB region has a dedicated authentication key.
- 4672 • Write Counter
  - 4673 ○ Type: Read only
  - 4674 ○ Size: 4 bytes
  - 4675 ○ 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). Each RPMB region has dedicated write counter.
- 4681 • Result Register
  - 4682 ○ Type: Read only
  - 4683 ○ Size: 2 bytes
  - 4684 ○ Description: This register provides the result of an authenticated operation. Result register values are described in section 12.4.3.6. Each RPMB region has dedicated result register.
- 4686 • RPMB Data Area
  - 4687 ○ Type: Readable and writable
  - 4688 ○ Size: Multiples of 128 Kbytes defined in RPMB Unit Descriptor
    - 4689 ■ 128 Kbytes minimum, 16 Mbytes maximum.
    - 4690 ■ Each RPMB region size is defined as bRPMBRegion0Size – bRPMBRegion3Size in the RPMB Unit Descriptor.
  - 4692 ○ 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.
- 4694 • Secure Write Protect Configuration Block
  - 4695 ○ Type: Readable and writable
  - 4696 ○ Size : 256 Bytes
  - 4697 ○ Description: Secure Write Protect Configuration Block is supported by RPMB region 0 only. 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.

4703  
4704

### 12.4.3.1 RPMB Resources (cont'd)

Table 12.1 — Secure Write Protect Configuration Block

Bit Byte <sup>(1)</sup>	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.								

4705

4706 **12.4.3.1 RPMB Resources (cont'd)**

4707 **a) LUN**

4708 The LUN field indicates the logical unit to which secure write protection shall apply. Valid values are  
4709 from 0 to the number of LU specified by bMaxNumberLU.

4710 **b) DATA LENGTH**

4711 The DATA LENGTH field specifies the length in bytes of the Secure Write Protect Entries (0 for no  
4712 entry, 16 for one entry, 32 for two entries, 48 for three entries and 64 for four entries). In a write  
4713 request, the device shall ignore the bytes from DATA LENGTH + 16 to 255 and set these bytes of the  
4714 Secure Write Protect Configuration Block to zero.

4715 **c) Secure Write Protect Entry 0 to Entry 3**

4716 The Secure Write Protect Configuration Block may contain only the Entry 0, the Entries 0 and 1, the  
4717 Entries 0, 1 and 2, or all four Entries. If the Secure Write Protect Configuration Block does not  
4718 contain any entry (DATA LENGTH = 00h), all entries in the specified logical unit will be removed.

4719 Table 12.2 defines the structure of Secure Write Protect Entry.

4720 **Table 12.2 — Secure Write Protect Entry**

Bit Byte <sup>(1)</sup>	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)

4723 **12.4.3.1 RPMB Resources (cont'd)**

4724 **d) WPT (Write Protect Type)**

4725 The write protect type field (WPT) specifies how WPF bit may be modified.

4726 **Table 12.3 — Write Protect Type field**

<b>Code</b>	<b>Definition</b>
00b	NV-type WPF bit is persistent through power cycle and hardware reset. WPF 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

4727 **e) WPF (Write Protect Flag)**

4728 0b : Secure Write Protection is disabled.

4729 1b : Secure Write Protection is enabled.

4730 A WPF set to one specifies that the logical unit shall inhibit alteration of the medium for LBA within  
4731 the range indicated by LOGICAL BLOCK ADDRESS field and NUMBER OF LOGICAL BLOCKS  
4732 field. Commands requiring writes to the medium shall be terminated with CHECK CONDITION  
4733 status, with the sense key set to DATA PROTECT, and the additional sense code set to WRITE  
4734 PROTECTED.

4735 Logical units that contain cache shall write all cached logical blocks to the medium (e.g., as they  
4736 would do in response to a SYNCHRONIZE CACHE command with the LOGICAL BLOCK  
4737 ADDRESS field and the NUMBER OF LOGICAL BLOCKS field set to the values indicated in the  
4738 Secure Write Protect Entry) prior to enabling the write protection.

4739 A WPF bit set to zero specifies that the logical unit may allow writing to the medium, depending on  
4740 other write inhibit mechanisms implemented by the logical unit.

4741 WPF shall be set to zero after device manufacturing.

4742 **f) LOGICAL BLOCK ADDRESS**

4743 This field specifies the LBA of the first logical block of the Secure Write Protect area.

4744 **g) NUMBER OF LOGICAL BLOCKS**

4745 This field specifies the number of contiguous logical blocks that belong to the Secure Write Protect  
4746 area.

4747 If the NUMBER OF LOGICAL BLOCKS field is set to zero, then the secure write protection shall  
4748 apply to the entire logical unit. In that case, only Entry-0 needs to be configured to enable secure  
4749 write protection for the entire logical unit.

4750 **12.4.3.2 Algorithm and Key for MAC Calculation**

4751 The message authentication code (MAC) is calculated using HMAC SHA-256 as defined in [HMAC-  
4752 SHA]. The HMAC SHA-256 calculation takes as input a key and a message. The resulting MAC is 256  
4753 bits (32 bytes), which are embedded in the data frame as part of the request or response.

4754 The key used for the MAC calculation is always the 256-bit Authentication Key stored in the target  
4755 RPMB region. The message used as input to the MAC calculation is the concatenation of the fields in the  
4756 RPMB packet.

4757

4758 **12.4.3.3 RPMB Message Components**

4759 Each RPMB message includes specific components. These components are displayed in Table 12.4.

4760 **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. See 12.4.3.6.
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 region.
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.

4761 **12.4.3.4 Request Message Types**

4762 The following request message types are defined to support RPMB. These messages are sent from the  
4763 host to the device.

- 4764 • Authentication Key programming request  
4765 • Write Counter read request  
4766 • Authenticated data write request  
4767 • Authenticated data read request  
4768 • Result read request  
4769 • Secure Write Protect Configuration Block write request  
4770 • Secure Write Protect Configuration Block read request

4771 Table 12.5 defines the Request Message Type codes for the various messages.

4772 **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

4773

4774 **12.4.3.5 Response Message Types**

4775 The following response message types are defined to support RPMB. These messages are sent from the  
4776 device to the host.

- 4777 • Authentication Key programming response  
4778 • Write Counter read response  
4779 • Authenticated data write response  
4780 • Authenticated data read response  
4781 • Secure Write Protect Configuration Block write response  
4782 • Secure Write Protect Configuration Block read response

4783 Table 12.6 defines the Response Message Type codes for the various messages.

4784  
4785

**Table 12.6 — Response Message Types**

<b>Code</b>	<b>Response Message Types</b>
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

4786

4787 **12.4.3.6 RPMB Operation Result**

- 4788 • Result component of an RPMB message is composed of two bytes. The most significant byte is  
4789 reserved and shall be set to zero.
- 4790 • Bit 7 of Result field shall indicate if the write counter has expired (i.e., reached its maximum  
4791 value) or not
- 4792     ○ Value of one will represent an expired write counter
- 4793     ○ Value of zero will represent a valid write counter
- 4794 • The other bits shall indicate the operation status
- 4795     ○ Operation Okay (00h)
- 4796         ■ General Failure (01h)
- 4797     ○ Authentication Failure (02h)
- 4798         ■ MAC comparison not matching, MAC calculation failure
- 4799     ○ Counter Failure (03h)
- 4800         ■ Counters not matching in comparison, counter incrementing failure
- 4801     ○ Address Failure (04h)
- 4802         ■ Address out of range, wrong address alignment
- 4803     ○ Write Failure (05h)
- 4804         ■ Data, Counter or Result write failure
- 4805     ○ Read Failure (06h)
- 4806         ■ Data, Counter or Result read failure
- 4807     ○ Authentication key not yet programmed (07h)
- 4808         ■ This value is the only valid result until the Authentication Key has been  
4809             programmed in the target RPMB region, after which it can never occur again
- 4810     ○ Secure Write Protect Configuration Block access failure (08h).
- 4811         ■ Secure Write Protect Configuration read or write failure.
- 4812     ○ Invalid Secure Write Protect Block Configuration parameter (09h).
- 4813         ■ Invalid LUN (or logical unit not enabled), DATA LENGTH, LOGICAL BLOCK  
4814             ADDRESS, NUMBER OF LOGICAL BLOCKS, or overlapped areas.
- 4815     ○ Secure Write Protection not applicable (0Ah).
- 4816         ■ Logical unit configured with other write protection modes (permanent or power-on)

4817 **Table 12.7 — Result data structure**

Bit[15:8]	Bit[7]	Bit[6:0]
Reserved	Write Counter Status	Operation Status

4820 **12.4.3.6 RPMB Operation Result (cont'd)**

4821 **Table 12.8 — Result code definition**

Code	Description
0000h (0080h)	Operation OK
0001h (0081h)	General failure
0002h (0082h)	Authentication failure <ul style="list-style-type: none"><li>• MAC comparison not matching, MAC calculation failure</li></ul>
0003h (0083h)	Counter failure <ul style="list-style-type: none"><li>• Counters not matching in comparison, counter incrementing failure</li></ul>
0004h (0084h)	Address failure <ul style="list-style-type: none"><li>• Address out of range, wrong address alignment</li></ul>
0005h (0085h)	Write failure <ul style="list-style-type: none"><li>• Data / Counter / Result write failure</li></ul>
0006h (0086h)	Read failure <ul style="list-style-type: none"><li>• Data / Counter / Result read failure</li></ul>
0007h	Authentication Key not yet programmed. <ul style="list-style-type: none"><li>• 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.</li></ul>
0008h (0088h)	Secure Write Protect Configuration Block access failure <ul style="list-style-type: none"><li>• Secure Write Protect Configuration read or write failure</li></ul>
0009h (0089h)	Invalid Secure Write Protect Block Configuration parameter <ul style="list-style-type: none"><li>• Invalid LUN or logical unit not enabled, DATA LENGTH, LOGICAL BLOCK ADDRESS, NUMBER OF LOGICAL BLOCKS, or overlapped areas</li></ul>
000Ah (008Ah)	Secure Write Protection not applicable <ul style="list-style-type: none"><li>• Logical unit configured with other write protection modes (permanent or power-on)</li></ul>
NOTE The values in parenthesis are valid when Write Counter has expired.	

4823 **12.4.4 Implementation**

4824 **12.4.4.1 RPMB Message**

4825 An RPMB Message may be composed of one or more RPMB Message Data Frames.

4826 RPMB Message Data Frame size is 512 bytes and it is organized as shown in Table 12.9.

4827 **Table 12.9 — RPMB Message Data Frame**

Bit Byte	7	6	5	4	3	2	1	0
0	(MSB)							
195								(LSB)
196	(MSB)							
227								(LSB)
228					Data [255]			
483					Data [0]			
484	(MSB)							
499								(LSB)
500	(MSB)							
503					Write Counter			(LSB)
504	(MSB)							
505								(LSB)
506	(MSB)							
507					Block Count			(LSB)
508	(MSB)							
509								(LSB)
510	(MSB)							
511					Request Message Type / Response Message Type			(LSB)

4828

4829 **12.4.4.2 MAC Calculation**

4830 The key used for the MAC calculation is always the 256 bit Authentication Key stored in the device.  
4831 Input to the MAC calculation is the concatenation of the fields in the RPMB Message Data Frames from  
4832 byte 228 to byte 511 (stuff bytes and the MAC are excluded).

4833 If RPMB Message is composed by several RPMB Message Data Frames then the input message to MAC  
4834 is the concatenation of bytes [228:511] of each data frame in the order in which the data frames are sent.  
4835 The MAC is valid only in the last data frame.

4836

4837 **12.4.4.3 RPMB Message Data Frame Delivery**

4838 The RPMB messages are delivered using SCSI security protocol commands:

- 4839 • SECURITY PROTOCOL IN is used to send request messages to the device  
4840 • SECURITY PROTOCOL OUT is used to request to the device the sending of response messages.

4841

4842 **12.4.5 SECURITY PROTOCOL IN/OUT Commands**

4843 SECURITY PROTOCOL IN command and SECURITY PROTOCOL OUT command defined in [SPC]  
4844 are used to encapsulate and deliver data packets of any security protocol between host and device without  
4845 interpreting, dis-assembling or re-assembly the data packets for delivery.

4846 The SECURITY PROTOCOL IN command and SECURITY PROTOCOL OUT command contain a  
4847 SECURITY PROTOCOL field. A unique security protocol ID is assigned by T10 for JEDEC UFS  
4848 application.

- 4849 • SECURITY PROTOCOL = ECh (JEDEC Universal Flash Storage)

4850

4851 **12.4.5.1 CDB format of SECURITY PROTOCOL IN/OUT commands**

4852 **Table 12.10 — CDB format of SECURITY PROTOCOL IN/OUT commands**

Bit Byte	7	6	5	4	3	2	1	0															
0	OPERATION CODE <sup>(1)</sup>																						
1	SECURITY PROTOCOL																						
2	SECURITY PROTOCOL SPECIFIC																						
3																							
4	INC_512 = 0b	Reserved																					
5	Reserved																						
6	(MSB)	ALLOCATION / TRANSFER LENGTH <sup>(2)</sup>																					
9																							
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.																							

4853 The RPMB well known logical unit shall support the following SECURITY PROTOCOL field values:

- 4854 • 00h: Security protocol information  
4855 • ECh: JEDEC Universal Flash Storage (security protocol ID assigned for JEDEC UFS application)

4856 Other values are invalid.

4857 SECURITY PROTOCOL IN/OUT commands shall consider the unique Security Protocol ID assigned to  
4858 JEDEC UFS application as the only valid Security Protocol ID.

4859 When the SECURITY PROTOCOL field is set to ECh (i.e., the JEDEC Universal Flash Storage),

- 4860 • INC\_512 bit shall be set to zero to specify that the ALLOCATION LENGTH or the TRANSFER  
4861 LENGTH field expresses the number of bytes to be transferred.  
4862 • If the ALLOCATION LENGTH field in a SECURITY PROTOCOL IN command is not equal to an  
4863 integer multiple of 512, then the command shall be terminated with CHECK CONDITION status.  
4864 • If the TRANSFER LENGTH field in a SECURITY PROTOCOL OUT command is not equal to an  
4865 integer multiple of 512, then the command shall be terminated with CHECK CONDITION status.  
4866 • The SECURITY PROTOCOL SPECIFIC field specifies the RPMB Protocol ID.

4867 **12.4.5.1 CDB format of SECURITY PROTOCOL IN/OUT commands (cont'd)**

4868 The RPMB Protocol ID indicates the RPMB region as defined in Table 12.11

4869 **Table 12.11 — SECURITY PROTOCOL SPECIFIC field for protocol ECh**

SECURITY PROTOCOL SPECIFIC: RPMB Protocol ID		Description
CDB Byte 2	CDB Byte 3	
00h	01h	RPMB Region 0
01h	01h	RPMB Region 1
02h	01h	RPMB Region 2
03h	01h	RPMB Region 3
Other values		Reserved

4870 If the SECURITY PROTOCOL SPECIFIC field is set to an invalid value or the corresponding RPMB  
4871 region is not enabled, then SECURITY PROTOCOL IN/OUT command shall be terminated with  
4872 CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST and the additional sense  
4873 code set to INVALID FIELD IN CDB.

4874 Secure Write Protect Configuration Block write and Secure Write Protect Configuration Block read  
4875 request are supported only by RPMB region 0.

4876 As required by [SPC], the SECURITY PROTOCOL value of 00h (security protocol information) shall be  
4877 supported if the SECURITY PROTOCOL IN command is supported by the device. The security protocol  
4878 information security protocol (i.e., the SECURITY PROTOCOL field set to 00h in a SECURITY  
4879 PROTOCOL IN command) is used to transfer security protocol related information from the logical unit.

4880 When the SECURITY PROTOCOL field is set to 00h in a SECURITY PROTOCOL IN command, the  
4881 two bytes SECURITY PROTOCOL SPECIFIC field shall contain a numeric value as defined in Table  
4882 12.11.

4883 **Table 12.12 — Security Protocol specific field values for protocol 00h**

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	

4884

#### 4885 12.4.5.2 Supported security protocols list description

4886 According to [SPC], if the SECURITY PROTOCOL field is set to 00h and the SECURITY PROTOCOL  
4887 SPECIFIC field is set to 0000h in a SECURITY PROTOCOL IN command, the parameter data shall have  
4888 the format shown in Table 12.13.

4889 **Table 12.13 — Supported security protocols SECURITY PROTOCOL IN parameter data**

Bit Byte	7	6	5	4	3	2	1	0
0								
5								Reserved
6	(MSB)							
7								SUPPORTED SECURITY PROTOCOL LIST LENGTH (m-7) (LSB)
8								SUPPORTED SECURITY PROTOCOL [first] (00h)
								.
m								SUPPORTED SECURITY PROTOCOL [last]
m+1								
n								Pad Bytes (optional)

4890 Security protocol information (00h) and the JEDEC Universal Flash Storage (ECh) are the only valid  
4891 security protocol ID's supported by the RPMB well known logical unit, therefore Table 12.13 shall be  
4892 implemented as defined in Table 12.14.

4893 **Table 12.14 — UFS Supported security protocols SECURITY PROTOCOL IN parameter data**

Bit Byte	7	6	5	4	3	2	1	0
0								
5								Reserved
6	(MSB)							
7								0002h (SUPPORTED SECURITY PROTOCOL LIST LENGTH) (LSB)
8								00h (Security protocol information)
9								ECh (JEDEC Universal Flash Storage)
10								
n								Pad bytes (optional)

4894

4895 **12.4.5.3 Certificate data description**

4896 If the SECURITY PROTOCOL field is set to 00h and the SECURITY PROTOCOL SPECIFIC field is  
4897 set to 0001h in a SECURITY PROTOCOL IN command, the parameter data shall have the format shown  
4898 in Table 12.14.

4899 **Table 12.15 — Certificate data SECURITY PROTOCOL IN parameter data**

Byte	Bit 7	6	5	4	3	2	1	0
0								
1								
2	(MSB)							
3								(LSB)
4								
m								
m+1								
n								

4900 The Device Server does not have a certificate to transfer, the CERTIFICATE LENGTH field shall be set  
4901 to 0000h. therefore Table 12.15 shall be implemented as defined in Table 12.16.

4902 **Table 12.16 — UFS certificate data SECURITY PROTOCOL IN parameter data**

Byte	Bit 7	6	5	4	3	2	1	0
0								
1								
2	(MSB)							
3								(LSB)
4								
n								

4903

4904    **12.4.6 RPMB Operations**

4905    **12.4.6.1 Request Type Message Delivery**

- 4906    • Only one RPMB operation can be executed at any time.
- 4907      • An initiator sends request type message to RPMB well known logical unit to request the  
4908       execution of an operation.
- 4909    • To deliver a request type message, the initiator sends a SECURITY PROTOCOL OUT command  
4910       with SECURITY PROTOCOL field is set to ECh (i.e., the JEDEC Universal Flash Storage) and  
4911       indicating the target RPMB region in the SECURITY PROTOCOL SPECIFIC field.
- 4912    • For an authenticated data write request, the data to be written into the RPMB data area is included in  
4913       the request message. The maximum data size in a single Authenticated Data Write request is equal to  
4914        $bRPMB\_ReadWriteSize \times 256$  bytes; multiple Authenticated Data Write operations should be  
4915       executed if the desired data size exceeds this value.
- 4916    • For SECURITY PROTOCOL OUT command, the Flags.W in the COMMAND UPIU is set to one  
4917       since data is transferred from the host to the device.
- 4918    • Table 12.17 defines the Expected Data Transfer Length field value in the COMMAND UPIU for the  
4919       various cases.

4920            **Table 12.17 — Expected Data Transfer Length value for Request Type Messages**

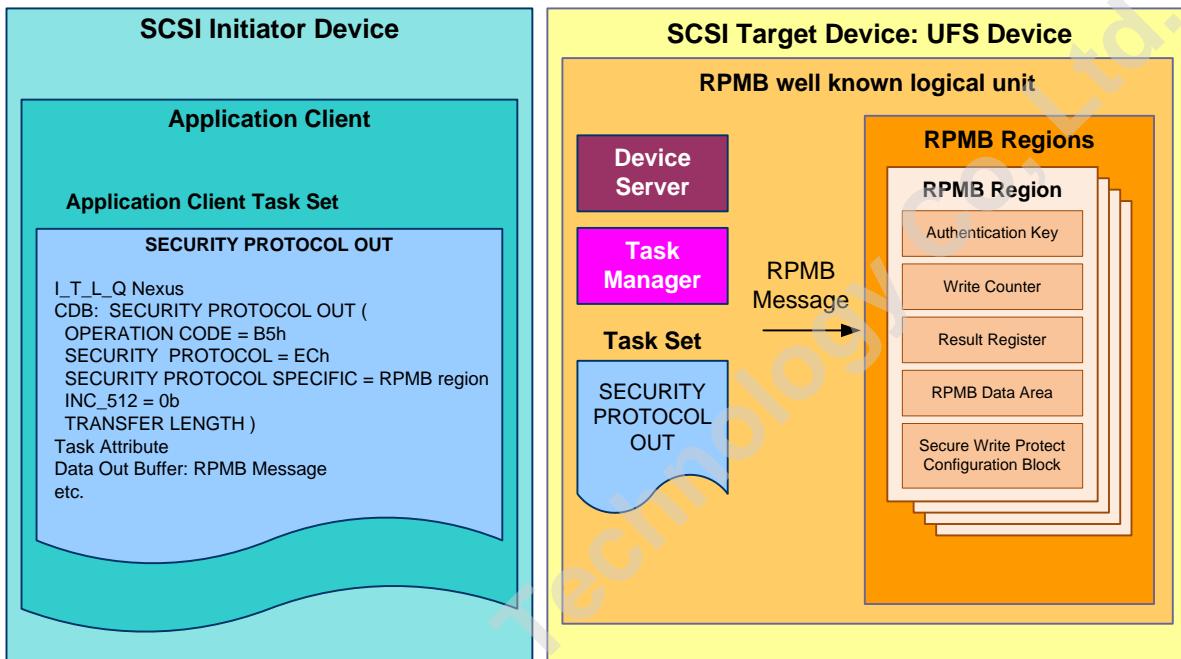
RPMB Message	Value
Authentication Key programming request, Result read request, Write Counter read request, Authenticated data read request, Secure Write Protect Configuration Block write request, Secure Write Protect Configuration Block read request	512
Authenticated data write request	$512 \times \text{Block Count}$

NOTE Block Count is equal to the data size divided by 256.

- 4921    • The device indicates to the host that it is ready to receive the request type message sending READY  
4922       TO TRANSFER UPIU. If the Expected Data Transfer Length is 512 byte, then Data Buffer Offset  
4923       field shall be set to a value of zero and Data Transfer Count field shall be set to a value of 512.
- 4924    • The number of bytes requested in a single READY TO TRANSFER UPIU shall not be greater than  
4925       the value indicated by bMaxDataOutSize attribute. A single READY TO TRANSFER UPIU may  
4926       request the transfer of one or more RPMB Messages.
- 4927    • In response to each READY TO TRANSFER UPIU, the host delivers the requested portion of the  
4928       message sending DATA OUT UPIU. See 10.7.13 for details about data transfer.

4929 **12.4.6.1 Request Type Message Delivery (cont'd)**

- 4930 • To complete the SECURITY PROTOCOL OUT command, the device returns a RESPONSE UPIU  
4931 with the status.
- 4932 • Figure 12.2 depicts a request type message delivery. The application client loads the RPMB Message  
4933 in the Data Out Buffer and indicates the target RPMB Region in SECURITY PROTOCOL SPECIFIC  
4934 field.



4935 **Figure 12.2 — Request type message delivery**

4936  
4937

4938 **12.4.6.2 Response Type Message Delivery**

- 4939 • A initiator requests the RPMB well known logical unit to send a response type message to retrieve the  
4940 result of a previous operation, to retrieve the Write Counter, to retrieve data from the RPMB data  
4941 area, or to retrieve the contents of a Secure Write Protect Configuration Block.
- 4942 • To request the delivery of a response type message, the host sends a SECURITY PROTOCOL IN  
4943 command with SECURITY PROTOCOL field is set to ECh (i.e., the JEDEC Universal Flash  
4944 Storage) and indicating the RPMB region in the SECURITY PROTOCOL SPECIFIC field.
- 4945 • For an authenticated data read the data from the RPMB data area is included in the response message.
- 4946 • For SECURITY PROTOCOL IN command, the Flags.R in the COMMAND UPIU is set to one since  
4947 data is transferred from the device to the host.
- 4948 • Table 12.18 defines the Expected Data Transfer Length field value in the COMMAND UPIU for the  
4949 various cases.

4950 **Table 12.18 — Expected Data Transfer Length value for Response Type Messages**

RPMB Message	Value
Response for Result read request <ul style="list-style-type: none"><li>- Authentication Key programming response</li><li>- Authenticated data write response</li><li>- Secure Write Protect Configuration Block write response</li></ul>	512
Write Counter read response	
Secure Write Protect Configuration Block read response	
Authenticated data read response	512 × Block Count
NOTE Block Count is equal to the data size divided by 256.	

- 4951 • The device returns the result or data requested in the RPMB message. The RPMB message is  
4952 delivered by sending one or more DATA IN UPIU in the data phase. A single DATA IN UPIU may  
4953 deliver one or more RPMB Messages.
- 4954 • The data size in DATA IN UPIU shall not exceed the value indicated by bMaxDataInSize attribute.
- 4955 • To complete the SECURITY PROTOCOL IN, the device sends a RESPONSE UPIU with the status.
- 4956 • Figure 12.3 depicts a response type message delivery. An application client requests a RPMB  
4957 Region to transfer the RPMB Message in the Data In Buffer specifying the RPMB Region ID in  
4958 SECURITY PROTOCOL SPECIFIC field of the CDB.

4959

4960 12.4.6.2 Response Type Message Delivery (cont'd)

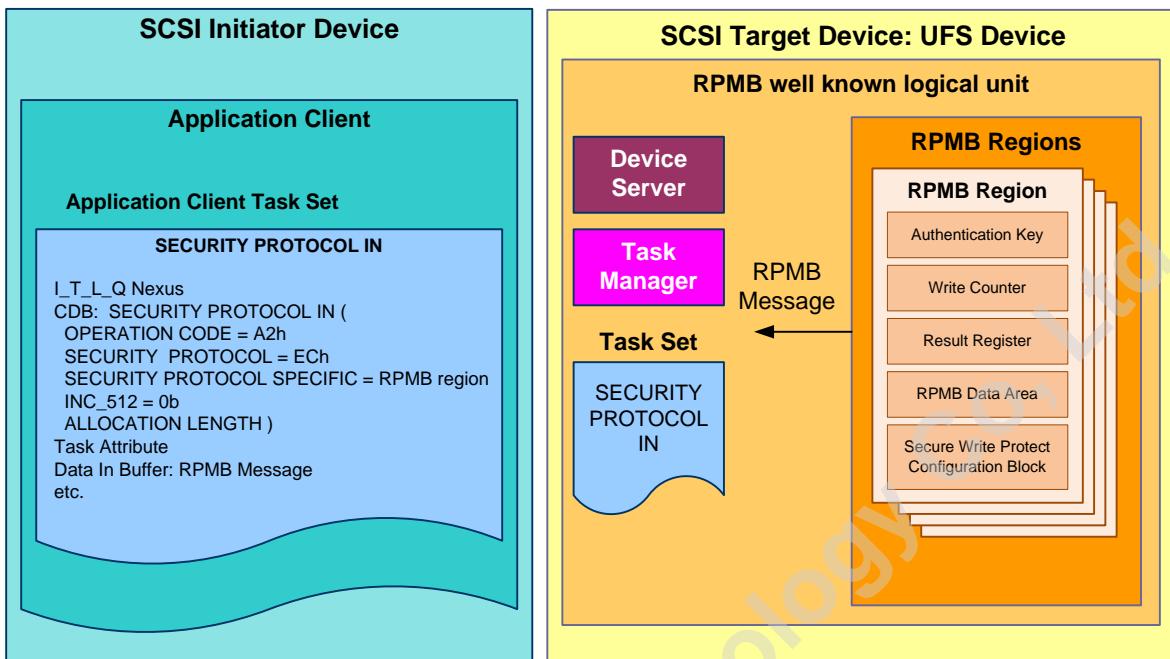


Figure 12.3 — Response Type Message delivery

4961  
4962  
4963

4964   **12.4.6.3 Authentication Key Programming**

- 4965   • The Authentication Key programming is initiated by a SECURITY PROTOCOL OUT command
- 4966      ○ An initiator sends the SECURITY PROTOCOL OUT command with SECURITY PROTOCOL
- 4967        field set to ECh and indicating the RPMB region in the SECURITY PROTOCOL SPECIFIC
- 4968        field. The RPMB data frame includes the Request Message Type = 0001h and the Authentication
- 4969        Key.
- 4970      ○ The device returns GOOD status in status response when Authentication Key programming is
- 4971        completed.
- 4972   • The Authentication Key programming verification process starts by issuing a SECURITY
- 4973        PROTOCOL OUT command
- 4974      ○ An initiator sends a SECURITY PROTOCOL OUT command with SECURITY PROTOCOL
- 4975        field set to ECh and indicating the RPMB region in the SECURITY PROTOCOL SPECIFIC
- 4976        field. The RPMB data frame contains the Request Message Type = 0005h (Result read request).
- 4977        Note that any request other than the Result read request from any initiator will overwrite the
- 4978        Result register of the RPMB Region.
- 4979      ○ The device returns GOOD status in status response when the operation result is ready for
- 4980        retrieval.
- 4981   • An initiator retrieves the operation result by issuing a SECURITY PROTOCOL IN command.
- 4982      ○ The SECURITY PROTOCOL field is set to ECh and the SECURITY PROTOCOL SPECIFIC
- 4983        field indicates the RPMB region.
- 4984      ○ Device returns the RPMB data frame containing the Response Message Type = 0100h and the
- 4985        Result code.
- 4986      ○ If programming of Authentication Key failed then returned result is “Write failure” (0005h). If
- 4987        some other error occurred during Authentication Key programming then returned result is
- 4988        “General failure” (0001h).

4989   Access to RPMB data area is not possible before the Authentication Key is programmed in the

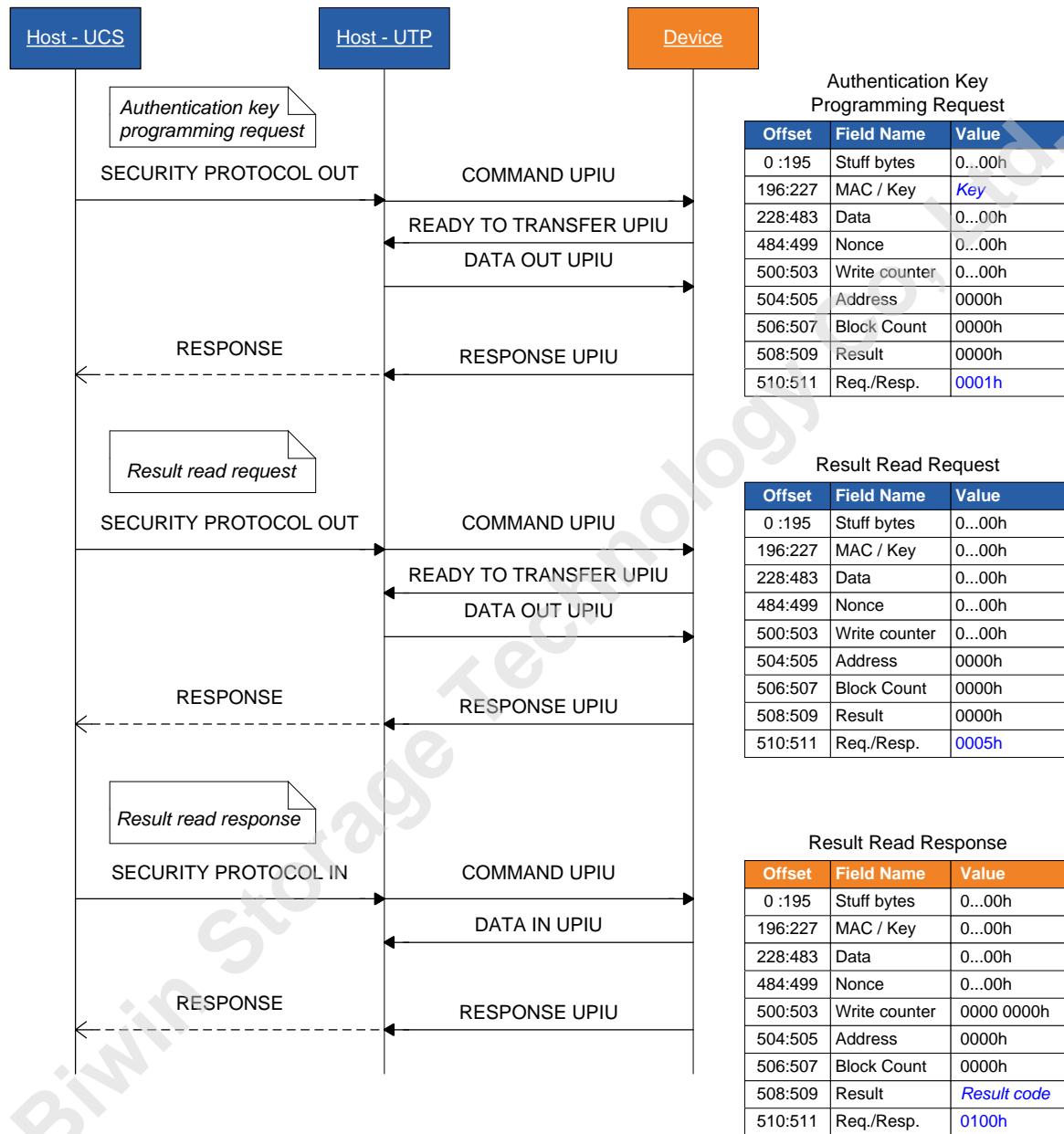
4990    corresponding RPMB region. The state of the device can be checked by trying to write/read data to/from

4991    the RPMB data area: if the Authentication Key is not programmed then the Result field in the response

4992    message will be set to “Authentication Key not yet programmed” (0007h).

4993

4994 12.4.6.3 Authentication Key Programming (cont'd)



4995

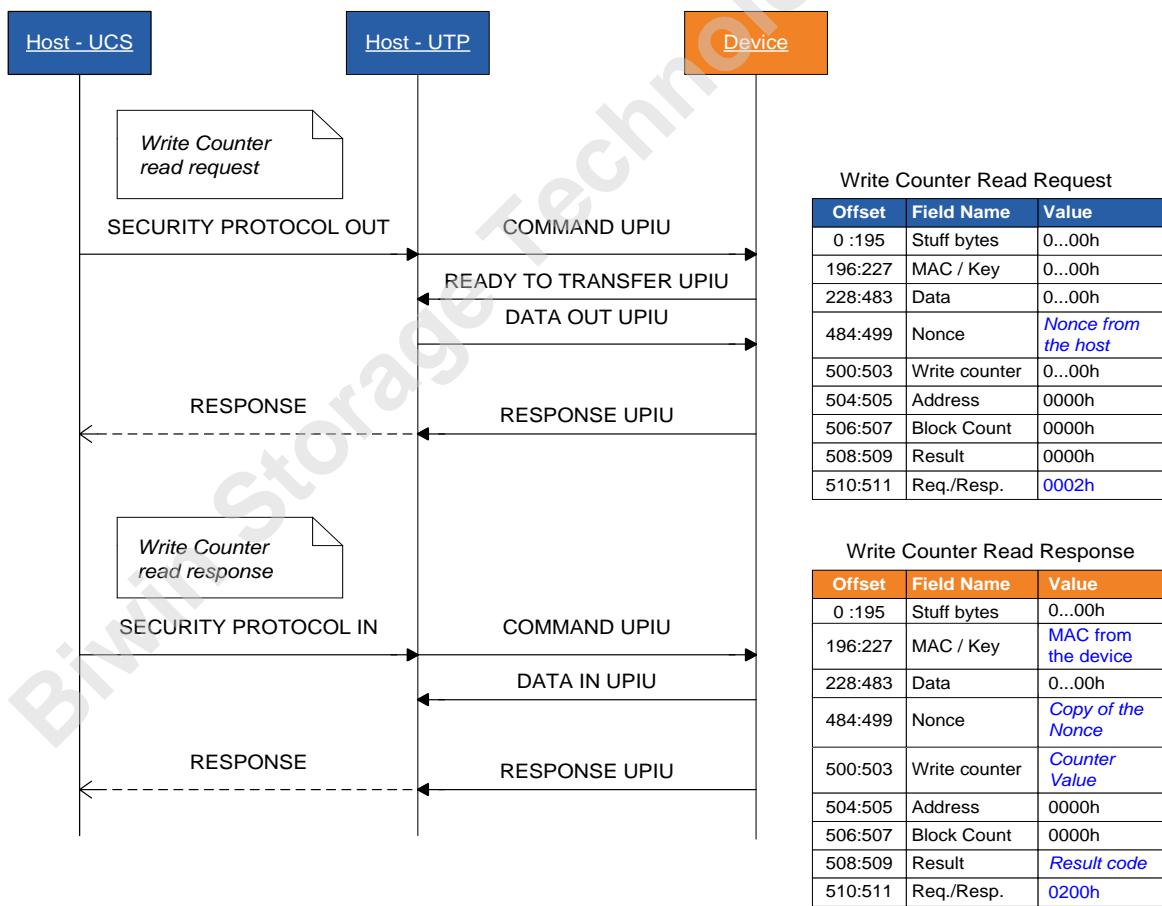
4996

4997

Figure 12.4 — Authentication Key Programming Flow

#### 4998 12.4.6.4 Read Counter Value

- 4999 • The Read Counter Value sequence is initiated by a SECURITY PROTOCOL OUT command.
  - 5000 ○ An initiator sends the SECURITY PROTOCOL OUT command with SECURITY PROTOCOL field set to ECh and indicating the RPMB region in the SECURITY PROTOCOL SPECIFIC field. The RPMB data frame includes the Request Message Type = 0002h and the Nonce.
- 5003 • When a GOOD status in the status response is received, the write counter value is retrieved sending a SECURITY PROTOCOL IN command.
  - 5005 ○ An initiator sends the SECURITY PROTOCOL IN command with the SECURITY PROTOCOL field is set to ECh and indicating the RPMB region in the SECURITY PROTOCOL SPECIFIC field.
  - 5008 ○ 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.
- 5010 If reading of the counter value fails then returned result is “Read failure” (0006h/0086h).
- 5011 If some other error occurs then Result is “General failure” (0001h/0081h).
- 5012 If counter has expired also bit 7 is set to 1 in returned results.



5013  
5014

Figure 12.5 — Read Counter Value Flow

5015 **12.4.6.5 Authenticated Data Write**

- 5016 • The Authenticated Data Write sequence is initiated by a SECURITY PROTOCOL OUT command.
- 5017     ○ An initiator sends the SECURITY PROTOCOL OUT command with SECURITY PROTOCOL field set to ECh and indicating the RPMB region in the SECURITY PROTOCOL SPECIFIC field. The RPMB message 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.
- 5022     ○ 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 "Write failure, write counter expired" (0085h). No data is written to the RPMB data area.
- 5025     ○ Next the address is checked. If the Address value is equal to or greater than the size of target RPMB region which is defined as bRPMBRegion0Size – bRPMBRegion3Size parameter value in the RPMB Unit Descriptor, then the Result is set to "Address failure" (0004h). No data is written to the RPMB data area.
- 5029     ○ If the Address value plus the Block Count value is greater than the size of target RPMB region which is defined as bRPMBRegion0Size – bRPMBRegion3Size parameter value, then the Result is set to "Address failure" (0004h). No data is written to the RPMB data area.
- 5032     ○ If the Block Count indicates a value greater than bRPMB\_ReadWriteSize, then the authenticated data write operation fails and the Result is set to "General failure" (0001h).
- 5034     ○ 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 "Authentication failure" (0002h). No data is written to the RPMB data area.
- 5038     ○ 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 "Counter failure" (0003h). No data is written to the RPMB data area.
- 5042     ○ 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.
- 5044     ○ The write counter is incremented by one if the write operation is successfully executed.
- 5045     ○ If write fails then returned result is "Write failure" (0005h).
- 5046     ○ If some other error occurs during the write procedure then returned result is "General failure" (0001h).
- 5048     ○ In an authenticated data write request with Block Count greater than one
- 5049         ■ 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.
- 5052         ■ 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).

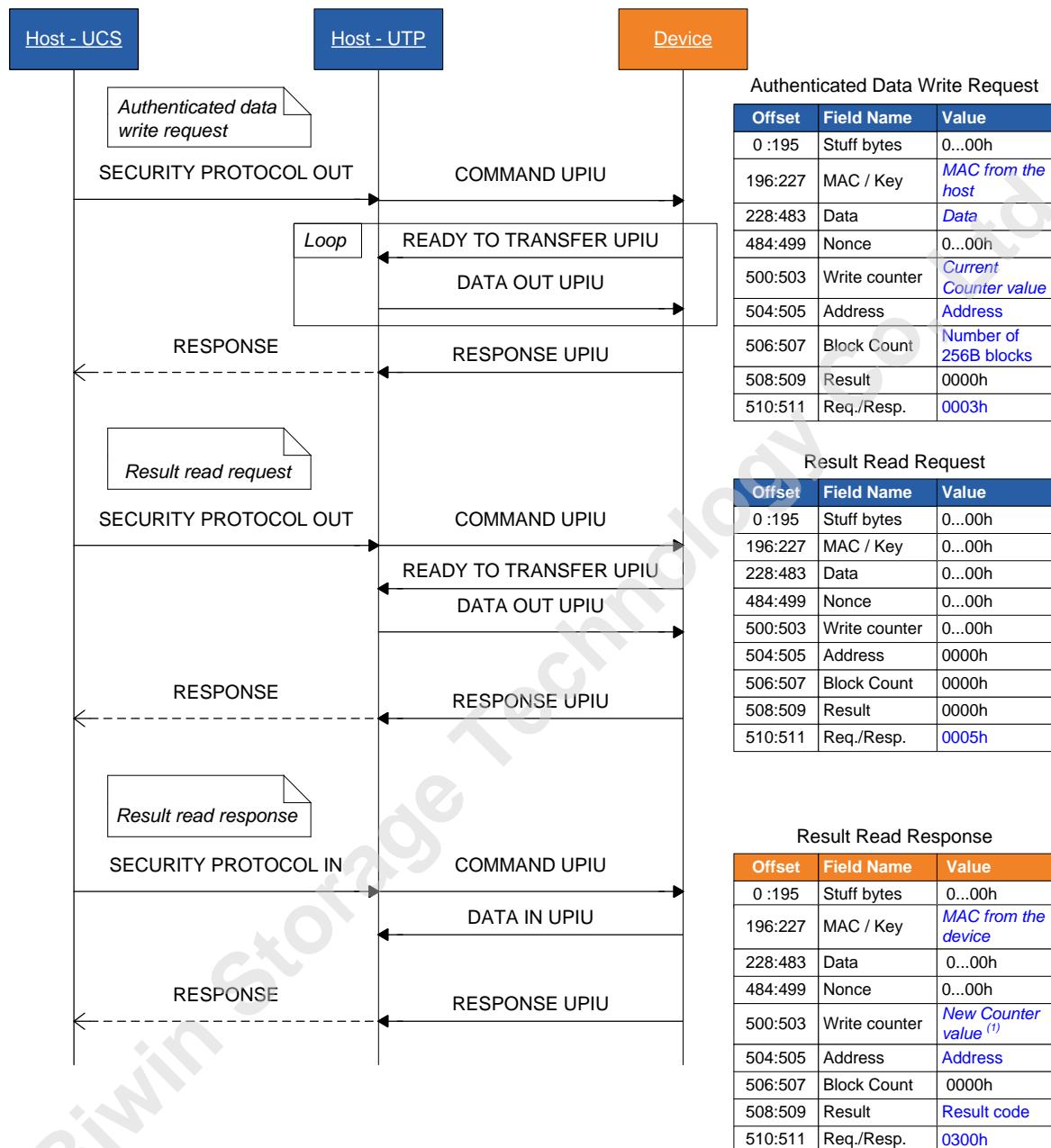
5056      **12.4.6.5 Authenticated Data Write (cont'd)**

- 5057            ○ When the authenticated data write operation is completed, the device may return GOOD status in  
5058            response to the SECURITY PROTOCOL OUT command regardless of whether the  
5059            Authenticated data write was successful or not.
- 5060        • The authenticated data write verification process starts by issuing a SECURITY PROTOCOL OUT  
5061            command.
- 5062            ○ An initiator sends a SECURITY PROTOCOL OUT command with SECURITY PROTOCOL  
5063            field set to ECh and indicating the RPMB region in the SECURITY PROTOCOL SPECIFIC  
5064            field. The RPMB data frame contains the Request Message Type = 0005h (Result read request).  
5065            Note that any request other than the Result read request from any initiator will overwrite the  
5066            Result register of the RPMB Region.
- 5067            ○ The device returns GOOD status when the operation result is ready for retrieval.
- 5068        • An initiator retrieves the operation result by issuing a SECURITY PROTOCOL IN command.
- 5069            ○ The SECURITY PROTOCOL field is set to ECh and the SECURITY PROTOCOL SPECIFIC  
5070            field indicates the RPMB region.
- 5071        • Device returns the RPMB data frame containing the Response Message Type = 0300h, the counter  
5072            value (incremented if the write operation is successfully executed), the address received in the  
5073            Authenticated data write request, the MAC and result of the authenticated data write operation.

Biwin Storage Technology

5075 12.4.6.5 Authenticated Data Write (cont'd)

5076



NOTE 1 The Write Counter is incremented only if the operation was successfully completed

5077  
5078  
5079

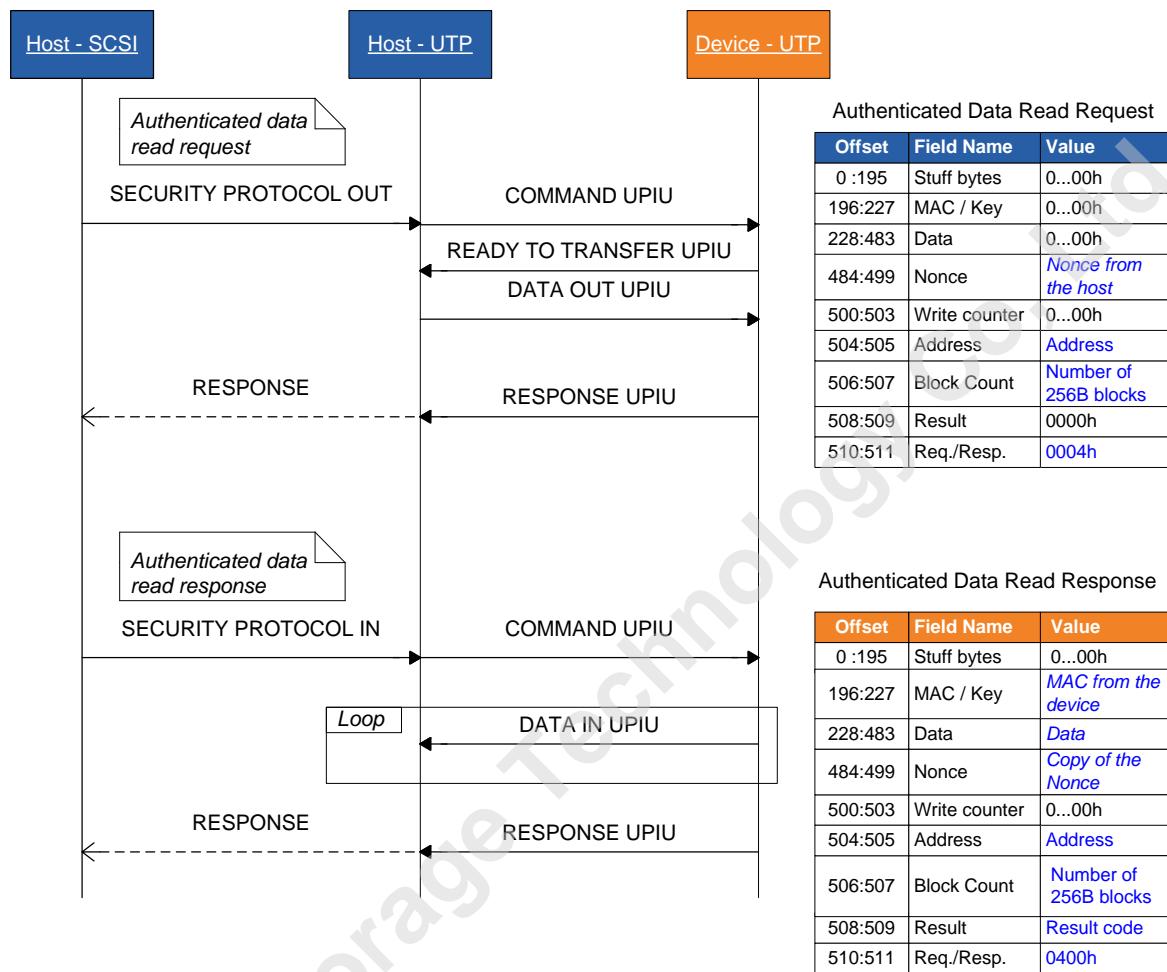
Figure 12.4 — Authenticated Data Write Flow

5080 **12.4.6.6 Authenticated Data Read**

- 5081 • The Authenticated Data Read sequence is initiated by a SECURITY PROTOCOL OUT command.
  - 5082 ○ An initiator sends the SECURITY PROTOCOL OUT command with SECURITY PROTOCOL field set to ECh and indicating the RPMB region in the SECURITY PROTOCOL SPECIFIC field. The RPMB data frame includes the Request Message Type = 0004h, the nonce, the data address, and the block count.
  - 5086 ○ When the device receives this request it first checks the address. If the Address value is equal to or greater than the size of target RPMB region which is defined as bRPMBRegion0Size – bRPMBRegion3Size parameter value in the RPMB Unit Descriptor, then Result is set to “Address failure” (0004h/0084h). The data read is not valid.
  - 5090 ○ If the Address value plus the Block Count value is greater than the size of target RPMB region which is defined as bRPMBRegion0Size – bRPMBRegion3Size parameter value, then the Result is set to “Address failure” (0004h/0084h). No data is read from the RPMB data area.
  - 5093 ○ 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 “Authentication failure” (0002h/0082h).
- 5096 • If the SECURITY PROTOCOL OUT command completes with GOOD status, data can be retrieved sending a SECURITY PROTOCOL IN command.
  - 5098 ○ An initiator sends the SECURITY PROTOCOL IN command with SECURITY PROTOCOL field set to ECh and indicating the RPMB region in the SECURITY PROTOCOL SPECIFIC field.
  - 5101 ○ The device returns a RPMB message 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.
  - 5104 ○ In an authenticated data read response with Block Count greater than one,
    - 5105 ■ the MAC is included only in the last RPMB message data frame. The MAC field is zero in all previous data frames.
    - 5107 ■ 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).
- 5110 • 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.
- 5113 • If data fetch from addressed location inside device fails then returned result is “Read failure” (0006h/0086h). If some other error occurs during the read procedure then returned result is “General failure” (0001h/0081h).

5117 **12.4.6.6 Authenticated Data Read (cont'd)**

5118



5119

5120

5121

**Figure 12.6 — Authenticated Data Read Flow**

5122 **12.4.6.7 Authenticated Secure Write Protect Configuration Block Write**

- 5123 • Authenticated Secure Write Protect Configuration Block write operation is supported by RPMB  
5124 region 0 only. If Authenticated Secure Write Protect Configuration Block write operation is issued to  
5125 the RPMB region other than RPMB region 0, then returned result is “General failure” (0001h/0081h).
- 5126 • The Authenticated Secure Write Protect Configuration Block write sequence is initiated by a  
5127 SECURITY PROTOCOL OUT command.
  - 5128 ○ An initiator sends the SECURITY PROTOCOL OUT command with SECURITY PROTOCOL  
5129 field set to ECh and indicating the RPMB region 0 in the SECURITY PROTOCOL SPECIFIC  
5130 field.
  - 5131 ○ If the INC\_512 bit and TRANSFER LENGTH field are not set to zero and 512 respectively, then  
5132 the command shall be terminated with CHECK CONDITION status with the sense key set to  
5133 ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.
  - 5134 ○ The SECURITY PROTOCOL OUT command delivers a single RPMB message data frame  
5135 which contains the Secure Write Protect Configuration Block in the Data field. The Secure Write  
5136 Protect Configuration Block is specific of the logical unit indicated by the LUN field (byte 228 of  
5137 the data frame).
  - 5138 ○ The other fields of RPMB data frame are set as specified in the following: Request Message Type  
5139 = 0006h, Result = 0000h, Block Count = 0001h, Address = 0000h, Write Counter = current  
5140 counter value,Nonce = 00...0h, and MAC = see 12.4.4.2.
  - 5141 ○ When the device receives the RPMB message data frame, it first checks whether the write counter  
5142 has expired. If the write counter is expired then the device sets the result to “Write failure, write  
5143 counter expired” (0085h). The Secure Write Protect Configuration Block is not updated.
  - 5144 ○ If the write counter was not expired, then the device calculates the MAC of request type, block  
5145 count, write counter, address and data, and compares this with the MAC in the request. If the two  
5146 MAC’s are different, then the device sets the result to “Authentication failure” (0002h). The  
5147 Secure Write Protect Configuration Block is not updated.
  - 5148 ○ If the MAC in the request and the calculated MAC are equal, then the device compares the write  
5149 counter in the request with the write counter stored in the device. If the two counters are different  
5150 then the device sets the result to “Counter failure” (0003h). The Secure Write Protect  
5151 Configuration Block is not updated.
  - 5152 ○ If the MAC and write counter comparisons are successful then the write request is considered to  
5153 be authenticated.
  - 5154 ○ If the LUN field indicates a logical unit with bLUWriteProtect set to a value different from zero,  
5155 then the device sets the result to “Secure Write Protection not applicable” (000Ah). The Secure  
5156 Write Protect Configuration Block is not updated.

5157

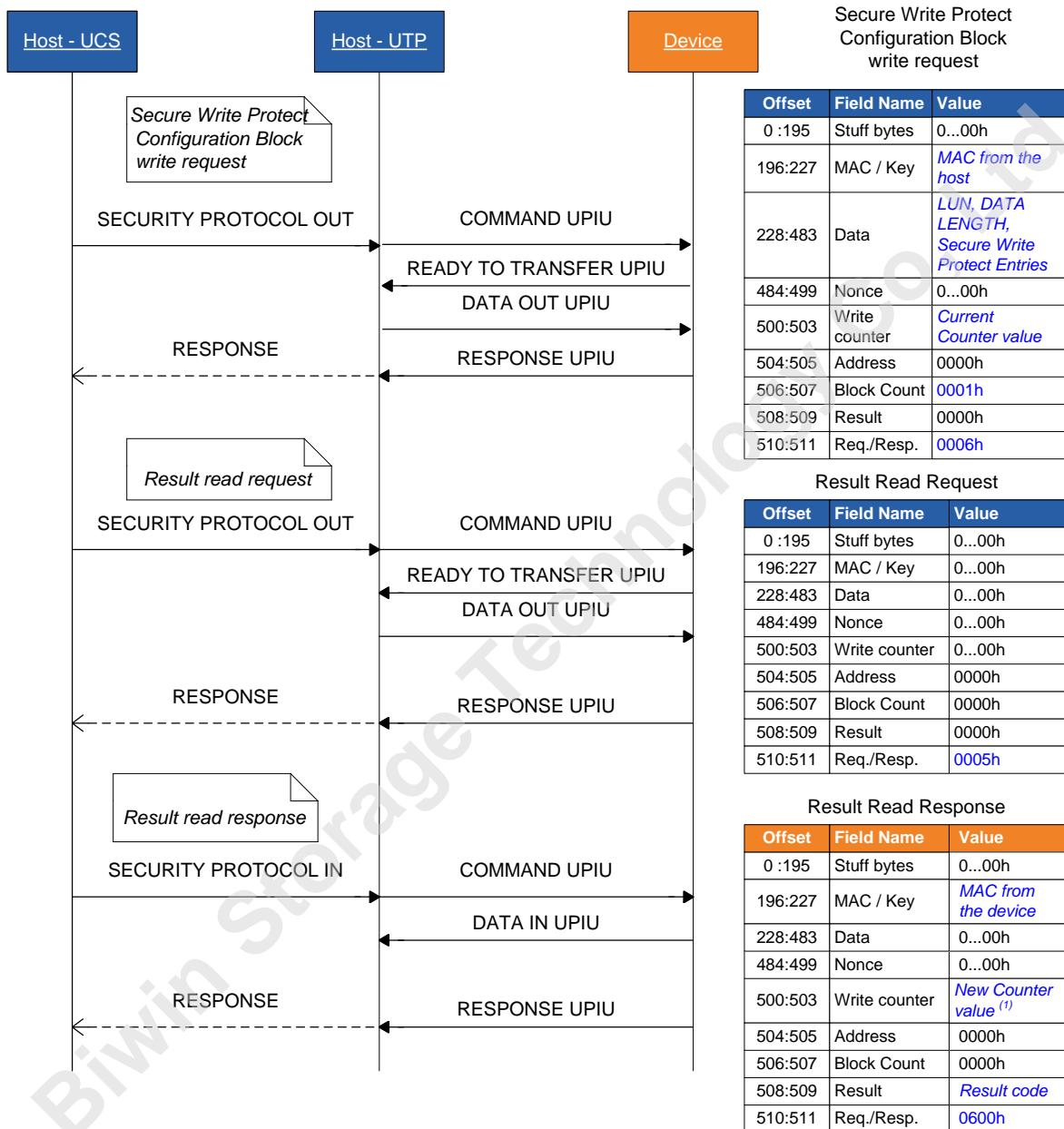
5158   **12.4.6.7 Authenticated Secure Write Protect Configuration Block Write (cont'd)**

5159

- 5160   ○ The device sets the result to "Invalid Secure Write Protect Block Configuration parameter"  
5161   (0009h) and it does not update the Secure Write Protect Configuration Block if one or more of the  
5162   following conditions occurs.
- 5163   ■ the LUN field is invalid: greater than the value specified by bMaxNumberLU or if the logical  
5164   unit is not enabled (bLUEnable=00h).
- 5165   ■ the DATA LENGTH is set to a value different from the following ones: 0, 16, 32, 48, 64.
- 5166   ■ the LOGICAL BLOCK ADDRESS in a Secure Write Protect entry exceeds the logical unit  
5167   capacity,
- 5168   ■ the LOGICAL BLOCK ADDRESS plus the NUMBER OF LOGICAL BLOCKS in a Secure  
5169   Write Protect entry exceeds the logical unit capacity,
- 5170   ■ two or more Secure Write Protect entries specify overlapping areas,
- 5171   ■ with this request, the number of Secure Write Protect areas set in the entire device is  
5172   increased to a value greater than what indicated by bNumSecureWPArea.
- 5173   ○ If some other error occurs during the write procedure then returned result is "Secure Write Protect  
5174   Configuration Block access failure" (0008h). The Secure Write Protect Configuration Block is  
5175   not updated.
- 5176   ○ If no error occurred, then the Secure Write Protect Configuration Block is updated overwriting  
5177   the former configuration and the write counter is incremented by one.
- 5178   ● The device may return GOOD status in response to the SECURITY PROTOCOL OUT command  
5179   regardless of whether the Authenticated Secure Write Protect Configuration Block Write was  
5180   successful or not.
- 5181   ● The successfulness of the programming of the data can be checked by retrieving the result register of  
5182   the RPMB.
- 5183   ● The verification process starts by issuing a SECURITY PROTOCOL OUT command.
- 5184   ○ An initiator sends a SECURITY PROTOCOL OUT command with SECURITY PROTOCOL  
5185   field set to ECh and indicating the RPMB region 0 in the SECURITY PROTOCOL SPECIFIC  
5186   field. The RPMB data frame contains the Request Message Type = 0005h (Result read request).  
5187   Note that any request other than the Result read request from any initiator will overwrite the  
5188   Result register of the RPMB Region.
- 5189   ○ The device returns "Good" status when the operation result is ready for retrieval.
- 5190   ● An initiator retrieves the operation result by issuing a SECURITY PROTOCOL IN command.
- 5191   ● The SECURITY PROTOCOL field is set to ECh and the SECURITY PROTOCOL SPECIFIC  
5192   field indicates the RPMB region.
- 5193   ● Device returns the RPMB data frame containing the Response Message Type = 0600h, the  
5194   incremented counter value, the MAC and result of the Authenticated Secure Write Protect  
5195   Configuration Block write operation.

5196

5197 12.4.6.7 Authenticated Secure Write Protect Configuration Block Write (cont'd)



NOTE 1 The Write Counter is incremented only if the operation was successfully completed.

5198

5199

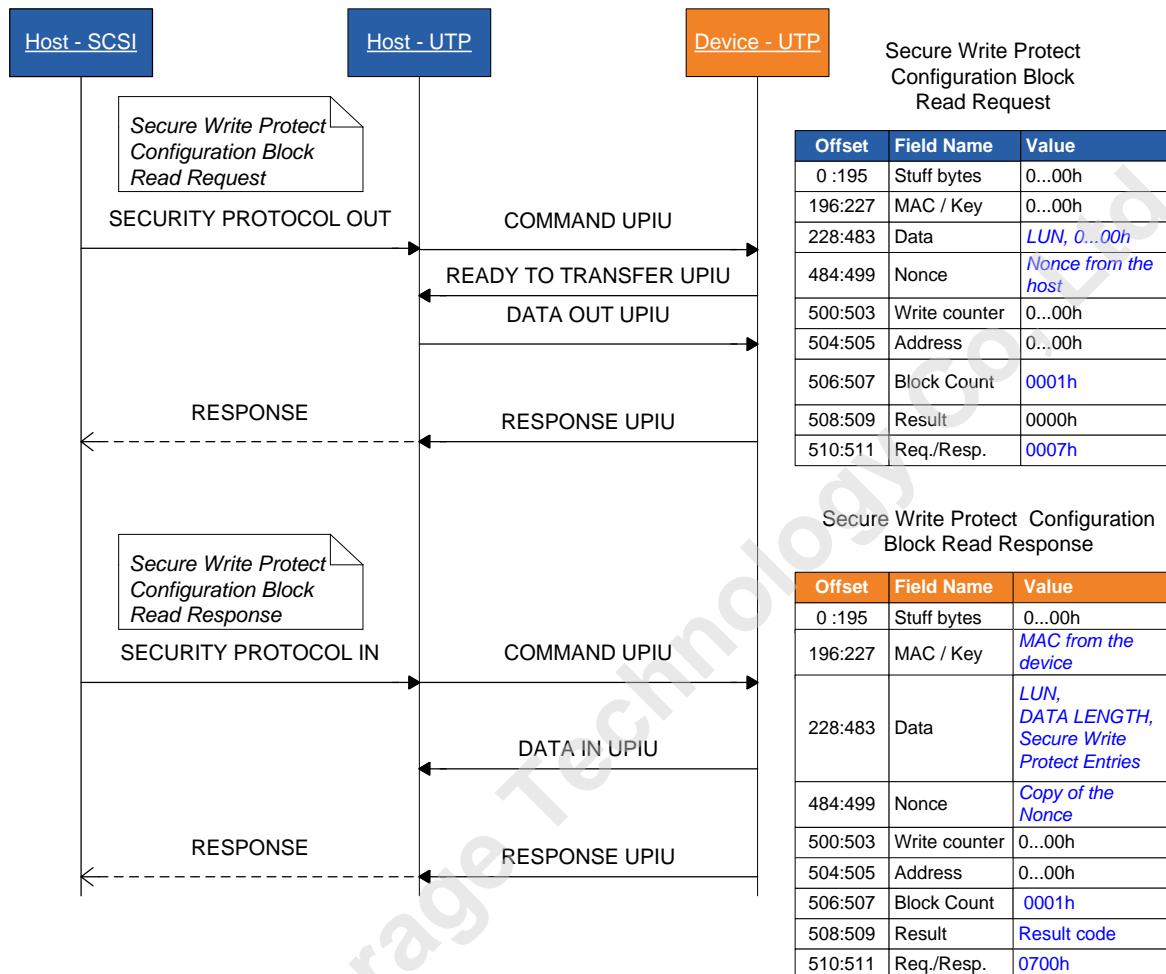
Figure 12.7 —Authenticated Secure Write Protect Configuration Block Write Flow

5200

#### 5201   **12.4.6.8 Authenticated Secure Write Protect Configuration Block Read**

- 5202   • Authenticated Secure Write Protect Configuration Block read operation is supported by RPMB region  
5203   0 only. If Authenticated Secure Write Protect Configuration Block read operation is issued to the  
5204   RPMB region other than RPMB region 0, then returned result is “General failure” (0001h/0081h).
- 5205   • The Authenticated Secure Write Protect Configuration Block Read sequence is initiated by a  
5206   SECURITY PROTOCOL OUT command.
  - 5207   ○ An initiator sends the SECURITY PROTOCOL OUT command with SECURITY PROTOCOL  
5208   field set to ECh and indicating the RPMB region 0 in the SECURITY PROTOCOL SPECIFIC  
5209   field.
  - 5210   ○ If the INC\_512 bit and TRANSFER LENGTH field are not set to zero and 512 respectively, then  
5211   the command shall be terminated with CHECK CONDITION status with the sense key set to  
5212   ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.
  - 5213   ○ The SECURITY PROTOCOL OUT command delivers a single RPMB message data frame  
5214   which contains in the Data field a LUN value (byte 228 of the data frame). The Secure Write  
5215   Protect Configuration Block that will be returned is specific of the logical unit indicated by the  
5216   LUN field.
  - 5217   ○ The RPMB data frame delivered from the host to the device includes the Request Message Type  
5218   = 0007h, Block Count=0001h, Address=0000h, the Data and Nonce. In this request, the LUN is  
5219   the only relevant byte of the Data field, all other bytes shall be considered reserved and they shall  
5220   be ignored.
  - 5221   ○ If the LUN field is not valid, then the device sets the result to “Invalid Secure Write Protect Block  
5222   Configuration parameter” (0009h/0089h). The LUN field is invalid if it is greater than the value  
5223   specified by bMaxNumberLU or if the logical unit is not enabled (bLUEnable=00h).
  - 5224   ○ If the LUN field indicates a logical unit with bLUWriteProtect set to a value different from zero,  
5225   then the device sets the result to “Secure Write Protection not applicable” (000Ah/008Ah).
  - 5226   ○ After successful fetch of the Secure Write Protect Configuration Block, the MAC is calculated  
5227   from response type, nonce, address, data and result. If the MAC calculation fails then returned  
5228   result is “Authentication failure” (0002h/0082h).
- 5229   • If the SECURITY PROTOCOL OUT command completes with GOOD status, then the Secure Write  
5230   Protect Configuration Block can be retrieved sending a SECURITY PROTOCOL IN command.
  - 5231   ○ An initiator sends the SECURITY PROTOCOL IN command with SECURITY PROTOCOL  
5232   field set to ECh, indicating the RPMB region 0 in the SECURITY PROTOCOL SPECIFIC field,  
5233   and in which INC\_512 bit and ALLOCATION LENGTH field indicate 512 bytes.
- 5234   • The device returns a RPMB data frame with Response Message Type = 0700h, the block count, a  
5235   copy of the nonce received in the request, the contents of the Secure Write Protect Configuration  
5236   Block in the Data field, the MAC and the result
- 5237   • If data fetch from addressed location inside device fails or some other error occurs during the read  
5238   procedure then returned result is “Secure Write Protect Configuration Block access failure”  
5239   (0008h/0088h).

5241    **12.4.6.8 Authenticated Secure Write Protect Configuration Block Read (cont'd)**  
5242



5243  
5244    **Figure 12.8 — Authenticated Secure Write Protect Configuration Block Read Flow**  
5245

5246    **12.5 Malware Protection**

5247 The UFS device will also have the option to protect boot, bus configuration settings and other important  
5248 device configuration settings so that once they are set they cannot be modified. The implementation of  
5249 the protection of these parameters is defined within the spec where the parameter is defined.

5250

---

5251    **13 UFS Functional Descriptions**

5252    **13.1 UFS Boot**

5253    **13.1.1 Introduction**

5254 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.

5258 Several features of the boot functionality can be configured in order to be adapted to different system requirements.

5260 Moreover specific features to ensure boot data integrity and no corruption of boot code are defined.

5261    **13.1.2 Boot Configuration**

5262 During boot operation the UFS host controller retrieves the system boot code stored in the UFS device.

5263 In this version of the standard, the boot mechanism is defined for a point-to-point topology (see Figure 5264 13.1).

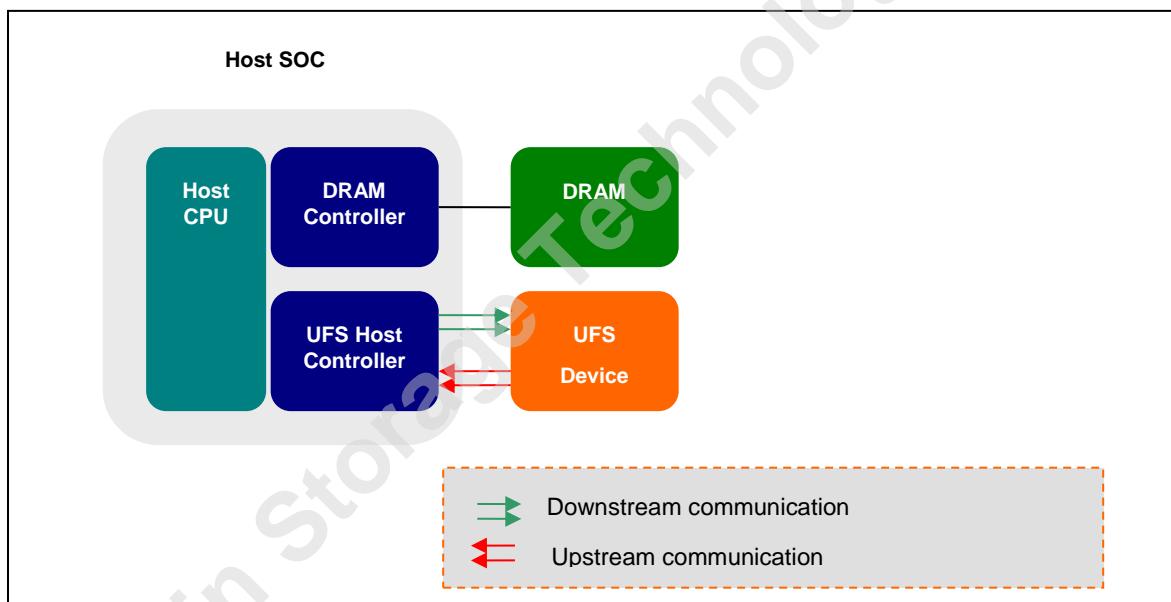


Figure 13.1 — UFS System Diagram

5265 Two logical units (Boot LU A, Boot LU B) can be used to store the boot code, but only one of them will  
5266 be active during the boot process. Any logical unit can be configured as “Boot LU A” or “Boot LU B”.  
5267 No more than one logical unit may be configured as “Boot LU A”, no more than one logical unit may be  
5268 configured as “Boot LU B”. The logical unit active during boot is mapped onto the Boot well known  
5269 logical unit (W-LUN = 30h) for read access. In this way, when the host updates the boot code, a fix  
5270 logical unit number is kept when the active logical unit is swapped from A to B or vice versa.  
5271

5272

5273 **13.1.2 Boot Configuration (cont'd)**

5274 Several configurable fields of the Device Descriptor and the Unit Descriptors determine the device  
5275 behavior during boot. Device Descriptor and Unit Descriptors are configured by writing the Configuration  
5276 Descriptor.

5277 For a UFS bootable device, the boot feature is enabled if bBootEnable field in the Device Descriptor is set  
5278 to 01h.

5279 The characteristics of the logical units used during boot are configured setting the corresponding fields of  
5280 the Configuration Descriptor; see 14.1.4.3, Configuration Descriptor, for details.

5281 The number of allocation units (dNumAllocUnits) field configures the logical unit size, and the boot  
5282 logical unit ID (bBootLunID) field allows to designate the logical unit as being "Boot LU A" or "Boot  
5283 LU B".

5284 The logical unit active during the boot shall be configured by writing the bBootLunEn attribute, as  
5285 described in Table 13.1.

5286 **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

5287 The host should not attempt to set bBootLunEn to 'Reserved' values, and UFS device shall generate an  
5288 error in case of an attempt to set 'Reserved' values and not execute the request.

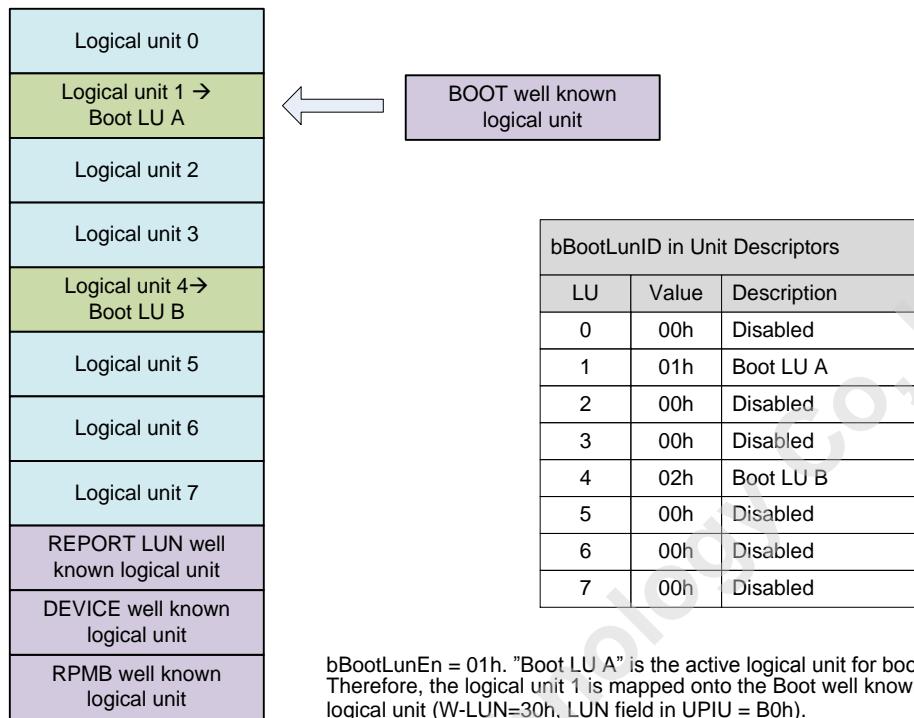
5289 When bBootLunEn attribute is 00h the boot feature is disabled, the device behaves as if bBootEnable  
5290 would be equal to zero.

5291 The active boot logical unit will be mapped onto the Boot well known boot logical unit (W-LUN = 30h)  
5292 once the bBootLunEn has been properly configured.

5293 Figure 13.2 shows an example of a UFS device having eight logical units: LU 1 and LU 4 are configured,  
5294 respectively, as "Boot LU A" and "Boot LU B". In particular, LU 1 is the active one (bBootLunEn =  
5295 01h).

5296

5297 **13.1.2 Boot Configuration (cont'd)**



5298

5299

5300

5301

**Figure 13.2 — Example of UFS Device Memory Organization for Boot**

5302 **13.1.3 Initialization and boot code download process**

5303 The initialization and boot code download process is made up of the following phases: partial  
5304 initialization, boot transfer and initialization completion.

5305 **13.1.3.1 Partial initialization**

5306 The partial initialization phase starts after power on, or hardware reset, or EndPointReset and involves the  
5307 entire UFS stack. At the end of this phase, the UniPro boot sequence shall be completed, and the UTP  
5308 layer shall be capable of accessing Device Descriptor (if the bDescrAccessEn field of the Device  
5309 Descriptor is ‘01h’) and exchanging UPIU for READ command and TEST UNIT READY command. If  
5310 the bDescrAccessEn field is ‘00h’ descriptors will be accessible only after the initialization completion  
5311 phase.

5312 Each single layer in the UFS protocol stack executes the initialization process on both UFS host and UFS  
5313 device sides.

5314 **a) Physical Layer (M-PHY)**

5315 After reset events, the physical layer will move from DISABLED state to HIBERN8 state.

5316 **b) Link Layer (UniPro)**

5317 On host and device side UniPro boot sequence takes place:

- 5318 1) The UniPro stack is reset using the DME\_RESET.req primitive.
- 5319 2) Wait until the reset completion is indicated by the DME\_RESET.cnf\_L primitive.
- 5320 3) The UniPro stack is enabled using the DME\_ENABLE.req primitive.
- 5321 4) Wait until the enable completion is indicated by the DME\_ENABLE.cnf\_L primitive.
- 5322 5) The UniPro Link StartUp sequence is initiated using the DME\_LINKSTARTUP.req primitive. The  
5323 UniPro Link Startup consists of a series of multiphase handshakes to establish initial link  
5324 communication in both directions between UFS host and device.
- 5325 6) Wait until the link startup completion is indicated by the DME\_LINKSTARTUP.cnf\_L primitive.

5326 **c) UFS Transport Layer (UTP)**

5327 At the end of the UFS Interconnect Layer initialization on both host and device side, the host shall  
5328 send a NOP OUT UPIU to verify that the device UTP Layer is ready.

5329 For some implementations, the device UTP layer may not be initialized yet, therefore the device may  
5330 not respond promptly to NOP OUT UPIU sending NOP IN UPIU.

5331 The host waits until it receives the NOP IN UPIU from the device. When the NOP IN UPIU is  
5332 received, the host is acknowledged that the UTP layer on the device is ready to execute UTP  
5333 transactions.

5334 **d) Link Configuration**

5335 The host may configure the Link Attributes (i.e., Gear, HS Series, PWM Mode in Rx and Tx) by  
5336 using DME primitives at UniPro level.

5337 **e) Device Descriptor Reading**

5338 The UFS host may optionally discover relevant device info for the boot process by accessing the  
5339 Device Descriptor (i.e., Device Class/Subclass, Boot Enable, Boot LUs size, etc.). The UFS host is  
5340 allowed to access the Device Descriptor only if the bDescrAccessEn is ‘01h’, otherwise this  
5341 descriptor can be accessed only after the device has fully completed its initialization.

5342 **13.1.3.2 Boot transfer**

5343 The following steps can be executed only if bBootEnable field is set.

5344 **Boot code download**

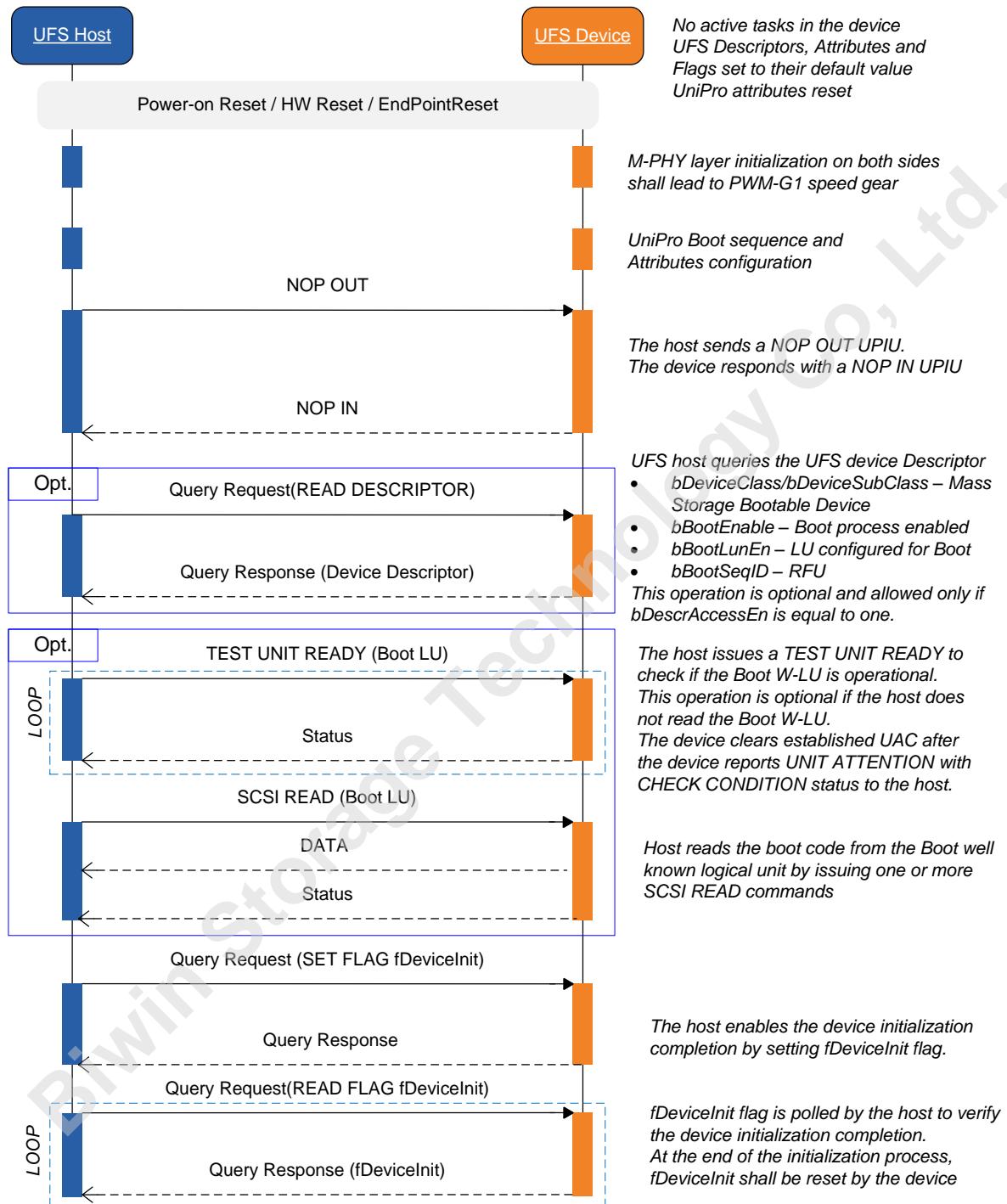
5345 At first, the UFS host issues a TEST UNIT READY command to the Boot well known logical unit to  
5346 verify if the latter can be accessed. If the command succeeds, the UFS host reads the Boot well known  
5347 logical unit by issuing SCSI READ commands and the UFS device will start to send the boot code on the  
5348 Upstream Link. During this phase only the Boot well known logical unit is accessible: this logical unit  
5349 shall accept read commands, while other logical units may not be ready.

5350 **13.1.3.3 Initialization completion**

5351 After the host has completed the boot code download from the Boot well known logical unit, the  
5352 initialization process proceeds as described in the following. The host sets the fDeviceInit flag to “01h” to  
5353 communicate to the UFS device that it can complete its initialization. The device shall reset the  
5354 fDeviceInit flag when the initialization is complete. The host polls the fDeviceInit flag to check the  
5355 completion of the process. When the fDeviceInit is reset, the device is ready to accept any command.

5356

5357 13.1.3.3 Initialization completion (cont'd)



5358

5359

5360

Figure 13.3 — Device Initialization and Boot Procedure Sequence Diagram

5361 **13.1.3.3 Initialization completion (cont'd)**

5362 **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) <sup>(1)</sup>	QUERY REQUEST UPIU (READ DESCRIPTOR Device Descriptor)	None
	Boot Transfer (Optional) <sup>(2)</sup>	COMMAND UPIU for Boot W-LU	INQUIRY, REQUEST SENSE, TEST UNIT READY, READ (6), READ (10), READ (16) <sup>(3)</sup>
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.

5363

5364 **13.1.4 Initialization process without boot code download**

5365 If the boot process is not enabled on the UFS device, or it is not supported by the device class, or the host  
5366 does not need to transfer the boot code, the host executes the initialization process as described in 13.1.3,  
5367 omitting the boot transfer phase.

5368 **13.1.5 Boot Logical Unit Operations**

5369 The Boot well known logical unit is read only, therefore the boot code can be stored only writing the boot  
5370 logical units (A or B).

5371 Boot logical units are written to store the boot code during the system manufacturing phase and they may  
5372 be also updated during the system lifecycle. These logical units can be read to verify their content.

5373 Therefore the following operations are permitted on the Boot logical units:

- 5374 1. boot code write - for boot code upload/update  
5375 2. boot code read - to verify the content programmed  
5376 3. boot code removal - to remove the content of the Boot logical unit

5377 These operations can be executed regardless the bBootEnable field value in the Device Descriptor.

5378 Boot logical units (A or B) can be write protected using the methods described in 12.3, Device Data  
5379 Protection.

5380 **13.1.6 Configurability**

5381 The boot process is configurable through several parameters in the Configuration Descriptor (see  
5382 14.1.4.3) to adapt it to different usage models and system features.

5383 The following parameters refer to boot capabilities.

- 5384 • Device Descriptor parameters:  
5385 - bBootEnable (Boot Enable)  
5386 - bDescrAccessEn (Descriptor Access Enable)  
5387 - bInitPowerMode (Initial Power Mode)  
5388 - bInitActiveICCLevel (Initial Active ICC Level)  
5389 • Unit Descriptor parameters for Boot LU A and Boot LU B:  
5390 - bLUEnable (Logical Unit Enable)  
5391 - bBootLunID (Boot LUN ID)  
5392 - bLUWriteProtect (Logical Unit Write Protect)  
5393 - bMemoryType (Memory Type)  
5394 - dNumAllocUnits (Number of Allocation Units)  
5395 - bDataReliability (Data Reliability)  
5396 - bLogicalBlockSize (Logical Block Size)  
5397 - bProvisioningType (Provisioning Type)

5398 NOTE These parameters are non volatile and they may be programmed during the system manufacturing phase.

5399 In addition to the parameters mentioned, the following attributes are relevant for device initialization and  
5400 boot

- 5401 • bBootLunEn (Boot LUN Enable)  
5402 • bRefClkFreq (Reference Clock Frequency value)

5403 **13.1.7 Security**

5404 **13.1.7.1 Boot Area Protection**

5405 Boot areas might be protected in order to avoid boot code alteration by a third party: the write protection  
5406 mechanism for the boot logical units can be defined configuring the corresponding bLUWriteProtect  
5407 parameter of the Unit Descriptor.

5408 In particular, the boot logical units may be permanently write protected or power-on write protected. In  
5409 case of power-on write protection, the boot logical units can be written only when the fPowerOnWPEn  
5410 flag is equal to zero.

5411

5412 **13.2 Logical Unit Management**

5413 **13.2.1 Introduction**

5414 The functionality aims to provide a mechanism to let an external application define and use a virtual  
5415 memory organization which could easily fit different usage models in a versatile way.

5416 Besides segmenting the available addressable space, the mechanism introduces the possibility of  
5417 differentiating each logical unit through dedicated functionalities and features.

5418 This section describes the procedure to configure the UFS device in terms of: number of logical units,  
5419 logical unit size, logical unit memory type, etc. Security features can be configured as described in clause  
5420 12, UFS Security.

5421 A UFS device can be organized in different logical units. Each one represents an autonomous computing  
5422 entity with independent logical address ranges and singularly accessible.

5423 Moreover, each logical unit can be defined for a specified use and with peculiar attributes (i.e., memory  
5424 type) in order to be adapted to different UFS host usage models and operating systems requirements.

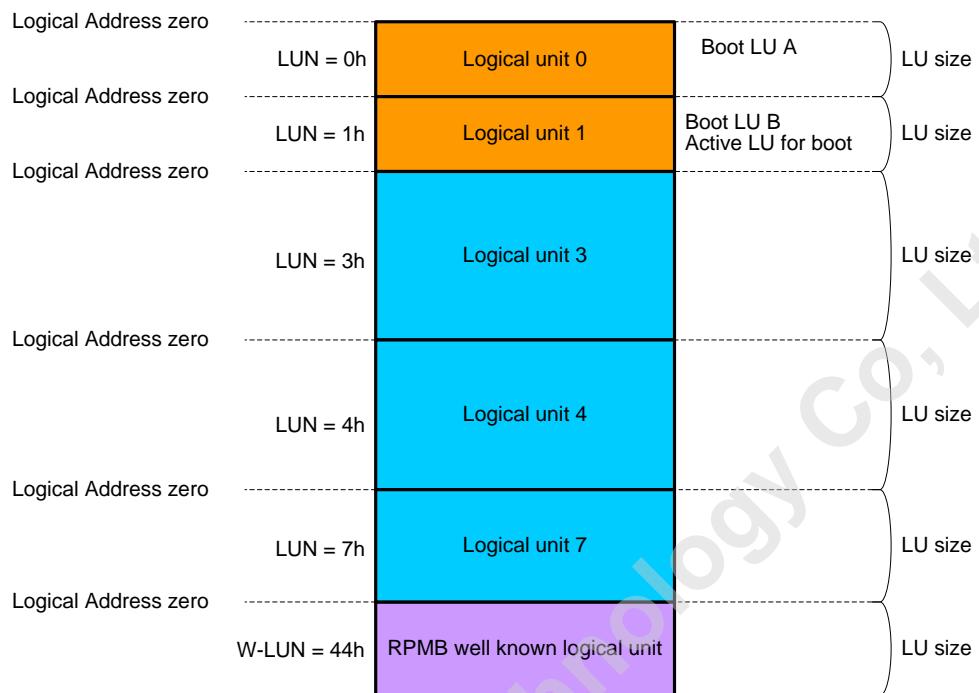
5425 **13.2.2 Logical Unit features**

5426 UFS device address space is organized in several memory areas configurable by the user. In particular,  
5427 such memory areas are denoted as logical units and characterized by the fact that they have independent  
5428 logical addressable spaces starting from the logical address zero.

5429 In addition to the logical units, the UFS device supports the following well known logical units for  
5430 specific purposes: UFS Device, REPORT LUNS, Boot and RPMB. Logical units are addressed by the  
5431 LUN (logical unit number), while well known logical unit are addressed by the W-LUN (well known  
5432 logical unit number).

5433

5434 13.2.2 Logical Unit features (cont'd)



5435

5436 NOTE 1 The figure shows an example of device configuration in which LU 0 and LU 1 are used as boot logical  
5437 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  
5438 accessed in read using the W-LUN = 30h (LUN field in UPIU = B0h).

**Figure 13.4 — Example of UFS Device Memory Organization**

5440 Each logical unit will have a physical implementation on the non-volatile storage media.

5441 In particular, the UFS device shall support:

- 5442 • The number of logical units specified by bMaxNumberLU. Each of them configurable as boot logical  
5443 units with a maximum of two.
- 5444 • One RPMB well known logical unit (W-LUN = 44h, LUN field in UPIU = C4h). RPMB well known  
5445 logical unit may be further configured into up to four separate RPMB regions (RPMB region 0 -  
5446 RPMB region 3).

5447 Two logical units can be configured as boot logical unit, with only one of them active and readable  
5448 through the Boot well known logical unit (W-LUN = 30h) for the execution of the system boot (see 13.1,  
5449 UFS Boot). The RPMB well known logical unit is accessed by authenticated operations by a well defined  
5450 security algorithm (see 12.4, RPMB). The other logical units will be used to fulfill other use cases.

5451

5452 **13.2.2 Logical Unit features (cont'd)**

5453 Common features of each logical unit are:

- 5454 • independent logical addressable spaces (starting from logical address zero up to the logical unit size),  
5455 • Configurable logical unit size.

5456 The size of each logical unit is determined by the number of allocation units assigned to it:  
5457 dNumAllocUnits parameter value of the Configuration Descriptor. The dNumAllocUnits is expressed in  
5458 terms of allocation unit size (bAllocationUnitSize).

5459 Moreover each logical unit is characterized by the memory type parameter which can be configured.  
5460 Examples of memory types to differentiate logical unit properties are the following ones:

- 5461 • Default type – regular memory characteristic  
5462 • System Code type – a logical unit that is rarely updated (e.g., system files or binary code executable  
5463 files or the host operating system image and other system data structures)  
5464 • Non-Persistent type – a logical unit that is used for temporary information (e.g., swap file extend the  
5465 host virtual memory space)  
5466 • Enhanced Memory type – vendor specific attribute

5467 The definition of the Enhanced Memory type is left open in order to accomplish different needs and  
5468 vendor specific implementations.

5469 Mechanisms of write protection can be configured for each logical unit. Write protection feature types  
5470 are:

- 5471 • Permanent write protection (permanent read only)  
5472 • Power on write protection (write protection can be cleared with a power cycle or a hardware reset  
5473 event)

5474 Write protection is not available in the RPMB well known logical unit.

5475

5476 **13.2.3 Logical Unit Configuration**

5477 The user shall configure the logical units of the UFS device according to the following rules:

- 5478 • Maximum number of logical units is specified by bMaxNumberLU.
- 5479 • One or two logical units can be configured as boot logical units.

5480 When a UFS device is shipped only the following well known logical units will be available: UFS  
5481 Device, REPORT LUNS and the RPMB. All other logical units shall be configured before they can be  
5482 accessed.

5483 The RPMB well known logical unit shall be configured by the device manufacturer before shipping the  
5484 device. Only RPMB region 0 shall be enabled and bRPMBRegion0Size shall be set to the same size as  
5485 the total RPMB well known logical unit size before shipping the device.

5486 Logical units and RPMB regions may be configured writing the Configuration Descriptors, see 14.1.3,  
5487 Accessing Descriptors and Device Configuration, for details. RPMB regions may be configured multiple  
5488 times until the Configuration Descriptor is locked by setting bConfigDescrLock attribute value to 01h.  
5489 RPMB key may be programmed any time when the target RPMB region is enabled, and may be  
5490 programmed independent to whether the value of bConfigDescrLock attribute is set or not. When RPMB  
5491 regions are re-configured with different configuration setting, the data which was previously written to  
5492 any RPMB region shall be erased while RPMB key and RPMB write counter are maintained. To keep the  
5493 compatibility to the previous version of the standard, RPMB region 0 is always enabled independent of  
5494 configuration value of bRPMBRegionEnable.

5495 The configuration of each logical unit and the configuration of RPMB regions may be retrieved by  
5496 reading the corresponding Unit Descriptor.

5497 It is recommended to execute logical unit configuration and RPMB region configuration during the  
5498 system manufacturing phase.

5499

5500 **13.2.3 Logical Unit Configuration (cont'd)**

5501 Table 13.3 summarizes the configurable parameters per logical unit. See 14.1.4, Descriptor Definitions,  
5502 for details about these parameters.

5503 **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
bRPMBRegionEnable	RPMB Region Enable	RPMB W-LU
bRPMBRegion0Size	RPMB Region 0 Size	RPMB W-LU
bRPMBRegion1Size	RPMB Region 1 Size	RPMB W-LU
bRPMBRegion2Size	RPMB Region 2 Size	RPMB W-LU
bRPMBRegion3Size	RPMB Region 3 Size	RPMB W-LU
dLUNumWriteBoosterBufferAllocUnits	WriteBooster Buffer size for the corresponding Logical Unit	Valid only for LU 0, ..., LU 7

5504     **13.2.3 Logical Unit Configuration (cont'd)**

5505     The following Geometry Descriptor parameters provide relevant information for configuring the logical  
5506     units:

- 5507     • qTotalRawDeviceCapacity (total raw device density in unit of 512 bytes)
- 5508     • dSegmentSize
- 5509     • bAllocationUnitSize (Allocation Unit Size, value expressed in number of segments)
- 5510     • wSupportedMemoryTypes
- 5511     • Maximum number of allocation unit for each memory type (dSystemCodeMaxNAllocU,  
5512        dNonPersistMaxNAllocU, etc.)
- 5513     • Capacity Adjustment Factor for each memory type (wSystemCodeCapAdjFac,  
5514        wNonPersistCapAdjFac, etc.)
- 5515     • bMinAddrBlockSize (this parameter indicates a value equal or greater than 4Kbyte)
- 5516     • bOptimalReadBlockSize and bOptimalWriteBlockSize
- 5517     • bMaxInBufferSize
- 5518     • bMaxOutBufferSize

5519     To enable the access to a logical unit, the user shall configure Unit Descriptor parameters as described in  
5520     the following:

- 5521     • bLUEnable  
5522        bLUEnable shall be set to 01h to enable the logical unit. If bLUEnable is equal to 00h the logical unit  
5523        is disabled and all Unit Descriptor parameters are don't care.
- 5524     • bMemoryType  
5525        bMemoryType shall be set to value corresponding to the desired memory type. The  
5526        wSupportedMemoryTypes parameter in the Geometry Descriptor indicates which memory types are  
5527        supported by the device.
- 5528     • bLogicalBlockSize  
5529        bLogicalBlockSize value shall adhere to the following rules:
  - 5530           o  $2^{bLogicalBlockSize} \geq bMinAddrBlockSize \times 512$ ,
  - 5531           o  $2^{bLogicalBlockSize} \leq bMaxInBufferSize \times 512$ ,
  - 5532           o  $2^{bLogicalBlockSize} \leq bMaxOutBufferSize \times 512$ .

5533     To optimize the device performance, it is recommended to configure the logical block size  
5534        (bLogicalBlockSize) to represent the value indicated by dOptimalLogicalBlockSize for the specific  
5535        logical unit memory type.

5536     Supported bLogicalBlockSize values are device specific, refer to the vendor datasheet for further  
5537        information.

5538

5539 **13.2.3 Logical Unit Configuration (cont'd)**

- 5540 • dNumAllocUnits

5541 dNumAllocUnits determines the size of the logical unit. If LUCapacity is the desired logical unit size  
5542 expressed in bytes, the dNumAllocUnits value shall be calculated using the following equation:

$$dNumAllocUnits = \text{CEILING} \left( \frac{LUCapacity \times CapacityAdjFactor}{bAllocationUnitSize \times dSegmentSize \times 512} \right)$$

5543 where:

- 5544 ○ CapacityAdjFactor = Capacity Adjustment Factor of the particular memory type

5545 The Capacity Adjustment Factor value for Normal memory type is one.

5546 The following example shows dNumAllocUnits calculation for two logical units (LU 1 and LU 4)  
5547 with the characteristics:

- 5548 ○ LU 1: 12 Gbyte, Normal memory type

- 5549 ○ LU 4: 32Mbyte, Enhanced memory type 1

5550 Assuming that the medium of the UFS device is composed by NAND flash memories which support  
5551 2 bit-per-cell and 1 bit-per-cell operation modes. The 2 bit-per-cell operation mode may be associated  
5552 with the Normal memory type, while the 1 bit-per-cell operation mode may be associated with the  
5553 Enhanced memory type 1.

5554 The Capacity Adjustment Factor for the Enhanced memory type 1 will be equal to 2.

5555 If:

- 5556 ○ dSegmentSize = 1024

- 5557 ○ bAllocationUnitSize = 8

5558 Then dNumAllocUnits for LU 1 and LU 4 are:

$$dNumAllocUnits_{LU\ 1} = \text{CEILING} \left( \frac{12\ \text{Gbyte} \times 1}{8 \times 1024 \times 512\ \text{byte}} \right) = \text{CEILING} \left( \frac{12\ \text{Gbyte}}{4\ \text{Mbyte}} \right) = 3072$$

$$dNumAllocUnits_{LU\ 4} = \text{CEILING} \left( \frac{32\ \text{Mbyte} \times 2}{8 \times 1024 \times 512\ \text{byte}} \right) = \text{CEILING} \left( \frac{64\ \text{Mbyte}}{4\ \text{Mbyte}} \right) = 16$$

5559  
5560 The logical unit capacity can be retrieved by either reading the qLogicalBlockCount parameter in the  
5561 Unit Descriptor or issuing the READ CAPACITY command.

5562 In particular, the relations between the parameters returned by READ CAPACITY (RETURNED  
5563 LOGICAL BLOCK ADDRESS and LOGICAL BLOCK LENGTH IN BYTES), and  
5564 bLogicalBlockSize and qLogicalBlockCount parameters in Unit Descriptors are:

- 5565 ○ RETURNED LOGICAL BLOCK ADDRESS = qLogicalBlockCount - 1,

- 5566 ○ LOGICAL BLOCK LENGTH IN BYTES =  $2^{bLogicalBlockSize}$

5568   **13.2.3 Logical Unit Configuration (cont'd)**

5569   • bBootLunID

5570    bBootLunID shall be set as described in the following:

- 5571      ○ 00h: if the logical unit is not a boot logical unit,
- 5572      ○ 01h: to configure the logical unit as “Boot LU A”,
- 5573      ○ 02h: to configure the logical unit as “Boot LU B”,

5574    NOTE The 01h value and 02h value shall be assigned to no more than one logical unit.

5575   • bLUWriteProtect

5576    bLUWriteProtect shall be set as described in the following:

- 5577      ○ 00h: if the logical unit is not write protected,
- 5578      ○ 01h: to configure power on write protection,
- 5579      ○ 02h: to configure permanent write protection.

5580   • bDataReliability

5581    bDataReliability shall be set to configure the device behavior when a power failure occurs during a write operation to the logical unit:

- 5583      ○ 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,
- 5584      ○ 01h: logical unit is protected. Logical unit's data is protected against power failure.

5586   • bProvisioningType

5587    bProvisioningType shall be set to configure the logical unit provisioning type:

- 5588      ○ 00h: to disable thin provisioning,
- 5589      ○ 02h: to enable thin provisioning with TPRZ = 0,
- 5590      ○ 03h: to enable thin provisioning with TPRZ = 1.

5591   RPMB well known logical unit may be divided in multiple RPMB regions configuring the parameters described in the following.

5593   The total size of the RPMB well known logical unit is indicated by qLogicalBlockCount of RPMB Unit Descriptor. An attempt to configure the device setting the RPMB regions with total size that exceeds the value indicated by qLogicalBlockCount shall fail. An attempt to configure the device setting a RPMB Region with enable bit to one and size to zero shall fail. An attempt to configure the device setting a RPMB Region with enable bit to zero and size greater than zero shall fail.

5598   • bRPMBRegionEnable

5599    RPMB regions may be enabled setting the bits of bRPMBRegionEnable parameter of the RPMB Descriptor. To keep the compatibility to the previous version of the standard, RPMB region 0 is always enabled.

- 5602      ○ Bit-0: Don't care. RPMB region 0 is always enabled independent of this bit value.
- 5603      ○ Bit-1: Set to 1 to enable RPMB region 1
- 5604      ○ Bit-2: Set to 1 to enable RPMB region 2
- 5605      ○ Bit-3: Set to 1 to enable RPMB region 3
- 5606      ○ Bit-4 to Bit-7: Reserved

5607

5608    **13.2.3 Logical Unit Configuration (cont'd)**

5609

- 5610    • bRPMBRegion0Size  
5611    bRPMBRegion0Size configures the size of RPMB region 0 in 128KB unit (00h: 0KB, 01h: 128KB,  
5612    ... , 80h: 16384KB). The size is not directly configurable, instead it is determined by following  
5613    formula.

5614

5615     $bRPMBRegion0Size = qLogicalBlockCount / 512 - bRPMBRegion1Size \text{ (if enabled)} -$   
5616     $bRPMBRegion2Size \text{ (if enabled)} - bRPMBRegion3Size \text{ (if enabled)}$

5617    • bRPMBRegion1Size

5618    bRPMBRegion1Size configures the size of RPMB region 1 in 128KB unit (00h: 0KB, 01h: 128KB,  
5619    ... , 80h: 16384KB) if RPMB region 1 is enabled.

5620    • bRPMBRegion2Size

5621    bRPMBRegion2Size configures the size of RPMB region 2 in 128KB unit (00h: 0KB, 01h: 128KB,  
5622    ... , 80h: 16384KB) if RPMB region 2 is enabled.

5623    • bRPMBRegion3Size

5624    bRPMBRegion3Size configures the size of RPMB region 3 in 128KB unit (00h: 0KB, 01h: 128KB,  
5625    ... , 80h: 16384KB) if RPMB region 3 is enabled.

5626

5627

5628 **13.3 Logical Block Provisioning**

5629 **13.3.1 Overview**

5630 Logical Block Provisioning is the concept that describes the relationship between the logical block  
5631 address space and the physical memory resources that supports the logical address space.

5632 Logical units in a UFS device shall be either a Full Provisioned LU or a Thin Provisioned LU.

5633

5634 **13.3.2 Full Provisioning**

5635 Every LBA in a fully provisioned logical unit is mapped.

5636 A logical unit that is fully provisioned shall provide enough LBA mapping resources to contain all logical  
5637 blocks for the logical unit's capacity as reported by the device server in response to a READ CAPACITY  
5638 command.

5639 The device server shall not cause any LBA on a fully provisioned logical unit to become unmapped.

5640 A fully provisioned logical unit does not support logical block provisioning management – i.e., does not  
5641 support UNMAP command.

5642

5643 **13.3.3 Thin Provisioning**

5644 In thin provisioning there is no requirement that the available physical memory resources match the size  
5645 of the logical address space.

5646 A thin provisioned logical unit is not required to provide LBA mapping resources sufficient to contain all  
5647 logical blocks for the logical unit's capacity as reported by the device server in response to a READ  
5648 CAPACITY command.

5649 In UFS device, a thin provisioned logical unit shall have sufficient physical memory resources for all  
5650 addressable logical blocks when the logical unit is configured by writing the Configuration Descriptor:  
5651 the number of LBAs reported in READ CAPACITY shall not exceed the number of physical memory  
5652 blocks available.

5653 Logical address to physical resource allocation is managed by Logical Block Provisioning Management.  
5654 Every LBA in a thin provisioned logical unit shall be either mapped or deallocated.

5655 In UFS device, a thin provisioned logical unit shall support the Mapped and Deallocated states in the  
5656 Logical Block Provisioning State Machine. The anchored state defined in [SBC] is not supported. An  
5657 unmapped LBA is a deallocated LBA.

5658 UFS device shall support Thin Provisioned logical unit including: Logical Block Provisioning  
5659 Management, UNMAP command, erased, discard and purge functionalities as described in clause 12,  
5660 UFS Security.

5661

5662 **13.4 Host Device Interaction**

5663 **13.4.1 Overview**

5664 This sub-section describes several UFS features conceived to improve performance and/or reliability.  
5665 Some of them are related directly to commands present in the logical unit queues (e.g., inter-LU priority,  
5666 system data tag, context management), others are related to the device state (e.g., background operations,  
5667 dynamic device capacity) or focused on reliability (e.g., data reliability, real-time clock information).

5668 **13.4.2 Applicable Devices**

5669 All the features described 13.4 should be implemented on UFS devices. The extent to which the features  
5670 are implemented will be up to the device manufacturer. It is expected that poor implementations will  
5671 result in lower device performance or reliability and higher max power consumption in the low power  
5672 modes.

5673 **13.4.3 Command Queue: Inter-LU Priority**

5674 The specific details of how commands interact in a queue of a specific logical unit are handled in other  
5675 chapters. This section outlines how the queues within a device interact with each other. There can be  
5676 many different implementations of a UFS device. For example, it may be implemented as a single or  
5677 multithreaded processor. In order to make the UFS spec implementation agnostic this section outlines a  
5678 parameter that allow the host to communicate to the device its priorities so that the device can take this  
5679 into account when executing commands from the host.

5680 In the implementation case where several queues are serviced from a single execution unit, it is necessary  
5681 for the host to either designate all queues as having the same priority or designate a single Queue as  
5682 having a higher priority. A parameter value shall be defined to allow the host to designate a queue as  
5683 having high or no priority. In the case where a queue is designated as having a higher priority, whenever  
5684 a command enters the queue with the high priority it will be executed as soon as possible resulting in  
5685 commands from other queues being stalled.

5686 One example of where a host may want to take advantage of this feature is when the host has allocated  
5687 one logical unit to be a code execution unit and another unit to be a mass storage unit. In this example the  
5688 execution of code takes priority over mass storage transfers, so the host would set the parameter bit  
5689 associated with the code logical unit to be high priority and leave the remaining queues as lower priority.

5690 The logical unit supports two level of queue priorities:

- 5691 1. High priority – This is used for logical unit with high priority requests. Any commands sent to this logical  
5692 unit will have higher priority than commands sent to logical units with lower priority. For example, when  
5693 servicing demand-paging applications, read commands would have this priority.
- 5694 2. No priority – This is used for all regular logical units which do not belong to priority #1

5695 In addition to the execution of commands explicitly issued by the host, the device may execute  
5696 background operations for device house-keeping. In general those operations have a much lower priority  
5697 than the commands sent by the host and are implemented using a specific method described in 13.4.4,  
5698 Background Operations Mode.

5699

5700 **13.4.3.1 Implementation**

5701 The bHighPriorityLUN parameter in the Device Descriptor shall be set to configure which logical unit  
5702 has the command queue with the higher priority.

5703 bHighPriorityLUN shall be set to:

- 5704 • 7Fh: if all logical units have the same priority,  
5705 • LUN of the higher priority logical unit.

Name	Description	Valid Values	Default <sup>(1)</sup>
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.

5706 **13.4.4 Background Operations Mode**

5707 **13.4.4.1 Introduction**

5708 A managed device requires time to execute management tasks. The background operations mode grants  
5709 the device time to execute the commands associated with these flash management tasks. Flash  
5710 management operations may include, but are not limited to, wear leveling, bad block management, wipe  
5711 and garbage collection. The operations completed during the background operations period are  
5712 determined by the device manufacturer and are not covered by the UFS spec.

5713 **13.4.4.2 Purpose**

5714 **Performance Improvement**

5715 The intent of this mode is to improve a device response to host commands by allowing the device to  
5716 postpone device management activities that occur as a result of host initiated operations to periods when  
5717 the host is not using the device.

5718 Systems that will use a UFS device tend to have peak periods of activity, in which the best response  
5719 possible is needed from the UFS device. These peak periods are followed by idle periods, where the host  
5720 can allow the device to do device management operations. Allowing better communication between the  
5721 host and the device on when these idle periods will occur allows the UFS device to perform more  
5722 optimally in the system.

5723 The device will still be permitted to do device management when the host initiates operations, however  
5724 the downside of doing this may be poorer device performance. This mode will just give device  
5725 manufacturers the option to delay device management operations to improve performance. It is  
5726 recommended that devices postpone as many tasks as possible to take full advantage of the possible  
5727 performance improvements associated with this feature.

5728 **Host Power Management**

5729 This mode will also allow the host to control when the device uses power to perform management  
5730 activities. The host will have more knowledge about the power consumed by the system and can make the  
5731 appropriate tradeoffs about when to use system power and when to conserve it.

5732 An example of where host control over power consumed by the device could be an advantage would be  
5733 the case where the system has very little battery power and the UFS device has a lot of unused memory.

5734 **13.4.4.2 Purpose (cont'd)**

5735 In this case the host may not wish for the device to perform clean up operations but conserve the power  
5736 for more critical system functions.

5737 Allowing the host to communicate with the device on when activities can be performed will allow better  
5738 system power management, which can be controlled by the host.

5739 **13.4.4.3 Background Operations Status**

5740 The device signals to the host that the device has a need for background operations using the Exception  
5741 Event mechanism, and in particular the URGENT\_BKOPS bit in wExceptionEventStatus

- 5742 • ‘0’: No immediate need to execute background operation (corresponds to no operations or non-  
5743 critical)
- 5744 • ‘1’: Immediate need to execute background operation (corresponds to performance being impacted or  
5745 critical)

5746 When the host detects a request for executing background operations (URGENT\_BKOPS set to one) it  
5747 may read the bBackgroundOpStatus attribute to find out the need level, as follows:

- 5748 • 00h : No operations required
- 5749 • 01h : Operations outstanding (non-critical)
- 5750 • 02h : Operations outstanding (performance being impacted)
- 5751 • 03h : Operations outstanding (critical)

5752 The URGENT\_BKOPS bit is set to zero if the bBackgroundOpStatus is set to zero or one, otherwise it is  
5753 set to one.

5754 It is expected that the host will respond as soon as possible when the status changes to service, since if the  
5755 background operations are not properly managed, then the device could fail to operate in an optimal way.

5756 If device status is operations outstanding (critical), commands other than mode sense and mode select  
5757 may be terminated with CHECK CONDITION status. The host should put in all possible measures to  
5758 ensure the device never reaches this state since it means the device is no longer able to operate.

5759 The point at which the device enters each of these states is up to the manufacturer and is not defined in  
5760 this standard.

5761 **13.4.4.4 Operation Initiation**

5762 There is no explicit command to start the background operations. A mode enable bit indicates whether the  
5763 device is allowed to execute background operations. If the background operations enable bit is set and the  
5764 device is in Active power mode or Idle power mode, then the device is allowed to execute any internal  
5765 operations.

5766 When the device receives a command which requires a data transfer and if all command queues are  
5767 empty, then the device shall start the data transfer sending DATA IN UPIU or RTT UPIU within the time  
5768 declared in bBackgroundOpsTermLat. The host can minimize the device response time by disabling  
5769 background operations mode during critical performance times.

5770 In the case where the background operations status is “operations outstanding (critical)”, the  
5771 aforementioned latency limit does not apply.

5772

5773 **13.4.4.5 Power Failure**

5774 It is the device's responsibility to ensure that the data in the device is not corrupted if a power failure  
5775 occurs during a background operation.

5776 **13.4.4.6 Implementation**

5777 **13.4.4.6.1 Background Operations Enable**

5778 fBackgroundOpsEn is the Flag to be used to enable or disable the execution of background operations.  
5779 This Flag is defined as follows:

- 5780 • 0 = Device is not permitted to run background operations  
5781 • 1 = Device is permitted to run background operations.

5782 The default value of this Flag is one: background operations permitted. The device shall terminate  
5783 ongoing background operations when this Flag is cleared by the host. For more details see Table 14.25,  
5784 Flags.

5785 **13.4.4.6.2 Background Operations Status**

5786 bBackgroundOpStatus is an attribute defined as follows:

- 5787 • 00h = not required  
5788 • 01h = required, not critical  
5789 • 02h = required, performance impact  
5790 • 03h = critical.

5791 For more details see 14.3, Attributes.

5792 **13.4.4.6.3 Background Operations Termination Latency**

5793 bBackgroundOpsTermLat defines the maximum latency for starting data transmission when background  
5794 operations are ongoing. The termination latency limit applies to two cases:

- 5795 • When the device receives a COMMAND UPIU with a transfer request. The device shall start the data  
5796 transfer and send a DATA IN UPIU or a RTT UPIU within the latency limit.  
5797 • When the device receives QUERY REQUEST UPIU clearing fBackgroundOpsEn Flag. The device is  
5798 expected to terminate background operations within the latency limit.

5799 The termination limit does not apply in the case where the background operations status is "operations  
5800 outstanding (critical)".

5801 bBackgroundOpsTermLat is a parameter of the Device Descriptor and its granularity is 10msec. It is  
5802 expected that transitions between link states (e.g., HIBERNATE to ACTIVE) or temporary congestion on  
5803 the link occur in shorter timescales and are therefore negligible in comparison to the background  
5804 operations timescale.

5805 **13.4.5 Power Off Notification**

5806 A UFS host will notify the device when it is going to power the device off by requesting the device to  
5807 move to UFS-PowerDown power mode. This will give the device time to cleanly complete any ongoing  
5808 operations. The device will respond to the host when it is ready for power off, meaning that the device  
5809 entered the UFS-PowerDown power mode. Host can then power off the device without the risk of data  
5810 loss.

5811

### 5812 13.4.6 Dynamic Device Capacity

5813 Common storage devices assume a fixed capacity. This presents a problem as the device ages and gets  
5814 closer to its end of life. When some blocks of the device become too old to be used reliably, spare blocks  
5815 are reallocated to replace them. A device contains some spare blocks for this purpose, as well as for some  
5816 housekeeping operations. However, when all spare blocks are consumed, the device can no longer meet  
5817 its fixed capacity definition and it stops being functional (some become read-only, some stop responding  
5818 completely).

5819 A simple solution to enable the host to continue operation at the end of the device lifetime, would be to  
5820 allow the capacity to be dynamic. If the device is allowed to reduce its reported capacity, it can reallocate  
5821 blocks that were used to store data as new spare blocks to compensate for aging. To support this, the  
5822 device needs to report to the host how much more spare blocks it needs for each logical unit, and then the  
5823 host needs to relinquish some used blocks to let the device reallocate them as spares.

5824 A device may implement a spare blocks resource management policy either per logical unit, allocating a  
5825 fixed amount of spare blocks per logical unit, or per memory type, allocating a fixed amount of spare  
5826 blocks for all logical units with the same memory type (bMemoryType).

5827 bDynamicCapacityResourcePolicy parameter in the Geometry Descriptor indicates which spare blocks  
5828 resource management policy is implemented.

#### 5829 13.4.6.1 Implementation

##### 5830 13.4.6.1.1 Initial Device Requirements

- 5831 • Only logical units that support thin provisioning and logical block provisioning management  
5832 functions can be involved in the dynamic device capacity process. The host can discover if thin  
5833 provisioning is enabled and if a logical unit supports logical block provisioning management  
5834 functions at UTP level, reading the bProvisioningType parameter in the Unit Descriptor of each  
5835 logical unit, or at SCSI level through the TPE bit in the READ CAPACITY (16) parameter data.
  - 5836 ○ bProvisioningType shall be either 02h or 03h.
  - 5837 ○ TPE bit shall be set to one.
- 5838 • The Unit Descriptor of each logical unit includes the following two parameters: qLogicalBlockCount  
5839 and qPhyMemResourceCount.
  - 5840 ○ The qLogicalBlockCount is equal to the total number of addressable logical blocks in the logical unit.  
5841 Its value is established when the logical unit is configured and never changes during the device life  
5842 time. In particular, the qLogicalBlockCount value shall be equal to RETURNED LOGICAL BLOCK  
5843 ADDRESS + 1 (RETURNED LOGICAL BLOCK ADDRESS is a field included in the Read Capacity  
5844 Parameter Data and corresponds to the last addressable block on medium under control of logical  
5845 unit).
  - 5846 ○ The qPhyMemResourceCount is equal to the total physical memory resources available in the logical  
5847 unit, expressed in  $2^{b\text{LogicalBlockSize}}$  unit. Its value decreases with the execution of dynamic capacity  
5848 process.
- 5849 • UFS requires that there shall be sufficient resource in the physical memory resources pool to support  
5850 the logical addressable memory space reported in READ CAPACITY when the device is first  
5851 configured. Therefore, qPhyMemResourceCount shall be equal to qLogicalBlockCount in the Unit  
5852 Descriptor initially.
- 5853 • Dynamic device capacity feature does not involve the following logical units: RPMB well known  
5854 logical unit, power-on write protected logical units (independently of fPowerOnWPEn flag value),  
5855 permanently write protected logical units (independently of fPermanentWPEn flag value), or logical  
5856 units configured as boot logical unit (Boot LUN ID = 01h or 02h, independently of bBootLunEn  
5857 value).

5858   **13.4.6.1.2 Dynamic Capacity Procedural Flow**

- 5859   1) When the physical memory resources necessary for proper operation in a logical unit has been drawn  
5860   down, the device may request to the host to remove some resources from the physical memory  
5861   resources pool serving the logical address space of the logical units. This is achieved setting to one  
5862   the DYNCAP\_NEEDED bit in the wExceptionEventStatus attribute. If DYNCAP\_EVENT\_EN bit in  
5863   wExceptionEventControl attribute is one, then the EVENT\_ALERT bit of Device Information field  
5864   included in the RESPONSE UPIU will be set to one.
- 5865   2) Device shall indicate the amount of physical memory to be removed from the resource pool in  
5866   dDynCapNeeded attribute. Each element of the dDynCapNeeded[LUN] shall be equal to the amount  
5867   of bytes to be removed divided by the optimal write block size (bOptimalWriteBlockSize is a  
5868   parameter in the Geometry Descriptor). Therefore, the host can calculate the amount of physical  
5869   memory to be removed from the resource pool for each logical unit multiplying the  
5870   dDynCapNeeded[LUN] attribute by bOptimalWriteBlockSize parameter. UFS device shall not  
5871   request to remove physical memory from the resource pool of logical units which are write protected  
5872   or configured as boot logical unit (dDynCapNeeded[LUN] shall be zero).
- 5873   3) The host ensures that all outstanding tasks in the device queues have been completed or aborted.
- 5874   4) If the device spare blocks resource management policy is per memory type  
5875   (bDynamicCapacityResourcePolicy = 01h), then the host should ensure that the amount of LBAs in  
5876   the deallocated state in all logical units with the same memory type (bMemoryType) is equal to or  
5877   greater than the amount of requested physical memory resources from all logical units with the same  
5878   memory type (bMemoryType).

5879   For example, assuming that bDynamicCapacityResourcePolicy = 01h and the device asks to remove  
5880   logical blocks from the resource pool of a single logical unit, the host may remove blocks from one or  
5881   more logical units having that particular memory type as long as the total amount of logical blocks in  
5882   a deallocated state results be greater than or equal to the total amount of logical blocks specified by  
5883   the device.

5884   However, if the device spare blocks resource management policy is per logical unit  
5885   (bDynamicCapacityResourcePolicy = 00h), then host ensures that the amount of LBAs in the  
5886   deallocated state is equal or greater than the amount of requested physical memory resources for each  
5887   logical unit.

5888   The host deallocates LBAs sending one or more UNMAP commands.

- 5889   a) The range(s) of LBA in the deallocate state shall be aligned to a bOptimalWriteBlockSize  
5890   boundary, and their size shall be an integer multiple of bOptimalWriteBlockSize .
- 5891   b) The LBA range(s) shall be equal or greater than the memory resources amount that has been  
5892   requested by the device for each logical unit.
- 5893   c) The LBA range(s) does not need to be at the end of the logical address space.

- 5894   5) The host sets fPhyResourceRemoval flag to one and triggers an EndPointReset or a device hardware  
5895   reset to initiate the dynamic capacity operation.
- 5896   6) The host may either execute the complete boot process as described in 13.1, UFS Boot, or may skip  
5897   reading the boot logical unit and set fDeviceInit flag to one to start the device initialization. During  
5898   this phase the device will execute the dynamic capacity operation. The host waits until the device  
5899   clears fDeviceInit flag. The device initialization may take a time longer than normal initialization due  
5900   to internal processes necessary to re-arrange physical memory resources. If a power loss occurs, the  
5901   dynamic capacity operation will proceed at the next power up during the device initialization.

5902 **13.4.6.1.2 Dynamic Capacity Procedural Flow (cont'd)**

- 5903 7) When the dynamic capacity operation is completed, the device will clear fDeviceInit flag and the  
5904 fPhyResourceRemoval flag. If the operation is completed successfully, the DYNCAP\_NEEDED bit  
5905 is cleared too, therefore the EVENT\_ALERT bit in Device Information will return to zero, if no other  
5906 exception events are active. The qPhyMemResourceCount parameter in the Unit Descriptors is  
5907 updated with a new value reflecting the amount of physical memory resources remaining in the  
5908 resource pool.

5909 NOTE The qLogicalBlockCount parameter in the Unit Descriptors and the RETURNED LOGICAL BLOCK  
5910 ADDRESS parameter in Read Capacity Parameter Data will stay the same as initially configured. Therefore,  
5911 the updated qPhyMemResourceCount value will be less than those two parameters after dynamic capacity  
5912 operation.

- 5913 8) If the dynamic capacity operation does not succeed, for example because LBA range(s) does not meet  
5914 one or more of the requirements described in point 4, the DYNCAP\_NEEDED bit shall remain set to  
5915 one. Therefore, the EVENT\_ALERT bit in the Device Information field of the RESPONSE UPIU  
5916 will remain set to one to notify the host that further dynamic capacity operation is needed.

5917 NOTE The dDynCapNeeded[LUN] attribute value may have been updated.

- 5918 9) To account for the reduction of the physical memory resources pool after dynamic capacity operation, the  
5919 host maintains a range(s) of LBA's in deallocated state that are aligned in address and size to integer  
5920 multiples of bOptimalWriteBlockSize, with the total equal to or greater than qLogicalBlockCount minus  
5921 qPhyMemResourceCount. Otherwise, a write error will result when the host attempts to write (map) more  
5922 LBA's than the available physical memory resources.

5923 NOTE The host may change the LBA range(s) that are in deallocate state during the use of the device.

5924 **13.4.6.1.3 Dynamic Device Capacity Notification**

5925 The dynamic device capacity is one of the exception events that may set the EVENT\_ALERT bit of the  
5926 Device Information field included in the RESPONSE UPIU.

5927 To enable the setting of this bit, the DYNCAP\_EVENT\_EN bit in the wExceptionEventControl attribute  
5928 shall be set to one.

5929 When the host detects that the EVENT\_ALERT bit is set to one, it should read the  
5930 wExceptionEventStatus attribute to discover if the source of this event is a request for reducing the device  
5931 capacity. In particular, if DYNCAP\_NEEDED bit is set to one, the host should process the request as  
5932 described in this standard.

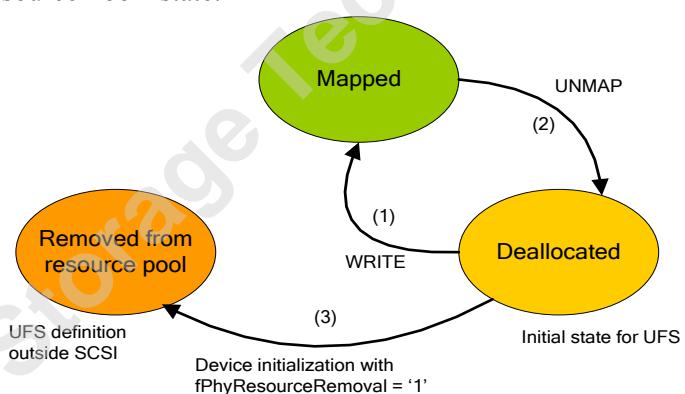
5933 The DYNCAP\_EVENT\_EN bit shall be set to zero if the host is not capable or does not intend to use the  
5934 dynamic device capacity feature. Otherwise, a dynamic capacity request event will set the  
5935 EVENT\_ALERT bit to one, masking out notification of other exception events.

5937 **13.4.6.1.4 Error handling**

- 5938 • Success/failure to unmap (release) LBA's is as defined in UNMAP command.
- 5939 • If the host ignores the dynamic device capacity notification and continues to write to the device  
5940 without unmapping LBA's to free up physical memory, the device may become non-writeable over  
5941 time and cause a WRITE command to fail. In which case, the device server shall return the following  
5942 error condition in response to the WRITE command.
- 5943     ○ The device server shall terminate the command requesting the operation with CHECK  
5944       CONDITION status with the sense key set to DATA PROTECT and the additional sense code set  
5945       to SPACE ALLOCATION FAILED WRITE PROTECT.
- 5946 • When the qPhyMemResourceCount is less than qLogicalBlockCount in Unit Descriptors, it indicates  
5947 that the physical memory resources pool is smaller than the logical addressable memory space  
5948 (LBA's) in the logical unit. It is the host's responsibility to keep track of the amount of physical  
5949 memory available. If the host attempts to write more data than the available physical memory  
5950 resources and the device is unable to complete the write operation successfully, the device shall return  
5951 the following error condition.
- 5952     ○ The device server shall terminate the command requesting the operation with CHECK  
5953       CONDITION status with the sense key set to DATA PROTECT and the additional sense code set  
5954       to SPACE ALLOCATION FAILED WRITE PROTECT.

5955 **13.4.6.1.5 Physical Memory Resource State Machine**

5956 Figure 13.5 shows the state machine for the physical memory resources. In addition to the "Mapped" state  
5957 and the "Deallocated" state, which are defined in logical block provisioning state machine too, there is the  
5958 "Removed from the Resource Pool" state.



5959  
5960 **Figure 13.5 — Physical Memory Resource State Machine**

- 5961 1) Write operation: physical memory resource from the resource pool is mapped to LBA containing  
5962 valid data.
- 5963 2) UNMAP operation for erase/discard:
- 5964     a) Physical memory resource is unmapped (deallocated) from LBA and returned to the resource  
5965       pool.
- 5966     b) Residual data in unmapped physical memory resource is not valid.
- 5967 3) Device re-initialization with fPhyResourceRemoval flag previously set to one causes some physical  
5968 memory resources to be removed from the resource pool servicing the logical address space. After  
5969 conversion the qPhyMemResourceCount is updated by UFS device to indicate the amount of physical  
5970 memory resources remaining in the resource pool of each logical unit.

5971 **13.4.7 Data Reliability**

5972 **13.4.7.1 Description**

5973 The UFS host has the ability to define the level of data reliability during normal operation and power  
5974 failure per logical unit.

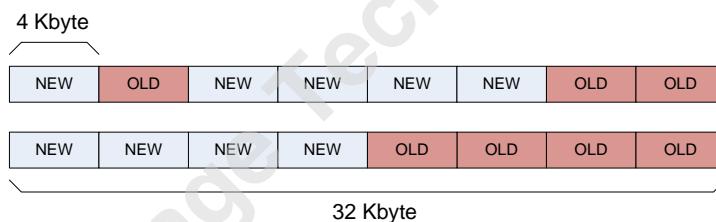
5975 There are two components to the data reliability that are defined for UFS. The first component deals with  
5976 the data currently being written and the second deals with the data that has previously been stored in the  
5977 medium.

5978 The first component, also known as Reliable Write, means that if the device losses power during a write  
5979 operation to the medium (when executing a SCSI write command, a host initiated flush of data to the  
5980 medium, etc.), the data in the range affected by the operation will be either the old data or the new written  
5981 data when the device recovers from power failure.

5982 Keeping either the old or new data is important for the file system to recover after power loss. The file  
5983 system may keep its structures segmented and signed with CRC or equivalent mechanism. The device  
5984 insures that a large enough granularity of data is authentic either with an old or new copy, to make sure  
5985 the file system can validate or invalidate these segments.

5986 The resolution for the old and new data shall be aligned to the logical block size. Figure 13.6 shows some  
5987 of the possible scenarios that the host will see when it recovers from power failure during a 32 Kbyte  
5988 write operation in a logical unit with 4 Kbyte logical block size.

5989 For RPMB well known logical unit, the data reliability granularity shall be equal to  
5990  $b_{RPMB\_ReadWriteSize} \times 256$  bytes.



5991

5992 **Figure 13.6 — Example of data status after a power failure during reliable write operation**

5993 The second component, also known as Data Reliability, allows the host to define the level of protection to  
5994 be applied to existing data on the device. In some technologies that will be used to implement UFS  
5995 devices, the device has the option to potentially sacrifice some of the existing data on a device during a  
5996 power failure in order to provide better write performance. Depending on the application for the end  
5997 device the host can select per logical unit whether the data in that logical unit shall be protected during  
5998 power failure, which may have a performance impact, or to select device performance with the risk of  
5999 losing data during a power failure.

6000 Data Reliability feature for each logical unit can be set when the device is configured during system  
6001 integration.

6002 In particular, if Data Reliability is enabled, the logical unit will execute Reliable Write operation and the  
6003 data already stored in the medium will not be corrupted by a power failure occurred during the execution  
6004 of a write operation to the medium.

6005 The data reliability feature will be configurable only for logical units and not for well known logical units.

6006 The RPMB well known logical unit will automatically select data reliability.

6007 **13.4.7.2 Implementation**

6008 bDataReliability parameter in the Unit Descriptor shall be used by the host to configure the logical unit  
6009 data reliability.

6010 bDataReliability values are defined as in the following:

6011 00h: Data reliability disabled  
6012     Corruption of the existing data in the medium of the specific logical unit may occur if a power loss  
6013     happens during device activity like writing of data to medium.

6014 01h: Data reliability enabled  
6015     The existing data stored in the medium of the specific logical unit shall not be corrupted if a power  
6016     loss occurs, and the memory range that was accessed by the interrupted write command shall  
6017     contain the old data or new data (or a mixture of old and new data as explained in 13.4.7.1) once  
6018     power is restored.

6019 **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

6020 **13.4.8 Real-Time Clock Information**

6021 Providing Real Time Clock (RTC) information to a storage device could be useful for the device internal  
6022 maintenance operations (execution of RTC internal operations is not affected by fBackgroundOpsEn flag  
6023 value).

6024 Host may provide either absolute time if available, or relative time information. This feature provides a  
6025 mechanism for the host to update either absolute or relative time.

6026 The host sends RTC information when a wPeriodicRTCUpdate has passed since the last RTC information  
6027 update. In case the device is not powered up or asleep when the period has expired, the host wakes it up  
6028 and update RTC information.

6029 NOTE When device is configured with wPeriodicRTCUpdate as ‘undefined’ (see Device Descriptor) a  
6030 wPeriodicRTCUpdate could not expire and the host may update RTC information considering vendor  
6031 recommendations.

6032 It is recommended to provide RTC information after transitioning from UFS-Sleep, UFS-DeepSleep or  
6033 UFS-PowerDown power mode to Active power mode.

6034 Updating RTC information is done by writing to dSecondsPassed attribute.

6035 While the device is busy handling an RTC update event and the related background operation, the device  
6036 may keep the fBusyRTC flag is set to one.

6037 The device may perform operations in the background as a result of receiving RTC update. In order to  
6038 allow optimized efficiency of these background operations, it is recommended that the host refrains from  
6039 sending commands, other than Query Request to the device, and keep the device powered and in Active  
6040 power mode as long as fBusyRTC flag is set to one. The device may consume active power during that  
6041 time. Therefore, it would be advisable to update RTC in times where the system is usually idle and has no  
6042 specific power limitations, e.g., at night time when battery is being charged.

### 6043 13.4.9 Context Management

6044 To better differentiate between large sequential operations and small random operations and to improve  
6045 multitasking support, contexts can be associated with groups of read or write commands. Associating a  
6046 group of commands with a shared context allows the device to optimize data handling.

6047 A context can be seen as an active session, configured for a specific read/write pattern (e.g., sequential in  
6048 some granularity). Multiple read or write commands are associated with this context to create some  
6049 logical association between them, to allow device performance optimization. For example, a large  
6050 sequential write pattern may have better performance by allowing the device to improve internal locality  
6051 and reduce some overhead (e.g., if some large unit of data is allowed to be affected by a power failure as a  
6052 whole while it is being written, all of the commands that fill this unit can work faster because they can  
6053 reduce the overhead usually required to protect each write individually in case of a power failure).  
6054 Furthermore, handling of multiple concurrent contexts allows the device to better recognize each of the  
6055 write patterns when they are all mixed together.

6056 The maximum number of contexts the device can support is reported in the bMaxContextIDNumber field  
6057 of the Geometry Descriptor. When the host configures the device, it divides this number across the  
6058 created LUs and writes the number of contexts to be supported in each LU to wContextCapabilities field  
6059 in the Configuration Descriptor.

6060 To use a context, an available Context ID shall be picked. Then, it shall be initialized by writing the  
6061 configuration attribute of the relevant LU (wContextConf). Then, data can be read/written associated to  
6062 the context by specifying the Context ID in GROUP NUMBER field of the CDB of the read/write  
6063 command. When the context is no longer used, the configuration attribute (wContextConf) should be  
6064 written as all ‘00’ by the host to close the context. A context shall be closed prior to re-configuring it for  
6065 another configuration/use.

6066 No ContextID shall be open after power cycle.

#### 6067 13.4.9.1 Context configuration

6068 Before any context can be used it shall be configured.

6069 Configuration is done by setting the context configuration attribute of the relevant LU (setting  
6070 wContextConf with INDEX equal to LUN and SELECTOR equal to ContextID, SELECTOR ‘0’ is  
6071 reserved.). Then, all read commands or write commands that are associated with this ContextID shall be  
6072 sent using GROUP NUMBER set to ContextID.

6073 When the context is no longer needed, it should be closed by writing a zero byte to the configuration  
6074 attribute.

6075 To configure a specific ContextID for a specific LU, the following fields shall be written to the context  
6076 configuration attribute of the specific context needed:

- 6077 • Activation and direction mode (read-only, write-only or read/write)  
6078     The direction indicates if all following accesses to this context would be either read-only, write-only  
6079     or both read/write. Writing a non-zero direction to this field is the ‘activation code’ for the context. A  
6080     zero in this field means a closed context which can no longer be addressed by its ID until re-  
6081     configured.
- 6082 • Large Unit context flag  
6083     This indicates if the context is following Large Unit rules or not
  - 6084         ○ If Large Unit context flag is set, then the Large Unit multiplier is used to specify the larger unit  
6085         size.
- 6086 • Reliability mode  
6087     Controls how data written to a context should respond to interruptions.

6088 **13.4.9.2 Activation and direction mode**

6089 A non-zero context can be configured as a read-only context, a write-only context or a read/write context.  
6090 Any read command associated with any context to an address that is part of an active write-only context is  
6091 not allowed, and may either fail the command or return dummy data.  
6092 Any write command associated with any context to an address that is part of an active read-only context is  
6093 not allowed, and may either fail the command, ignore the data written or cause unexpected data to return  
6094 in the read commands.  
6095 A context that is configured as read/write context may be read or written, as long as the writing follows  
6096 the context rules.

6097 NOTE read/write context may have reduced performance compared to read-only or write-only contexts.

6098

6099 **13.4.9.3 Large-Unit mode**

6100 The Large Unit is the smallest unit that can be used for large sequential read/write operations, in order to  
6101 reduce internal overhead and improve performance.

6102 Accessing a Large Unit (both read and write) shall:

- 6103 • Use a ContextID configured to operate in Large Units
- 6104 • Always access a full Large Unit, in order and from beginning to end
  - 6105 • Multiple read/write commands with TRANSFER LENGTH smaller than the Large Unit size may  
6106 be used to read or write the Large Unit. Read/write commands may be interleaved with other  
6107 accesses and commands, as long as the specific Large Unit is being accessed with its own  
6108 separate Context ID
  - 6109 • A Large Unit that is being written shall not be modified outside the scope of the context (e.g., no  
6110 other writes from other contexts to the address range of the Large Unit shall be used, no  
6111 erases/trims to that range, etc.)
  - 6112 • Different Large Units belonging to the same context may be located in non consecutive addresses  
6113 on the media, as long as alignment is kept (a Large Unit shall be accessed in order from  
6114 beginning to end, but only within the range of the specific Large Unit – the next Large Unit can  
6115 be non-consecutive and even in a lower address)
- 6116 • When writing a Large Unit context, data shall always be aligned and in multiples of  
6117 bOptimalWriteBlockSize

6118 When writing a Large Unit context, the last Large Unit before closing the context may be partially  
6119 written, as long as it is written from the beginning, in order and up to a specific point where it is closed.  
6120 The rest of the Large Unit may be padded by the device to the end of the Large Unit with random data.

6121

6122 **13.4.9.4 Reliability mode**

6123 In case a write command to a Context ID is interrupted, the device behaves as if all the writes to the  
6124 context from its configuration were written in one large write command.

6125 In case of a power failure or software reset before closing an active context – even if not in the middle of  
6126 a write command to the specific context (even if not in the middle of any command) – is considered as if  
6127 the event occurred during writing the entire context.

6128 A context behavior is determined as part of its configuration and is applied to all writes to this context  
6129 until it is closed. Interruption during any of the writes may cause some of the data not to be fully  
6130 programmed on the device. Still no partial Logical Block (of bLogicalBlockSize size) shall exist – any  
6131 Logical Block written as part of the context shall contain either the new data written or its old data before  
6132 the context was configured. The scope of data that may be affected by the interruption depends on the  
6133 mode configured:

- 6134 • For non-Large Unit contexts:

- 6135     ○ MODE0 – Normal mode – Any data written to the context from the time it was configured may  
6136       be affected
- 6137     ○ MODE1 – Non-Large Unit, reliable mode – Only data written by a specifically interrupted write  
6138       command may be affected. Any previously completed write to the context shall not be changed  
6139       because of any interruption.

- 6140 • For Large Unit contexts:

6141     NOTE In the following cases, the unit N refers to the current Large Unit which is being written when  
6142       interruption occurs, unit N-1 refers to the last Large Unit of the context that was written completely before the  
6143       current one and unit N-2 and earlier are Large Units that were completed before the N-1 unit.

- 6145     ○ MODE0 – Normal mode – Any unit may be affected: Any data written to the context from the  
6146       time it was configured may be affected.
- 6147     ○ MODE1 – Large Unit, unit-by-unit mode – Unit N may be affected, units N-1 and earlier are not:  
6148       Any data written to a Large Unit context may affect the entire specific Large Unit accessed. Any  
6149       previously completed Large Units in the context shall not be changed because of any interruption.
- 6150     ○ MODE2 – Large Unit, one-unit-tail mode – Unit N and N-1 may be affected, units N-2 and  
6151       earlier are not: Any data written to a Large Unit context may affect the entire specific Large Unit  
6152       accessed and the entire completed Large Unit that was accessed before the current one. Any other  
6153       completed Large Units in the context shall not be changed because of any interruption.

6154 In case the host sends a Task Management Request to abort a write command to a non-zero context or the  
6155       write command fails with an error, the write may still be interrupted like any context-less write. In case  
6156       these scenarios are interrupting a write to a Large Unit context, the device shall always stop writing on a  
6157       bOptimalWriteBlockSize boundary.

6158

### 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.

### 6201 13.4.11 Exception Events Mechanism

6202 The Exception Events Mechanism is used by the device to report occurrence of certain events to the host.  
6203 It consists of three components EVENT\_ALERT bit, the wExceptionEventStatus attribute and  
6204 wExceptionEventControl attribute:

- 6205 • A bit in wExceptionEventStatus attribute is assigned to each exception event. The device shall set the  
6206 wExceptionEventStatus bits to one when the corresponding exception events are active, otherwise  
6207 they shall be set to zero.
- 6208 • A bit in wExceptionEventControl attribute is assigned to each exception event.  
6209 EVENT\_ALERT bit shall be set if there is at least one wExceptionEventStatus bit and  
6210 wExceptionEventControl bit pair set to one.  
6211 The setting of an wExceptionEventStatus bit to one will not force the EVENT\_ALERT bit to one if  
6212 the corresponding bit in the wExceptionEventControl is zero.
- 6213 • The EVENT\_ALERT is a bit in the Device Information field of the RESPONSE UPIU which is the  
6214 logical OR of all bits in the wExceptionEventStatus masked by the bits of the  
6215 wExceptionEventControl. The EVENT\_ALERT bit is set to one when at least one bit in the  
6216 wExceptionEventStatus is set and the corresponding wExceptionEventControl bit is one. The  
6217 EVENT\_ALERT is set to zero if all exception events that are enabled in the wExceptionEventControl  
6218 are not active.

6219 There are seven defined exception events: dynamic device capacity, system pool exhausted , background  
6220 operation, too high temperature, too low temperature, performance throttling and WriteBooster Buffer  
6221 flush. The bits in the wExceptionEventStatus associated with those exception events are described as  
6222 follows:

- 6223 • DYNCAP\_NEEDED – the device requests a Dynamic Capacity operation (see 13.4.6, Dynamic  
6224 Device Capacity). This bit is cleared once a Dynamic Capacity operation has completed successfully,  
6225 releasing the entire capacity that the device had requested to release.
- 6226 • SYSPOOL\_EXHAUSTED – the device ran out of resources to treat further host data as System Data  
6227 (see 13.4.10, System Data Tag Mechanism). This bit is cleared once the host has turned enough  
6228 memory areas that were previously handled as System data areas, to non-system data areas.
- 6229 • URGENT\_BKOPS – the device requests host attention for the level of need in Background  
6230 Operations (see 13.4.4, Background Operations Mode). This bit is cleared once  
6231 bBackgroundOpStatus returns to 00h or 01h.
- 6232 • TOO\_HIGH\_TEMP – the device requests that the host takes action to reduce the device's Tcase  
6233 temperature.
- 6234 • TOO\_LOW\_TEMP – the device requests that the host takes action to increase the device's Tcase  
6235 temperature.
- 6236 • PERFORMANCE\_THROTTLING – the device is operating at reduced performance. The host may  
6237 read bThrottlingStatus to determine the cause of reduced performance.
- 6238 • WRITEBOOSTER\_FLUSH\_NEEDED – Buffer for WriteBooster needs to be flushed. The host is  
6239 expected to issue a flush command by setting fWriteBoosterBufferFlushEn as '1'.

6240 In the Device Information field of the RESPONSE UPIU, the device will only indicate the events that  
6241 were enabled by the host through writing to the wExceptionEventControl attribute. The event bits in the  
6242 wExceptionEventStatus attribute and in the Device Information field of the Response UPIU are cleared  
6243 by the device when the clear conditions are met.

6244 **13.4.12 Queue Depth Definition**

6245 Each logical unit is responsible for managing its own task set. Independently from the task set, the  
6246 resources used for queueing tasks may either be statically allocated to each LU, so that the LU is capable  
6247 of queueing new tasks up to a certain depth, or be shared by all LUs, so that queueing resources are  
6248 dynamically allocated to LUs, depending on tasks received.

6249 A device may implement one of the two queueing architectures described above. The device informs the  
6250 host software on the policy implemented using read only parameters in the Device Descriptor and the  
6251 Unit Descriptors.

6252 The depth of a queue is defined as the number of pending commands which can be stored in the queue.

6253 **13.4.12.1 Shared Queue**

6254 In the shared queue architecture, the device has a fixed-depth queue where tasks are queued as they are  
6255 received, regardless of their LUN designation.

6256 When a COMMAND UPIU is received, resources are allocated from the shared queue and the command  
6257 is accounted towards the queue depth limit. The host is expected to track the queue depth and not issue  
6258 more commands than can be stored in the queue. If queue resources are unavailable, the device shall  
6259 return a response with TASK SET FULL status.

6260 QUERY REQUEST UPIUs, NOP OUT UPIUs, and TASK MANAGEMENT REQUEST UPIUs are not  
6261 stored in the shared queue.

6262 When this queueing architecture is implemented, the parameter bQueueDepth in the Device Descriptor  
6263 shall indicate the depth of the queue. The value of bQueueDepth shall be equal to, or larger than, 1. The  
6264 bLUQueueDepth parameters in all Unit Descriptors shall all be equal to 0, while bLUQueueDepth in  
6265 RPMB Descriptor may be 0 or 1 (see 13.4.12.3).

6266 **13.4.12.2 Per-Logical Unit Queues**

6267 In the per-LU queueing architecture, the device implements separate fixed-depth queues, one queue for  
6268 each LU, or, in other words, allocates a fixed number of queueing resources for each LU.

6269 When a COMMAND UPIU is received, resources are allocated from the queue associated with its logical  
6270 unit, as indicated by the LUN field in the UPIU Header. The command is accounted towards the depth  
6271 limit of the respective queue. The host is expected to track the queue depths and not issue more  
6272 commands than can be stored in their designated queue. If resources are unavailable for the designated  
6273 LU, the device shall return a response with TASK SET FULL status (even if queueing resources are  
6274 available for other LUs).

6275 QUERY REQUEST UPIUs, NOP OUT UPIUs, and TASK MANAGEMENT REQUEST UPIUs are not  
6276 stored in the LU queues.

6277 When this queueing architecture is implemented, the bLUQueueDepth parameters in Unit Descriptors  
6278 shall indicate the depth of the queue of each logical unit. If a logical unit is enabled, the value of  
6279 bLUQueueDepth in its Unit Descriptor shall be equal to, or larger than, 1.

6280 bQueueDepth parameter in the Device Descriptor shall be equal to 0.

6281 NOTE For backward compatibility with previous revisions of the standard, if bLUQueueDepth for an LU is 0, and  
6282 bQueueDepth is also 0, the queue depth should be treated by the host as unknown. The host is expected to infer the  
6283 queue depth using software algorithms.

6284

### 6285 **13.4.12.3 RPMB Well Known Logical Unit Queue**

6286 RPMB logical unit may use shared queue resources or may use its own separate queue, as implemented  
6287 by the device manufacturer.

6288 If the RPMB logical unit uses the shared queue resources, its bLUQueueDepth parameter shall be equal to  
6289 0. When a COMMAND UPIU is received, resources are allocated from the shared queue, and the  
6290 command is accounted towards the queue depth limit. The host is expected to track the queue depth and  
6291 not issue more commands than can be stored in the queue. If queue resources are unavailable, the device  
6292 shall return a response with TASK SET FULL status.

6293 If the RPMB logical unit uses a separate queue, its bLUQueueDepth parameter shall be equal to 1. The  
6294 host is expected to not issue more than one command to RPMB LU at any given time. If more than one  
6295 command is issued to RPMB LU, the device shall return a response with TASK SET FULL status.

6296 It should be noted that, unlike other LUs, it is permitted that RPMB LU has a fixed depth queue while  
6297 other LUs use a shared queue.

### 6298 **13.4.13 Device Life Span Mode**

6299 The intent of this mode is to improve the device life span by increasing the device endurance. Devices use  
6300 mechanism like wear leveling etc. for improving the device life time which is limited to P/E cycle count  
6301 given by a memory vendor. In this mode, device may use technology like lower programming voltage etc.  
6302 for operations to increase the P/E cycle count and result in improving the device life.

6303 Read and write operation performance in UFS are very high, However maximum operation speed is not  
6304 required always e.g., when user is sleeping, device is in screen-off mode, downloading large size  
6305 files/video and so on. During such scenarios, UFS host may indicate to device to use technology like  
6306 lower programming voltage etc. for operations by enabling fDeviceLifeSpanModeEn flag. On disabling  
6307 this flag, device uses normal voltage for operations. It is expected that the device will respond normally as  
6308 soon as the flag is disabled.

6309 There may be performance degradation in this mode, therefore fDeviceLifeSpanModeEn should be set  
6310 only when the device is not actively used.

6311 The improvement of device life span is dependent on device implementation.

#### 6312 **13.4.13.1 Implementation**

##### 6313 **Device Life Span Mode Enable**

6314 fDeviceLifeSpanModeEn is the Flag to be used to enable or disable the execution of technology like  
6315 lower programming voltage etc. for operations.

6316 0 = Device Life Span Mode is disabled.

6317 1 = Device Life Span Mode is enabled.

6318 The default value of this flag is zero. Host can enable this flag depending on scenarios like screen-off,  
6319 downloading large files etc.

6320

6321 **13.4.14 Refresh Operation**

6322 In order to improve reliability (e.g., retention), the device data can be refreshed at the physical block level  
6323 by erasing and re-programming its physical blocks.

6324 The refresh operation mechanism provides a better capability for the host to control refresh operations  
6325 explicitly with the refresh configuration, initiation, interruption, and progress management.

6326 This feature is optional. The device indicates whether the feature is supported or not in  
6327 bUFSFeaturesSupport parameter in Device descriptor. It aims at addressing reliability requirements.

6328 It depends on device implementations how the reliability improves on this feature.

6329 **13.4.14.1 Configuration**

6330 **13.4.14.1.1 Refresh Method**

6331 From refresh perspective, there are three types of blocks:

- 6332 – Type 1: Physical blocks that don't contain data (i.e., unmapped physical blocks). Blocks of this type  
6333 will not be refreshed.
- 6334 – Type 2: Physical blocks that contain data but the device doesn't consider them to be in need of refresh
- 6335 – Type 3: Physical blocks that contain data and the device considers them to be in need of refresh

6336 The device shall support two refresh methods: Manual-Force or Manual-Selective.

- 6337 – Manual-Force

6338 The device is obliged to refresh the amount of physical blocks as requested by the host, regardless  
6339 whether these blocks need refresh or not. Only type 2 and type 3 blocks will be refreshed.

- 6340 – Manual-Selective

6341 The device only refreshes the physical blocks that it considers to be in need of refresh. Specifically  
6342 these are blocks of type 3.

6343 The attribute bRefreshMethod needs to be initially set to either Manual-Force or Manual-Selective mode  
6344 before initiating a device refresh operation.

6345 Upon the host request, the device shall perform a refresh operation following the specified method  
6346 (Manual-Force or Manual-Selective). In addition to the host initiated refresh modes, the device may  
6347 implement an additional automatic refresh mode which is transparent to the host.

6348 **13.4.14.1.2 Refresh Unit**

6349 The host may configure bRefreshUnit attribute to specify the amount of physical blocks to be refreshed  
6350 upon a single refresh request.

6351 The device refreshes the amount of physical blocks specified by bRefreshUnit starting from the first  
6352 physical block. In order to refresh the whole device (i.e., all physical blocks), the host has to send the  
6353 refresh command one (bRefreshUnit = 01h: 100%) or more times (bRefreshUnit = 00h: Minimum refresh  
6354 capability of Device).

6355 In any case the device stops refreshing when dRefreshProgress reaches 100000 (100.000%).

6356

6357 **13.4.14.1.3 Refresh Frequency**

6358 The device provides the refresh frequency indicator (bRefreshFreq attribute) in unit of month. The host  
6359 should make sure that the whole device has been refreshed within the time period specified by the  
6360 bRefreshFreq attribute. dRefreshTotalCount is incremented to indicate that the whole device refresh is  
6361 completed.

6362 Since the required refresh frequency depends on the use condition (e.g., temperature), bRefreshFreq  
6363 attribute may be configured by OEM to let the host know how often it needs to send refresh requests.

6364 **13.4.14.2 Initiation and Interruption**

6365 The host initiates a refresh operation by setting fRefreshEnable to 1b and interrupts it by clearing  
6366 fRefreshEnable to 0b. After interruption, fRefreshEnable can be also used to resume the refresh operation  
6367 where it last stopped by setting it to 1b.

6368 bRefreshStatus attribute provides information about a single refresh operation status.

6369 The host should send a query request to set fRefreshEnable flag to one only if command queues are  
6370 empty. A query request to set fRefreshEnable flag which is processed when device command queues are  
6371 not empty may fail. If the request fails,, Query Response field in the QUERY RESPONSE UPIU shall be  
6372 set to FFh (“General Failure”), the refresh operation shall not start, and the bRefreshStatus shall be set to  
6373 04h.

6374 If a refresh operation is in progress, any request, other than Query Request (READ) and the refresh  
6375 interruption described above, will fail.

6376 Host should check the Device lifetime remained by reading bDeviceLifeTimeEst of Device Health  
6377 Descriptor. Host should consider more carefully whether refresh operation will be issued or not since the  
6378 refresh operation is consuming remained life time of device.

6379

6380 **13.4.14.3 Progress Management**

6381 The device shall indicate the refresh progress with respect to its entire physical blocks. The two  
6382 parameters are defined as progress monitors in Device Health Descriptor.

- 6383 – dRefreshTotalCount  
6384 Indicate how many times the device complete refresh for the entire device. Incremented by 1 when  
6385 dRefreshProgress reach 100%.
- 6386 – dRefreshProgress  
6387 Indicate the refresh progress with respect to its entire physical blocks in %.

6388 They work both in Manual-Force and Manual-Selective methods, and will not be changed by any  
6389 operation (e.g., WRITE(10)) other than refresh operations.

6390 It is host responsibility to keep pace of sending refresh requests based on:

- 6391 – Refresh Frequency (bRefreshFreq)  
6392 – Refresh Unit (bRefreshUnit)  
6393 – Progress monitor (dRefreshProgress, dRefreshTotalCount)

6394 For example, the host should send refresh request 1000 times during 6 month if

- 6395 – bRefreshFreq = 06h (6 month)  
6396 – bRefreshUnit = 00h (e.g., 0.100% as Minimum refresh capability of Device)

6397 bRefreshMethod (00h: Manual-Force, 01h: Manual-Selective) does not affect on how the progress  
6398 monitor grows. Regardless of the actually refreshed blocks, dRefreshProgress is increased by the same  
6399 amount. In particular this means that even if type 1 blocks (see chapter 13.4.14.1.1) are not refreshed in  
6400 Manual-Force mode and neither type 1 nor type 2 blocks are refreshed in Manual-Selective mode,  
6401 dRefreshProgress is increased by the same amount.

6402 When bRefreshMethod = 02h (Manual-Selective), even though some of physical blocks are not refreshed  
6403 by device choice, dRefreshProgress should be incremented by the same amount.

6404 For example, dRefreshProgress will be increased by 2% both in Manual-Force and Manual-Selective  
6405 mode in the following conditions.

- 6406 • Device has 100 physical blocks in total
- 6407 • bRefreshUnit is set to 00h (e.g., 2% as Minimum refresh capability of Device)
- 6408 • dRefreshProgress initially indicates 0%
- 6409 • 1st physical block is empty
- 6410 • 2nd physical block has data but does not need refresh

6411 **Case 1: bRefreshMethod = Manual-Force**

6412 Upon a single refresh command completion, dRefreshProgress will be increased by 2% even though the  
6413 device ignores 1st physical block and refreshes 2nd physical block.

6414 **Case 2: bRefreshMethod = Manual-Selective**

6415 Upon a single refresh command completion, dRefreshProgress will be increased by 2% even though the  
6416 device ignores both 1st and 2nd physical blocks.

6417 **13.4.15 Temperature Event Notification**

6418 The purpose of this feature is to provide notification to host in advance when UFS device temperature  
6419 approaches defined upper and lower boundary of temperature. This feature is optional. Two bits in  
6420 bUFSFeaturesSupport parameter specify the temperature event notification support (see 14.1.4.2 “Device  
6421 Descriptor”).

6422 When temperature of device is too high or too low that host’s awareness is needed, device shall notify this  
6423 situation to host by using wExceptionEventStatus Attribute in exception event mechanism defined in  
6424 13.4.11. When TOO\_HIGH\_TEMP in wExceptionEventStatus is raised, it is recommended for host to do  
6425 throttling or other cooling activities for lowering device Tcase temperature. When TOO\_LOW\_TEMP in  
6426 wExceptionEventStatus is raised, it is recommended for host to do activities for increasing device’s Tcase  
6427 temperature.

6428 When this temperature alarming is raised, host may want to know temperature reported by device even  
6429 though UFS device only could give rough temperature. In this purpose, device case rough temperature  
6430 shall be provided through bDeviceCaseRoughTemperature attribute. Since the temperature sensor inside  
6431 semiconductor device is not expected enough accurate, this temperature information shown in  
6432 bDeviceCaseRoughTemperature is to provide the rough temperature range only. And there could some  
6433 variation depending on location of measurement also, host should assume that this temperature  
6434 information from device have around +/- 10 °C error range. Therefore, this temperature information should  
6435 be referred by host only as rough device case temperature.

6436 To give a temperature boundary information for too high or too low temperature alarming,  
6437 bDeviceTooHighTempBoundary and bDeviceTooLowTempBoundary attribute is defined. The  
6438 temperature alarming status field in wExceptionEventStatus shall be automatically cleared by device  
6439 when device case temperature is going within boundary temperature range.

6440 **13.4.16 Performance Throttling Event Notification**

6441 The purpose of this feature is to provide notification to the host if the device is limiting performance. If  
6442 the device needs to reduce performance, the host will be notified through the Exception Event Mechanism  
6443 defined in 13.4.11. While the PERFORMANCE\_THROTTLING exception event bit is set, the host  
6444 should expect reduced performance from the device.

6445 The host may read the device bThrottlingStatus attribute to discover why the device is operating at lower  
6446 performance. Bits in bThrottlingStatus attribute will remain set while the condition exists.

6447 This feature is optional. The device indicates whether the feature is supported or not by  
6448 bExtendedUFSFeaturesSupport parameter in Device Descriptor.

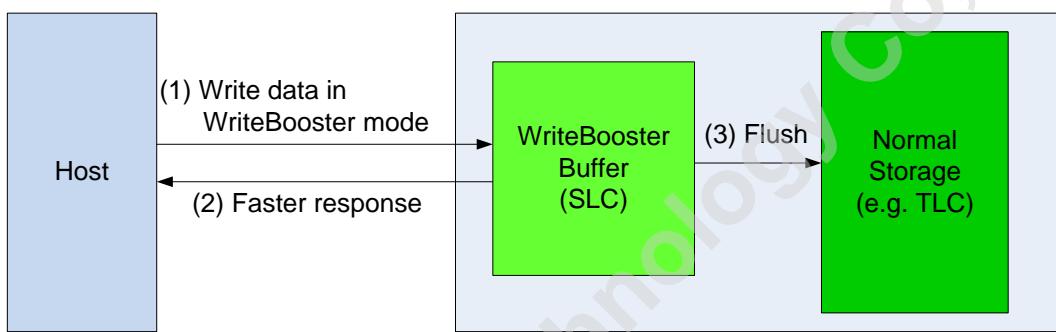
6449 How much the performance of the UFS device is reduced when a performance throttling event is notified  
6450 depends on the device implementation.

6451  
6452

6453 **13.4.17 WriteBooster**

6454 **13.4.17.1 Overview**

6455 The write performance of TLC NAND is considerably lower than SLC NAND because the logically  
6456 defined TLC bits require more programming steps and have higher error correction probability. To  
6457 improve the write performance, part of the TLC NAND (normal storage) is configured as SLC NAND  
6458 and used as write buffer, temporarily or permanently. Using SLC NAND as a WriteBooster Buffer  
6459 enables the write request to be processed with lower latency and improves the overall write performance.  
6460 Some portions of TLC NAND allocated for the user area are assigned as the WriteBooster Buffer. The  
6461 data written in the WriteBooster Buffer can be flushed into TLC NAND storage by an explicit host  
6462 command or implicitly while in hibernate (HIBERN8) state. Technologies other than TLC and SLC  
6463 NAND may be used as normal storage and WriteBooster Buffer.  
6464



6465  
6466 **Figure 13.7 — Concept of WriteBooster feature**

6467 Bit[8] of dExtendedUFSTFeaturesSupport indicates if the device supports the WriteBooster feature.

6468 There are two WriteBooster mode of operations: “LU dedicated buffer” mode and “shared buffer” mode.  
6469 In the “LU dedicated buffer” mode, the WriteBooster Buffer is dedicated to a logical unit, while in the  
6470 “shared buffer” mode all logical units share the same WriteBooster Buffer except well-known logical  
6471 units. bSupportedWriteBoosterBufferTypes indicates which modes are supported by the device. In both  
6472 WriteBooster mode of operations, the WriteBooster Buffer size is configurable.

6473 There are two user space configuration options: “user space reduction” and “preserve user space”. With  
6474 the “user space reduction”, the WriteBooster Buffer reduces the total configurable user space; while with  
6475 the “preserve user space”, the total space is not reduced.

6476 The WriteBooster feature is enabled when fWriteBoosterEn flag is set to one.

6477 bAvailableWriteBoosterBufferSize attribute indicates the available space in the WriteBooster Buffer. An  
6478 exception event is triggered when there is the need to flush the WriteBooster Buffer: bit[5] of  
6479 wExceptionEventStatus is set to indicate that data in WriteBooster Buffer should be flushed to normal  
6480 storage.

6481 There are two flags for controlling the WriteBooster Buffer flush operation. fWriteBoosterBufferFlushEn  
6482 flag enables the flush operation: when it is set to one, the device shall flush the WriteBooster Buffer.  
6483 fWriteBoosterBufferFlushDuringHibernate enables the flush operation during hibernate: the device  
6484 initiates a WriteBooster Buffer flush operation whenever the link enters in the hibernate state.

6485 bWriteBoosterBufferFlushStatus attribute provides the flush operation status, while  
6486 bWriteBoosterBufferLifeTimeEst attribute indicates the estimated lifetime of the WriteBooster Buffer.

6487 **13.4.17.2 WriteBooster configuration**

6488 Bit[8] of dExtendedUFSFeaturesSupport indicates if the device supports the WriteBooster feature. If the  
6489 device does not support this feature, a query request that attempts to set a WriteBooster parameter in a  
6490 Configuration Descriptor to a value different from zero shall fail, and the Query Response field in  
6491 QUERY RESPONSE UPIU shall be set to “General Failure”.

6492 The WriteBooster Buffer can be configured in “LU dedicated buffer” mode or “shared buffer” mode  
6493 according to the device capability. bSupportedWriteBoosterBufferTypes indicates which modes are  
6494 supported by the device.

6495 If bWriteBoosterBufferPreserveUserSpaceEn is set to 00h, the WriteBooster Buffer reduces the total user  
6496 space that can be configured at provisioning. The amount of the reduction can calculated multiplying the  
6497 WriteBooster Buffer size by the value indicated by bWriteBoosterBufferCapAdjFac. For example, the  
6498 bWriteBoosterBufferCapAdjFac value for a TLC NAND storage device with a SLC NAND WriteBooster  
6499 Buffer is 3; therefore, the total user capacity that can be configured is reduced by  
6500  $3 \times \text{WriteBoosterBufferCapacity}$ .

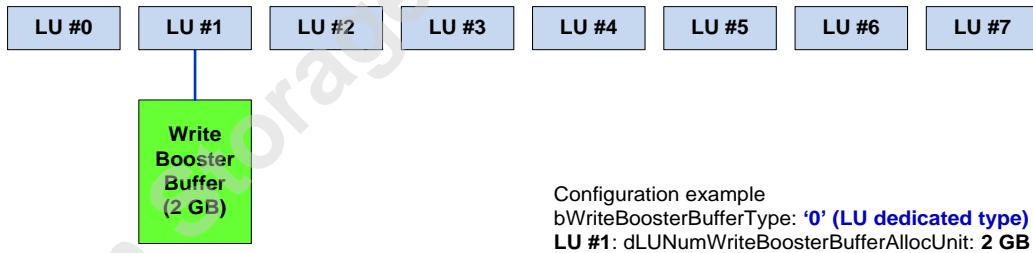
6501 Setting bWriteBoosterBufferPreserveUserSpaceEn to 01h avoids the reduction of the total user space that  
6502 can be configured at provisioning, but it may result in lower performance, see 13.4.17.5.

6503 ***LU dedicated buffer mode***

6504 If the device supports the “LU dedicated buffer” mode, this mode is configured by setting  
6505 bWriteBoosterBufferType to 00h. The logical unit WriteBooster Buffer size is configured by setting the  
6506 dLUNumWriteBoosterBufferAllocUnits field of the related Unit Descriptor. Only a value greater than  
6507 zero enables the WriteBooster feature in the logical unit. When bConfigDescrLock attribute is set to 01h,  
6508 logical unit configuration can no longer be changed.

6509 The maximum number of supported WriteBooster Buffers is defined in the bDeviceMaxWriteBoosterLUs  
6510 parameter of the Geometry Descriptor. bDeviceMaxWriteBoosterLUs is 01h, therefore the WriteBooster  
6511 Buffer can be configured in only one logical unit.

6512 Figure 13.8 shows an example of device configuration with a 2 GB WriteBooster Buffer.



6513  
6514 **Figure 13.8 — Example of “LU dedicated buffer” mode configuration**

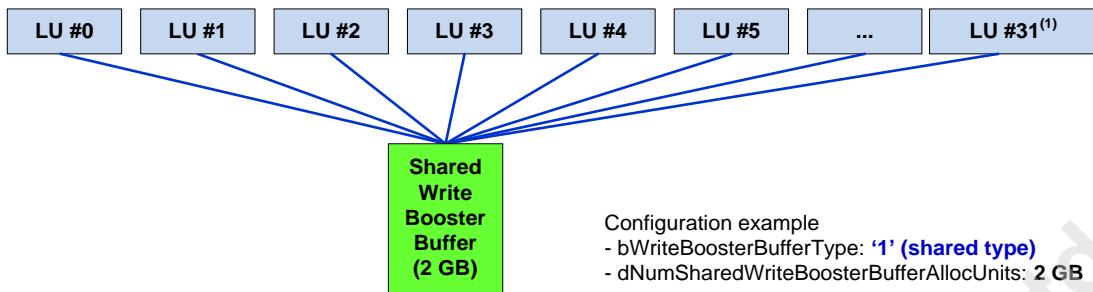
6515 The WriteBooster Buffer is available only for the logical units from 0 to 7 which are configured as  
6516 “normal memory type” (bMemoryType = 00h) and “not Boot well known logical unit” (bBootLunID =  
6517 00h), otherwise the Query Request shall fail and the Query Response field shall be set to “General  
6518 Failure”.

6519 ***Shared buffer mode***

6520 If the device supports the “shared buffer” mode, this mode is configured by setting  
6521 bWriteBoosterBufferType to 01h.

6522 The WriteBooster Buffer size is configured by setting the dNumSharedWriteBoosterBufferAllocUnits  
6523 field of the Device Descriptor. Figure 13.9 shows an example of device configuration with a 2 GB  
6524 WriteBooster Buffer.

6525 13.4.17.2 WriteBooster configuration (cont'd)



Configuration example  
- bWriteBoosterBufferType: **'1' (shared type)**  
- dNumSharedWriteBoosterBufferAllocUnits: **2 GB**

6526 NOTE 1 Number of supported logical unit is indicated by bMaxNumberLU

6527 **Figure 13.9 — Example of “shared buffer” mode configuration**

6528 Note that, if bWriteBoosterBufferType is set to 01h but dNumSharedWriteBoosterBufferAllocUnits is set  
6529 to zero, the WriteBooster feature is disabled.

6530 13.4.17.3 Writing data to WriteBooster Buffer

6531 If the fWriteBoosterEn flag is set to zero, data written to any logical unit is written in normal storage.

6532 If the fWriteBoosterEn flag is set to one and the device is configured in “shared buffer” mode, data  
6533 written to any logical unit is written in the shared WriteBooster Buffer.

6534 If the fWriteBoosterEn flag is set to one and the device is configured in “LU dedicated buffer” mode,  
6535 data written to the logical unit configured to use a dedicated buffer is written in the logical unit  
6536 WriteBooster Buffer. Data written to any logical unit not configured to use a dedicated buffer is written in  
6537 normal storage.

6538 Writes to the WriteBooster Buffer may decrease the lifetime and the availability of the WriteBooster  
6539 Buffer.

6540 In the “LU dedicated buffer” mode, the device may write data from other LUs to the WriteBooster Buffer  
6541 in case there are multiple pending commands while fWriteBoosterEn is set to one.

6542 Whenever the endurance of the WriteBooster Buffer is consumed completely, a write command is  
6543 processed as if WriteBooster feature was disabled. Therefore, it is recommended to set fWriteBoosterEn  
6544 to one, only when WriteBooster performance is needed, so that WriteBooster feature can be used for a  
6545 longer time.

6546 In the shared buffer configuration, the data that may not need the performance of WriteBooster will be  
6547 written to the WriteBooster Buffer when fWriteBoosterEn is one, there is higher probability to fill the  
6548 WriteBooster Buffer and consume its endurance earlier.

6549 If there is no available buffer, write data in WriteBooster mode will be stored in normal storage with  
6550 normal write performance. The host can identify the available WriteBooster Buffer size by referring to the  
6551 bAvailableWriteBoosterBufferSize attribute. This available buffer size is decreased by the WriteBooster  
6552 operation and increased by the WriteBooster Buffer flush operation. The available buffer size can be  
6553 increased by UNMAP commands.

6554 The WriteBooster Buffer lifetime is indicated by the bWriteBoosterBufferLifeTimeEst attribute. If the  
6555 value of bWriteBoosterBufferLifeTimeEst is equal to 0Bh (Exceeded its maximum estimated  
6556 WriteBooster Buffer lifetime), a write command shall be processed as if the WriteBooster feature was  
6557 disabled.

#### 6558   **13.4.17.4 Flushing WriteBooster Buffer**

6559   When the entire buffer for WriteBooster is consumed, data will be written in normal storage instead of in  
6560   the WriteBooster Buffer. The device informs the host when the WriteBooster Buffer is full or near full  
6561   with the exception event WRITEBOOSTER\_FLUSH\_NEEDED, see 13.4.11.

6562   The WRITEBOOSTER\_FLUSH\_NEEDED event mechanism is enabled by setting the  
6563   WRITEBOOSTER\_EVENT\_EN bit of the wExceptionEventControl attribute.

6564   There are two methods for flushing data from the WriteBooster Buffer to the normal storage: one is using  
6565   an explicit flush command, the other enabling the flushing during link hibernate state. If the  
6566   fWriteBoosterBufferFlushEn flag is set to one, the device shall flush the data stored in the WriteBooster  
6567   Buffer to the normal storage. If fWriteBoosterBufferFlushDuringHibernate is set to one, the device  
6568   flushes the WriteBooster Buffer data automatically whenever the link enters the hibernate (HIBERN8)  
6569   state.

6570   The time needed to flush the WriteBooster Buffer depends on the amount of data to be flushed. The  
6571   bWriteBoosterBufferFlushStatus attribute indicates the status of the WriteBooster flush operation. The  
6572   device shall execute the WriteBooster flush operation only when the command queue is empty. If the  
6573   device receives a command while flushing the WriteBooster Buffer, the device may suspend the flush  
6574   operation to expedite the processing of that command. Note that bWriteBoosterBufferFlushStatus will  
6575   still indicate “Flush operation in progress” (01h) even if it has been temporarily suspended. After  
6576   completing the host command, the device will resume flushing the data from the WriteBooster Buffer  
6577   automatically. While the flushing operation is in progress, the device is in Active power mode.

6578   The device shall stop the flushing operation if both fWriteBoosterBufferFlushEn and  
6579   fWriteBoosterBufferFlushDuringHibernate are set to zero.

#### 6580   **13.4.17.5 Preserve user space option**

6581   There are two user space configuration options: “user space reduction” and “preserve user space”. With  
6582   the “user space reduction”, the WriteBooster Buffer reduces the total configurable user space. While with  
6583   the “preserve user space”, the total space is not reduced. However, the physical storage allocation may be  
6584   smaller than total capacity of all logical units, since part of the physical storage is used for the  
6585   WriteBooster Buffer.

6586   When the physical storage allocated for the logical units is fully used, the device will start using the  
6587   physical storage allocated for the WriteBooster Buffer. Therefore, the WriteBooster Buffer size may be  
6588   less than what initially configured. The current size of the WriteBooster Buffer can be discovered reading  
6589   the dCurrentWriteBoosterBufferSize attribute. Approximate available space in the WriteBooster Buffer  
6590   can be calculated by multiplying the value of bAvailableWriteBoosterBufferSize (percentage of available  
6591   WriteBooster Buffer) by the value of dCurrentWriteBoosterBufferSize (current WriteBooster Buffer  
6592   size).

6593   The bSupportedWriteBoosterBufferUserSpaceReductionTypes parameter of the Geometry Descriptor  
6594   indicates which options are supported. Setting bWriteBoosterBufferPreserveUserSpaceEn parameter of  
6595   the Device Descriptor to 01h enables “preserve user space”.

6596   The disadvantage of this mode is that there could be performance degradation when the physical storage  
6597   used for the WriteBooster Buffer is returned to user space, since the device has to make internal data  
6598   structure adjustments as well as flush the WriteBooster Buffer data. If the free storage is increased  
6599   enough, the device can build the WriteBooster Buffer from the physical storage again. If the remaining  
6600   storage is repeatedly increased and decreased, the WriteBooster Buffer may be repeatedly built from and  
6601   returned to the user storage space and performance degradation can occur.

6602    **13.4.17.5 Preserve user space option (cont'd)**

6603    The amount of performance degradation due to returning the storage used for WriteBooster Buffer to user  
6604    storage space is device specific. This is because the condition for migration between user space and the  
6605    WriteBooster Buffer depends on device capacity and vendor's migration policy, which includes the  
6606    granularity and frequency of migration, what percentage of free user space remains from which the  
6607    migration starts, etc. For more details, see device data sheet.

6608

Biwin Storage Technology Co., Ltd.

6609 **13.5 UFS Cache**

6610 Cache is a temporary storage space in a UFS device. The cache should in typical case reduce the access  
6611 time (compared to an access to the medium) for both write and read. The cache is not directly accessible  
6612 by the host but is a separate element in a UFS device. This temporary storage space may be utilized also  
6613 for some implementation specific operations like as an execution memory for the memory controller  
6614 and/or as storage for an address mapping table etc. but which definition is out of scope of this standard.

6615 The implementation of the cache is optional for the UFS devices but the related commands shall be  
6616 implemented so that compatibility with host software driver is seamless independent from the  
6617 implementation. Devices may explicitly indicate that cache is supported by setting INQUIRY Data VPD  
6618 page (see [SPC]) parameter V\_SUP=1b. V\_SUP=0b means that there may or may not be cache  
6619 implemented. INQUIRY Data VPD page parameter NV\_SUP shall be set to 0b.

6620 The cache is a device level cache and applies for all LUs generically. Data written to and read from a  
6621 Boot W-LU and RPMB W-LU shall not be cached due to the specific nature of these LUs.

6622 The cache is expected to be volatile by nature. Data in the cache is not expected to remain valid over  
6623 power cycles or HW/SW resets. The UFS device is expected to manage the cache so that it shall not be  
6624 possible to read stale data from the device.

6625 While the cache is implemented the device server may utilize the cache during write and read operations  
6626 for storing data which an application client may request later. The algorithm to manage the cache is out of  
6627 scope of this standard and is left for the implementation. There are parameters related to the management  
6628 of the cache defined in CDBs (see bullets) and in 11.4.2.3, Caching Mode Page.

- 6629 • The disable page out (DPO) bit in the CDB of write, read and verify commands allows the application  
6630 client to influence the replacement of the logical blocks in the cache (e.g., in case the cache is full).  
6631 Setting the DPO bit to 1 means that the device server should not replace the existing logical blocks in  
6632 the cache with the new logical blocks written or read. When the DPO and FUA bits are set to one,  
6633 write and read operations effectively bypass the cache.
- 6634 • The force unit access (FUA) bit in the CDB of write and read commands enables the application  
6635 client to access the medium. Setting the FUA bit to 1 means that the device server shall perform the  
6636 write to the medium before completing the command and to read logical blocks from the medium (not  
6637 from the cache).

6638 During write operations the device server may use the cache to store data that is to be written to the  
6639 medium at a later time (write-back caching) and thus the command may complete prior to logical blocks  
6640 being written to the medium. This means also that such data may get lost if sudden power loss or reset  
6641 occurs. There is also possibility of an error occurring during the actual write operation to the medium  
6642 later. If an error occurred during such write operation it may be reported as a deferred error on a later  
6643 command.

6644 An UNMAP operation or any other operation which affects data of a logical block in the medium shall  
6645 cause the device server to update potential data related to the logical block in the cache accordingly.

6646 It is recommended that the host synchronizes the cache before initiating a PURGE operation.

6647 When a VERIFY command is processed both force unit access and synchronize cache operation are  
6648 implied.

6649 Following commands shall be implemented by the device server to enable application client to control the  
6650 behavior of the cache:

- 6651 • PRE-FETCH commands: see 11.3.19 and 11.3.20
- 6652 • SYNCHRONIZE CACHE commands: see 11.3.24 and 11.3.25

6653 **13.6 Production State Awareness (PSA)**

6654 **13.6.1 Introduction**

6655 UFS device can utilize knowledge about its production status and adjust internal operations accordingly.

6656 For example, content which was loaded into the storage device prior to device soldering might be  
6657 corrupted, at a higher probability than in regular mode. The UFS device could use “special” internal  
6658 operations for loading content prior to device soldering which would reduce production failures and use  
6659 “regular” operations post-soldering.

6660 The sensitivity for device soldering is a property of the logical unit, some logical units may be sensitive to  
6661 device soldering while some logical units may not be sensitive to it. Before loading the data to the device  
6662 the host should read bPSASensitive, to identify the LU which are sensitive to device soldering.

6663 Pre-loaded data is data which is loaded on the device after device configuration is completed  
6664 (bConfigDescrLock is set and reset of the device), and before device soldering to the host platform. The  
6665 combined maximum amount of data which could be pre-loaded to all sensitive LUs is device specific and  
6666 defined by dPSAMaxDataSize attribute.

6667 **13.6.2 PSA flow**

6668 PSA feature is based on the device capability to uniquely identify which data is written before the  
6669 soldering process. The PSA flow may be initiated only if all LBAs in logical units with bPSASensitive =  
6670 01h are unmapped. In case the host does not know if LBAs are unmapped, then it should set bPSAState  
6671 ‘Off’, send an UNMAP command for the entire LBA range of each LU with bPSASensitive set to 01h, to  
6672 unmap that data and re-start the PSA flow as described in the following.

6673 Depicted in Figure 13.10 is the PSA flow. To start the PSA flow, the host first checks if PSA feature is  
6674 supported by the device (see bUFSFeaturesSupport parameter in Device descriptor).

6675 The host is expected to set dPSADataSize to indicate amount of data it plans to pre-load in all logical  
6676 units with bPSASensitive = 01h. In case the host tries to set dPSADataSize > dPSAMaxDataSize, then  
6677 the device shall return a General failure error.

6678 The host writes bPSAState attribute to ‘Pre-soldering’ and then pre-loads the data in the logical units  
6679 through WRITE commands.

6680 The host should count towards dPSAMaxDataSize limit only data written through a WRITE command to  
6681 a LU with bPSASensitive descriptor set to ‘1’. During PSA flow the host is not expected to write data to  
6682 the same LBA more than once, in such case device behavior may be undefined.

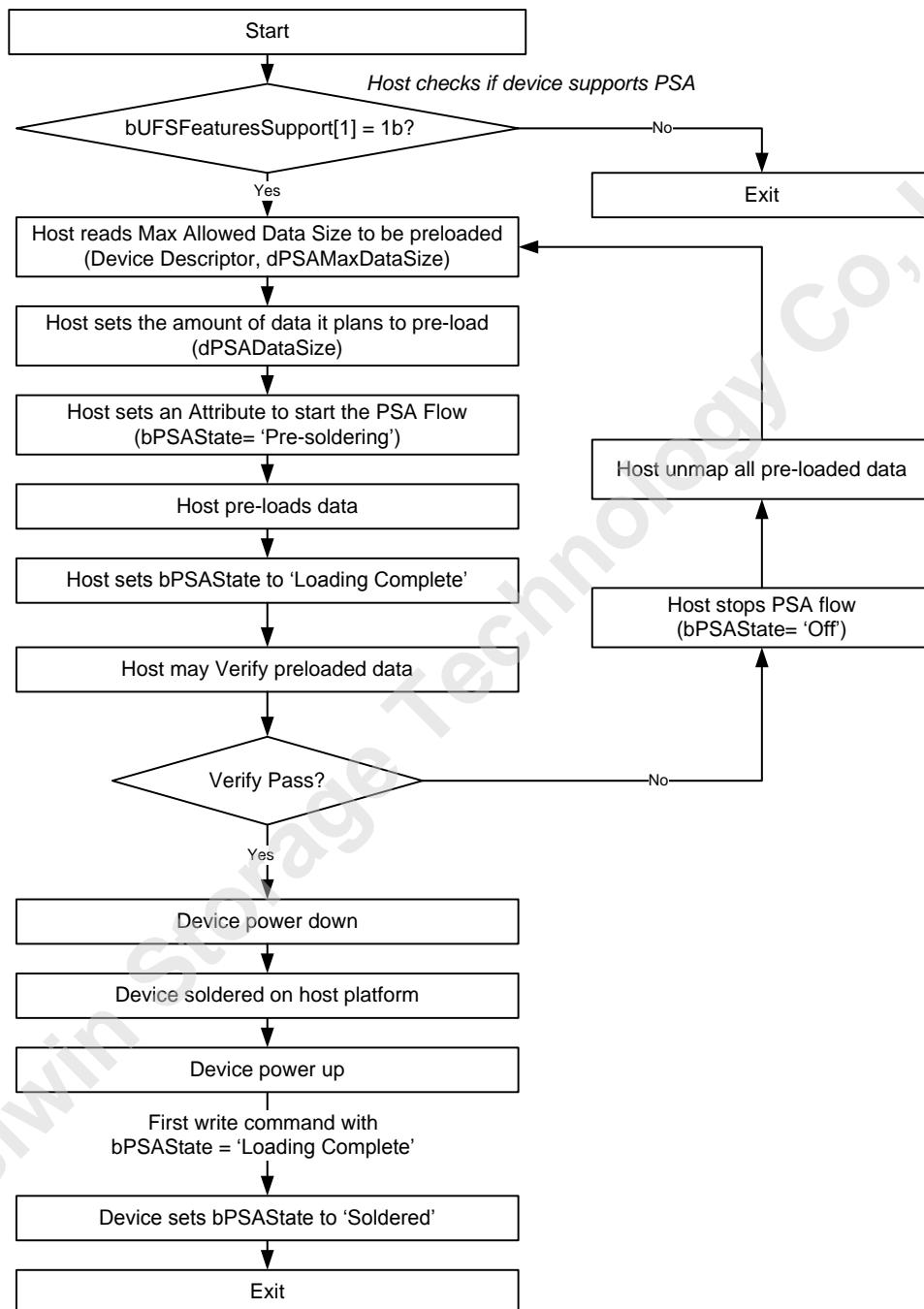
6683 Once the host finishes pre-loading all LU (wrote total of dPSADataSize amount of data) the host is  
6684 expected to change the state of bPSAState from ‘Pre-soldering’ to ‘Loading Complete’ to indicate to the  
6685 device that pre-loading of data is complete.

6686 Prior to soldering, at ‘Loading Complete’ bPSAState state, device may stop using special internal  
6687 operations and resume regular operations. Therefore, the host should not write data to the device as data  
6688 may be corrupted during soldering; a WRITE Command in this situation may result in an error.

6689

6690 **13.6.2 PSA flow(cont'd)**

6691 After the setting of bPSAState to 'Loading Complete', the device may be soldered. The device shall set  
6692 bPSAState to 'Soldered' during the processing of the first WRITE command after a power-up occurred  
6693 with bPSAState = 'Loading Complete'.



6694  
6695  
6696

**Figure 13.10 — PSA flow**

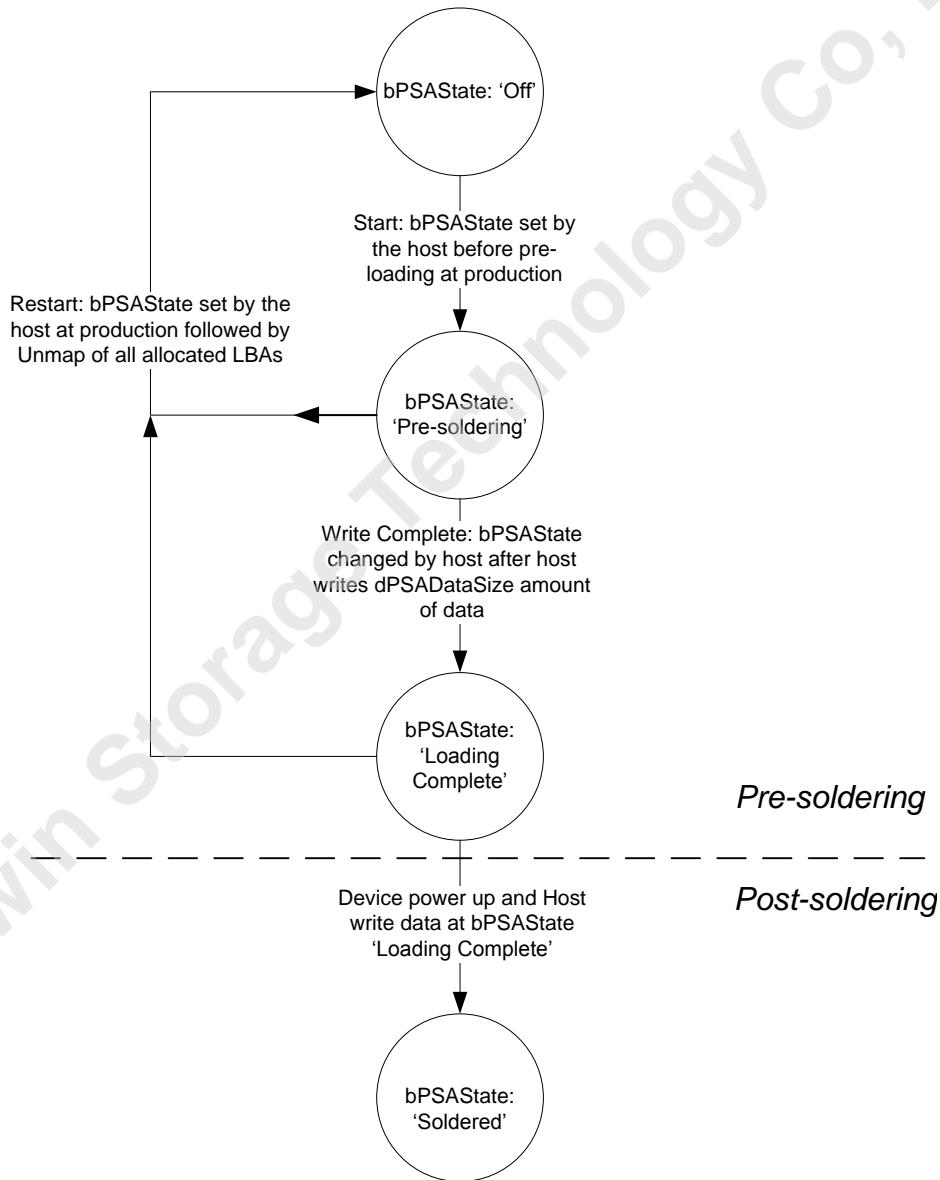
6697 **13.6.2 PSA flow(cont'd)**

6698 Depicted in Figure 13.11 is the PSA state machine which describes the different states of the bPSAState  
6699 attribute and the transitions between its states.

6700 Host misbehavior of writing, during pre-soldering phase, more data than indicated by dPSADataSize may  
6701 result in data corruption during device soldering.

6702 At any time before setting bPSAState to ‘Soldered’ the host may re-start the PSA flow by switching  
6703 bPSAState to ‘Off’, unmap all sensitive data and set bPSAState to ‘Pre-soldering’.

6704 A change in bPSAState attribute may involve additional operations by the device which may require  
6705 some time. bPSAStateTimeout indicates the maximum allowed timeout in which the device may return a  
6706 response.



6707 **Figure 13.11 — PSA state machine**

6708

---

## 6709 14 UFS Descriptors, Flags And Attributes

---

### 6710 14.1 UFS Descriptors

6711 A descriptor is a data structure with a defined format. Descriptors are accessed via QUERY REQUEST  
6712 UPIU packets. Descriptors are independently addressable data structures. Descriptors may be stand alone  
6713 and unique per device or they may be interrelated and linked in a hierarchical fashion to other descriptors  
6714 by parameters defined within the top-level descriptor. Descriptors can range in size from 2 bytes through  
6715 255 bytes. A 2 byte descriptor is an empty descriptor. All descriptors have a length value as their first  
6716 element. This length represents the entire length of the descriptor, including the length byte. All  
6717 descriptors have a type identification as their second byte. If a parameter in a Descriptor has a size greater  
6718 than one byte, the byte containing the MSB is stored at the lowest offset and the byte containing the LSB  
6719 is stored at the highest offset (i.e., big-endian byte ordering). A descriptor can be partially read, but  
6720 starting point is always the offset 00h.

6721 A Descriptor is a block or page of parameters that describe something about a Device. For example, there  
6722 are Device Descriptors, Configuration Descriptors, Unit Descriptors, etc.

6723 In general, all Descriptors are readable, some may be write once, others may have a write protection  
6724 mechanism.

6725 The Configuration Descriptor is writeable and allows modification of the device configuration set by the  
6726 manufacturer.

6727 Table 14.1 specifies the descriptor identification values.

6728

**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

6729 **14.1.1 Descriptor Types**

6730 Descriptors are classified into types, as indicated in Table 14.1. Some descriptors are singular entities,  
6731 such as a Device descriptor. Others can have multiple copies, depending upon the quantity defined, such  
6732 as UNIT descriptors. See Figure 14.1.

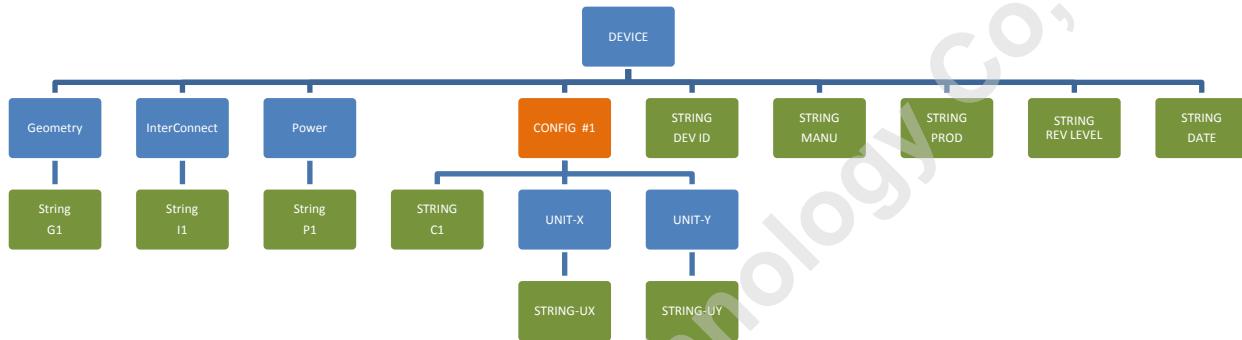
6733

#### 6734 14.1.2 Descriptor Indexing

6735 Each descriptor type has an index number associated with it. The index value starts at zero and increments  
6736 by one for each additional descriptor that is instantiated. For example, the first and only Device  
6737 Descriptor has an index value of 0. The first String Descriptor will have an index value of 0. The Nth  
6738 String Descriptor will have an index value of N-1.

#### 6739 14.1.3 Accessing Descriptors and Device Configuration

6740 Descriptors are accessed by requesting a descriptor IDN and a descriptor INDEX. For example, to request  
6741 the fifth Unit descriptor, the request would reference TYPE UNIT (numeric value = 2) and INDEX 4  
6742 (numeric value = N-1).



6743  
6744 **Figure 14.1 — Descriptor Organization**

6745

6746 All descriptors are read-only, except the Configuration Descriptors and the OEM\_ID String Descriptor  
6747 which are: readable, writeable if bConfigDescrLock attribute value is equal to 00h.

6748 In particular, the Configuration Descriptors allow modification of the device configuration set by the  
6749 manufacturer. Parameter settings in the Configuration Descriptors are used to calculate and populate the  
6750 parameter fields in the Device Descriptor and the Unit Descriptors – an internal operation by the device.

6751 The host may write the Configuration Descriptors multiple times if bConfigDescrLock attribute is equal  
6752 to zero.

6753 A device that supports 8 logical units has only one Configuration Descriptor, while a device that supports  
6754 32 logical units has four Configuration Descriptors. The host may write only the Configuration  
6755 Descriptors related to the logical units to be configured, and avoid to send write descriptor query requests  
6756 for the Configuration Descriptors that do not need to be changed.

6757 The bConfDescContinue parameter in Configuration Descriptor indicates the end of a sequence of write  
6758 descriptor query requests during device configuration. In particular, if bConfDescContinue is set to one,  
6759 the current query request will be followed by another one, and the device shall not start internal  
6760 operations to implement the new configuration. If bConfDescContinue is set to zero, the current query  
6761 request is the last one, and the device shall start internal operations to implement the new configuration.

6762

6763 **14.1.3 Accessing Descriptors and Device Configuration (cont'd)**

6764 For example, to configure LU 0, LU 1, LU 2, LU 18 and LU 20 the following write descriptor query  
6765 request may be sent.

6766 1) write descriptor query request with INDEX = 0, bConfDescContinue = 01h, Device Descriptor  
6767 parameters, logical unit parameters from 0 to 7

6768 2) write descriptor query request with INDEX = 2, bConfDescContinue = 00h, Device Descriptor  
6769 parameters, logical unit parameters from 16 to 23

6770 The latency of a write descriptor request with bConfDescContinue is set to zero may be significantly  
6771 longer than a query request with bConfDescContinue set to one. If bConfDescContinue = 00h, then the  
6772 device shall send a QUERY RESPONSE UPIU only after completing the configuration process.

6773 If bConfDescContinue = 0h, then the Query Response field in the QUERY RESPONSE UPIU shall be set  
6774 to "Success" only if

- 6775 • parameters in Device Descriptor and Unit Descriptors have been updated successfully  
6776 • logical units configuration has been completed successfully  
6777 • logical units are ready for operation (read, write, etc.)

6778 If the query request fails, it shall be possible to repeat the configuration procedure sending a new  
6779 sequence of write descriptor query requests.

6780 The device may not be able to properly process SCSI commands while updating its configuration,  
6781 therefore Host should not send them.

6782 The Configuration Descriptor for same index may be sent more than once by host (eg, Configuration  
6783 Descriptor with Index = 00h) with bConfDescContinue = 01h. The last received values of each  
6784 Configuration Descriptor index before bConfDescContinue set to zero, shall be considered as valid  
6785 configuration.

6786 If power cycle happens while bConfDescContinue is 01h, then all the configuration descriptor values  
6787 shall be reset to previous successful configuration value or if there is no previous configuration then they  
6788 shall be reset to MDV values as per UFS Specification.

6789 Once the device has been configured, the setting of bConfigDescLock to one permanently locks the  
6790 device configuration. Some devices may be configured multiple times during system setup or system  
6791 development. The details and restrictions related to multiple device configurations are vendor specific and  
6792 out of scope for this standard.

6793 NOTE This version of the standard does not require a power cycle to complete the device configuration.

6795 **14.1.3.1 Read Descriptor**

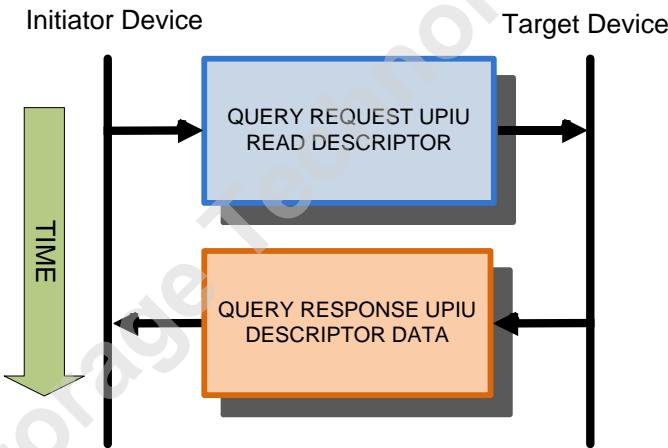
6796 A Query Request operation with READ DESCRIPTOR opcode is sent by the host to the device. The host  
6797 builds a QUERY REQUEST UPIU places a READ DESCRIPTOR opcode within the UPIU, sets the  
6798 appropriate values in the required fields and sends that UPIU to the target device.

6799 Upon reception of the QUERY REQUEST UPIU the device will decode the READ DESCRIPTOR  
6800 opcode field and retrieve the descriptor indicated by the DESCRIPTOR IDN field, the INDEX field and  
6801 the SELECTOR field . When the device is ready to return data to the host the device will construct a  
6802 QUERY RESPONSE UPIU, set the appropriate fields and place the entire retrieved descriptor within a  
6803 single Data Segment area of the QUERY RESPONSE UPIU.

6804 Upon transmission of the QUERY RESPONSE UPIU the device will consider the Query Request  
6805 operation complete.

6806 Upon reception of the QUERY RESPONSE UPIU the host will retrieve the requested descriptor from the  
6807 Data Segment area, and decode the Status and other fields within the UPIU and take the appropriate  
6808 completion action. Upon reception of the QUERY RESPONSE UPIU the host will consider that the  
6809 Query Request operation is complete.

6810



6811  
6812  
6813

**Figure 14.2 — Read Request Descriptor**

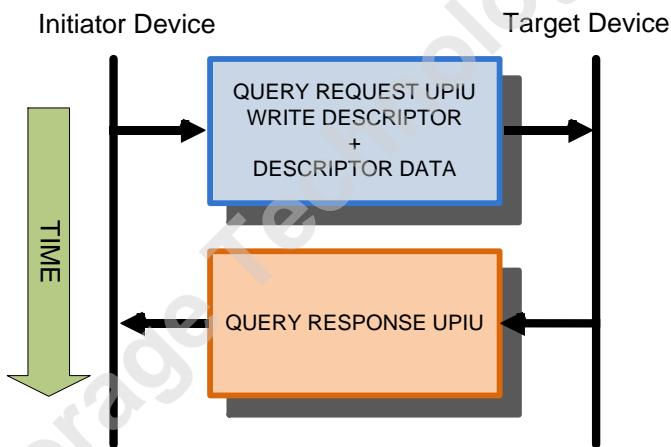
6814 **14.1.3.2 Write Descriptor**

6815 A Query Request operation with WRITE DESCRIPTOR opcode is sent by the host to the device. The  
6816 host builds a QUERY REQUEST UPIU places a WRITE DESCRIPTOR opcode within the UPIU, sets  
6817 the appropriate values in the required fields and sends that UPIU to the target device. A QUERY  
6818 REQUEST UPIU with a WRITE DESCRIPTOR opcode will additionally include a data segment that  
6819 contains the descriptor data the host is sending to the device.

6820 Upon reception of the QUERY REQUEST UPIU the device will decode the WRITE DESCRIPTOR  
6821 opcode field and access the descriptor indicated by the DESCRIPTOR IDN field, the INDEX field and  
6822 the SELECTOR field. The device will extract the descriptor data within the data segment of the UPIU and  
6823 will internally overwrite the addressed descriptor with the extracted data. When the device has completed  
6824 this process it will then notify the host of the success or failure of this operation by returning to the host a  
6825 QUERY RESPONSE UPIU with the RESPONSE field containing the response code.

6826 After the transmission of the QUERY RESPONSE UPIU the device will consider the operation complete.

6827 Upon reception of the QUERY RESPONSE UPIU the host will decode the RESPONSE field and other  
6828 fields within the UPIU and take the appropriate completion action. Upon reception of the QUERY  
6829 RESPONSE UPIU the host will consider that the operation is complete.



6830  
6831 **Figure 14.3 — Write Request Descriptor**  
6832  
6833

6834 **14.1.4 Descriptor Definitions**

6835 **14.1.4.1 Generic Descriptor Format**

6836 The format of all descriptors begins with a header which contains the length of the descriptor and a type  
6837 value that identifies the specific type of descriptor. The length value includes the length of the header  
6838 plus any additional data.

6839 **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

6840

6841 For unit descriptors, the format includes an indication for the unit being addressed.

6842

6843 **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

6844

#### 14.1.4.2 Device Descriptor

This is the main descriptor and should be the first descriptor retrieved as it specifies the device class and sub-class and the protocol (command set) to use to access this device and the maximum number of logical units contained within the device. The Device Descriptor is read only, some of its parameters may be changed writing the corresponding parameter of the Configuration Descriptor.

In a QUERY REQUEST UPIU, the Device Descriptor is addressed setting: DESCRIPTOR IDN = 00h, INDEX = 00h and SELECTOR = 00h.

**Table 14.4 — Device Descriptor**

DEVICE DESCRIPTOR					
Offset	Size	Name	MDV <sup>(1)</sup>	User Conf.	Description
00h	1	bLength	59h	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 <sup>(3)</sup>	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 <sup>(1)</sup>	User Conf.	Description
09h	1	bDescrAccessEn	00h	Yes	<p>Descriptor Access Enable Indicate whether the Device Descriptor can be read after the partial initialization phase of the boot sequence</p> <p>00h: Device Descriptor access disabled 01h: Device Descriptor access enabled Others: Reserved</p>
0Ah	1	bInitPowerMode	01h	Yes	<p>Initial Power Mode bInitPowerMode defines the Power Mode after device initialization or hardware reset</p> <p>00h: UFS-Sleep Mode 01h: Active Mode Others: Reserved</p>
0Bh	1	bHighPriorityLUN	7Fh	Yes	<p>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.</p>
0Ch	1	bSecureRemovalType	00h	Yes	<p>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</p>
0Dh	1	bSecurityLU	01h	No	<p>Support for security LU 00h: not supported 01h: RPMB Others: Reserved</p>

DEVICE DESCRIPTOR					
Offset	Size	Name	MDV <sup>(1)</sup>	User Conf.	Description
0Eh	1	bBackgroundOpsTermLat	Device specific	No	<p>Background Operations Termination Latency  <b>bBackgroundOpsTermLat</b> defines the maximum latency for the termination of ongoing background operations.</p> <p>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 <b>bBackgroundOpsTermLat</b>.</p> <p>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.</p>
0Fh	1	bInitActiveICCLevel	00h	Yes	<p>Initial Active ICC Level  <b>bInitActiveICCLevel</b> defines the <b>bActiveICCLevel</b> value after power on or reset.  Valid range from 00h to 0Fh.</p>
10h	2	wSpecVersion	0310h	No	<p>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: version 3.21 = 0321h</p>
12h	2	wManufactureDate	Device specific	No	<p>Manufacturing Date  BCD version of the device manufacturing date, i.e., August 2010 = 0810h</p>
14h	1	iManufacturerName	Device specific	No	<p>Manufacturer Name  Index to the string which contains the Manufacturer Name.</p>
15h	1	iProductName	Device specific	No	<p>Product Name  Index to the string which contains the Product Name.</p>
16h	1	iSerialNumber	Device specific	No	<p>Serial Number  Index to the string which contains the Serial Number.</p>
17h	1	iOemID	Device specific	No	<p>OEM ID  Index to the string which contains the OEM ID.</p>

DEVICE DESCRIPTOR					
Offset	Size	Name	MDV <sup>(1)</sup>	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	16h	No	Unit Descriptor 0 Base Offset Offset of the Unit Descriptor 0 configurable parameters within the Configuration Descriptor.
1Bh	1	bUDConfigPLength	1Ah	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	<p>Frequency and method of Real-Time Clock update.</p> <p>Bits [15:10] Reserved</p> <p>Bits [9] TIME_BASELINE</p> <ul style="list-style-type: none"> <li>0b: Time elapsed from the previous dSecondsPassed update.</li> <li>1b: Absolute time elapsed from January 1st 2010 00:00.</li> </ul> <p>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'.</p> <p>Bits [8:6] TIME_UNIT</p> <ul style="list-style-type: none"> <li>000b = Undefined</li> <li>001b = Months</li> <li>010b = Weeks</li> <li>011b = Days</li> <li>100b = Hours</li> <li>101b = Minutes</li> <li>110b = Reserved</li> <li>111b = Reserved</li> </ul> <p>Bits [5:0] TIME_PERIOD</p> <p>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.</p>

DEVICE DESCRIPTOR					
Offset	Size	Name	MDV <sup>(1)</sup>	User Conf.	Description
1Fh	1	bUFSFeaturesSupport	Device specific	No	<p>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.</p> <ul style="list-style-type: none"> <li>bit[0]: Field Firmware Update (FFU)</li> <li>bit[1]: Production State Awareness (PSA)</li> <li>bit[2]: Device Life Span</li> <li>bit[3]: Refresh Operation</li> <li>bit[4]: TOO_HIGH_TEMPERATURE</li> <li>bit[5]: TOO_LOW_TEMPERATURE</li> <li>bit[6]: Extended Temperature</li> <li>bit[7]: Reserved for Host Performance Booster(HPB) Extension Standard</li> <li>Others: Reserved</li> </ul> <p>Bit 0 shall be set to one.</p>
20h	1	bFFUTimeout	Device specific	No	<p>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.</p> <p>A value of zero indicates that no timeout is provided.</p>
21h	1	bQueueDepth	Device specific	No	<p>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.</p> <p>If bLUQueueDepth&gt;0 for any LU (except RPMB LU), then bQueueDepth shall be 0.</p>
22h	2	wDeviceVersion	Device specific	No	<p>Device Version  This field provides the device version.</p>
24h	1	bNumSecureWPArea	Device specific	No	<p>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).</p>

DEVICE DESCRIPTOR					
Offset	Size	Name	MDV <sup>(1)</sup>	User Conf.	Description
25h	4	dPSAMaxDataSize	Device specific	No	<p>PSA Maximum Data Size</p> <p>This parameter specifies the maximum amount of data that may be written during the pre-soldering phase of the PSA flow.</p> <p>The value indicates the total amount of data for all logical units with bPSASensitive = 01h.</p> <p>Value expressed in units of 4 Kbyte.</p>
29h	1	bPSAStateTimeout	Device specific	No	<p>PSA State Timeout</p> <p>This parameter specifies the command maximum timeout for a change in bPSAState state.</p> <p>00h means undefined.</p> <p>Otherwise, the formula to calculate the max timeout value is:</p> <p>Production State Timeout = 100us * <math>2^{bPSAStateTimeout}</math></p> <p>For example:</p> <p>01h means 100us x <math>2^1</math> = 200us</p> <p>02h means 100us x <math>2^2</math> = 400us</p> <p>17h means 100us x <math>2^{23}</math> = 838.86s</p>
2Ah	1	iProductRevisionLevel	Device specific	No	<p>Product Revision Level</p> <p>Index to the string which contains the Product Revision Level</p>
2Bh	5	Reserved	-	-	Reserved
30h	16	Reserved	-	-	Reserved for Unified Memory Extension standard
40h	3	Reserved	-	-	Reserved for Host Performance Booster (HPB) Extension Standard
43h	12	Reserved	-	-	Reserved

DEVICE DESCRIPTOR					
Offset	Size	Name	MDV <sup>(1)</sup>	User Conf.	Description
4Fh	4	dExtendedUFSTFeaturesSupport	Device specific	No	<p>Extended UFS Features Support  This field indicates which features are supported by the device. This field value will be exactly same value and same functionality as defined in the bit[0~7] of bUFSFeaturesSupport device descriptor.  Since bUFSFeaturesSupport will be obsoleted, it is recommended to refer this descriptor to find out device feature support. A feature is supported if the related bit is set to one.</p> <p>bit[0]: Field Firmware Update (FFU)  bit[1]: Production State Awareness (PSA)  bit[2]: Device Life Span  bit[3]: Refresh Operation  bit[4]: TOO_HIGH_TEMPERATURE  bit[5]: TOO_LOW_TEMPERATURE  bit[6]: Extended Temperature  bit[7]: Reserved for Host-aware Performance Booster (HPB)  bit[8]: WriteBooster  bit[9]: Performance Throttling  bit[10~31] : Reserved</p>
53h	1	bWriteBoosterBufferPreserveUserSpaceEn	0	Yes	<p>Preserve User Space mode  00h: User space is reduced if WriteBooster Buffer is configured. The WriteBooster Buffer reduces the user space that can be configured at provisioning.  01h: User space shall not be reduced if WriteBooster Buffer is configured.  If the user space is almost consumed the WriteBooster Buffer space may be used as the user space. During the migration of the WriteBooster Buffer space to the user space, there could be performance degradation.</p> <p>Others: Reserved</p>
54h	1	bWriteBoosterBufferType	0	Yes	<p>WriteBooster Buffer Type  00h: LU dedicated buffer type  01h: Single shared buffer type</p>

DEVICE DESCRIPTOR					
Offset	Size	Name	MDV <sup>(1)</sup>	User Conf.	Description
55h	4	dNumSharedWriteBoosterBufferAlloc Units	0	Yes	<p>The WriteBooster Buffer size for the shared WriteBooster Buffer configuration.</p> <p>The dNumSharedWriteBoosterBufferAllocUnits value shall be calculated using the following equation:</p> $\text{dNumSharedWriteBoosterBufferAllocUnits} = \text{CEILING}\left(\frac{\text{WriteBoosterBufferCapacity} \times 1}{\text{bAllocationUnitSize} \times \text{dSegmentSize} \times 512}\right),$ <p>where WriteBoosterBufferCapacity is the desired WriteBooster Buffer size expressed in bytes. For example, to configure 4 GB WriteBooster Buffer if bAllocationUnitSize = 8, and dSegmentSize = 1024, then the value for the dNumSharedWriteBoosterBufferAllocUnits is 400h.</p> <p>If this value is zero, then the shared WriteBooster is not configured for this device.</p>

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.

6854 **14.1.4.2 Device Descriptor (cont'd)**

6855 **a) wManufacturerID**

6856 This parameter contains manufacturer identification information for the device manufacturer. The  
6857 Manufacturer ID is defined by JEDEC in Standard Manufacturer's identification code [JEP106]. The  
6858 wManufacturerID consists of two fields: Manufacturer ID Code and Bank Index.

6859 **Table 14.5 — wManufacturerID definition**

Byte	7	6	5	4	3	2	1	0
0	Bank Index							
1	Manufacturer ID Code							

6860 **b) Bank Index**

6861 This field contains an index value of the bank that contains the manufacturer identification code. The  
6862 Bank Index value shall be equal to the number of the continuation fields that precede the  
6863 manufacturer identification code as specified by [JEP106].

6864 **c) Manufacturer ID Code**

6865 Manufacturer identification code as defined by JEDEC in Standard Manufacturer's identification  
6866 code [JEP106].

6867

6868 **14.1.4.3 Configuration Descriptor**

6869 The device configuration set by the manufacturer can be modified by writing the Configuration  
6870 Descriptor. In particular, the Configuration Descriptor allows to configure parameters included in the  
6871 Device Descriptor, RPMB Unit Descriptor, and Unit Descriptors. The Configuration Descriptor can be  
6872 written if bConfigDescrLock attribute value is equal to 00h. If bConfigDescrLock attribute value is 01h  
6873 the Configuration Descriptor is locked and a write request shall fail.

6874 There are up to four Configuration Descriptors.

6875 The first Configuration Descriptor is addressed by setting DESCRIPTOR IDN = 01h, INDEX = 00h and  
6876 SELECTOR = 00h in a QUERY REQUEST UPIU, and used to change configurable parameters of  
6877 Device Descriptor, RPMB Unit Descriptor, and the first eight Unit Descriptors (LU 0 to LU 7).

6878 The second Configuration Descriptor is addressed by setting DESCRIPTOR IDN = 01h, INDEX = 01h  
6879 and SELECTOR = 00h in a QUERY REQUEST UPIU, and used to change configurable parameters of the  
6880 next eight Unit Descriptors (LU 8 to LU 15).

6881 The third Configuration Descriptor is addressed by setting DESCRIPTOR IDN = 01h, INDEX = 02h and  
6882 SELECTOR = 00h in a QUERY REQUEST UPIU, and used to change configurable parameters of the  
6883 next eight Unit Descriptors (LU 16 to LU 23).

6884 The fourth Configuration Descriptor is addressed by setting DESCRIPTOR IDN = 01h, INDEX = 03h  
6885 and SELECTOR = 00h in a QUERY REQUEST UPIU, and used to change configurable parameters of  
6886 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, Device Descriptor configurable parameters, and RPMB Unit Descriptor configurable parameters. 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, Device Descriptor configurable parameters, and RPMB Unit 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

6904

6905 **14.1.4.3 Configuration Descriptor (cont'd)**

6906 Table 14.9 shows the Configuration Descriptor format with DESCRIPTOR IDN = 01h, INDEX = 03h  
6907 and SELECTOR = 00h: the lower address space is used for Configuration Descriptor header, then there  
6908 are eight address spaces for user configurable parameters included in the Unit Descriptors of LU 24 to LU  
6909 32.

6910 Parameter offsets are defined based on:

- 6911 • Offset of the Unit Descriptor 0 configurable parameters within the Configuration Descriptor (B).  
6912 • Length of Unit Descriptor configurable parameters (L)

6913 Values for B and L are stored in the Device Descriptor parameters bUD0BaseOffset and  
6914 bUDConfigPLength respectively.

6915 **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

6917  
6918

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.

6919 14.1.4.3 Configuration Descriptor (cont'd)

6920

6921 Table 14.10 defines Configuration Descriptor header, Device Descriptor configurable parameters, and  
6922 RPMB Unit Descriptor configurable parameters within the Configuration Descriptor with INDEX = 00h.

6923 See 14.1.4.2, Device Descriptor, for details about configurable parameters.

6924

Biwin Storage Technology Co., Ltd.

6925

#### 14.1.4.3 Configuration Descriptor (cont'd)

6926

6927

**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 <sup>(1,2)</sup>	Description
00h	1	bLength	E6h	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. Others: Reserved.
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	1	Reserved	-	Reserved for Host Performance Booster (HPB) Extension Standard
0Ch	1	bRPMBRegionEnable		RPMB Region Enable Configures which RPMB regions are enabled in RPMB well known logical unit.
0Dh	1	bRPMBRegion1Size		RPMB Region 1 Size Configures the size of RPMB region 1 if RPMB region 1 is enabled.
0Eh	1	bRPMBRegion2Size		RPMB Region 2 Size Configures the size of RPMB region 2 if RPMB region 2 is enabled.
0Fh	1	bRPMBRegion3Size		RPMB Region 3 Size Configures the size of RPMB region 3 if RPMB region 3 is enabled.
10h	1	bWriteBoosterBufferPreserveUserSpaceEn	00h	Enable preserve user space when WriteBooster Buffer is configured.
11h	1	bWriteBoosterBufferType	00h	Configure the WriteBooster Buffer type
12h	4	dNumSharedWriteBoosterBufferAllocUnits	00h	Configure the WriteBooster Buffer size for a shared WriteBooster Buffer configuration.

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.

### 14.1.6.3 Configuration Descriptor (cont'd)

Table 14.11 defines Configuration Descriptor header within the Configuration Descriptor with INDEX = 01h, 02h, or 03h.

**Table 14.11— Configuration Descr. Header with INDEX = 01h/02h/03h**

Configuration Descriptor Header				
Offset	Size	Name	MDV <sup>(1)</sup>	Description
00h	1	bLength	E6h	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. Others: Reserved.
03h	19	Reserved		

NOTE 1 The column "MDV" (Manufacturer Default Value) specifies parameter values after device manufacturing.

Table 14.12 defines the Unit Descriptors user configurable parameters within the Configuration Descriptor. See 14.1.4.5, Unit Descriptor.

**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	3	Reserved	

Unit Descriptor configurable parameters			
Offset	Size	Name	Description
10h	6	Reserved	See Host Performance Booster (HPB) Extension Standard
16h	4	dLUNumWriteBooster BufferAllocUnits	The WriteBooster Buffer size for the Logical Unit. The value is expressed in the unit of Allocation Units. If this value is '0', then the WriteBooster is not supported for this LU. The Logical unit among LU0 ~ LU7 can be configured for WriteBooster Buffer.

#### 14.1.4.4 Geometry Descriptor

The Geometry Descriptor describes the device geometric parameters. In a QUERY REQUEST UPIU, the Geometry Descriptor is addressed setting: DESCRIPTOR IDN = 07h, INDEX = 00h and SELECTOR = 00h.

Table 14.13 — Geometry Descriptor

GEOMETRY DESCRIPTOR				
Offset	Size	Name	Value	Description
00h	1	bLength	57h	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.

GEOMETRY DESCRIPTOR				
Offset	Size	Name	Value	Description
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.
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

GEOMETRY DESCRIPTOR				
Offset	Size	Name	Value	Description
1Ah	1	bMaxContextIDNumber	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 = $2^{bSysDataTagUnitSize} \times bMinAddrBlockSize \times 512$
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: System Data Tag Resource Size = $\text{Tag Unit Size} \times \text{floor} \left( \frac{qTotalRawDeviceCapacity \times 2^{bSysDataTagResSize-10}}{\text{Tag Unit Size}} \right)$ 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	<p>Supported Memory Types Bit map which represents the supported memory types.</p> <ul style="list-style-type: none"> <li>bit 0: Normal memory type</li> <li>bit 1: System code memory type</li> <li>bit 2: Non-Persistent memory type</li> <li>bit 3: Enhanced memory type 1</li> <li>bit 4: Enhanced memory type 2</li> <li>bit 5: Enhanced memory type 3</li> <li>bit 6: Enhanced memory type 4</li> <li>bit 7: Reserved</li> <li>...</li> <li>bit 14: Reserved</li> <li>bit 15: RPMB memory type</li> </ul> <p>A value one means that the corresponding memory type is supported. Bit 0 and bit 15 shall be one for all UFS device.</p>
20h	4	dSystemCodeMaxNAllocU	Device specific	<p>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</p>
24h	2	wSystemCodeCapAdjFac	Device specific	<p>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.</p> $\text{CapacityAdjFactor} = \text{Capacity}_{\text{NormalMem}} / \text{Capacity}_{\text{SystemCode}}$ $\text{wSystemCodeCapAdjFac} = \text{INTEGER}(256 \times \text{CapacityAdjFactor})$
26h	4	dNonPersistMaxNAllocU	Device specific	<p>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</p>

GEOMETRY DESCRIPTOR				
Offset	Size	Name	Value	Description
2Ah	2	wNonPersistCapAdjFac	Device specific	<p>Capacity Adjustment Factor for the Non-Persistent memory type</p> <p>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.</p> $\text{CapacityAdjFactor} = \text{Capacity}_{\text{NormalMem}} / \text{Capacity}_{\text{NonPersist}}$ $\text{wNonPersistCapAdjFac} = \text{INTEGER}(256 \times \text{CapacityAdjFactor})$
2Ch	4	dEnhanced1MaxNAllocU	Device specific	<p>Max Number of Allocation Units for the Enhanced memory type 1</p> <p>Maximum available quantity of Enhanced memory type 1 for the entire device</p> <p>Value expressed in number of Allocation Unit</p>
30h	2	wEnhanced1CapAdjFac	Device specific	<p>Capacity Adjustment Factor for the Enhanced memory type 1</p> <p>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.</p> $\text{CapacityAdjFactor} = \text{Capacity}_{\text{NormalMem}} / \text{Capacity}_{\text{Enhanced1}}$ $\text{wEnhanced1CapAdjFac} = \text{INTEGER}(256 \times \text{CapacityAdjFactor})$
32h	4	dEnhanced2MaxNAllocU	Device specific	<p>Max Number of Allocation Units for the Enhanced memory type 2</p> <p>Maximum available quantity of Enhanced memory type 2 for the entire device</p> <p>Value expressed in number of Allocation Unit</p>
36h	2	wEnhanced2CapAdjFac	Device specific	<p>Capacity Adjustment Factor for the Enhanced memory type 2</p> <p>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.</p> $\text{CapacityAdjFactor} = \text{Capacity}_{\text{NormalMem}} / \text{Capacity}_{\text{Enhanced2}}$ $\text{wEnhanced2CapAdjFac} = \text{INTEGER}(256 \times \text{CapacityAdjFactor})$

GEOMETRY DESCRIPTOR				
Offset	Size	Name	Value	Description
38h	4	dEnhanced3MaxNAllocU	Device specific	<p>Max Number of Allocation Units for the Enhanced memory type 3</p> <p>Maximum available quantity of Enhanced memory type 3 for the entire device</p> <p>Value expressed in number of Allocation Unit</p>
3Ch	2	wEnhanced3CapAdjFac	Device specific	<p>Capacity Adjustment Factor for the Enhanced memory type 3</p> <p>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.</p> $\text{CapacityAdjFactor} = \frac{\text{Capacity}_{\text{NormalMem}}}{\text{Capacity}_{\text{Enhanced3}}}$ $\text{wEnhanced3CapAdjFac} = \text{INTEGER}(256 \times \text{CapacityAdjFactor})$
3Eh	4	dEnhanced4MaxNAllocU	Device specific	<p>Max Number of Allocation Units for the Enhanced memory type 4</p> <p>Maximum available quantity of Enhanced memory type 4 for the entire device</p> <p>Value expressed in number of Allocation Unit</p>
42h	2	wEnhanced4CapAdjFac	Device specific	<p>Capacity Adjustment Factor for the Enhanced memory type 4</p> <p>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.</p> $\text{CapacityAdjFactor} = \frac{\text{Capacity}_{\text{NormalMem}}}{\text{Capacity}_{\text{Enhanced4}}}$ $\text{wEnhanced4CapAdjFac} = \text{INTEGER}(256 \times \text{CapacityAdjFactor})$

GEOMETRY DESCRIPTOR				
Offset	Size	Name	Value	Description
44h	4	dOptimalLogicalBlockSize	Device specific	<p>Optimal Logical Block Size</p> <p>bit [3:0]: Normal memory type</p> <p>bit [7:4]: System code memory type</p> <p>bit [11: 8]: Non-Persistent memory type</p> <p>bit [15:12]: Enhanced memory type 1</p> <p>bit [19:16]: Enhanced memory type 2</p> <p>bit [23:20]: Enhanced memory type 3</p> <p>bit [27:24]: Enhanced memory type 4</p> <p>bit [31:28]: Reserved</p> <p>The optimal logical block size for each memory type can be calculated from the related dOptimalLogicalBlockSize field as indicated in the following:</p> $\text{Optimal Logical Block Size} = 2^{\text{(dOptimalLogicalBlockSize field)}} \times \text{bMinAddrBlockSize} \times 512 \text{ byte}$
48h	5	Reserved	-	Reserved for Host Performance Booster (HPB) Extension Standard
4Dh	2	Reserved	-	Reserved
4Fh	4	dWriteBoosterBufferMaxNAllocUnits	Device specific	Maximum total WriteBooster Buffer size which is supported by the entire device. The summation of the WriteBooster Buffer size for all LUs should be equal to or less than size value indicated by this descriptor.
53h	1	bDeviceMaxWriteBoosterLUs	Device specific	<p>Number of maximum WriteBooster Buffer supported by the device.</p> <p>In this version of the standard, the valid value of this field is 1.</p> <p>Other values are reserved.</p>

GEOMETRY DESCRIPTOR				
Offset	Size	Name	Value	Description
54h	1	bWriteBoosterBufferCapAdjFac	Device specific	<p>Capacity Adjustment Factor for the WriteBooster Buffer memory type.</p> <p>This value provides the LBA space reduction multiplication factor when WriteBooster Buffer is configured in user space reduction mode.</p> <p>Therefore, this parameter applies only if bWriteBoosterBufferPreserveUserSpaceEn is 00h.</p> <p>For “LU dedicated buffer” mode, the total user space is decreased by the following amount:  <math>bWriteBoosterBufferCapAdjFac * dLUNumWriteBoosterBufferAllocUnits * bAllocationUnitSize * dSegmentSize * 512 \text{ byte}^*</math>.</p> <p>For “shared buffer” mode, the total user space is decreased by by the following amount:  <math>bWriteBoosterBufferCapAdjFac * dNumSharedWriteBoosterBufferAllocUnits * bAllocationUnitSize * dSegmentSize * 512 \text{ byte}^*</math>.</p> <p>The value of this parameter is 3 for TLC NAND when SLC mode is used as WriteBooster Buffer. 2 for MLC NAND.</p>
55h	1	bSupportedWriteBoosterBufferUserSpaceReductionTypes	Device specific	<p>The supportability of user space reduction mode and preserve user space mode.</p> <p>00h: WriteBooster Buffer can be configured only in user space reduction type.</p> <p>01h: WriteBooster Buffer can be configured only in preserve user space type.</p> <p>02h: Device can be configured in either user space reduction type or preserve user space type.</p> <p>Others : Reserved</p>
56h	1	bSupportedWriteBoosterBufferTypes	Device specific	<p>The supportability of WriteBooster Buffer type.</p> <p>00h: LU based WriteBooster Buffer configuration</p> <p>01h: Single shared WriteBooster Buffer configuration</p> <p>02h: Supporting both LU based WriteBooster Buffer and Single shared WriteBooster Buffer configuration</p> <p>Others: Reserved</p>
NOTE 1 The Capacity Adjustment Factor value for Normal memory type is one.				

#### 14.1.4.5 Unit Descriptor

This page describes specific characteristics and capabilities of an individual logical unit, for example geometry of the device and maximum addressable item. There are up to thirty two unit descriptors. In a QUERY REQUEST UPIU, an Unit Descriptor is addressed setting: DESCRIPTOR IDN = 02h, INDEX = unit index, and SELECTOR = 00h.

**Table 14.14 — Unit Descriptor**

UNIT DESCRIPTOR					
Offset	Size	Name	MDV <sup>(1)</sup>	User Conf. <sup>(2)</sup>	Description
00h	1	bLength	2Dh	No	Size of this descriptor
01h	1	bDescriptorID N	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 02h: Reserved for Host Performance Booster (HPB) Extension Standard 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	bPSASensitivity	Device specific	No <sup>(3)</sup>	00h: LU is not sensitive to soldering 01h: LU is sensitive to soldering Others: Reserved

UNIT DESCRIPTOR					
Offset	Size	Name	MDV <sup>(1)</sup>	User Conf. <sup>(2)</sup>	Description
08h	1	bMemoryType	00h	Yes	<p>Memory Type bMemoryType defines logical unit memory type.</p> <p>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</p>
09h	1	bDataReliability	00h	Yes	<p>Data Reliability bDataReliability defines the device behavior when a power failure occurs during a write operation to the logical unit</p> <p>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</p>
0Ah	1	bLogicalBlockSize	0Ch	Yes	<p>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: <math>2^{b\text{LogicalBlockSize}}</math> (i.e., bLogicalBlockSize = 0Ch corresponds to 4 Kbyte Logical Block Size). Its minimum value is 0Ch, which corresponds to 4 Kbyte</p>
0Bh	8	qLogicalBlockCount	00h	Yes <sup>(4)</sup>	<p>Logical Block Count Total number of addressable logical blocks in the logical unit</p>
13h	4	dEraseBlockSize	00h <sup>(5)</sup>	No	<p>Erase Block Size Optimal granularity for erase and discard operations. Value in number of Logical Blocks</p>
17h	1	bProvisioningType	00h	Yes	<p>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</p>

UNIT DESCRIPTOR					
Offset	Size	Name	MDV <sup>(1)</sup>	User Conf. <sup>(2)</sup>	Description
18h	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 bMaxContextIDNumber. 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: $1 \leq \text{multiplier} \leq (\text{LARGE\_UNIT\_MAX\_MULTIPLIER\_M1} + 1)$ 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)
23h	6	Reserved	-	-	Reserved for Host Performance Booster (HPB) Extension Standard

UNIT DESCRIPTOR					
Offset	Size	Name	MDV <sup>(1)</sup>	User Conf. <sup>(2)</sup>	Description
29h	4	dLUNumWriteBoosterBufferAllocUnits	00h	Yes	<p>The WriteBooster Buffer size for the Logical Unit.</p> <p>The dLUNumWriteBoosterBufferAllocUnits value shall be calculated using the following equation:</p> $\text{dLUNumWriteBoosterBufferAllocUnits} = \text{CEILING}\left(\frac{\text{WriteBoosterBufferCapacity} \times 1}{\text{bAllocationUnitSize} \times \text{dSegmentSize} \times 512}\right),$ <p>where WriteBoosterBufferCapacity is the desired WriteBooster Buffer size expressed in bytes. To configure 4 GB WriteBooster Buffer, if bAllocationUnitSize = 8, and dSegmentSize = 1024, then the value for the dLUNumWriteBoosterBufferAllocUnits is 0400h. If this value is '0', then the WriteBooster is not supported for this LU.</p> <p>The Logical unit among LU0 ~ LU7 can be configured for WriteBooster Buffer. Otherwise, whole WriteBooster Buffer configuration in this device is invalid.</p>

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.

6948 **14.1.4.6 RPMB Unit Descriptor**

6949 In a QUERY REQUEST UPIU, the RPMB Unit Descriptor is addressed setting: DESCRIPTOR IDN =  
6950 02h, INDEX = C4h, and SELECTOR = 00h.

6951

**Table 14.15 — RPMB Unit Descriptor**

RPMB UNIT DESCRIPTOR					
Offset	Size	Name	MDV <sup>(1)</sup>	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	bRPMBRegionEnable	00h	Yes	RPMB Region Enable Bit-0: Don't care. RPMB region 0 is always enabled independent of this bit value. Bit-1: If set to 1, RPMB region 1 is enabled. Bit-2: If set to 1, RPMB region 2 is enabled. Bit-3: If set to 1, RPMB region 3 is enabled. Bit-4 to Bit-7: 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)

RPMB UNIT DESCRIPTOR					
Offset	Size	Name	MDV <sup>(1)</sup>	User Conf.	Description
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	1	bRPMBRegion0Size	Device Specific	Yes	RPMB Region 0 Size RPMB region 0 size is defined in 128KB unit (00h: 0KB, 01h: 128KB, ..., 80h: 16384KB). The MDV value multiplied by 512 shall match with the MDV value of LogicalBlockCount in the RPMB Unit Descriptor.
14h	1	bRPMBRegion1Size	00h	Yes	RPMB Region 1 Size RPMB region 1 size is defined in 128KB unit (00h: 0KB, 01h: 128KB, ..., 80h: 16384KB).
15h	1	bRPMBRegion2Size	00h	Yes	RPMB Region 2 Size RPMB region 2 size is defined in 128KB unit (00h: 0KB, 01h: 128KB, ..., 80h: 16384KB).
16h	1	bRPMBRegion3Size	00h	Yes	RPMB Region 3 Size RPMB region 3 size is defined in 128KB unit (00h: 0KB, 01h: 128KB, ..., 80h: 16384KB).
17h	1	bProvisioningType	00h	No	Provisioning Type 00h:Thin Provisioning is disabled
18h	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	3	Reserved	0000h		

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.

#### 6953 14.1.4.7 Power Parameters Descriptor

6954 This descriptor contains information about the power capabilities and power states of the device. In a  
6955 QUERY REQUEST UPIU, the Power Parameters Descriptor is addressed setting: DESCRIPTOR IDN =  
6956 08h, INDEX = 00h, and SELECTOR = 00h.

6957

**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	2	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

6958

#### 6959 14.1.4.8 Interconnect Descriptor

6960 The Interconnect Descriptor contains the MIPI M-PHY® specification version number and the MIPI  
6961 UniPro® specification version number. In a QUERY REQUEST UPIU, the Interconnect Descriptor is  
6962 addressed setting: DESCRIPTOR IDN = 04h, INDEX = 00h, and SELECTOR = 00h.

6963 **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	0180h	MIPI UniPro® version number in BCD format Example: version 3.21 = 0321h
04h	2	bcdMphyVersion	0410h	MIPI M-PHY® version number in BCD format Example: version 3.21=0321h

6964

#### 6965 14.1.4.9 Manufacturer Name String Descriptor

6966 This descriptor contains the UNICODE, left justified, manufacturer name string.

6967 The content of the descriptor shall be identical to the content of the “VENDOR IDENTIFICATION” field  
6968 in Inquiry Response Data. The length of the descriptor shall be 12h (18 decimal), containing exactly 8  
6969 UNICODE characters, to match “VENDOR IDENTIFICATION” field in Inquiry Response Data.

6970 In a QUERY REQUEST UPIU, the Manufacturer Name String Descriptor is addressed setting:  
6971 DESCRIPTOR IDN = 05h, INDEX = iManufacturerName (Device Descriptor parameter), and  
6972 SELECTOR = 00h.

6973 **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

6974

#### 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

#### 6993 14.1.4.12 Serial Number String Descriptor

6994 This descriptor contains the UNICODE serial number string that may consist of up to 126 UNICODE  
6995 characters. Number of UNICODE characters is calculated by  $(LENGTH - 2) \div 2$ .

6996 In a QUERY REQUEST UPIU, the Serial Number String Descriptor is addressed setting:  
6997 DESCRIPTOR IDN = 05h, INDEX = iSerialNumber (Device Descriptor parameter), and SELECTOR =  
6998 00h.

6999 **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

#### 7000 14.1.4.13 Product Revision Level String Descriptor

7001 This descriptor contains the UNICODE, left justified, product revision level string.

7002 The content of the descriptor shall be identical to the content of the “PRODUCT REVISION LEVEL”  
7003 field in Inquiry Response Data. The length of the descriptor shall be Ah, containing exactly 4 UNICODE  
7004 characters, to match “PRODUCT REVISION LEVEL” field in Inquiry Response Data.

7005 In a QUERY REQUEST UPIU, the Product Revision Level String Descriptor is addressed setting:  
7006 DESCRIPTOR IDN = 05h, INDEX = iProductRevisionLevel (Device Descriptor parameter), and  
7007 SELECTOR = 00h.

7008 **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

7009

7010 **14.1.4.14 Device Health Descriptor**

7011 Device Health Descriptor provides information related to the health of the device.

7012 In a QUERY REQUEST UPIU, the Device Health Descriptor is addressed by setting: DESCRIPTOR  
7013 IDN = 09h, INDEX = 00h and SELECTOR = 00h.

7014 **Table 14.23 — Device Health Descriptor**

<b>Offset</b>	<b>Size</b>	<b>Name</b>	<b>Value</b>	<b>Description</b>
00h	1	bLength	2Dh	Size of this descriptor
01h	1	bDescriptorIDN	09h	Device Health Descriptor Type Identifier
02h	1	bPreEOLInfo	Device specific	<p>Pre End of Life Information This field provides indication about device life time reflected by average reserved blocks.</p> <p>00h: Not defined 01h: Normal 02h: Warning. Consumed 80% of reserved blocks. 03h: Critical. Consumed 90% of reserved blocks. Others: Reserved</p>
03h	1	bDeviceLifeTimeEstA	Device specific	<p>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.</p> <p>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</p>

Offset	Size	Name	Value	Description
04h	1	bDeviceLifeTimeEstB	Device specific	<p>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.</p> <p>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</p>
05h	32	VendorPropInfo	Device specific	Reserved for Vendor Proprietary Health Report
25h	4	dRefreshTotalCount	Device specific	<p>Total Refresh Count</p> <p>Indicate how many times the device complete refresh for the entire device. Incremented by 1 when dRefreshProgress reach 100000 (100.000%).</p>
29	4	dRefreshProgress	Device specific	<p>Refresh Progress</p> <p>Indicate the refresh progress in %.</p> <p>dRefreshProgress will indicate 0.000% ~ 100.000% in dec.</p> <p>dRefreshProgress = 100000 (dec) when it complete 100.000%.</p> <p>dRefreshProgress = 1000 (dec) when it complete 1.000%.</p> <p>When this value reach 100000 (100.000%)</p> <ol style="list-style-type: none"> <li>1. Device stops refreshing even if it did not complete the number of units specified by bRefreshUnit</li> <li>2. dRefreshProgress shall be reset to 0</li> <li>3. dRefreshTotalCount shall be incremented by 1</li> </ol> <p>When bRefreshMethod = 02h (Manual-Selective), even though some of physical blocks are not refreshed by device choice, dRefreshProgress should be incremented just as much as bRefreshUnit.</p>

7016 **14.2 Flags**

7017 A flag is a single Boolean value that represents a TRUE or FALSE, '0' or '1', ON or OFF type of value.  
7018 A flag can be cleared or reset, set, toggled or read. Flags are useful to enable or disable certain functions  
7019 or modes or states within the device.

7020 Read access property (read or write only) and write access property (read only, write only, persistent, etc.)  
7021 are defined for each flag. Table 14.24 describes the supported access properties for flags.

7022 **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.

7023

7024 **14.2 Flags (cont'd)**

7025

**Table 14.25 — Flags**

FLAGS					
IDN	Name	Type	Type <sup>1</sup>	Default	Description
# Ind. <sup>2</sup>	# Sel. <sup>3</sup>				
00h	Reserved				
01h	fDeviceInit	Read / Set only	D	0	<p>Device Initialization Host set fDeviceInit flag to initiate device initialization after boot process is completed. Device resets flag when device initialization is completed.</p> <p>0b: Device initialization completed or not started yet.</p> <p>1b: Device initialization in progress.</p>
02h	fPermanentWPEn	Read / Write once	D	0	<p>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.</p> <p>00h: Permanent write protection disabled 01h: Permanent write protection enabled</p>
03h	fPowerOnWPEn	Read / Power on reset	D	0	<p>Power On Write Protection Enable fPowerOnWPEn enables the write protection on all logical units configured as power on write protected.</p> <p>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).</p> <p>The device shall set fPowerOnWPEn to zero in the event of power cycle or hardware reset.</p> <p>0b: Power on write protection disabled. 1b: Power on write protection enabled.</p>
04h	fBackgroundOpsEn	Read / Volatile	D	1	<p>Background Operations Enable 0b: Device is not permitted to run background operations. 1b: Device is permitted to run background operations.</p>

FLAGS					
IDN	Name	Type	Type <sup>1</sup>	Default	Description
			# Ind. <sup>2</sup>		
			# Sel. <sup>3</sup>		
05h	fDeviceLifeSpanModeEn	Read / Volatile	D	0	<p>Device Life Span Mode</p> <p>0b: Device Life Span Mode is disabled.</p> <p>1b: Device Life Span Mode is enabled.</p> <p>For more details see 13.4.13, Device Life Span Mode.</p>
06h	fPurgeEnable	Write only / Volatile	D	0	<p>Purge Enable</p> <p>0b: Purge operation is disabled.</p> <p>1b: Purge operation is enabled.</p> <p>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.</p>
07h	fRefreshEnable	Write only / Volatile	D	0	<p>Refresh Enable</p> <p>0b: Refresh operation is disabled.</p> <p>1b: Refresh operation is enabled.</p> <p>This flag shall only be set when the command queue of all logical units are empty and the bRefreshStatus is 00h (Idle). fRefreshEnable is automatically cleared by the UFS device when the operation completes or an error condition occurs. fRefreshEnable can be cleared by the host to interrupt an ongoing refresh operation.</p>
08h	fPhyResourceRemoval	Read / Persistent	D	0	<p>Physical Resource Removal</p> <p>The host sets this flag to one to indicate that the dynamic capacity operation shall commence upon device EndPointReset or hardware reset.</p> <p>The device shall reset this flag to zero after completion of dynamic capacity operation. The host cannot reset this flag.</p>

FLAGS					
IDN	Name	Type	Type <sup>1</sup>	Default	Description
			# Ind. <sup>2</sup>		
			# Sel. <sup>3</sup>		
09h	fBusyRTC	Read Only	D	0	<p>Busy Real Time Clock</p> <p>0b : Device is not executing internal operation related to RTC</p> <p>1b: Device is executing internal operation related to RTC</p> <p>When this flag is set to one, it is recommended for the host to not send commands to the device</p>
0Ah	Reserved	-	-	-	Reserved for Unified Memory Extension standard
0Bh	fPermanentlyDisableFw Update	Read / Write once	D	0	<p>Permanently Disable Firmware Update</p> <p>0b: The UFS device firmware may be modified</p> <p>1b: The UFS device shall permanently disallow future firmware updates to the UFS device</p>
0Ch	Reserved	-	-	-	Reserved for Unified Memory Extension standard
0Dh	Reserved	-	-	-	Reserved for Unified Memory Extension standard
0Eh	fWriteBoosterEn	Read / Volatile	A/LU/0 or D <sup>4</sup>	0	<p>WriteBooster Enable</p> <p>0b: WriteBooster is not enabled.</p> <p>1b: WriteBooster is enabled.</p>
0Fh	fWriteBoosterBufferFlushEn	Read / Volatile	A/LU/0 or D <sup>4</sup>	0	<p>Flush the data in WriteBooster Buffer to the user area of storage.</p> <p>0b: Flush operation is not performed.</p> <p>1b: Flush operation is performed.</p>
10h	fWriteBoosterBufferFlushDuringHibernate	Read / Volatile	A/LU/0 or D <sup>4</sup>	0	<p>Flush WriteBooster Buffer during hibernate state.</p> <p>0b: Device is not allowed to flush the WriteBooster Buffer during link hibernate state.</p> <p>1b: Device is allowed to flush the WriteBooster Buffer during link hibernate state.</p>
11h	Reserved	-	-	-	Reserved for Host Performance Booster (HPB) Extension Standard

FLAGS									
IDN	Name	Type	Type <sup>1</sup>	Default	Description				
			# Ind. <sup>2</sup>						
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 by 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 by 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.									
NOTE 4 If the bWriteBoosterBufferType is configured as 01h (shared type), the WriteBooster Buffer is configured as a single shared buffer for the whole device. In this case, the value of LU is does not matter.									

7026

7027 **14.3 Attributes**

7028 An Attribute is a parameter that represents a specific range of numeric values that can be written or read.  
7029 For example, the maximum Data In data packet size would be an attribute. Attribute size can be from 1-  
7030 bit to 32-bit. Attributes of the same type can be organized in arrays, each element of them identified by  
7031 an index. For example, in case of parameter that is logical unit specific, the LUN would be used as index.

7032 Read access property (read or write only) and write access property (read only, write once, persistent,  
7033 etc.) are defined for each attribute. Table 14.26 describes the supported access properties for attributes.

7034  
7035

**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 be 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.

7036

7037 **14.3 Attributes (cont'd)**

7038

**Table 14.27 — Attributes**

ATTRIBUTES							
IDN	Name	Access Property	Size	Type <sup>1</sup>	MDV <sup>4</sup>	Description	Notes
				# Ind. <sup>2</sup>			
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 <sup>1</sup>	MDV <sup>4</sup>	Description	Notes
				# Ind. <sup>2</sup>			
05h	bBackgroundOp Status	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 <sup>1</sup>	MDV <sup>4</sup>	Description	Notes			
				# Ind. <sup>2</sup>						
08h	bMaxDataOutSize	Read / Persistent	1 byte	D		<p>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.</p> <p>Value expressed in number of 512-byte units.</p> <p>bMaxDataOutSize shall not exceed the bMaxOutBufferSize parameter.</p> <p>bMaxDataOutSize = bMaxOutBufferSize when the UFS device is shipped.</p> <p>This parameter can be written by the host only when all LU task queues are empty.</p>	8			
09h	dDynCapNeeded	Read only	4 bytes	<table border="1"> <tr> <td>A</td> </tr> <tr> <td>Number of LU specified by bMaxNumberLU (LUN)</td> </tr> <tr> <td>0</td> </tr> </table>	A	Number of LU specified by bMaxNumberLU (LUN)	0	0000 0000h	<p>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.</p>	9
A										
Number of LU specified by bMaxNumberLU (LUN)										
0										
0Ah	bRefClkFreq	Read / Persistent	1 byte	D	01h	<p>Reference Clock Frequency value</p> <p>0h: 19.2 MHz</p> <p>1h: 26 MHz</p> <p>2h: 38.4 MHz</p> <p>3h: Obsolete</p> <p>Others: Reserved</p>	10			
0Bh	bConfigDescrLock	Read / Write once	1 byte	D	00h	<p>Configuration Descriptor Lock</p> <p>0h: Configuration Descriptor not locked</p> <p>1h: Configuration Descriptor locked</p> <p>Others: Reserved</p>				

ATTRIBUTES							
IDN	Name	Access Property	Size	Type <sup>1</sup>	MDV <sup>4</sup>	Description	Notes
				# Ind. <sup>2</sup>			
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 occurred (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: TOO_HIGH_TEMP_EN Bit 4: TOO_LOW_TEMP_EN Bit 5: WRITEBOOSTER_EVENT_EN Bit 6: PERFORMANCE_THROTTLING_EN Bit 7 -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: TOO_HIGH_TEMP Bit 4: TOO_LOW_TEMP Bit 5: WRITEBOOSTER_FLUSH_NEEDED Bit 6: PERFORMANCE_THROTTLING Bit 7 -15: Reserved	
0Fh	dSecondsPassed	Write only / Volatile	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 <sup>1</sup>	MDV <sup>4</sup>	Description	Notes
				# Ind. <sup>2</sup>			
# Sel. <sup>3</sup>				A			
10h	wContextConf	Read / Volatile	2 bytes	Number of LU specified by bMaxNumberLU (LUN) 15 (ID)	0000h	<p>INDEX specifies the LU number. SELECTOR specifies the Context ID within the LU. Valid values are 01h – Fh.</p> <p><b>Bit[15:8]: Reserved</b></p> <p><b>Bit[7:6]: Reliability mode</b></p> <ul style="list-style-type: none"> <li>00b: MODE0 (normal)</li> <li>01b: MODE1 (non-Large Unit, reliable mode or Large Unit unit-by-unit mode)</li> <li>10b: MODE2 (Large Unit, one-unit-tail mode)</li> <li>11b: Reserved</li> </ul> <p><b>Bit[5:3]: Large Unit multiplier</b></p> <p>If Large Unit context is set, this field defines the Large Unit size, else it is ignored</p> <p><b>Bit[2]: Large Unit context</b></p> <ul style="list-style-type: none"> <li>0b: Context is not following Large Unit rules</li> <li>1b: Context follows Large Unit rules</li> </ul> <p><b>Bit [1:0]: Activation and direction mode</b></p> <ul style="list-style-type: none"> <li>00b: Context is closed and it is no longer active</li> <li>01b: Context is configured and activated as a write-only context.</li> <li>10b: Context is configured and activated as a read-only context</li> <li>11b: Context is configured and activated as a read/write context</li> </ul>	
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 <sup>1</sup>	MDV <sup>4</sup>	Description	Notes
				# Ind. <sup>2</sup>			
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 OFFh: 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	4 bytes	D	00 ... 00h	The amount of data that the host plans to load to all logical units with bPSASensitive set to 1. Value expressed in units of 4 Kbyte.	
17h	bRefClkGatingWaitTime	Read only	1 byte	D	Device specific	Minimum time for which the reference clock is required by device during transition to LS-MODE or HIBERN8 state. The larger time requirement among the transition to LS-MODE and HIBERN8 states should be set for this attribute. 00h: Undefined 01h: 1 micro second ... FFh: 255 micro seconds	12, 13

ATTRIBUTES							
IDN	Name	Access Property	Size	Type <sup>1</sup>	MDV <sup>4</sup>	Description	Notes
				# Ind. <sup>2</sup>			
18h	bDeviceCaseRoughTemperaure	Read only	1 byte	D	00h	Device's rough package case surface temperature. This value shall be valid when (TOO_HIGH_TEMPERATURE is supported and TOO_HIGH_TEMP_EN is enabled) or ( TOO_LOW_TEMPERATURE is supported and TOO_LOW_TEMP_EN is enabled ). 0 : Unknown Temperature 1~250 : ( this value – 80 ) degrees in Celsius. ( i.e., -79 °C ~ 170 °C ) Others: Reserved	
19h	bDeviceTooHighTempBoundary	Read only	1 byte	D	Device specific	High temperature boundary from which TOO_HIGH_TEMP in wExceptionEventStatus is turned on. 0 : Unknown 100~250: ( this value – 80 ) degrees in celcius. ( i.e., 20 °C ~ 170 °C ) Others: Reserved	
1Ah	bDeviceTooLowTempBoundary	Read only	1 byte	D	Device specific	Low temperature boundary from which TOO_LOW_TEMP in wExceptionEventStatus is turned on. 0 : Unknown 1~80 : ( this value – 80 ) degrees in celcius. ( i.e., -79 °C ~ 0 °C ) Others: Reserved	
1Bh	bThrottlingStatus	Read only	1 byte	D	00h	Each set bit represents an existing situation resulting in performance throttling. Bit 0: Temperature Others: Reserved	

ATTRIBUTES							
IDN	Name	Access Property	Size	Type <sup>1</sup>	MDV <sup>4</sup>	Description	Notes
				# Ind. <sup>2</sup>			
1Ch	bWriteBoosterBufferFlushStatus	Read only	1 byte	A/LU/0 or D	0	<p>Flush operation status of WriteBooster Buffer.</p> <p>00h: idle. Device is not flushing the WriteBooster Buffer: either the WriteBooster Buffer is empty or a flush has not been initiated</p> <p>01h: Flush operation in progress. The WriteBooster Buffer is not yet empty and a flush has been initiated.</p> <p>02h: Flush operation stopped prematurely. The WriteBooster Buffer is not empty and the host stopped the in-progress flush.</p> <p>03h: Flush operation completed successfully.</p> <p>04h: Flush operation general failure Others : Reserved</p> <p>When the bWriteBoosterBufferFlushStatus is equal to the one of values 02h, 03h or 04h, value of the bWriteBoosterBufferFlushStatus is automatically cleared as 00h right after the bWriteBoosterBufferFlushStatus is read. A write to the WriteBooster Buffer when the status is 03h will cause automatic transition to either 00h or 01h.</p>	Note 15

ATTRIBUTES							
IDN	Name	Access Property	Size	Type <sup>1</sup>	MDV <sup>4</sup>	Description	Notes
				# Ind. <sup>2</sup>			
# Sel. <sup>3</sup>							
1Dh	bAvailableWriteBoosterBufferSize	Read only	1 byte	A/LU/0 or D	0	<p>Available WriteBooster Buffer Size  This available buffer size is decreased by WriteBooster operation and increased by flush operation.</p> <p>Value expressed in unit of 10% granularity</p> <p>00h: 0% buffer remains,  01h: 10% buffer remains.  02h~09h: 20%~90% buffer remains  0Ah: 100% buffer remains  Others : Reserved</p> <p>The % reported by the attributes is remaining portion of the current WriteBooster Buffer size indicated by the dCurrentWriteBoosterBufferSize attribute.</p>	Note 15

ATTRIBUTES							
IDN	Name	Access Property	Size	Type <sup>1</sup>	MDV <sup>4</sup>	Description	Notes
				# Ind. <sup>2</sup>			
1E	bWriteBoosterBufferLifeTimeEst	Read only	1 byte	A/LU/0 or D	Device specific	<p>This field provides an indication of the WriteBooster Buffer lifetime based on the amount of performed program/erase cycles. In cases of preserve user space configuration for WriteBooster Buffer, this lifetime will be reduced by writing on normal user level space, since WriteBooster Buffer is shared with the user level space.</p> <p>The detailed calculation method is vendor specific.</p> <p>00h: Information not available (WriteBooster Buffer is disabled)</p> <p>01h: 0% - 10% WriteBooster Buffer life time used</p> <p>02h: 10% - 20% WriteBooster Buffer life time used</p> <p>03h: 20% - 30% WriteBooster Buffer life time used</p> <p>04h: 30% - 40% WriteBooster Buffer life time used</p> <p>05h: 40% - 50% WriteBooster Buffer life time used</p> <p>06h: 50% - 60% WriteBooster Buffer life time used</p> <p>07h: 60% - 70% WriteBooster Buffer life time used</p> <p>08h: 70% - 80% WriteBooster Buffer life time used</p> <p>09h: 80% - 90% WriteBooster Buffer life time used</p> <p>0Ah: 90% - 100% WriteBooster Buffer life time used</p> <p>0Bh: Exceeded its maximum estimated WriteBooster Buffer life time (write commands are processed as if WriteBooster feature was disabled)</p> <p>Others: Reserved</p>	Note 15

ATTRIBUTES							
IDN	Name	Access Property	Size	Type <sup>1</sup>	MDV <sup>4</sup>	Description	Notes
				# Ind. <sup>2</sup>			
1Fh	dCurrentWriteBoosterBufferSize	Read only	4 byte	A/LU/0 or D	0	The current WriteBooster Buffer size. In the case of preserve user space mode, depending on available user space remained, the storage block for the WriteBooster Buffer may be used for the user space. Therefore, the WriteBooster Buffer size can be less than initially configured WriteBooster Buffer size. Host can check the current WriteBooster Buffer size by checking this attribute. Value expressed in unit of Allocation Units. If this value is 0, then the current WriteBooster Buffer size is 0. In the case of user space reduction mode, this value shall be same to the value of dLUNumWriteBoosterBufferAllocUnits or dNumSharedWriteBoosterBufferAllocUnits depending on buffer configuration mode.	Note 15
...	Reserved	-	-	-	-	-	
2Bh	Reserved	-	-	-	-	-	
2Ch	bRefreshStatus	Read only	1 byte	D	00h	Refresh Operation Status 00h: Idle (refresh operation disabled) 01h: Refresh operation in progress 02h: Refresh operation stopped prematurely 03h: Refresh operation completed successfully 04h: Refresh operation failed due to logical unit queue not empty 05h: Refresh operation general failure. Others: Reserved When the bRefreshStatus is equal to the values 02h, 03h, 04h or 05h, the bRefreshStatus is automatically cleared to 00h (Idle) the first time that it is read.	

ATTRIBUTES							
IDN	Name	Access Property	Size	Type <sup>1</sup>	MDV <sup>4</sup>	Description	Notes
				# Ind. <sup>2</sup>			
2Dh	bRefreshFreq	Read / Persistent	1 byte	D	Device Specific	Refresh Frequency Host should make sure that dRefreshTotalCount will be incremented on this frequency. 00h: Not defined 01h: 1 month 02h: 2 month ... FFh: 255 month	
2Eh	bRefreshUnit	Read / Persistent	1 byte	D	Device Specific	Refresh Operation Unit This attribute may be set to adjust the minimum physical block numbers to be refreshed upon a single request (i.e., fRefreshEnable set to 1) 00h: Minimum refresh capability of Device 01h: 100.000% (i.e., entire device) Others: Reserved	

ATTRIBUTES							
IDN	Name	Access Property	Size	Type <sup>1</sup>	MDV <sup>4</sup>	Description	Notes
				# Ind. <sup>2</sup>			
2Fh	bRefreshMethod	Read / Persistent	1 byte	D	00h	<p>Refresh Method</p> <p>This parameter specifies the refresh operation method.</p> <p>00h : Not defined</p> <p>01h: Manual-Force</p> <p>The device is obliged to refresh the amount of physical blocks as requested by the host, regardless whether these blocks need refresh or not. The refresh command refreshes the amount of physical blocks given in bRefreshUnit. Refresh starts at the next physical block from where it stopped (or the first block if refresh was never triggered before).</p> <p>02h: Manual-Selective</p> <p>The refresh command refreshes the amount of physical blocks given in bRefreshUnit. Refresh starts at the next physical block from where it stopped (or the first block if refresh was never triggered before). The device only refreshes the blocks that it considers to be in need of refresh. Regardless of the actually refreshed blocks, dRefreshProgress is increased by bRefreshUnit once the refresh command is completed.</p> <p>Others: Reserved</p>	14

ATTRIBUTES													
IDN	Name	Access Property	Size	Type <sup>1</sup>	MDV <sup>4</sup>	Description	Notes						
				# Ind. <sup>2</sup>									
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.													
NOTE 12 UFS Host may start a timer when DME_POWERMODE.ind is received for HS-MODE to LS-MODE transition or DME_HIBERNATE_ENTER.ind is received for HS-MODE to HIBERN8 transition. In addition to bRefClkGatingWaitTime, Device PA_MinRxTrailingClocks and Host PA_MinRxTrailingClocks should be considered to determine when the reference clock may be stopped.													
NOTE 13 If this attribute is set to value '00h', it means the minimum wait time for reference clock removal is not specified by the device.													
NOTE 14 If the host attempts to write bRefreshMethod when dRefreshProgress is not zero, the operation shall fail, and Response field in the QUERY RESPONSE UPIU shall be set to FFh ("General failure").													
NOTE 15 If the bWriteBoosterBufferType is configured as 01h (shared type), the WriteBooster Buffer is configured as a single shared buffer for the whole device. In this case, the value of LU does not matter.													

7039

7040

---

7041 **15 UFS Mechanical Standard**

---

7042 UFS discrete and multichip ballouts are defined in [JESD21C].

Biwin Storage Technology Co., Ltd.

---

7043 **Annex A (informative) Dynamic Capacity Host Implementation Example**

---

7044 **A.1 Overview**

7045 This standard defines the Dynamic Capacity feature to enable a UFS device to regain at least partial  
7046 functionality at the end-of-life where the defect level of the storage medium has accumulated to the point  
7047 where the device can no longer maintain normal functionality.

7048 Dynamic Capacity operation allows the device, at the direction of the host system, to remove physical  
7049 memory resources from the resource pool dedicated for data storage and re-task the resources for device  
7050 internal utility, thus restoring device functionality.

7051 The net result of the Dynamic Capacity operation is a reduction of the usable storage space in the physical  
7052 medium while the logical address space remains the same as before – i.e., the physical storage is less than  
7053 the logical address space. It is the responsibility of the host system to keep track of the reduction in  
7054 physical storage to maintain normal operation in its file system.

7055 This application note outlines a method for the host file system to account for the reduction in physical  
7056 storage with minimal impact.

7057 **A.2 Method Outline**

- 7058 1. The host system receives notification from the device in the Device Information parameter in  
7059 Response UPIU that the device requires physical memory to be freed up from the storage space to  
7060 continue operation.
- 7061 2. The host reads the bDynCapNeeded[LUN] attributes and the bOptimalWriteBlockSize parameter in  
7062 the Device Descriptor to determine how much physical memory resources needs to be freed up in  
7063 each logical unit.
- 7064 3. The host identifies the logical block address range(s) in the file system where the data can be  
7065 discarded/erased to free up the physical memory resources. The host then uses the UNMAP command  
7066 to unmap (deallocate) the LBA range(s), and initiates the Dynamic Capacity operation by setting the  
7067 fPhyResourceRemoval flag and resetting the UFS device.
- 7068 4. The host can mark the particular LBA range(s) as unusable in its file system by the means of dummy  
7069 file(s) to ensure these LBA's will not be used in future write operations. The unusable LBA's marked  
7070 by dummy file(s) match the reduction of physical storage, therefore from the host system perspective,  
7071 the file system is intact.
- 7072 5. The host can further backup the unusable LBA information by storing the information in the system  
7073 area in case the file system of the main data storage logical unit is corrupted.

---

## Annex B (informative) Differences Between Specification Revisions

---

### B.1 Changes between JESD220E and its predecessor JESD220D (January 2018)

#### New features or new definitions

The following items were added:

- UFS-DeepSleep Power Mode, see 7.4 “UFS Device Power Modes and LU Power Condition”
- Performance Throttling, see 13.4.16 “Performance Throttling Event Notification”
- WriteBooster, see 13.4.17 “Write”

#### B.1.2 Changes in section 2 “Normative Reference”

Added the following documents:

- Application Note for UniPro® v1.8
- UFS Host Performance Booster (HPB) Extension Specification

#### B.1.3 Changes in features already defined in UFS 2.1

Changes related to features already defined in the previous version of the standard are summarized in the following:

- Added the following Exception Events: PERFORMANCE\_THROTTLING and WRITEBOOSTER\_FLUSH\_NEEDED. See 13.4.11 “Exception Events Mechanism”.

Several clarifications were added and editorial changes were implemented in addition to what summarized in this annex.

## B.2 Changes between JESD220D and its predecessor JESD220C (March 2016)

### B.2.1 New features or new definitions

The following items were added:

- VCC = 2.5V, see 4.2 “Interface Features” and 6 “UFS ELECTRICAL: CLOCK, RESET, SIGNALS AND SUPPLIES”
- M-PHY<sup>(R)</sup> HS-GEAR4, see 6.4 “Reference Clock” and 8.7 “UFS PHY Attributes”
- bRefClkGatingWaitTime attribute, see 6.4 “Reference Clock”
- Extended temperature ranges, see 6.6 “Absolute Maximum DC Ratings and Operating Conditions”
- M-PHY<sup>(R)</sup> Adapt, see 8.6
- Error History Mode (MODE = 1Ch) in READ BUFFER Command, see 11.3.27.3
- RPMB Regions, see 12.4 RPMB
- Refresh Operation, see 13.4.14
- Temperature Event Notification, see 13.4.15

### B.2.2 Changes in section 2 “Normative Reference”

- M-PHY<sup>(R)</sup> specification: from version 3.00 to version 4.1.
- Unified Protocol (UniPro) specification: from version 1.6 to version 1.8.

Added the following specifications:

- JEDEC Standard Manufacturer’s identification code, JEP106
- JEDEC Multichip Packages (MCP) and Discrete eMMC, e2MMC, and UFS, JESD21C

### B.2.3 Changes in features already defined in UFS 2.1

Changes related to features already defined in the previous version of the standard are summarized in the following:

- Support for HS-GEAR3 has been changed to mandatory, see 4.1 “General Features”
- Removed VCC = 1.8V, see 6 “UFS ELECTRICAL: CLOCK, RESET, SIGNALS AND SUPPLIES”
- Support for READ BUFFER command has been changed to mandatory, see Table 11.1
- Removed 52 MHz reference clock frequency, see 6.4
- Support for some M-PHY features has been changed to optional (PWM-G2 to PWM-G4, small amplitude, LCC functionality), see 8
- Added Unit Attention Condition details, see 10.8.9

Several clarifications were added and editorial changes were implemented in addition to what summarized in this annex.

### B.3 Changes between JESD220C and its predecessor JESD220B (September 2013)

#### B.3.1 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.3.2 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.3.3 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.3.3 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.3.3 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.4 Changes between JESD220B and its predecessor JESD220A (June 2012)

### B.4.1 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.4.2 Changes in section 2 “Normative Reference”

- M-PHY<sup>SM</sup> specification: from version 2.00.00 to version 3.0.
- Unified Protocol (UniPro<sup>SM</sup>) specification: from version 1.41.00 to version 1.6.

### B.4.3 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.4.3 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<sup>SM</sup> 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<sup>SM</sup> 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.4.3 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 initialization 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 ( $bRPMB\_ReadWriteSize \times 256$  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 10<sup>th</sup> 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:

Page 10

- ### 3. Other suggestions for document improvement:

\_\_\_\_\_  
\_\_\_\_\_  
\_\_\_\_\_

### Submitted by

Name: \_\_\_\_\_

Phone: \_\_\_\_\_

Company:

E-mail:

**Address:**

City/State/Zip:

Date:



Biwin Storage Technology Co., Ltd.