JEDEC STANDARD

Universal Flash Storage (UFS)

Version 5.0

JESD220H

(Revision of JESD220G, December 2024)

February 2026

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.

All risk and liability relating to the use of JEDEC standards is assumed by the user, who agrees to indemnify and hold JEDEC harmless.

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

PRICE: Contact JEDEC
3103 10th Street North, Suite 240S, Arlington, VA 22201

For information, contact:

JEDEC Solid State Technology Association
3103 10th Street North
Suite 240S
Arlington, VA 22201
https://www.jedec.org/contact

This page intentionally left blank.

UNIVERSAL FLASH STORAGE (UFS), VERSION 5.0

Contents

Page

Foreword ... -xiii-
Introduction ... -xiii-
1 Scope ... 1
2 Normative References ... 2
3 Terms and Definitions ... 3
3.1 Acronyms ... 4
3.2 Conventions ... 5
3.3 Keywords ... 6
3.4 Abbreviations ... 6
4 Introduction ... 7
4.1 General Features ... 7
4.2 Interface Features ... 8
4.3 Functional Features ... 8
5 UFS Architecture Overview ... 9
5.1 UFS Top Level Architecture ... 9
5.1.1 Application Layer ... 9
5.1.2 UFS Device Manager ... 9
5.1.3 Service Access Points ... 10
5.1.4 UIO_SAP ... 11
5.1.5 UDM_SAP ... 11
5.1.6 UFS Transport Protocol Layer ... 11
5.1.7 UFS Interconnect Layer ... 11
5.1.8 UFS Topology ... 11
5.2 UFS System Model ... 12
5.3 System Boot and Enumeration ... 12
5.4 UFS Interconnect (UIC) Layer ... 13
5.4.1 UFS Physical Layer Signals ... 13
5.4.2 MIPI UniPro ... 13
5.4.3 MIPI UniPro Related Attributes ... 14
5.5 UFS Transport Protocol (UTP) Layer ... 14
5.5.1 Architectural Model ... 15
5.6 UFS Application and Command Layer ... 17
5.7 Mechanical ... 18
6 UFS Electrical: Clock, Reset, Signals And Supplies ... 19
6.1 UFS Signals ... 19
6.2 Reset Signal and LSS Signal ... 22
6.3 Power Supplies ... 22
6.4 Reference Clock ... 23
6.4.1 HS Gear Rates ... 27
6.4.2 Host Controller Requirements for Reference Clock Generation ... 28
6.5 External Charge Pump Capacitors (Optional) ... 30
6.6 ZQ Calibration Resistor (optional) ... 31
6.7 Absolute Maximum DC Ratings and Operating Conditions ... 32
6.8 AC and DC Operating Conditions ... 33

UNIVERSAL FLASH STORAGE (UFS), VERSION 5.0

Contents

Page

7 Reset, Power-Up And Power-Down 34
7.1 Reset 34
7.1.1 Power-on Reset 34
7.1.2 Hardware Reset 35
7.1.3 EndPointReset 36
7.1.4 Logical Unit Reset 37
7.1.5 Host UniPro Warm Reset 37
7.1.6 Summary of Resets and Device Behavior 38
7.2 Power Up Ramp 39
7.3 Power Off Ramp 40
7.4 UFS Device Power Modes and LU Power Condition 41
7.4.1 Device Power Modes 41
7.4.2 Power Management Command: START STOP UNIT 51
7.4.3 Power Mode Control 52
7.4.4 Logical Unit Power Condition 54
8 UFS UIC Layer: MIPI M-PHY 55
8.1 Termination 55
8.2 Drive Levels 55
8.3 PHY State Machine 55
8.4 HS Burst 56
8.4.1 HS Prepare Length Control 56
8.4.2 HS Sync Length Control 56
8.5 PWM Burst 56
8.5.1 LS Prepare Length Control 56
8.6 Adapt 56
8.7 UFS PHY Attributes 57
8.8 Electrical Characteristics 58
8.8.1 Transmitter Characteristics 58
8.8.2 Receiver Characteristics 58
9 UFS UIC Layer: MIPI Unipro 59
9.1 Overview 59
9.2 Architectural Model 59
9.3 UniPro/UFS Transport Protocol Interface (Data Plane) 60
9.3.1 Flow Control 60
9.3.2 Object Sizes 60
9.4 UniPro/UFS Control Interface (Control Plane) 61
9.5 UniPro/UFS Transport Protocol Address Mapping 62
9.6 Options and Tunable Parameters of UniPro 63
9.6.1 UniPro PHY Adapter 63
9.6.2 UniPro Data Link Layer 64
9.6.3 UniPro Network Layer 64
9.6.4 UniPro Transport Layer 65
9.6.5 UniPro Device Management Entity Transport Layer 65
9.6.6 UniPro Attributes 66

UNIVERSAL FLASH STORAGE (UFS), VERSION 5.0

Contents

Page

10 UFS Transport Protocol (UTP) Layer 67
10.1 Overview 67
10.2 UTP and UniPro Specific Overview 68
10.2.1 Phases 68
10.2.2 Data Pacing 68
10.2.3 UniPro 68
10.3 UFS Transport Protocol Transactions Overview 68
10.4 Service Delivery Subsystem 69
10.5 UPIU Transactions 69
10.6 General UFS Protocol Information Unit Format 71
10.6.1 Overview 72
10.6.2 Basic Header Format 72
10.7 UFS Protocol Information Units 78
10.7.1 COMMAND UPIU 78
10.7.2 RESPONSE UPIU 81
10.7.3 DATA OUT UPIU 89
10.7.4 DATA IN UPIU 93
10.7.5 READY TO TRANSFER UPIU 99
10.7.6 TASK MANAGEMENT REQUEST UPIU 105
10.7.7 TASK MANAGEMENT RESPONSE UPIU 108
10.7.8 QUERY REQUEST UPIU 110
10.7.9 QUERY RESPONSE UPIU 123
10.7.10 REJECT UPIU 137
10.7.11 NOP OUT UPIU 140
10.7.12 NOP IN UPIU 141
10.7.13 Data Out Transfer Rules 142
10.7.14 Overview - Data Out of Order Transfer 146
10.8 Logical Units 148
10.8.1 UFS SCSI Domain 148
10.8.2 UFS Logical Unit Definition 148
10.8.3 Well Known Logical Unit Definition 149
10.8.4 Logical Unit Addressing 149
10.8.5 Well Known Logical Units Defined in UFS 150
10.8.6 Translation of 8-bit UFS LUN to 64-bit SCSI LUN Address 151
10.8.7 SCSI Write Command 152
10.8.8 SCSI Read Command 153
10.8.9 Unit Attention Condition 154
10.9 Application Layer and Device Manager Transport Protocol Services 155
10.9.1 UFS Initiator Port and Target Port Attributes 155
10.9.2 Execute Command Procedure Call Transport Protocol Services 156
10.9.3 SCSI Command Transport Protocol Service 156
10.9.4 SCSI Command Received Transport Protocol 156
10.9.5 Send Command Complete Transport Protocol Service 157
10.9.6 Command Complete Received Transport Protocol Service 157
10.9.7 Data Transfer SCSI Transport Protocol Services 157
10.9.8 Task Management Function Procedure Calls 161
10.9.9 Query Function Transport Protocol Services 163

UNIVERSAL FLASH STORAGE (UFS), VERSION 5.0

Contents

Page

11 UFS Application (UAP) Layer – SCSI Commands 165
11.1 Universal Flash Storage Command Layer (UCL) Introduction 165
11.1.1 The Command Descriptor Block (CDB) 165
11.2 Universal Flash Storage Native Commands (UNC) 165
11.3 Universal Flash Storage SCSI Commands 166
11.3.1 General information about SCSI commands in UFS 167
11.3.2 INQUIRY Command 167
11.3.3 MODE SELECT (10) Command 168
11.3.4 MODE SENSE (10) Command 169
11.3.5 READ (10) Command 169
11.3.6 READ (16) Command 170
11.3.7 READ CAPACITY (10) Command 171
11.3.8 READ CAPACITY (16) Command 171
11.3.9 START STOP UNIT Command 172
11.3.10 TEST UNIT READY Command 172
11.3.11 REPORT LUNS Command 172
11.3.12 VERIFY (10) Command 173
11.3.13 WRITE (10) Command 174
11.3.14 WRITE (16) Command 175
11.3.15 REQUEST SENSE Command 175
11.3.16 FORMAT UNIT Command 176
11.3.17 PRE-FETCH (10) Command 177
11.3.18 PRE-FETCH (16) Command 177
11.3.19 SECURITY PROTOCOL IN Command 177
11.3.20 SECURITY PROTOCOL OUT Command 178
11.3.21 SEND DIAGNOSTIC Command 178
11.3.22 SYNCHRONIZE CACHE (10) Command 179
11.3.23 SYNCHRONIZE CACHE (16) Command 179
11.3.24 UNMAP Command 179
11.3.25 READ BUFFER Command 180
11.3.26 WRITE BUFFER Command 180
11.3.27 EXTENDED COPY Command 182
11.3.28 POPULATE TOKEN Command 183
11.3.29 WRITE USING TOKEN Command 183
11.3.30 RECEIVE ROD TOKEN INFORMATION Command 183
11.3.31 GET STREAM STATUS Command 184
11.4 Mode Pages 184
11.4.1 Mode Page Overview 184
11.4.2 UFS Supported Pages 184
11.5 Vital Product Data Parameters 186

UNIVERSAL FLASH STORAGE (UFS), VERSION 5.0

Contents

Page

12 UFS Security 187
12.1 UFS Security Feature Support Requirements 187
12.2 Secure Mode 187
12.2.1 Description 187
12.2.2 Requirements 187
12.2.3 Implementation 189
12.3 Device Data Protection 194
12.3.1 Description and Requirements 194
12.3.2 Implementation 194
12.4 RPMB 195
12.4.1 Introduction 195
12.4.2 RPMB Well Known Logical Unit Description 195
12.4.3 Requirements 196
12.4.4 Implementation in Normal RPMB Mode 208
12.4.5 Implementation in Advanced RPMB Mode 209
12.4.6 SECURITY PROTOCOL IN/OUT Commands 210
12.4.7 RPMB Operations 212
12.5 Malware Protection 265
13 UFS Functional Descriptions 266
13.1 UFS Boot 266
13.1.1 Introduction 266
13.1.2 Boot Configuration 266
13.1.3 Initialization and Boot Code Download Process 269
13.1.4 Initialization Process without Boot Code Download 272
13.1.5 Boot Logical Unit Operations 272
13.1.6 Configurability 273
13.1.7 Security 274
13.2 Logical Unit Management 274
13.2.1 Introduction 274
13.2.2 Logical Unit Features 274
13.2.3 Logical Unit Configuration 277
13.3 Logical Block Provisioning 283
13.3.1 Overview 283
13.3.2 Full Provisioning 283
13.3.3 Thin Provisioning 283
13.4 Host Device Interaction 284
13.4.1 Overview 284
13.4.2 Applicable Devices 284
13.4.3 Command Queue: Inter-LU Priority 284
13.4.4 Background Operations Mode 285
13.4.5 Power Off Notification 288
13.4.6 Dynamic Device Capacity 288
13.4.7 Data Reliability 293
13.4.8 Real-Time Clock Information 294
13.4.9 Timestamp Information 295
13.4.10 Context Management 296
13.4.11 System Data Tag Mechanism 299
13.4.12 Exception Events Mechanism 300

UNIVERSAL FLASH STORAGE (UFS), VERSION 5.0

Contents

Page

13.4.13 Queue Depth Definition 302
13.4.14 Device Life Span Mode 304
13.4.15 Refresh Operation 304
13.4.16 Temperature Event Notification 307
13.4.17 Performance Throttling Event Notification 308
13.4.18 WriteBooster 308
13.4.19 Host Initiated Defragmentation 317
13.4.20 Fast Recovery Mode 323
13.4.21 Copy Offloading 324
13.4.22 Pre-Erase 324
13.4.23 Zoned Logical Units 326
13.5 UFS Cache 328
13.6 Production State Awareness (PSA) 329
13.6.1 Introduction 329
13.6.2 PSA flow 329
14 UFS Descriptors, Flags And Attributes 333
14.1 UFS Descriptors 333
14.1.1 Descriptor Types 334
14.1.2 Descriptor Indexing 334
14.1.3 Accessing Descriptors and Device Configuration 334
14.1.4 Descriptor Definitions 338
14.2 Flags 375
14.3 Attributes 380
15 UFS Mechanical Standard 405
Annex A (Informative) Dynamic Capacity Host Implementation Example 406
A.1 Overview 406
A.2 Method Outline 406
Annex B (Informative) Reference Clock Measurement Procedure 407
B.1 Random RMS Jitter Measurement Procedure 407
B.2 Deterministic Jitter Measurement Procedure 408
B.3 Equipment Requirements 409
B.4 Test Setup 409
Annex C (Informative) Reference Clock Detection in HS-LSS : An Implementation Example 410
C.1 Overview 410
C.2 Method Outline 410
Annex D (Informative) Bibliography 411
Annex E (Informative) Differences between Revisions 412
E.1 Differences Between JESD220H and its Predecessor JESD220G (December 2024) 412
E.2 JESD220G and its Predecessor JESD220F (August 2022) 413
E.3 JESD220F and its Predecessor JESD220E (January 2020) 414
E.4 JESD220E and its Predecessor JESD220D (January 2018) 416
E.5 JESD220D and its Predecessor JESD220C (March 2016) 417
E.6 JESD220C and its Predecessor JESD220B (September 2013) 418
E.7 JESD220B and its Predecessor JESD220A (June 2012) 421

UNIVERSAL FLASH STORAGE (UFS), VERSION 5.0

Contents (cont'd)

Page

Figures

Figure 5.1 — UFS Top Level Architecture 11
Figure 5.2 — Usage of UDM_SAP 12
Figure 5.3 — Usage of UIO_SAP 12
Figure 5.4 — UFS System Model 14
Figure 6.1 — Conceptual UFS Device Block Diagram 21
Figure 6.2 — Supply Voltage Power Up Timings 25
Figure 6.3 — bRefClkGatingWaitTime 28
Figure 6.4 — Clock Input Levels, Rise Time, and Fall Time 29
Figure 6.5 — Test Load Impedance 30
Figure 6.6 — Output Driver and Input Receiver Levels 30
Figure 6.7 — Clock Output Levels, Rise Time and Fall Time 31
Figure 6.8 — DC and AC voltage Range for VCC/VCCQ/VCCQ_M /VCCQ2 35
Figure 7.1 — Power-on Reset 36
Figure 7.2 — Hardware Reset 37
Figure 7.3 — Reset AC timings 37
Figure 7.4 — EndPointReset 38
Figure 7.5 — Logical Unit Reset 39
Figure 7.6 — Power Up Ramps 41
Figure 7.7 — Power Off Ramps 42
Figure 7.8 — Power Mode State Machine 47
Figure 8.1 — Simplified Example for I/O Termination 57
Figure 9.1 — UniPro Internal Layering View (a) and UniPro Black Box View (b) 61
Figure 9.2 — Physical Lane Connections 65
Figure 10.1 – EHS Entry Format 78
Figure 10.2 — Data Out Transfer Example 93
Figure 10.3 — Data In Transfer Example 98
Figure 10.4 — DATA IN UPIU with Retransmission Sequence Example 99
Figure 10.5 — READY TO TRANSFER UPIU Sequence Example 104
Figure 10.6 — READY TO TRANSFER UPIU with Retransmission Sequence Example 105
Figure 10.7 — Aggregated Data Packet Example 136
Figure 10.8 — Example for Data Out Transfer Rule 1 143
Figure 10.9 — Example for Data Transfer Count Mismatch 144
Figure 10.10 — Example for Data Out Transfer Rule 2 145
Figure 10.11 — Example for Data Out Transfer Rule 3 146
Figure 10.12 — Example for Data Out of Order Transfer - 1 147
Figure 10.13 — Example for Data Out of Order Transfer - 2 148
Figure 10.14 — UFS SCSI domain 149
Figure 10.15 — Logical Unit Addressing 150
Figure 10.16 — SCSI Write 153
Figure 10.17 — SCSI Read 154
Figure 10.18 — Command without Data Phase 157
Figure 10.19 — Command + Read Data Phase 1/2 159
Figure 10.20 — Command + Read Data Phase 2/2 159

UNIVERSAL FLASH STORAGE (UFS), VERSION 5.0

Contents (cont'd)

Page

Figure 10.21 — Command + Write Data Phase 1/2 161
Figure 10.22 — Command + Write Data Phase 2/2 161
Figure 10.23 — Task Management Function 163
Figure 10.24 — UFS Query Function 164
Figure 12.1 — Purge Operation State Machine 192
Figure 12.2 — Advanced RPMB Message Structure in EHS Field 210
Figure 12.3 — Request Type Message Delivery 215
Figure 12.4 — Response Type Message Delivery 217
Figure 12.5 — Authentication Key Programming Flow 219
Figure 12.6 — Read Counter Value Flow 220
Figure 12.7 — Authenticated Data Write Flow 223
Figure 12.8 — Authenticated Data Read Flow 225
Figure 12.9 —Authenticated Secure Write Protect Configuration Block Write Flow 228
Figure 12.10 — Authenticated Secure Write Protect Configuration Block Read Flow 230
Figure 12.11 — RPMB Purge Enable Flow 232
Figure 12.12 — RPMB Purge Status Read Flow 233
Figure 12.13 — Authenticated Vendor Specific Command Request Flow 235
Figure 12.14 — Authenticated Vendor Specific Command Read Request Flow 237
Figure 12.15 — Authentication Key Programming Flow (in Advanced RPMB Mode) 238
Figure 12.16— Read Counter Value Flow (in Advanced RPMB Mode) 240
Figure 12.17 — Authenticated Data Write Flow (in Advanced RPMB Mode) 243
Figure 12.18 — Authenticated Data Read Flow (in Advanced RPMB Mode) 246
Figure 12.19 — Authenticated Secure Write Protect Configuration Block Write Flow (in Advanced RPMB Mode) 249
Figure 12.20 — Authenticated Secure Write Protect Configuration Block Read Flow (in Advanced RPMB Mode) 253
Figure 12.21 — RPMB Purge Enable Flow (in Advanced RPMB Mode) 256
Figure 12.22 — RPMB Purge Status Read Flow (in Advanced RPMB Mode) 258
Figure 12.23 — Authenticated Vendor Specific Command Request Flow (in Advanced RPMB Mode) 261
Figure 12.24 — Authenticated Vendor Specific Command Status Read Request Flow (in Advanced RPMB Flow) 264
Figure 13.1 — UFS System Diagram 267
Figure 13.2 — Example of UFS Device Memory Organization for Boot 269
Figure 13.3 — Device Initialization and Boot Procedure Sequence Diagram 272
Figure 13.4 — Example of UFS Device Memory Organization 276
Figure 13.5 — Physical Memory Resource State Machine 293
Figure 13.6 — Example of Data Status After a Power Failure During Reliable Write Operation 294
Figure 13.7 — Concept of WriteBooster Feature 309
Figure 13.8 — Example of "LU Dedicated Buffer" Mode Configuration 311
Figure 13.9 — Example of "shared buffer" Mode Configuration 311
Figure 13.10 – (Example) Initiating a WriteBooster Buffer Resize Operation 315
Figure 13.11 – (Example) Checking the Status of Resize Operation and Updated WriteBooster Buffer Size 315
Figure 13.12 – (Example) Transition of bHIDState Attribute 319

UNIVERSAL FLASH STORAGE (UFS), VERSION 5.0

Contents (cont'd)

Page

Figure 13.13 – (Example) HID Analysis Operation 321
Figure 13.14 – (Example) HID Defrag Operation 322
Figure 13.15 – (Example) HID Analysis & HID Defrag Operation 323
Figure 13.16 – (Example) Monitoring the Progress of Defragmenting 323
Figure 13.17 – FAST_RECOVERY_NEEDED from RESPONSE UPIU 324
Figure 13.18 – Pre-Erase Operation 324
Figure 13.19 — PSA Flow 332
Figure 13.20 — PSA State Machine 333
Figure 14.1 — Descriptor Organization 335
Figure 14.2 — Read Request Descriptor 337
Figure 14.3 — Write Request Descriptor 338
Figure B.1 — Test Setup 410
Figure C.1 — REF CLK Frequency Detection in HS-LSS 411

Tables

Table 5.1 — UFS Signals 16
Table 5.2 — ManufacturerID and DeviceClass Attributes 17
Table 6.1 — Signal Name and Definitions 23
Table 6.2 — Electrostatic Discharge Sensitivity Characteristics 24
Table 6.3 — Reset Signal and LSS Signal Electrical Parameters 25
Table 6.4 — UFS Power Supply Parameters 25
Table 6.5 — Reference Clock Parameters 27
Table 6.6 — Reference Clock Frequency and HS Operation 28
Table 6.7 — HS-BURST Rates 30
Table 6.8 — Host Controller Reference Clock Parameters 31
Table 6.9 — Charge Pump Capacitors Description 33
Table 6.10 — Charge Pump Related Ball Names 34
Table 6.11 — ZQ Calibration Resistors Description 34
Table 6.12 — Absolute Maximum DC ratings and Operating Conditions 35
Table 6.13 —AC and DC Voltage Operating Conditions 36
Table 7.1 — Reset timing parameters 38
Table 7.2 — Reset States 41
Table 7.3 — UniPro Attributes, UFS Attributes and UFS Flags reset 41
Table 7.4 — Allowed SCSI Commands and UPIU for Each Power Mode 51
Table 7.5 — Device Well Known Logical Unit Responses to SSU Command 52
Table 7.6 — Device Well Known Logical Unit Responses to Commands Other Than SSU 53
Table 7.7 — Pollable Sense Data for Each Power Modes 53
Table 7.8 — START STOP UNIT Fields 54
Table 7.9 — Attribute for Power Mode Control 55
Table 7.10 — Device Descriptor Parameters 55
Table 7.11 — Power Parameters Descriptor Fields 56
Table 7.12 — Format for Power Parameter Element 56
Table 7.13 — Logical Unit Response to SCSI Command 57

UNIVERSAL FLASH STORAGE (UFS), VERSION 5.0

Contents (cont'd)

Page

Table 8.1 — UFS PHY M-TX Capability Attributes(0) 60
Table 8.2 — UFS PHY M-RX Capability Attributes(0) 61
Table 9.1 — UFS Initiator and Target Port Identifiers 65
Table 9.2 — UniPro Attribute 68
Table 10.1 — UPIU Transaction Codes 71
Table 10.2 — UPIU Transaction Code Definitions 72
Table 10.3 — General Format of the UFS Protocol Information Unit 73
Table 10.4 — Basic Header Format 74
Table 10.5 — Transaction Type Format 74
Table 10.6 — UPIU Flags 75
Table 10.7 — UTP Response Values 76
Table 10.8 — UPIU Associated to a Single Task 76
Table 10.9 — Initiator ID Composition 77
Table 10.10 — Command Set Type 77
Table 10.11 — EHS Entry Format 79
Table 10.12 — COMMAND UPIU 80
Table 10.13 — Flags definition for COMMAND UPIU 81
Table 10.14 — RESPONSE UPIU 84
Table 10.15 — Flags Definition for RESPONSE UPIU 85
Table 10.16 — Flags and Residual Count Relationship 88
Table 10.17 — DATA OUT UPIU 91
Table 10.18 — Flags Definition for DATA OUT UPIU 92
Table 10.19 —DATA IN UPIU 95
Table 10.20 — Flags Definition for DATA IN UPIU 96
Table 10.21 — READY TO TRANSFER UPIU 101
Table 10.22 — Flags Definition for RTT UPIU 102
Table 10.23 — Task Management Request UPIU 107
Table 10.24 — Task Management Function Values 108
Table 10.25 — Task Management Input Parameters 109
Table 10.26 — Task Management Response UPIU 110
Table 10.27 — Task Management Output Parameters 111
Table 10.28 — Task Management Service Response 111
Table 10.29 — QUERY REQUEST UPIU 112
Table 10.30 — Query Function Field Values 114
Table 10.31 — Transaction Specific Fields 115
Table 10.32 — Query Function Opcode Values 115
Table 10.33 — Read Descriptor 116
Table 10.34 — Write Descriptor 117
Table 10.35 — Read Attribute 118
Table 10.36 — Write Attribute 119
Table 10.37 — Read Flag 120
Table 10.38 — Set Flag 121
Table 10.39 — Clear Flag 122
Table 10.40 — Toggle Flag 123
Table 10.41 — NOP 123

UNIVERSAL FLASH STORAGE (UFS), VERSION 5.0

Contents (cont'd)

Page

Table 10.42 — Aggregated Read 124
Table 10.43 — QUERY RESPONSE 125
Table 10.44 — Query Response Code 126
Table 10.45 — Transaction Specific Fields 127
Table 10.46 — Read Descriptor 128
Table 10.47 — Write Descriptor 129
Table 10.48 — Read Attribute Response Data Format 130
Table 10.49 — Write Attribute 131
Table 10.50 — Read Flag Response Data Format 132
Table 10.51 — Set Flag 133
Table 10.52 — Clear Flag 134
Table 10.53 — Toggle Flag 135
Table 10.54 — NOP 135
Table 10.55 — Aggregated Read 136
Table 10.56 — Aggregated Data Packet Group Header Format 137
Table 10.57 — Reject UPIU 139
Table 10.58 — Basic Header Status Description 141
Table 10.59 — E2E Status Definition 141
Table 10.60 — NOP OUT UPIU 142
Table 10.61 — NOP IN UPIU 143
Table 10.62 — Parameters # to Data Out Transfer Rules 147
Table 10.63 — Well Known Logical Unit Commands 152
Table 10.64 — Examples of Logical Unit Representation Format 153
Table 10.65 — Events for UAC Establishment 156
Table 10.66 — Commands for Exceptional Behavior on UAC 156
Table 10.67 — UFS Initiator Port and Target Port Attributes 157
Table 10.68 — Send Query Request UFS Transport Protocol Service 165
Table 10.69 — Query Request Received UFS Transport Protocol Service Indication 166
Table 10.70 — Query Function Executed UFS Transport Protocol Service Response 166
Table 10.71 — Received Query Function Executed UFS Transport Protocol Service Confirmation 166
Table 11.1 — UFS SCSI Command Set 168
Table 11.2 — Format of LUN field in UPIU 175
Table 11.3 — Well Known Logical Unit Numbers 175
Table 11.4 — UFS Supported Pages 186
Table 12.1 — Secure Write Protect Configuration Block for Normal RPMB 199
Table 12.2 — Secure Write Protect Configuration Block for Advanced RPMB 200
Table 12.3 — Secure Write Protect Entry 201
Table 12.4 — Write Protect Type Field 202
Table 12.5 — RPMB Purge Response Packet Format (Legacy RPMB Mode) 203
Table 12.6 — RPMB Purge Response Packet Format (Advanced RPMB Mode) 203
Table 12.7 — RPMB Message Components 206
Table 12.8 — Request Message Types 207
Table 12.9 — Response Message Types 207
Table 12.10 — Result Data Structure 208
Table 12.11 — Result Code Definition 209

UNIVERSAL FLASH STORAGE (UFS), VERSION 5.0

Contents (cont'd)

Page

Table 12.12 — RPMB Message Data Frame 210
Table 12.13 — Advanced RPMB Meta Information 212
Table 12.14 — SECURITY PROTOCOL SPECIFIC Field for Protocol ECh 213
Table 12.15 — Security Protocol Specific Field Values for Protocol 00h 214
Table 12.16 — Expected Data Transfer Length Value for Request Type Messages 215
Table 12.17 — Expected Data Transfer Length Value for Response Type Messages 217
Table 13.1 — bBootLunEn Attribute 269
Table 13.2 — Valid UPIUs and SCSI Commands for Each Initialization Phase 274
Table 13.3 — Logical Unit Configurable Parameters 280
Table 13.4 — Parameter for Controlling Logical Unit Data Reliability 296
Table 14.1 — Descriptor Identification Values 335
Table 14.2 — Generic Descriptor Format 340
Table 14.3 — Logical Unit Descriptor Format 340
Table 14.4 — Device Descriptor. 340
Table 14.5 — wManufacturerID Definition 350
Table 14.6 — Configuration Descriptor Format (INDEX = 00h) 351
Table 14.7 — Configuration Descriptor Format (INDEX = 01h) 352
Table 14.8 — Configuration Descriptor Format (INDEX = 02h) 352
Table 14.9 — Configuration Descriptor Format (INDEX = 03h) 352
Table 14.10 — Configuration Descr. Header and Device Descr. Conf. Parameters (INDEX = 00h) 353
Table 14.11 — Configuration Descr. Header with INDEX = 01h/02h/03h 354
Table 14.12 — Unit Descriptor Configurable Parameters 355
Table 14.13 — Geometry Descriptor. 355
Table 14.14 — Unit Descriptor 365
Table 14.15 — RPMB Unit Descriptor. 369
Table 14.16 — Power Parameters Descriptor 372
Table 14.17 — Interconnect Descriptor 372
Table 14.18 — Manufacturer Name String 373
Table 14.19 — Product Name String 373
Table 14.20 — OEM_ID String 374
Table 14.21 — Serial Number String Descriptor 374
Table 14.22 — Product Revision Level String 375
Table 14.23 — Device Health Descriptor 375
Table 14.24 — Vendor Specific Descriptor 377
Table 14.25 — Flags Access Properties 377
Table 14.26 — Flags 377
Table 14.27 — Attributes Access Properties 382
Table 14.28 — Attributes. 382
Table B.1 — Maximum Random RMS Jitter Amount 409

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 standards from other standards bodies, such as MIPI (M-PHY and UniPro) and INCITS/SCSI (SBC, SPC, SAM, and ZBC).

Introduction

The UFS electrical interface is a universal serial communication bus which can be utilized for different types of applications. MIPI M-PHY is used as the 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 such as smart phones, cameras, organizers, PDAs, digital recorders, MP3 players, internet tablets, electronic toys, etc.

This page intentionally left blank.

UNIVERSAL FLASH STORAGE (UFS), VERSION 5.0

(From JEDEC Board Ballot JCB-26-09, formulated under the cognizance of the JC-64.1 Subcommittee on Electrical Specifications and Command Protocols.)

1 Scope

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

2 Normative References

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

[MIPI-M-PHY], *MIPI Alliance Specification for M-PHY®, Version 6.0,* November 2025.

[MIPI-UniPro], *MIPI Alliance Specification for Unified Protocol (UniPro®), Version 3.0,* September 2025.

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

[SAM], *INCITS T10 standard: SCSI Architecture Model – 6 (SAM–6), INCITS 546-2021.*

[SPC], *INCITS T10 standard: SCSI Primary Commands – 6 (SPC-6), INCITS 566-2025.*

[SBC], *INCITS T10 standard: SCSI Block Commands – 5 (SBC–5), INCITS 571-2025.*

[ZBC], *INCITS, SCSI Zoned Block Commands - 2 (ZBC-2), INCITS 550-2023.*

[JESD8-12], *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, JESD8-12A.01,* September 2007.

[HBM-MM], *JEDEC Recommended ESD Target Levels for HBM/MM Qualification, JEP155B,* July 2019.

[CDM], *JEDEC Recommended ESD-CDM Target Levels, JEP157A,* April 2022.

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

[JEP106], *Standard Manufacturer's Identification Code, JEP106BM,* June 2025.

[JESD21], *Multi-chip Packages (MCP) and Discrete eMMC, e2MMC, and UFS, JESD21C, —* [(1)]

[JESD220-3], *UFS Host Performance Booster (HPB) Extension Specification, JESD220-3A,* September 2020.

[JESD220-4], *UFS File Based Optimization (FBO) Extension Specification, JESD220-4 Version 1.01,* November 2020.

[JESD230], *NAND Flash Interface Interoperability, JESD230G.01,* September 2025.

---

(1) To be Published.

3 Terms and Definitions

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

Application Client: See [SAM].

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

Command Descriptor Block: See [SAM]

Device ID: The bus address of a UFS device.

Device Server: See [SAM].

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

Dword: 32-bit data value, a Doubleword.

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

Host: See [SPC].

Initiator device: See [SAM].

Kilobyte: 1024 or 2^{10} bytes.

Logical Unit: See [SAM].

Logical Unit Number: See [SAM].

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

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

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

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

Target device: See [SAM].

Task: Synonymous with a SCSI command. See [SAM].

Task Tag: Synonymous with a SCSI command identifier. See [SAM].

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

Terabyte: 1.099.511.627.776 or 2^{40} bytes.

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

Unit: A bus device

Unit Attention: See [SPC].

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

3.1 Acronyms

CDB	Command Descriptor Block
CPort	A CPort is a Service Access Point on the UniPro Transport Layer (L4) within a Device that is used for Connection-oriented data transmission
DMA	Direct Memory Access
DSC	Digital Still Camera
FFU	Field Firmware Update
GB	Gigabyte
HCI	Host Controller Interface
HPB	UFS Host Performance Booster
HS-LSS	High Speed – Link Startup Sequence
KB	Kilobyte
LBA	Logical Block Address
LS-LSS	Low Speed – Link Startup Sequence
LSS	Link Startup Sequence
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
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

3.1 Acronyms (cont'd)

UFS	Universal Flash Storage
UMPC	Ultra-Mobile PC
UniPro	Unified Protocol
UPIU	UFS Protocol Information Unit
UTP	UFS Transport Protocol
ZBC	Zoned Block Commands

3.2 Conventions

This standard follows some conventions used in SCSI documents since it adopts several SCSI standards. See [SAM] for numeric conventions.

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

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

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

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

3.3 Keywords

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

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

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

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

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

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

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

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

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

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

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

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

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

3.4 Abbreviations

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

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

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

4 Introduction

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

4.1 General Features

- Target performance
  - High-Speed GEARs (see 6.4.1 for details)
    - Support for GEAR1 is mandatory
    - Support for GEAR2 is mandatory
    - Support for GEAR3 is mandatory
    - Support for GEAR4 is mandatory
    - Support for GEAR5 is mandatory
    - Support for GEAR6 is mandatory
- Target host applications
  - Mobile phone, UMPC, DSC, PMP, MP3 and any other applications that require mass storage, bootable mass storage, and external card
- Target device types
  - External card
  - Embedded device
    - Mass storage and bootable mass storage
  - Future expansion of device class types
    - I/O devices, camera, wireless, … , etc.
- Topology
  - One device per UFS port.
- UFS Layering
  - UFS Command Set Layer (UCS)
    - Simplified SCSI command set based on SBC, SPC, and ZBC. UFS will not modify these SBC, SPC, and ZBC compliant commands. Options for defining UFS Native command and future extensions exist.
  - UFS Transport Protocol Layer (UTP)
    - JEDEC-defined protocol layer, i.e., UTP for SCSI. This does not exclude support for other protocols in the UFS Transport Protocol Layer.
  - UFS Interconnect Layer (UIC)
    - MIPI UniPro® [MIPI-UniPro] is adopted for data link layer
      - MIPI M-PHY® [MIPI-M-PHY] is adopted for physical layer

4.2 Interface Features

- Three power supplies
  - VCCQ power supply: 1.2 V (nominal)
  - VCCQ_M power supply: 1.2 V (nominal)
  - VCCQ2 power supply: 1.8 V (nominal)
  - VCC power supply: 2.5 V (nominal)
- Signaling, refer to [MIPI-M-PHY]
- Two signaling schemes
  - Multiple gears defined for High-Speed burst mode
  - Low-speed mode, Gear1 only, with PWM signaling scheme
- Adapt (M-PHY® versions 4.1 and above)
  - M-RX equalizer training to adapt to the channel characteristic
- M-PHY TX EQ training feature is supported in M-PHY 6.0 and UniPro 3.0

4.3 Functional Features

UFS functional features are NAND management features. These include

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

5 UFS Architecture Overview

5.1 UFS Top Level Architecture

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

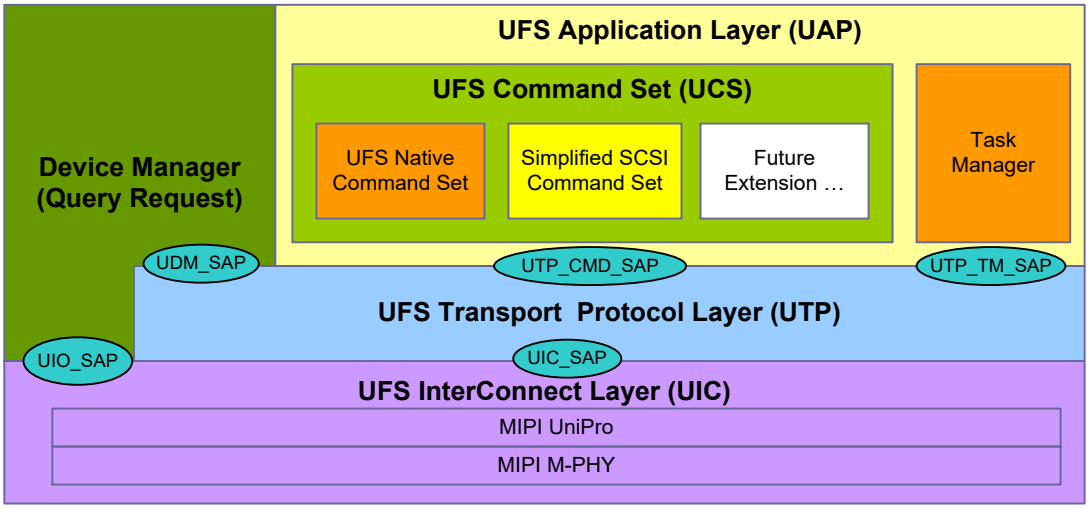


Figure 5.1 — UFS Top Level Architecture

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

5.1.1 Application Layer

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

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

5.1.2 UFS Device Manager

The device manager has the following two responsibilities:

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

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

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

5.1.3 Service Access Points

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

- UDM_SAP
- UIO_SAP

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

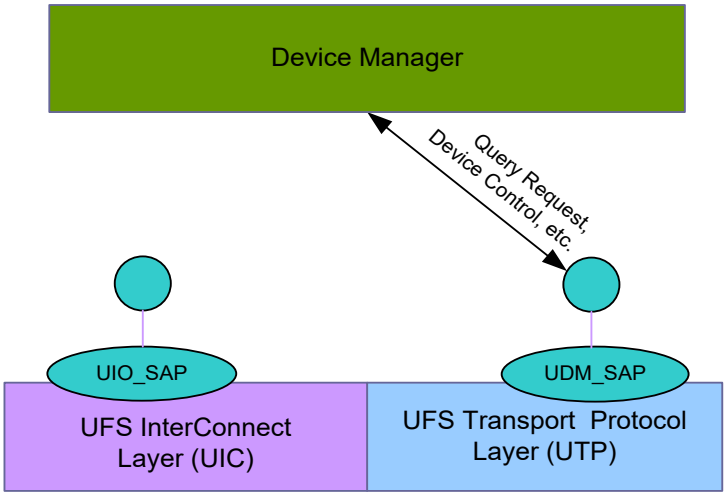


Figure 5.2 — Usage of UDM_SAP

UIO_SAP is the service access point exposed by the UIC layer for the device manager to trigger the reset of the UIC layer and to transfer requests and responses related to UIC management functions. Figure 5.3 depicts the usage of the service access point.

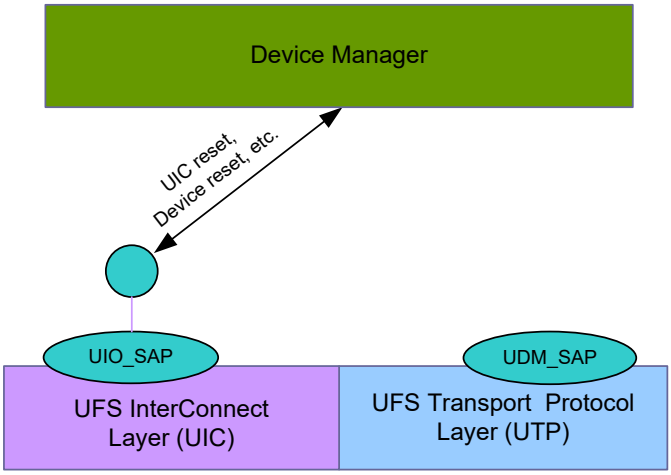


Figure 5.3 — Usage of UIO_SAP

5.1.4 UIO_SAP

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

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

For the detailed internal mechanism, refer to [MIPI-UniPro].

5.1.5 UDM_SAP

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

For further details refer to the following sub-clauses: 10.9.9, Query Function Transport Protocol Services, 10.7.8, QUERY REQUEST UPIU, and 10.7.9, QUERY RESPONSE UPIU.

5.1.6 UFS Transport Protocol Layer

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

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

5.1.7 UFS Interconnect Layer

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

5.1.8 UFS Topology

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

5.2 UFS System Model

Figure 5.4 shows an example of UFS system. It shows how a UFS host is connected to a UFS device, the position of UFS host controller and its related UFS HCI interface.

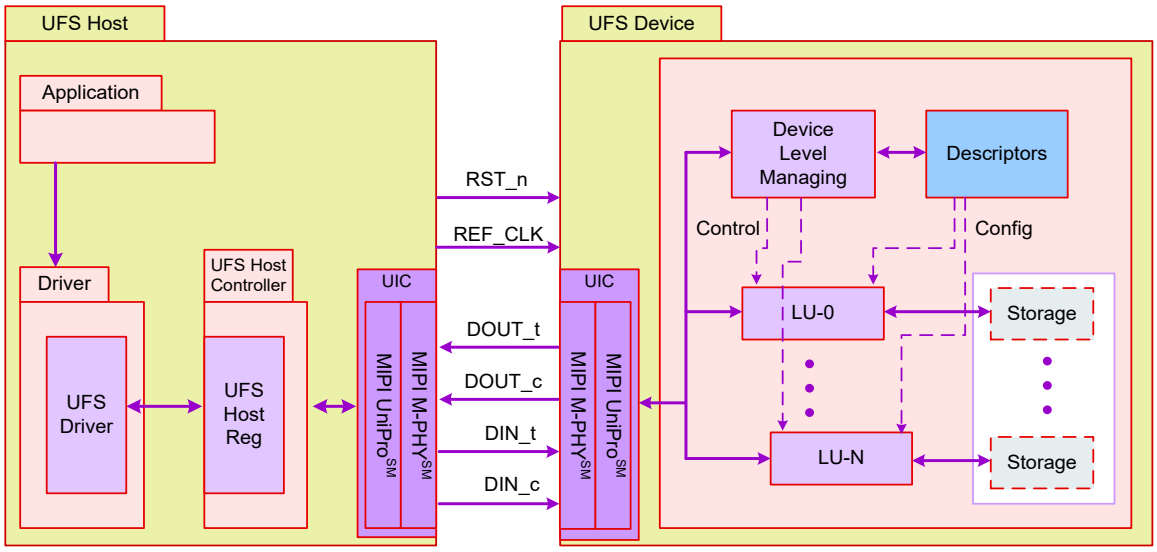


Figure 5.4 — UFS System Model

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

Figure 5.4 also indicates the UFS interface between the UFS host and the UFS device. The UFS Interconnect (UIC) Layer consists of MIPI UniPro and MIPI M-PHY. The physical layer M-PHY is differential, dual simplex PHY that includes TX and RX pairs.

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

5.3 System Boot and Enumeration

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

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

5.4 UFS Interconnect (UIC) Layer

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

5.4.1 UFS Physical Layer Signals

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

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

Table 5.1 summarizes the signals required for a UFS device. Only the single lane, per direction, per link, configuration is shown. See clause 6 and clause 8 for full details about UFS signals.

Table 5.1 — UFS Signals

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

5.4.2 MIPI UniPro

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

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

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

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

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

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

5.4.3 MIPI UniPro Related Attributes

In general, UniPro related attributes and values and their use are defined in [MIPI-UniPro]. Many attributes are generic to all UniPro applications and are not included this document. The following attributes have specific usage in UFS applications as indicated in Table 5.2.

Table 5.2 — ManufacturerID and DeviceClass Attributes

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

5.5 UFS Transport Protocol (UTP) Layer

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

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

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

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

5.5.1 Architectural Model

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

5.5.1.1 Client-Server Model

See [SAM] for the description of the SCSI distributed service model and client-server model.

Initiator device and Target device are mapped into UFS physical network devices.

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

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

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

See [SAM] for the interrelation between commands (sometimes called tasks in this standard), command identifiers (sometimes called task tags in this standard), logical units, task managers, and task sets.

See [SAM] for the definition of command descriptor block (CDB). The description of the CDB content and structure are defined in detail in [SAM], [SBC] and [SPC] INCITS T10 standards.

Responsibilities of each class are described in SCSI (see [SAM], [SPC], and [SBC]). Additional responsibilities for each class are as described as in this standard.

The Device Manager class description can be found in 5.1.2.

5.5.1.2 CDB, Status, Task Management

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

5.5.1.3 Nexus

Nexus and nexus notation are defined in [SAM]. Using nexus notation with a command identifier (or task tag) to uniquely identify a command or task is also defined in [SAM].

There shall be at least one initiator device in the UFS definition. There shall only one target device, the UFS device. Each UFS device shall include one or more logical units.

Overlapped commands are defined in [SAM]. The UFS device is not required to detect overlapped commands.

5.5.1.4 SCSI Command Model

The SCSI command model is defined in [SAM].

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

5.5.1.5 SCSI Task Management Functions

Task management functions are defined in [SAM].

Parameters fields in the UTP Task Management Request and UTP Task Management Response headers contain the requisite information for the input and output arguments as described in the task management function procedure calls in [SAM].

5.6 UFS Application and Command Layer

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

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

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

UFS SCSI command set includes:

1. SBC compliant commands [SBC]:
   - FORMAT UNIT
   - READ (10) and READ (16)
   - READ CAPACITY (10)
   - REQUEST SENSE
   - SEND DIAGNOSTIC
   - UNMAP
   - WRITE (10) and WRITE (16)
   - GET STREAM STATUS
2. SPC compliant commands [SPC]:
   - INQUIRY
   - REPORT LUNS
   - READ BUFFER
   - TEST UNIT READY
   - WRITE BUFFER
   - SECURITY PROTOCOL IN and SECURITY PROTOCOL OUT
   - RECEIVE COPY STATUS
3. Optionally supported ZBC compliant commands [ZBC] (see 13.4.23 for support requrements):
   - FINISH ZONE
   - REPORT ZONES
   - RESET WRITE POINTER

5.6 UFS Application and Command Layer (cont'd)

4. SCSI operational commands for UFS applications and compatible with existing SCSI driver:
   - MODE SELECT (10) and MODE SENSE (10)
   - PRE-FETCH (10)
   - START STOP UNIT
   - SYNCHRONIZE CACHE (10)
   - VERIFY (10)
5. Value-added optional commands for UFS:
   - PRE-FETCH (16), SYNCHRONIZE CACHE (16), and READ CAPACITY(16).

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

5.7 Mechanical

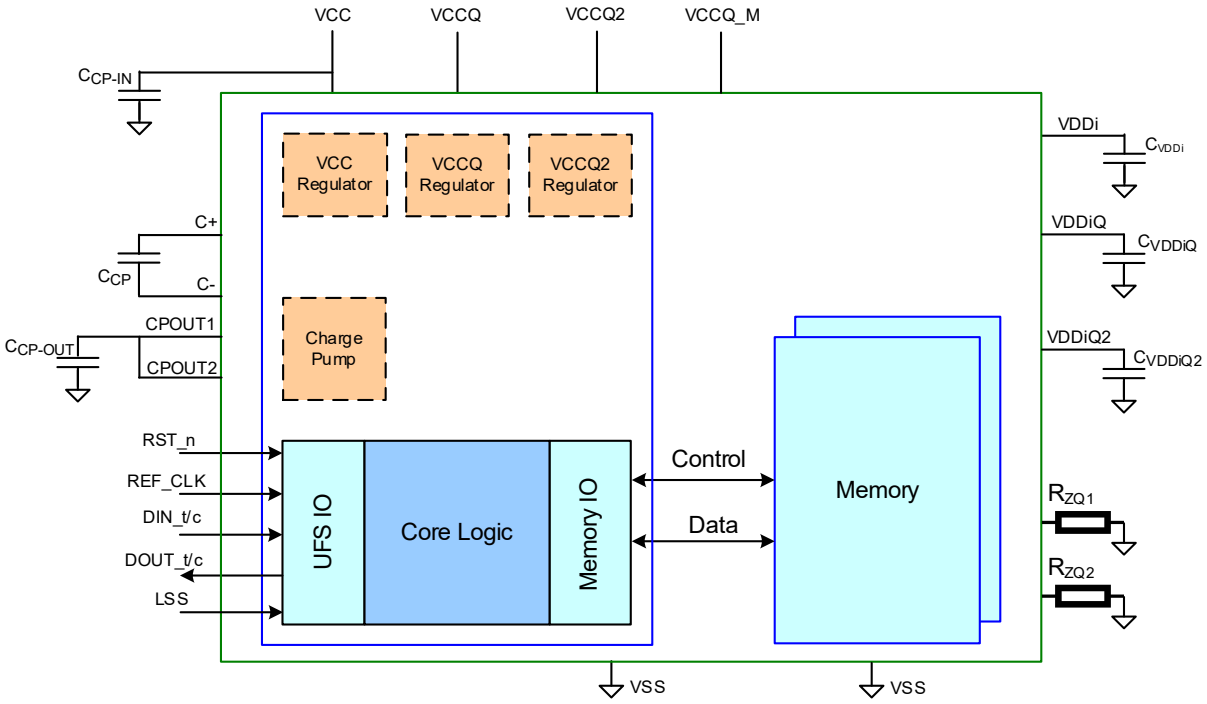
Packaging and requirements for UFS embedded device should adhere to the following guideline if possible:

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

6 UFS Electrical: Clock, Reset, Signals And Supplies

6.1 UFS Signals

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



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

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

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

Figure 6.1 — Conceptual UFS Device Block Diagram

6.1 UFS Signals (cont'd)

Table 6.1 — Signal Name and Definitions

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

6.1 UFS Signals (cont'd)

Table 6.1 — Signal Name and Definitions (cont'd)

Name	Type	Description
LSS(4)	Input	Input Link Startup Sequence (LSS) mode. 0: low speed link startup sequence (LS-LSS) 1: high speed link startup sequence (HS-LSS) This pin may be directly driven by the host or connected to external pull-up or pull-down according to the selected Link Startup Sequence mode. This value is expected to be fixed before RST_n is high to determine the Link Startup mode. After link startup detection has been completed, the device should take steps to minimize leakage current.
R_{ZQ1}/R_{ZQ2}	Input	External Calibration Resistor (optional). Refer to datasheet to determine if resistors are needed.

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 For the LSS pin, the recommended external pull-up/pull-down is between 0 and 2.2 kΩ.

It is recommended to apply [HBM-MM] and [CDM] to all signals described in Table 6.1. See Table 6.2 for ESD specification This standard does not require use of the Machine Model for ESD qualification.

Table 6.2 — Electrostatic Discharge Sensitivity Characteristics

Parameter	Symbol	Min	Max	Unit	NOTEs
Human body model (HBM)	ESD_{HBM}	1000	-	V	1
Charged-device model (CDM)	ESD_{CDM}	250	-	V	2

NOTE 1 Refer to ESDA/JEDEC Joint Standard JS-001-2024 for HBM measurement procedures.

NOTE 2 Refer to ESDA/JEDEC Joint Standard JS-002-2022 for CDM measurement procedures.

6.2 Reset Signal and LSS Signal

To meet the requirements of the JEDEC standard [JESD8-12A], the RST_n and LSS signal voltages shall be within the specified ranges for VCCQ in Table 6.3.

Table 6.3 — Reset Signal and LSS 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 6.4
Input LOW voltage	VIL	VSS-0.3	0.35*VCCQ	V	For VCCQ as defined in Table 6.4
Input Capacitance	Cin		10	pF	
Input leakage Current	Ilkg		10	µA	

6.3 Power Supplies

Table 6.4 — UFS Power Supply Parameters

Parameter	Symbol	Min	Max	Unit	NOTEs
VCC DC operating range	VCC	2.4	2.7	V	3
VCCQ DC operating range	VCCQ	1.14	1.26	V	1, 3
VCCQ_M DC operating range	VCCQ_M	1.14	1.26	V	1, 3
VCCQ2 DC operating range	VCCQ2	1.70	1.95	V	3
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_{VDDi}	1		µF	
VCCQ internal regulator capacitor	C_{VDDiQ}	1		µF	
VCCQ2 internal regulator capacitor	C_{VDDiQ2}	1		µF	

NOTE 1 See [JESD8-12A.01].

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.

6.3 Power Supplies (cont'd)

Figure 6.2 shows tPRUH, tPRUL and tPRUV timings.

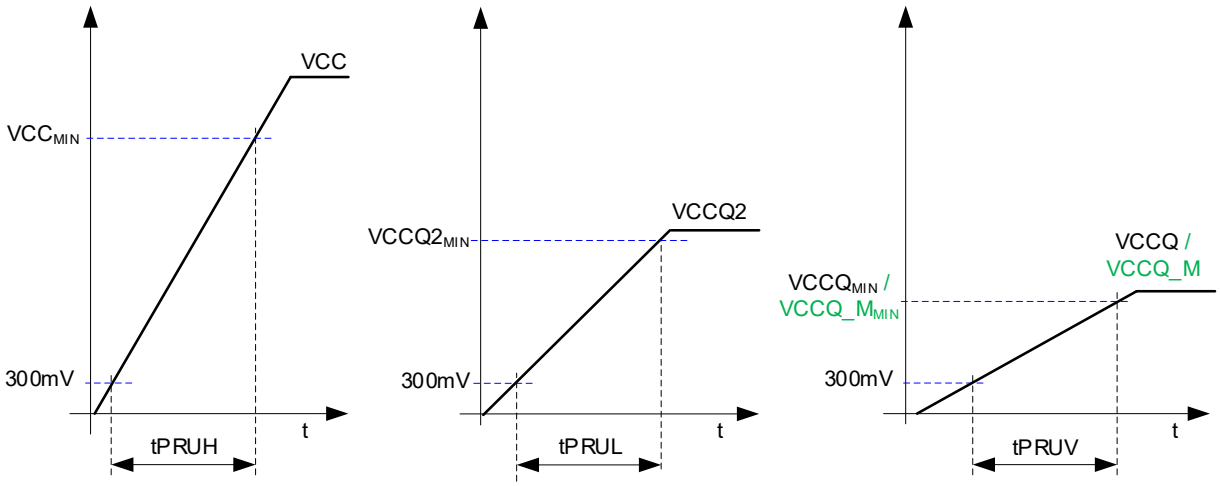


Figure 6.2 — Supply Voltage Power Up Timings

6.4 Reference Clock

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

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

6.4 Reference Clock (cont'd)

Table 6.5 — Reference Clock Parameters

Parameter	Symbol	Nominal		Unit	NOTEs
Frequency	f_{ref}	19.2, 26.0, 38.4, 52.0		MHz	1
		Min	Max		
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
Random Jitter (f_{ref} = 19.2)	RJ_{RMS}		5.9	ps	5
Random Jitter (f_{ref} = 26.0)			4.6		
Random Jitter (f_{ref} = 38.4)			3.5		
Random Jitter (f_{ref} = 52.0)			2.8		
Deterministic Jitter	$DJ_{\delta\delta}$		15	ps	6
Input Impedance	RL_{RX}	100		kΩ	7
	CL_{RX}		5	pF	

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

NOTE 2 Figure 6.4 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 The RJRMS magnitudes are based on the phase noise parameters of previous versions of this standard and replace them. RJRMS is computed with the integration range from 50 kHz to the Nyquist frequency of the reference clock.

NOTE 6 Reference Clock frequencies of 19.2 MHz and 26.0 MHz cannot be used for HS-G5 and above, because the corresponding RJRMS values are beyond the allowable limit for HS-G5 operation. The frequency of the reference clock shall be 38.4 MHz or 52.0 MHz for HS-G5 and above operation.

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

6.4 Reference Clock (cont'd)

bRefClkFreq attribute indicates to the device the frequency of the REF_CLK signal in LS-LSS mode, and its default value corresponds to 52.0 MHz.

Table 6.6 provides an overview of the relationship between reference clock frequencies and HS operation.

Table 6.6 — Reference Clock Frequency and HS Operation

Reference Clock Frequency [MHz]	HS-G1	HS-G2	HS-G3	HS-G4	HS-G5	HS-G6
19.2	Y	Y	Y	Y	N	N
26.0	Y	Y	Y	Y	N	N
38.4	Y	Y	Y	Y	Y	Y
52.0	Y	Y	Y	Y	Y	Y

bRefClkFreq attribute can be written only if both sub-links are in LS-MODE. This attribute indicates the REF_CLK frequency. In HS-LSS mode, the UFS device shall detect the frequency of the REF_CLK signal. A stable REF_CLK signal is expected when RST_n signal goes high. REF_CLK is not expected to change in HS-Mode. See Annex C for details.

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

- LS-MODE (SLEEP or PWM-BURST state)
- HIBERN8 state

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

6.4 Reference Clock (cont'd)

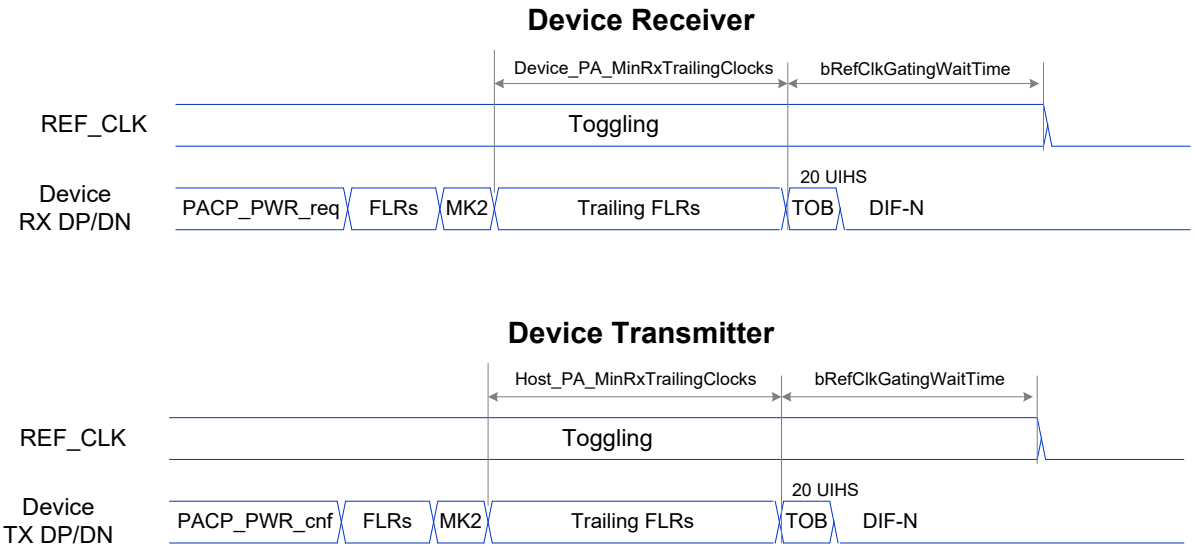


Figure 6.3 — bRefClkGatingWaitTime

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.

Device PA_MinRxTrailingClocks and Host PA_MinRxTrailingClocks should be considered, in addition to bRefClkGatingWaitTime, to determine when the reference clock may be stopped.

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

The reference clock shall be turned ON and stably running before initiation of the state transition to STALL from a SLEEP or HIBERN8 state. Reference Clock shall not be gated in HS modes.

6.4 Reference Clock (cont'd)

Figure 6.4 shows clock rise time and fall time measurements.

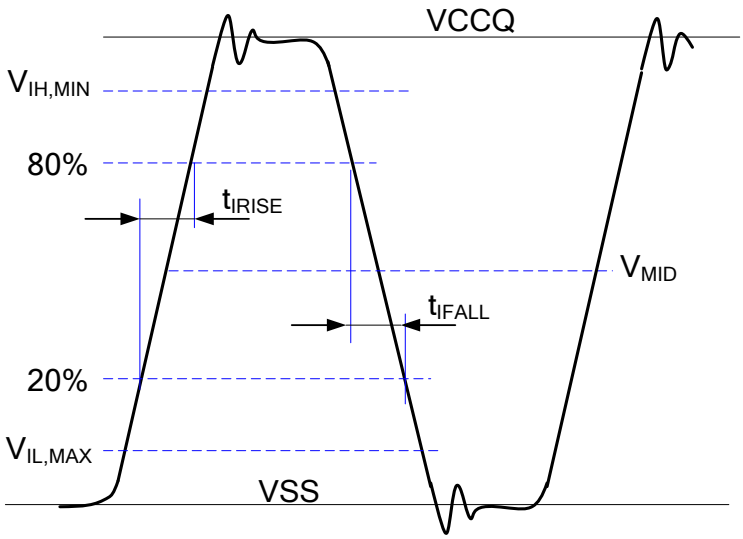


Figure 6.4 — Clock Input Levels, Rise Time, and Fall Time

6.4.1 HS Gear Rates

Table 6.7 defines the data rate values for the two rate series with respect REF_CLK frequency value (fref).

Table 6.7 — HS-BURST Rates

HS-GEAR	Rate A-series	Rate B-series		Unit
	f_{ref}	f_{ref}		
	19.2 / 26.0 / 38.4 / 52.0	19.2 / 38.4	26.0 / 52.0	MHz
HS-GEAR1	1248 (2)	1459.2	1456.0	Mbps
HS-GEAR2	2496	2918.4	2912.0	Mbps
HS-GEAR3	4992	5836.8	5824.0	Mbps
HS-GEAR4	9984	11673.6	11648.0	Mbps
HS-GEAR5	19968	23347.2	23296.0	Mbps
HS-GEAR6	39936	46694.4	46592.0	Mbps

NOTE 1 "Mbps" indicates 1,000,000 bits per second.

NOTE 2 1248 Mbps with fref = 38.4 MHz may be obtained using a prescaler:

fref * M /P, where M = 65 (PLL multiplier), P = 2 (Prescaler).

NOTE 3 In a system that supports HS-G5 or above, the REF_CLK frequency shall be 38.4 MHz or 52 MHz.

6.4.2 Host Controller Requirements for Reference Clock Generation

Table 6.8 — 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.

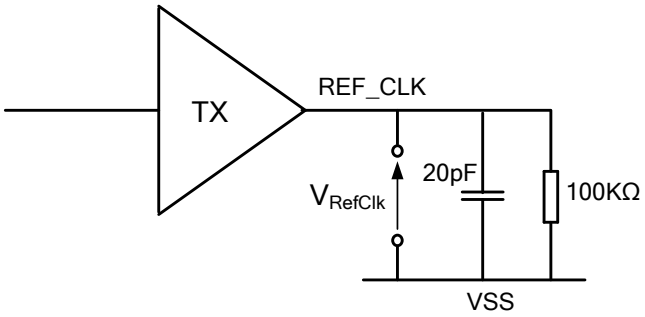


Figure 6.5 — Test Load Impedance

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.

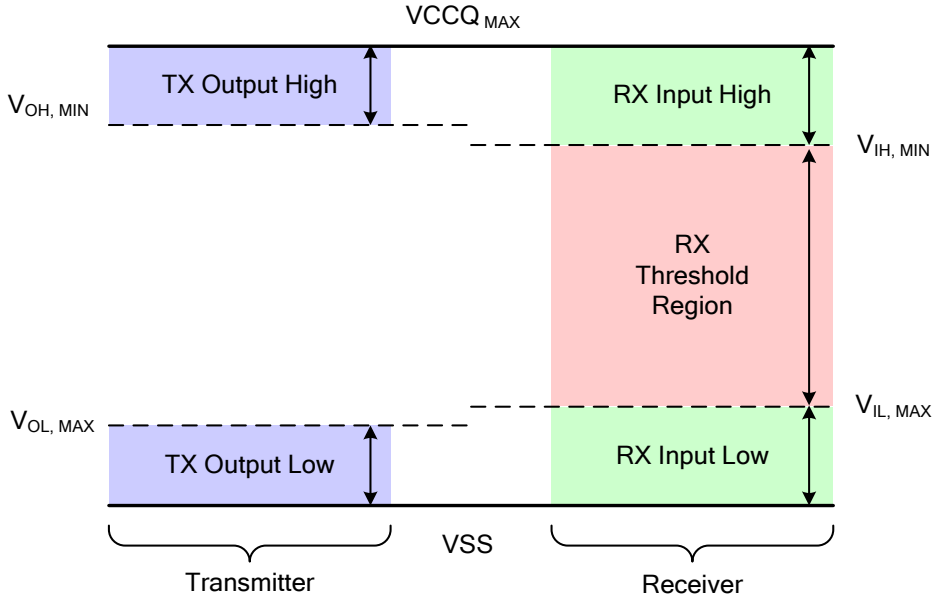


Figure 6.6 — Output Driver and Input Receiver Levels

6.4.2 Host Controller Requirements for Reference Clock Generation (cont'd)

Table 6.8 — 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 $V_{OH,MIN}$.

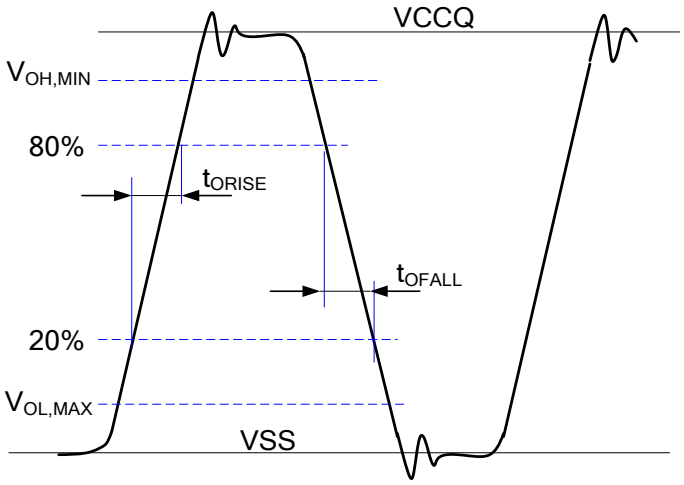


Figure 6.7 — Clock Output Levels, Rise Time and Fall Time

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.

6.5 External Charge Pump Capacitors (Optional)

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

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

Figure 6.1 shows the electrical connections required in case of charge pump implementation that uses external capacitors. Table 6.9 provides description of the capacitors to be used.

Table 6.9 — Charge Pump Capacitors Description

Capacitor Name	Min	Typ	Max	Description
C_{CP-IN}	Implementation Specific	4.7 µF	Implementation Specific	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}		4.7 µF		Charge pump output bypass capacitor
C_{CP}		0.22 µF		Charge pump flying capacitor

The charge pump capacitors are optional for UFS devices.

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

Table 6.10 specifies name and description of the balls for connecting external charge pump capacitors.

Table 6.10 — Charge Pump Related Ball Names

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

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

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

6.6 ZQ Calibration Resistor (optional)

In order to produce memory devices that can accommodate high performance, ZQ calibration resistors may be required.

A ZQ calibration resistor requires a passive component. An optional usage of external ZQ calibration resistors is provided.

Figure 6.1 shows an example of the electrical connections needed if the ZQ calibration implementation uses external resistors. Table 6.11 provides a description of the resistors to be used.

Table 6.11 — ZQ Calibration Resistors Description

Resistor Name	Type	Nominal	Accuracy	Description
R_{ZQ1}	Input	240 Ω	±1%	Optional resistor for device internal calibration
R_{ZQ2}	Input	300 Ω	±1%	Optional resistor for NAND interface's calibration

6.7 Absolute Maximum DC Ratings and Operating Conditions

Stresses greater than those listed in Table 6.12 may cause permanent damage to the device. This is a stress rating only, and functional operation of the device at these or any other conditions above those indicated in the operational sections of this standard is not implied. Exposure to absolute maximum rating conditions for extended periods may affect reliability.

Table 6.12 — Absolute Maximum DC ratings and Operating Conditions

Parameter	Symbol	Min	Max	Unit	Notes
Voltage on M-PHY signals		- 0.2	0.9	V	1
Voltage on REF_CLK, RST_n signals		- 0.2	1.5	V	1
VCC supply voltage	VCC	- 0.3	3.2	V	1
VCCQ supply voltage	VCCQ	- 0.2	1.5	V	1
VCCQ_M supply voltage	VCCQ_M	- 0.2	1.5	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.

6.8 AC and DC Operating Conditions

Table 6.13 shows the DC and AC voltage operating conditions for UFS.

Table 6.13 —AC and DC Voltage Operating Conditions

Symbol	Low Freq Voltage Range (2) (DC to 2 MHz)			
UFS	Min	Nominal	Max	Unit
VCC(1)	2.4	2.5	2.7	V
VCCQ(1)	1.14	1.2	1.26	V
VCCQ2(1)	1.70	1.8	1.95	V
VCCQ_M(3)	1.14	1.2	1.26	V

NOTE 1 At greater than 2 MHz, the AC noise shall be less than +/- 3% of the nominal voltage measured at any UFS device VCC/VCCQ/VCCQ2 power ball. This AC noise specification is aligned to the AC noise spec of the JEDEC NAND standard [JESD230].

NOTE 2 The operating voltage, including DC noise and AC noise, shall be between the Min and Max specified voltage values measured at any UFS device VCC/VCCQ/VCCQ_M/VCCQ2 power ball. Refer to Figure 6.8 illustrating that DC noise and AC noise shall be within the Min and Max specified voltage range.

NOTE 3 At greater than 2 MHz the AC noise shall be less than +/- 1.5% of the nominal voltage measured at any UFS device VCCQ_M power ball. This AC noise specification is aligned to the power stability needs of HS-Gear6 signaling [MIPI-M-PHY].

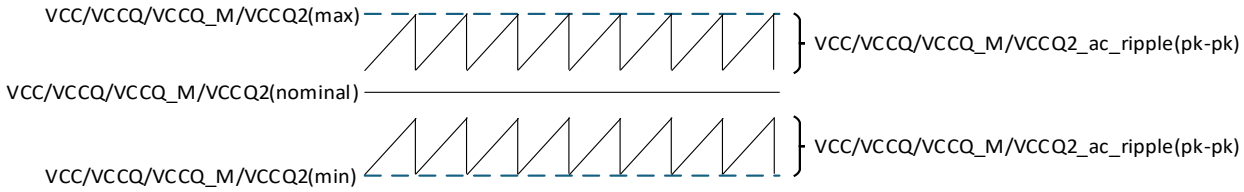


Figure 6.8 — DC and AC voltage Range for VCC/VCCQ/VCCQ_M /VCCQ2

7 Reset, Power-Up And Power-Down

7.1 Reset

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

7.1.1 Power-on Reset

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

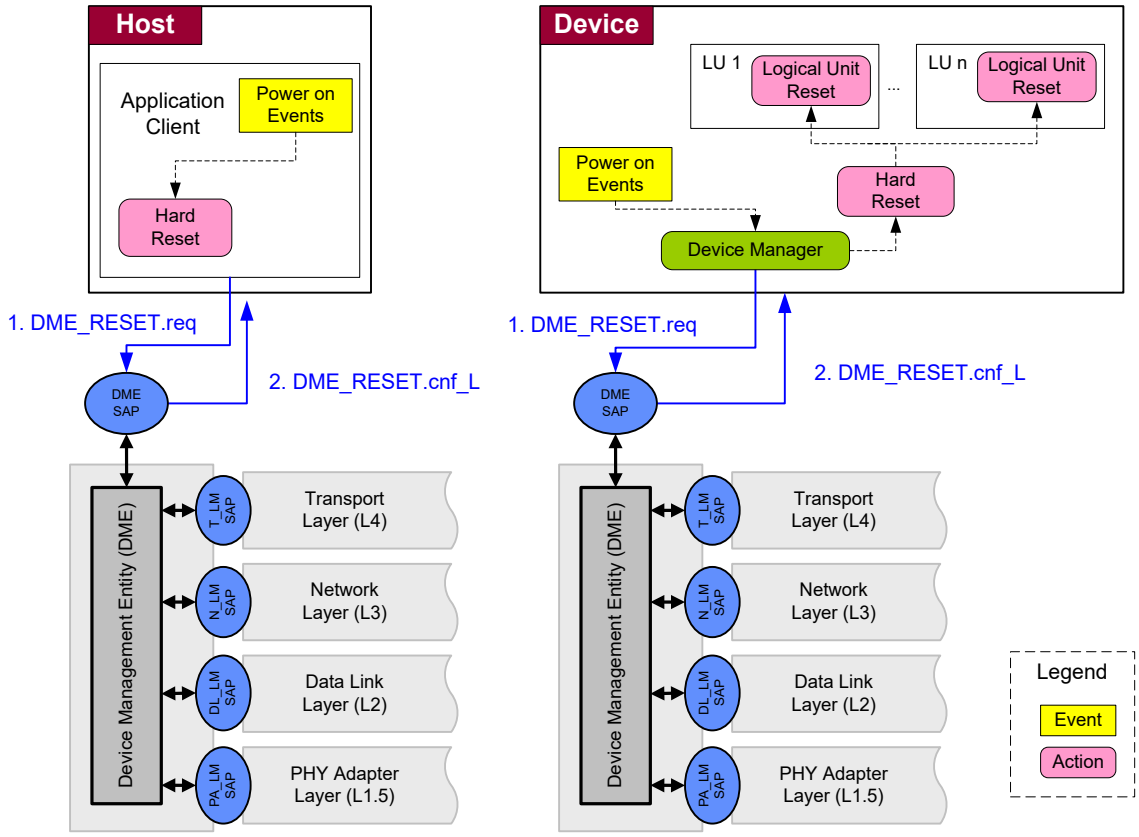


Figure 7.1 — Power-on Reset

7.1.2 Hardware Reset

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

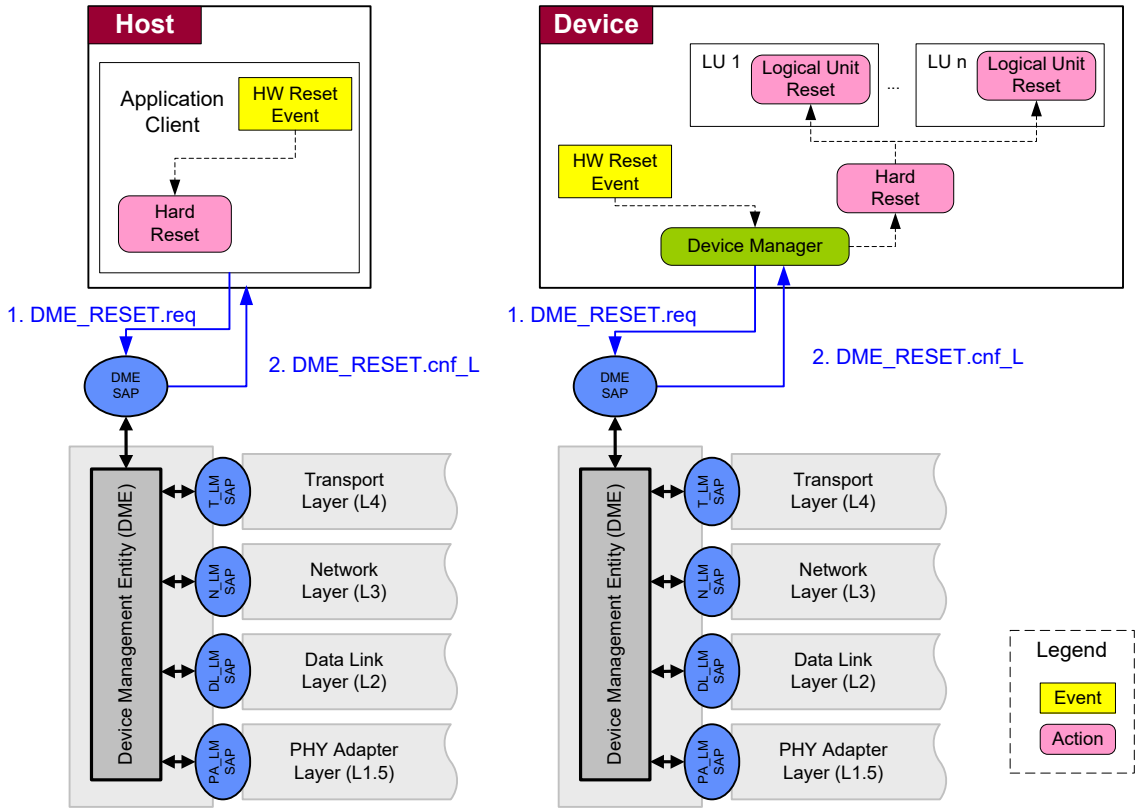


Figure 7.2 — Hardware Reset

Figure 7.3 shows the hardware reset AC timings.

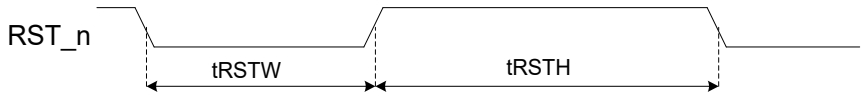


Figure 7.3 — Reset AC timings

Table 7.1 — Reset timing parameters

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

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

7.1.3 EndPointReset

The EndPointReset feature is defined in [MIPI-UniPro].

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 is expected to ignore the reception of an EndPointReset from a UFS device.

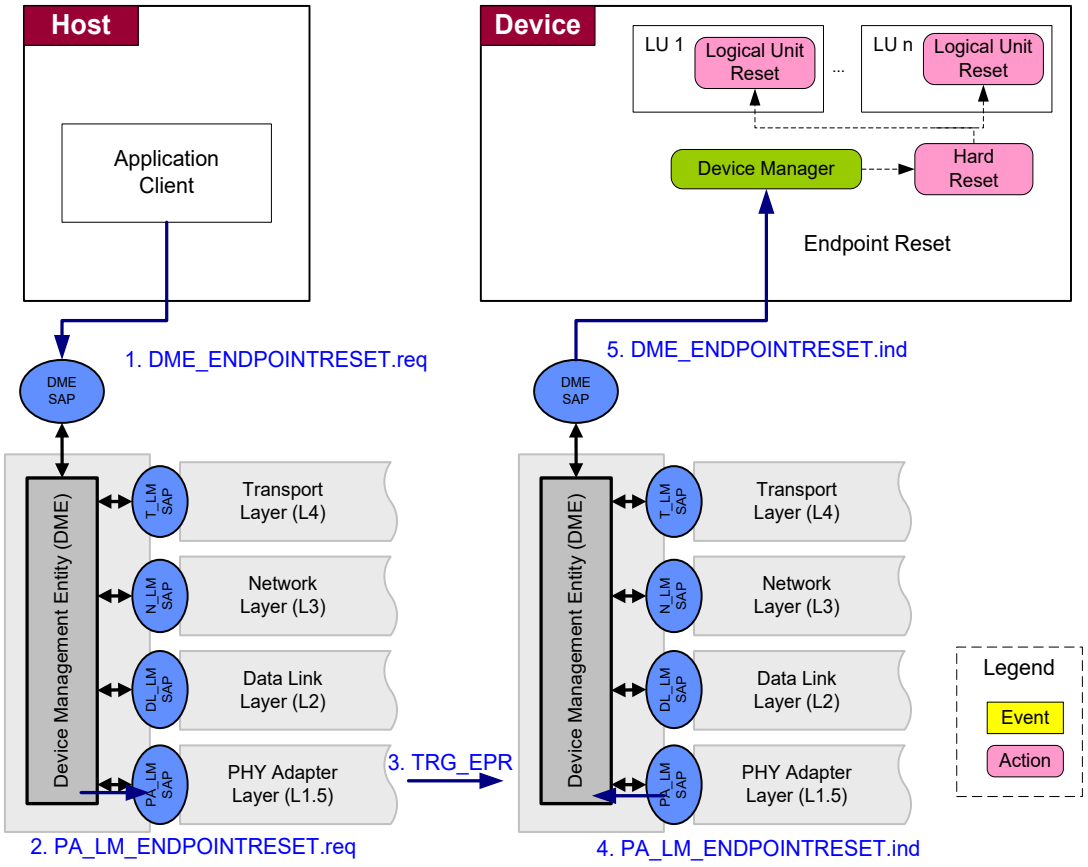


Figure 7.4 — EndPointReset

7.1.4 Logical Unit Reset

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

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

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

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

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

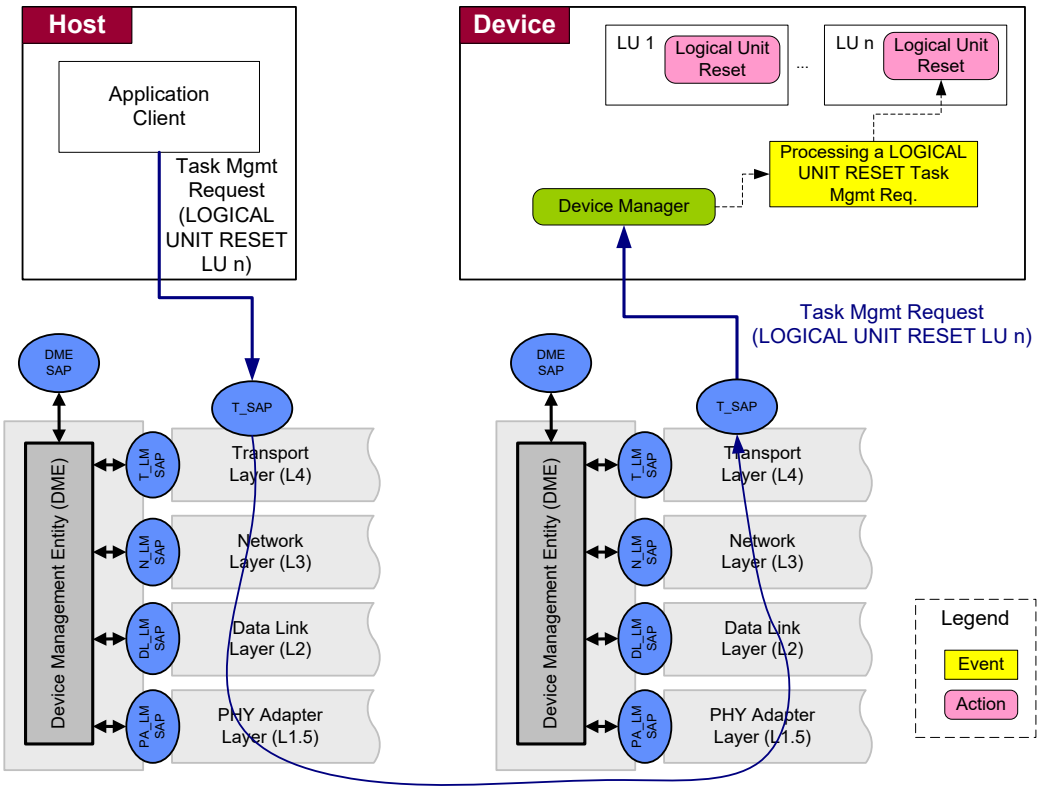


Figure 7.5 — Logical Unit Reset

7.1.5 Host UniPro Warm Reset

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

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

7.1.6 Summary of Resets and Device Behavior

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

Table 7.2 — Reset States

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

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

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

NOTE 1 See Table 14.25 and Table 14.27 for the definition of Flags and Attributes write access properties.

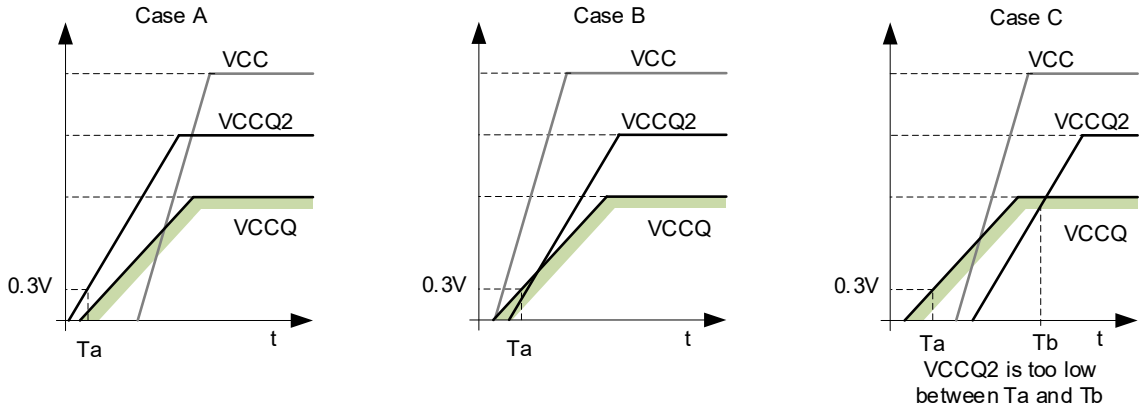
NOTE 2 Values of Attributes and Flags with “Write once” or “Persistent” access property are kept after power cycle or any type of reset event.

7.2 Power Up Ramp

During power up, VCC, VCCQ, VCCQ_M, and VCCQ2 should be applied as described in the following:

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

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



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

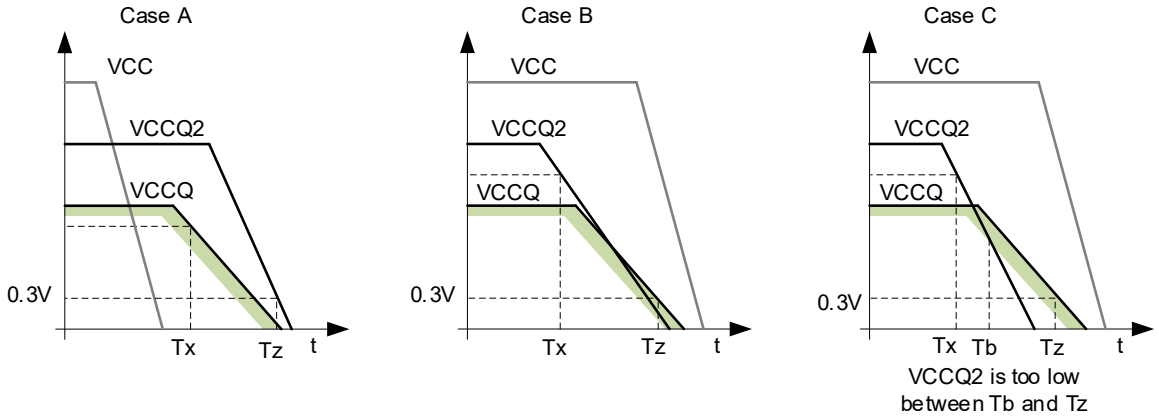
Figure 7.6 — Power Up Ramps

7.3 Power Off Ramp

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

- Tx is the point where VCCQ, VCCQ_M, or VCCQ2 decreases under its minimum operating condition value specified (see Table 6.4).
- Tz is the point where VCCQ, VCCQ_M, and VCCQ2 are below 300 mV.
- VCCQ2 and VCCQ_M should be greater than VCCQ - 200 mV between Tx and Tz.
- VCC can be ramped down independently from VCCQ, VCCQ_M, and VCCQ2.
- While powering off the device, RST_n signal and REF_CLK signal should be between VSS and VCCQ.

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



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

Figure 7.7 — Power Off Ramps

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

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

7.4 UFS Device Power Modes and LU Power Condition

7.4.1 Device Power Modes

The device supports multiple power modes, which are controlled by the START STOP UNIT command (see [SBC]) and additional attributes. The device power mode is independent of the bus state of the upstream or downstream links, which are controlled independently.

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

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

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

Power Condition (PC) values to transition between UFS power modes are as described in 7.4.4.

7.4.1.1 Active Power Mode

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

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

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

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

7.4.1.1 Active Power Mode (cont'd)

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

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

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

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

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

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

The following power mode may be: Idle, Pre-Sleep, or Pre-PowerDown.
All supported commands are available in Active Mode.

7.4.1.2 Idle Power Mode

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

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

7.4.1.3 Pre-Active Power Mode

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

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

7.4.1.3 Pre-Active Power Mode (cont'd)

While in Pre-Active power mode:

a) the Device well known logical unit may successfully complete only: START STOP UNIT command and REQUEST SENSE command; other commands may be terminated with CHECK CONDITION status, with the sense key set to NOT READY, with the additional sense code set to LOGICAL UNIT IS IN PROCESS OF BECOMING READY, see 7.4.4 for further details;

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

7.4.1.4 UFS-Sleep Power Mode

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

VCC power supply may be turned off in this power mode.The UFS-Sleep power mode is entered from Pre-Sleep power mode.

While in UFS-Sleep power mode:

a) the Device well known logical unit may successfully complete only: START STOP UNIT command and REQUEST SENSE command; other commands may be terminated with CHECK CONDITION status, with the sense key set to NOT READY and the additional sense code set to LOGICAL UNIT NOT READY, INITIALIZING COMMAND REQUIRED, see 7.4.4 for further details;

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

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

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

7.4.1.5 Pre-Sleep Power Mode

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

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

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

7.4.1.5 Pre-Sleep Power Mode (cont'd)

While in Pre-Sleep power mode:

a) the Device well known logical unit may successfully complete only: START STOP UNIT command, REQUEST SENSE command and task management functions; other commands may be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, see 7.4.4 for further details;

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

7.4.1.6 UFS-PowerDown Power Mode

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

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

While in UFS-PowerDown power mode:

a) the Device well known logical unit may successfully complete only: START STOP UNIT command and REQUEST SENSE command; other commands may be terminated with CHECK CONDITION status, with the sense key set to NOT READY and the additional sense code set to LOGICAL UNIT NOT READY, INITIALIZING COMMAND REQUIRED, see 7.4.4 for further details;

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

7.4.1.7 Pre-PowerDown Power Mode

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

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

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

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

7.4.1.7 Pre-PowerDown Power Mode (cont'd)

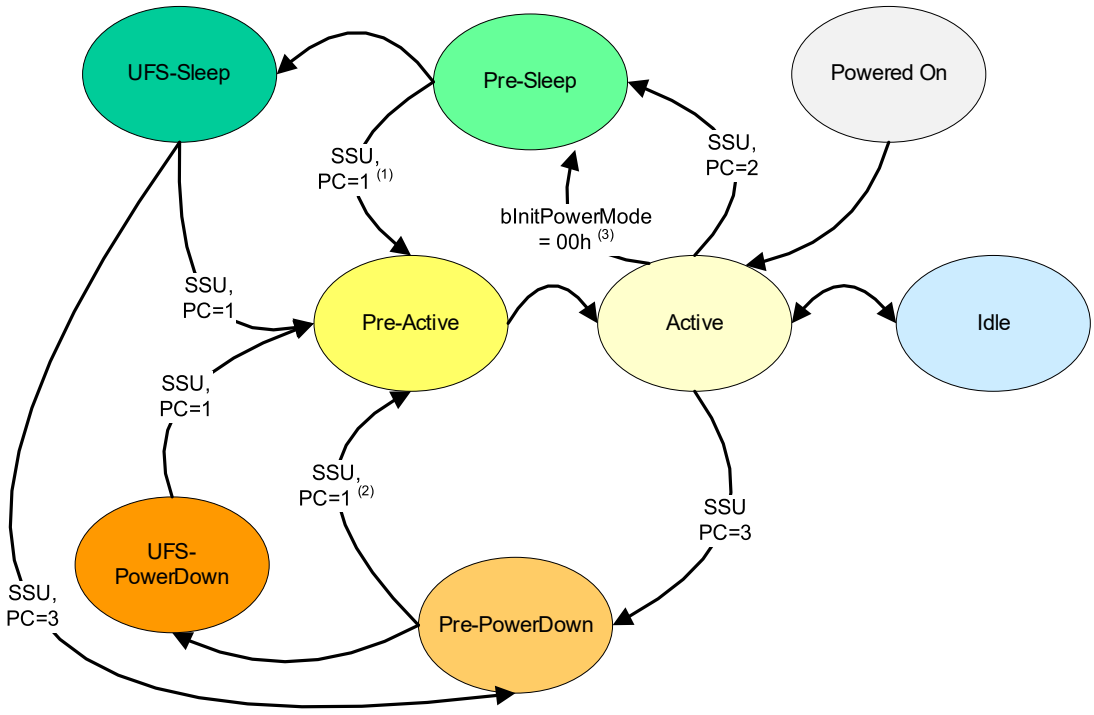
While in Pre-PowerDown power mode:

a) the Device well known logical unit may successfully complete only: START STOP UNIT command, REQUEST SENSE command and task management functions; other commands may be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, see 7.4.4 for further details;

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

7.4.1.8 Power Mode State Machine

The relationship amongst the different power modes is shown in Figure 7.8.



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

Figure 7.8 — Power Mode State Machine

7.4.1.8.1 Transitions from Powered On Power Mode

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

Transition from Powered_On to Active
This transition shall occur when the device is ready to begin power on initialization.

7.4.1.8.2 Transitions from Pre-Active Power Mode

Transition from Pre-Active to Active
This transition shall occur when the device meets the requirements for being in Active power mode.

7.4.1.8.3 Transitions from Active Power Mode

Transition from Active to Idle
This transition may occur when the device completes any ongoing operations.

Transition from Active to Pre-Sleep
This transition shall occur

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

Transition from Active to Pre-PowerDown
This transition shall occur if the device server processes a START STOP UNIT command with the POWER CONDITION field set to 3h.

7.4.1.8.4 Transitions from Idle Power Mode

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

7.4.1.8.5 Transitions from Pre-Sleep Power Mode

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

Transition from Pre-Sleep to Sleep
This transition shall occur when the device meets the requirements for being in Sleep power mode.

7.4.1.8.6 Transitions from UFS-Sleep Power Mode

Transition from UFS-Sleep to Pre-Active
This transition shall occur if the device server processes a START STOP UNIT command with the POWER CONDITION field set to 1h.

Transition from UFS-Sleep to Pre-PowerDown
This transition shall occur if the device server processes a START STOP UNIT command with the POWER CONDITION field set to 3h.

7.4.1.8.7 Transitions from Pre-PowerDown Power Mode

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

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

7.4.1.8.8 Transitions from UFS-PowerDown Power Mode

Transition from UFS-PowerDown to Pre-Active
This transition shall occur if the device server processes a START STOP UNIT command with the POWER CONDITION field set to 1h.

7.4.1.8.9 SCSI Command and UPIU Transactions

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

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

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

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

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

7.4.1.8.9 SCSI Command and UPIU Transactions (cont'd)

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

Table 7.4 — Allowed SCSI Commands and UPIU for Each Power Mode

Power Mode	SCSI Commands		UPIU Transactions
	Device well known logical unit	Other logical units	
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-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

7.4.1.9 Responses to SCSI Commands

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

Table 7.5 — Device Well Known Logical Unit Responses to SSU Command

Current Power Mode	PC	STATUS	SENSE KEY	ASC, ASCQ
Pre-Active	1h	GOOD (1)	-	-
	Others	CHECK CONDITION	NOT READY	LOGICAL UNIT NOT READY, START STOP UNIT COMMAND IN PROGRESS
Active	1h, 2h, 3h, 4h	GOOD (1)	-	-
	Others	CHECK CONDITION	ILLEGAL REQUEST	INVALID FIELD IN CDB
Pre-Sleep	2h	GOOD (1)	-	-
	Others	CHECK CONDITION	NOT READY	LOGICAL UNIT NOT READY, START STOP UNIT COMMAND IN PROGRESS
UFS-Sleep	1h, 2h, 3h, 4h	GOOD (1)	-	-
	Others	CHECK CONDITION	ILLEGAL REQUEST	INVALID FIELD IN CDB
Pre-PowerDown	3h	GOOD (1)	-	-
	Others	CHECK CONDITION	NOT READY	LOGICAL UNIT NOT READY, START STOP UNIT COMMAND IN PROGRESS
UFS-PowerDown	1h, 3h	GOOD (1)	-	-
	Others	CHECK CONDITION	ILLEGAL REQUEST	INVALID FIELD IN CDB

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

7.4.1.9 Responses to SCSI Commands (cont'd)

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

Table 7.6 — Device Well Known Logical Unit Responses to Commands Other Than SSU

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

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

Table 7.7 — Pollable Sense Data for Each Power Modes

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

7.4.2 Power Management Command: START STOP UNIT

The START STOP UNIT command is defined in [SBC].

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

Table 7.8 — START STOP UNIT Fields

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

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

7.4.3 Power Mode Control

Table 7.9 defines a series of attributes used to control the active current levels and power modes.

Table 7.9 — Attribute for Power Mode Control

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

Table 7.10 shows the Device Descriptor parameters that specify the power mode and the Active ICC level after power on or reset. See 14.1.5.2 for details about device configuration.

Table 7.10 — Device Descriptor Parameters

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

7.4.3 Power Mode control (cont'd)

Table 7.11 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 7.12.

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 7.11 — Power Parameters Descriptor Fields

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

Table 7.12 — Format for Power Parameter Element

Field Name	Bit Range	Description
Unit	bit [15:14]	00b:nA 01b:uA 10b:mA 11b: A
-	bit [13:12]	Reserved (00b)
Value	bit [11: 0]	The maximum current expected in each current consumption level

7.4.4 Logical Unit Power Condition

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

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

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

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

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

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

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

Table 7.13 defines the logical unit responses to SCSI commands for various device power modes, assuming that the logical unit is in active power condition. See 7.4.4 for details about Device well known logical unit.

Table 7.13 — Logical Unit Response to SCSI Command

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

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

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

8 UFS UIC Layer: MIPI M-PHY

8.1 Termination

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

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

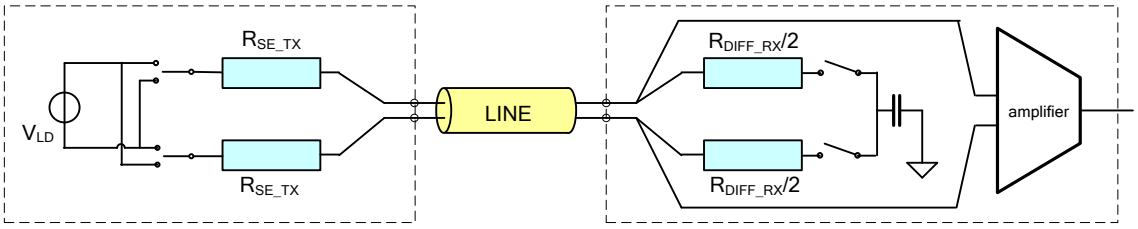


Figure 8.1 — Simplified Example for I/O Termination

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

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

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

8.2 Drive Levels

[MIPI-M-PHY] defines one drive amplitude, large amplitude (LA). The UFS interface utilizes large amplitude (LA).

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

SUBLINKS in a LINK shall communicate with same amplitude.

8.3 PHY State Machine

The UFS interface shall implement the Type I state machine.

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

8.4 HS Burst

A UFS device shall support the HS-GEAR1, HS-GEAR2, HS-GEAR3, HS-GEAR4, HS-GEAR5, and HS-GEAR6. The supported gears are indicated in the Capability Attribute Table 8.1 and Table 8.2.

SUBLINKS in a LINK may communicate with different HS-GEAR or with PWM-GEAR1.

HS-GEAR1 is active after power up or reset, when the LSS pin is high.

8.4.1 HS Prepare Length Control

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

8.4.2 HS Sync Length Control

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

8.5 PWM Burst

A UFS device shall support the PWM-G1. Other PWM gears are not supported. The supported PWM-GEARS are indicated in the Capability Attributes Table 8.1 and Table 8.2.

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

PWM-GEAR1 is active after power up or reset, when the LSS pin is low.

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

8.5.1 LS Prepare Length Control

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

8.6 Adapt

The MIPI M-PHY versions 4.1 and above support a new Adapt sequence that is used to train the filter characteristics of its equalizers in the M-RX module according to the channel. UFS devices shall and UFS hosts are expected to implement the equalizer and the M-TX shall be able to provide the Adapt sequence to its M-RX counterpart if needed.

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

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

8.7 UFS PHY Attributes

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

Table 8.1 — UFS PHY M-TX Capability Attributes(0)

Attribute Name	Attribute ID	Range defined in [MIPI-M-PHY]	UFS Value	Notes
TX_HSGEAR_Capability	0x02	HS_G1_ONLY = 1, HS_G1_TO_G2 = 2, HS_G1_TO_G3 = 3, HS_G1_TO_G3 = 4, HS_G1_TO_G5 = 5, HS_G1_TO_G6 = 6	6	1
TX_PWMGEAR_Capability	0x04	PWM_G1_ONLY = 1	1	2
TX_HS_Unterminated_LINE_Drive_Capability	0x07	0 = No, 1 = Yes	0	
TX_LS_Terminated_LINE_Drive_Capability	0x08	0 = No, 1 = Yes	1	3
TX_Hibern8Time_Capability	0x0F	1 to 128	1	4

NOTE 0 Attributes with the same values as in [MIPI-M-PHY] are not listed in this table. Refer to [MIPI-M-PHY] for other attributes not listed in this table.

NOTE 1 All HS gears from HS-GEAR1 to TX_HSGEAR_Capability shall be supported.

NOTE 2 Only PWM-GEAR1 shall be supported.

NOTE 3 A UFS device shall support terminated LS burst.

NOTE 4 Time defined in 0.1 msec steps.

8.7 UFS PHY Attributes (cont'd)

Table 8.2 — UFS PHY M-RX Capability Attributes(0)

Attribute Name	Attribute ID	Range defined in [MIPI-M-PHY]	UFS Value	Notes
RX_HSGEAR_Capability	0x82	HS_G1_ONLY = 1, HS_G1_TO_G2 = 2, HS_G1_TO_G3 = 3, HS_G1_TO_G4 = 4, HS_G1_TO_G5 = 5, HS_G1_TO_G6 = 6	6	1
RX_PWMGEAR_Capability	0x84	PWM_G1_ONLY = 1	1	2
RX_Hibern8Time_Capability	0x92	1 to 128	1	3
RX_HS_Equalizer_Setting_Capability	0x9D	B[0] = 0: No equalization B[0] = 1: Yes equalization	1	
RX_EYEMON_Capability	0xF1	For bit[0]: FALSE = 0, TRUE = 1	bit[0] = 1	4

NOTE 0 Attributes with the same values as in [MIPI-M-PHY] are not listed in this table. Refer to [MIPI-M-PHY] for other attributes not listed in this table.

NOTE 1 All HS gears from HS-GEAR1 to RX_HSGEAR_Capability shall be supported.

NOTE 2 Only PWM-GEAR1 shall be supported.

NOTE 3 Time defined in 0.1 msec steps.

NOTE 4 Specifies support for eye monitor function. UFS devices shall and UFS hosts are expected to support RX_EYEMON_capability. The rest of the associated attributes for EYEMON shall be taken from MPHY specification.

8.8 Electrical Characteristics

8.8.1 Transmitter Characteristics

As defined in [MIPI-M-PHY].

8.8.2 Receiver Characteristics

As defined in [MIPI-M-PHY].

9 UFS UIC Layer: MIPI Unipro

9.1 Overview

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

9.2 Architectural Model

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

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

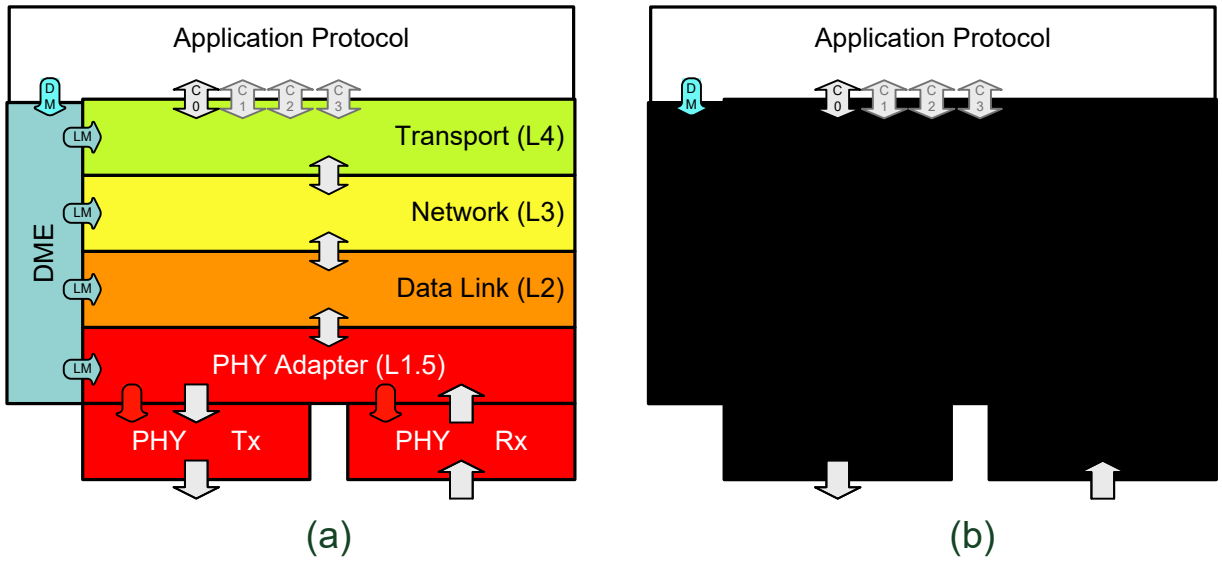


Figure 9.1 — UniPro Internal Layering View (a) and UniPro Black Box View (b)

9.3 UniPro/UFS Transport Protocol Interface (Data Plane)

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

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

The T_CO_SAP provides the following core data transfer service primitives (see [MIPI-UniPro]):

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

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

- T_CO_DATA.cnf_L(L4CPortResultCode)
  - Issued by UniPro to report the result of a Message (Fragment) transfer request
- T_CO_DATA.ind(MessageFragment, EOM, SOM, MsgStatus)
  - Issued by UniPro to deliver a received Message (Fragment) towards the service user
  - EOM informs the service user that this is the last Message Fragment (EndOfMessage)
  - SOM informs the service user that this is the first Message Fragment (StartOfMessage)
- T_CO_DATA.rsp_L( )
  - Issued by a service user of UniPro to report readiness to receive the next Message (Fragment)

9.3.1 Flow Control

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

9.3.2 Object Sizes

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

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

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

9.4 UniPro/UFS Control Interface (Control Plane)

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

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

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

DME Control Primitives

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

NOTE After exit from Hibernate all UniPro Transport Layer attributes (including L4 T_PeerDeviceID, L4 T_PeerCPortID, L4 T_ConnectionState, etc.) will be reset to their reset values. All required attributes must be restored properly on both ends before communication can resume.

- DME_POWERMODE
  - Allows to change the power mode of one or both directions of the M-PHY Link
- DME_TEST_MODE (optional)
  - Allows to set the peer UniPro Device on the Link in a specific test mode
- DME_LINKLOST
  - Indication of the UniPro stack towards higher layers that the Link has been lost
- DME_ERROR
  - Indication of the UniPro stack towards higher layers that an error condition has been encountered in one of the UniPro Layers

9.5 UniPro/UFS Transport Protocol Address Mapping

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

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

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

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

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

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

The Nexus (I_T_L_Q) is composed of:

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

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

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

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

- UFS Initiator Port Identifier (I) shall be 20 bits wide and UFS Target Port Identifier (T) shall be 20 bits wide and
    - UFS Initiator/Target Port Identifier shall contain the UniPro Network Layer Device ID of the entity (host or device) containing said UFS Port
        - The UniPro Network Layer Device ID reset value shall be 0 for the Host
        - The UniPro Network Layer Device ID reset value shall be 1 for the Device
    - UFS Initiator/Target Port Identifier contains the UniPro Transport Layer CPort ID which said UFS Port uses to communicate to the remote entity
        - The UniPro Transport Layer CPort ID reset value shall be 0 for the Host
        - The UniPro Transport Layer CPort ID reset value shall be 0 for the Device
- UFS Initiator Port Identifier shall contain the Initiator ID field according to Table 9.1.

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

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

Table 9.1 — UFS Initiator and Target Port Identifiers

UFS Port	UFS Port IDs																			
	19	18	17	16	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					Initiator ID							
													EXT_IID				IID			
Target Port Identifier (T)	Device ID = 000 0001b							CPort ID = 0 0000b					0000b				0000b			

The Initiator ID is comprised of two fields in the UPIU (as depicted in the figure above), IID as the least significant nibble and EXT_IID as the most significant nibble.

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

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

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

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

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

9.6 Options and Tunable Parameters of UniPro

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

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

9.6.1 UniPro PHY Adapter

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

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

9.7.1 UniPro PHY Adapter (cont'd)

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

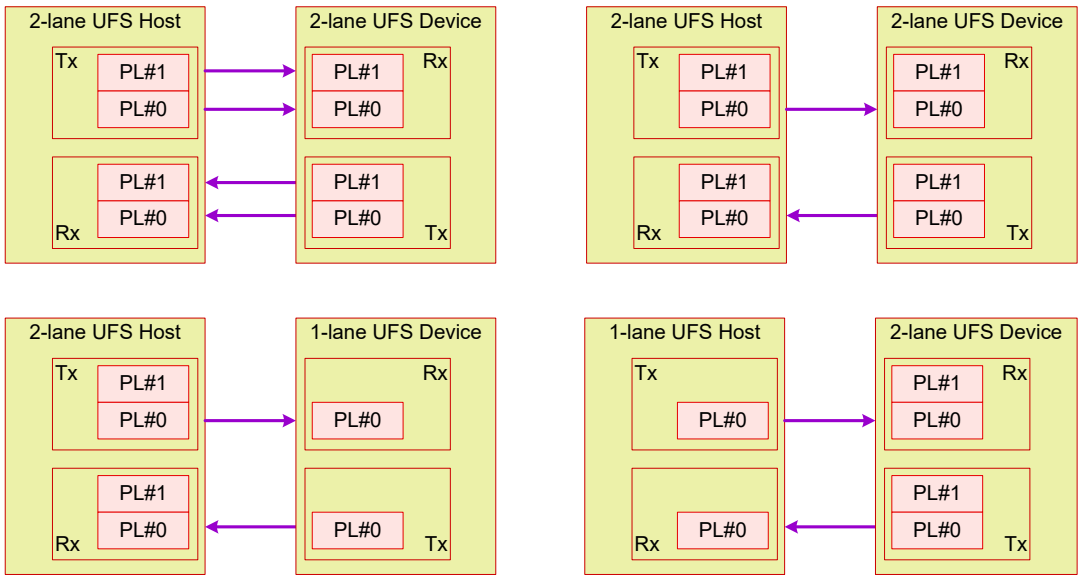


Figure 9.2 — Physical Lane Connections

9.6.2 UniPro Data Link Layer

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

9.6.3 UniPro Network Layer

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

9.6.4 UniPro Transport Layer

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

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

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

9.6.5 UniPro Device Management Entity Transport Layer

DME service primitives provide the means to

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

UFS Hosts are expected to and UFS Devices shall implement the following DME service primitives;

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

UFS Hosts

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

UFS Devices

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

9.6.6 UniPro Attributes

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

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

10 UFS Transport Protocol (UTP) Layer

10.1 Overview

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

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

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

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

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

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

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

10.2 UTP and UniPro Specific Overview

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

The UFS transactions will be grouped into data structures called UFS Protocol Information Units (UPIU). There are UPIUs defined for UFS SCSI commands, responses, data in and data out, task management, utility functions, vendor functions, transaction synchronization and control. The list is extensible for future additions.

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

10.2.1 Phases

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

10.2.2 Data Pacing

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

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

10.2.3 UniPro

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

10.3 UFS Transport Protocol Transactions Overview

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

10.3 UFS Transport Protocol Transactions Overview (cont'd)

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

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

The maximum UPIU size is defined as being 65600 bytes.

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

10.4 Service Delivery Subsystem

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

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

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

10.5 UPIU Transactions

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

Table 10.1 — UPIU Transaction Codes

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

10.5 UPIU Transactions (cont'd)

Table 10.2 — UPIU Transaction Code Definitions

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

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

10.6 General UFS Protocol Information Unit Format

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

Table 10.3 — General Format of the UFS Protocol Information Unit

General UPIU Format				
0 Transaction Type		1 Flags	2 LUN	3 Task Tag
4 IID	Command Set Type	5 EXT_IID / Query Function / Task Manag. Function	6 Response	7 EXT_IID / Status
8 Total EHS Length		9 Device Information	10 (MSB) Data Segment Length	11 (LSB)
12		13	14	15
16				19
20		Transaction Specific Fields		23
24				27
28				31
i		i+1	i+2	i+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 shall be supported only for COMMAND UPIU and RESPONSE UPIU, therefore the Total EHS Length may be set to a value greater than zero in COMMAND UPIU and RESPONSE UPIU, while it shall be set to zero in all other UPIU.

NOTE 2 EXT_IID is defined in byte 7 for host to device UPIUs (COMMAND, DATA OUT, TASK MANAGEMENT REQUEST). EXT_IID is defined in byte 5 for device to host UPIUs (RESPONSE, DATA IN, RTT, TASK MANAGEMENT RESPONSE, REJECT).

10.6.1 Overview

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

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

10.6.2 Basic Header Format

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

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

Table 10.4 — Basic Header Format

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

The basic header formats are defined as follows:

a) Transaction Type

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

Table 10.5 — Transaction Type Format

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

10.6.2 Basic Header Format (cont'd)

b) HD

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

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

c) DD

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

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

d) Transaction Code

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

e) Flags

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

Table 10.6 — UPIU Flags

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

NOTE 1 "-" denotes reserved values.
NOTE 2 CP = Command Priority

See the individual UPIU definitions in 10.7 for the meanings of the Table 10.6 encodings.

10.6.2 Basic Header Format (cont'd)

f) Response

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

Table 10.7 — UTP Response Values

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

g) Status

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

h) Reserved

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

i) LUN

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

j) Task Tag

The Task Tag is generated by the Initiator device when creating a task request. This field will be maintained by the Initiator device and Target device for all UPIU transactions relating to a single task. The Initiator device will contain a register or variable that represents the Task Tag value. The Initiator device will generate unique Task Tag by incrementing the internal variable when creating a new task request. When a task request is made up of or generates a series of UPIU transactions, all UPIU will contain the same value in the Task Tag field. In particular, the same Task Tag value shall be maintained for the UPIU grouped in each row of Table 10.8.

Table 10.8 — UPIU Associated to a Single Task

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

10.6.2 Basic Header Format (cont'd)

k) IID, EXT_IID (Nexus Initiator ID)

The Initiator ID field is 8 bits wide, comprised of the EXT_IID and IID fields as shown in Table 10.9.

Table 10.9 — Initiator ID Composition

Initiator ID							
7	6	5	4	3	2	1	0
EXT_IID [7:4] of byte 7 for host to device UPIUs [7:4] of byte 5 for device to host UPIUs				IID [7:4] of byte 4			

- IID
  - The IID field is 4 bits wide, encoded in bits [7:4] of byte 4. This field indicates identity fo the Initiator device which created the task request.
- EXT_IID
  - The EXT_IID field is 4 bits wide, encoded in bits [7:4] of byte 7 for host to device UPIUs (COMMAND, DATA OUT, TASK MANAGEMENT REQUEST)
  - The EXT_IID field is 4 bits wide, encoded in bits [7:4] of byte 5 for device to host UPIUs (RESPONSE, DATA IN, READY TO TRANSFER, TASK MANAGEMENT RESPONSE, REJECT)

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

UFS devices shall support all 256 Initiator ID values (when bEXTIIDEn = 01h) or 16 Initiator ID values (when bEXTIIDEn = 00h). The Initiator ID shall be encoded in this field by the Host when creating a request. This field is maintained by the Initiator device and Target device for all UPIU transactions relating to the same task.

l) Command Set Type

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

Table 10.10 — Command Set Type

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

10.6.2 Basic Header Format (cont'd)

m) Query Function, Task Manag. Function

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

n) Device Information

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

o) Total Extra Header Segment Length

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

$$Total\ EHS\ Length\ value = INTEGER\left(\frac{Total\ Extra\ Header\ Segment\ Bytes + 31}{32}\right)$$

The maximum size of all EHS fields combined is 96 bytes. A value of zero in this field indicates that there are no EHS contained within the UPIU. Therefore the valid value of this field shall be one of the following: 0, 1, 2 or 3.

p) Data Segment Length

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

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

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

q) Transaction Specific Fields

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

10.6.2 Basic Header Format (cont'd)

r) Extra Header Segments

The Extra Header Segments exist if the Total EHS Length field contains a non-zero value. For UTP, this area will start at byte address 32 within the UPIU. The UPIU may contain zero or more EHS. The length of each Extra Header Segment shall be a multiple of 32 bytes.The Extra Header Segments contain zero or more EHS Entry of the following format. In this version of the standard, up to one EHS entry shall be allowed.

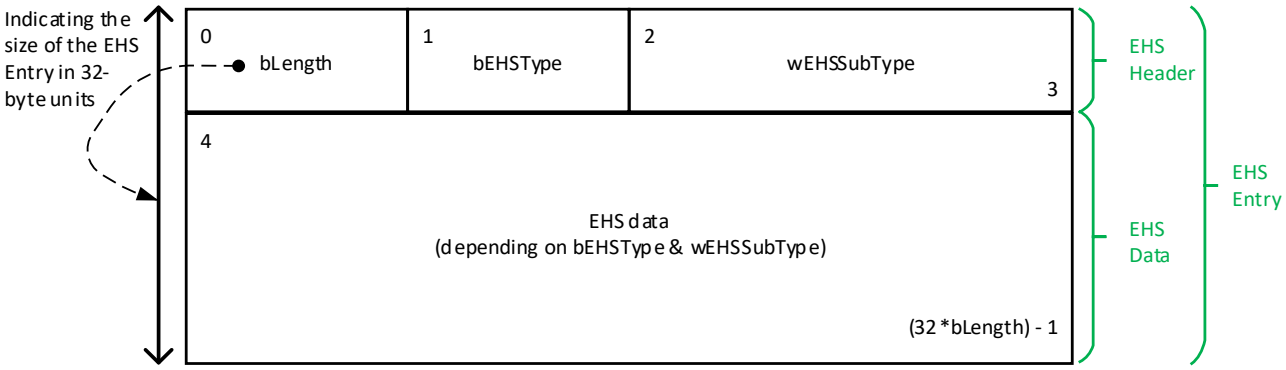


Figure 10.1 – EHS Entry Format

Table 10.11 describes the EHS Entry data format that gives detailed field information.

Table 10.11 — EHS Entry Format

Byte	Name	Description
0	bLength	Size of the EHS Entry in 32 byte units
1	bEHSType	EHS Type 00h : Reserved 01h: Advanced RPMB 02h-7Fh : Reserved 80h-FFh : Vendor specific
2:3	wEHSSubType	EHS Sub Type
4:(32*bLength)-1	EHS data	EHS Data Data in this field depends on the EHS Type and EHS Sub Type

10.6.2 Basic Header Format (cont'd)

s) Data Segment

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

10.7 UFS Protocol Information Units

This sub-clause provides the details of each UFS Protocol Information Unit.

10.7.1 COMMAND UPIU

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

Table 10.12 — COMMAND UPIU

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

10.7.1.1 Basic Header

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

a) Transaction Type

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

b) Flags

Table 10.13 describes the flags used in COMMAND UPIU.

Table 10.13 — Flags definition for COMMAND UPIU

<table>
<tr><th>Flag</th><th>Description</th></tr>
<tr><td>Flags.R</td><td>A value of ‘1’ in the .R flag indicates that the command requires a data transfer (incoming data) from Target device to Initiator device. If .R is set to ‘1’ then .W shall be set to ‘0’. If .R and .W are set to ‘0’ then no data transfer is required for this command and the Expected Data Transfer Length field is ignored.</td></tr>
<tr><td>Flags.W</td><td>A value of ‘1’ in the .W flag indicates that the command requires a data transfer (outgoing data) from Initiator device to Target device. If .W is set to ‘1’ then .R shall be set to ‘0’. If .W and .R are set to ‘0’ then no data transfer is required for this command and the Expected Data Transfer Length field is ignored.</td></tr>
<tr><td>Flags.ATTR</td><td>The .ATTR field contains the task attribute value as defined by [SAM].
ATTR Definition
<table>
<tr><th>Task Attribute</th><th>Bit 1</th><th>Bit 0</th></tr>
<tr><td>Simple</td><td>0</td><td>0</td></tr>
<tr><td>Ordered</td><td>0</td><td>1</td></tr>
<tr><td>Head of Queue</td><td>1</td><td>0</td></tr>
<tr><td>ACA (Not Used)</td><td>1</td><td>1</td></tr>
</table>
The relative order of execution between Head of Queue commands is left for implementation.</td></tr>
<tr><td>Flags.CP</td><td>The .CP field indicates the Command Priority; see [SAM] for details. In UFS, the .CP field supports only two values whereas [SAM] allows a larger range.
This 1-bit field specifies the relative scheduling importance of a command having a Simple task attribute in relation to other commands having Simple task attributes already in the task set. If the command has a task attribute other than Simple then this field has no meaning
A task manager may use command priority to determine an ordering to process commands with the Simple task attribute within the task set.
A value of “1” indicates high priority. A value of “0” indicates no priority.</td></tr>
<tr><td colspan="2">NOTE The bit assignment of the Flags field is shown in Table 10.6.</td></tr>
</table>

10.7.1.1 Basic header (cont'd)

c) Data Segment Length

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

d) Expected Data Transfer Length

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

For a data transfer from the Initiator device to the Target device, the .W flag shall be set to '1' and the .R flag shall be set to '0'. The value in the Expected Data Transfer Length field represents the size of the data in the number of bytes, excluding retransmits, that the Initiator device expects to send to the Target device. Upon completion of the transfer, the target device informs the initiator how many bytes have been processed (excluding retransmits) via the residual count.

For a data transfer from the Target device to the Initiator device, the .R flag shall be set to '1' and the .W flag shall be set to '0'. The value in the Expected Data Transfer Length field represents the size of the data in the number of bytes, excluding retransmits, that the Initiator device expects to receive from the Target device. Upon completion of the transfer, the target device informs the initiator how many bytes have been processed (excluding retransmits) via the residual count.

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

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

The RESPONSE UPIU is used to complete the requested task initiated by the COMMAND UPIU.

NOTE: Before the RESPONSE UPIU is sent to the host, some portion or all data may be retransmitted to update the data sent by previous DATA IN UPIUs or DATA OUT UPIUs if the out of order feature for DATA IN UPIU or DATA OUT UPIU is enabled (bOutOf OrderDataEn = 01h, 02h or 03h). See READY TO TRANSFER UPIU, DATA OUT UPIU, and DATA IN UPIU for more details.

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

10.7.1.1 Basic header (cont'd)

e) CDB

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

f) IID

This field is the LSB nibble of the Initiator ID nexus, as described in bullet k) of the Basic Header Format. The Initiator ID nexus indicates the identity of the Initiator device who created the task request. See Initiator ID description in 10.6.2 for details.

g) EXT_IID

This field is the MSB nibble of the Initiator ID nexus, as described in bullet k) of the Basic Header Format. The Initiator ID nexus indicates the identity of the Initiator device who created the task request. See Initiator ID description in 10.6.2 for details.

h) Command Set Type

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

10.7.2 RESPONSE UPIU

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

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

10.7.2 RESPONSE UPIU (cont'd)

Table 10.14 — RESPONSE UPIU

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

10.7.2.1 Basic Header

The first 12 bytes of the RESPONSE UPIU contain the Basic Header as described in 10.6.2. Details follow:

a) Transaction Type

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

b) Flags

Table 10.15 describes the flags used in RESPONSE UPIU.

Table 10.15 — Flags Definition for RESPONSE UPIU

Flag	Description
Flags.O	The Flags.O flag will be set to '1' to indicate that a data overflow occurred during the task execution: the Target device has more data bytes to transfer than the Initiator device requested. The Residual Transfer Count field will indicate the number of available bytes not transferred from the Target device to the Initiator device or vice versa. The Residual Transfer Count will be set to the value difference of the total number of bytes available to be transferred and the Expected Data Transfer Length value received in the COMMAND UPIU. See "j) Residual Transfer Count" for further explanation.
Flags.U	The Flags.U flag will be set to '1' to indicate that a data underflow occurred during the task execution: the Target device has less data bytes to transfer than the Initiator device requested. The Residual Transfer Count field will indicate the number of bytes that were not transferred from the Target device to the Initiator device or vice versa. The Residual Transfer Count will be set to the value difference of the Expected Data Transfer Length value received in the COMMAND UPIU and the actual number of bytes transferred. See "j) Residual Transfer Count" for further explanation.
Flags.D	The .D flag will be set to '1' to indicate that a UTP Data Out Mismatch error occurred during the task execution: the data buffer offset and/or the data transfer count parameter in the Data Out UPIU doesn't match the corresponding parameters in the RTT request. See 10.7.13 for further explanation.
NOTE The bit assignment of the Flags field is shown in Table 10.6.	

c) IID

This field is the LSB nibble of the Initiator ID nexus, as described in bullet k) of the Basic Header Format. The Initiator ID nexus indicates the identity of the Initiator device who created the task request. See Initiator ID description in 10.6.2 for details.

d) EXT_IID

This field is the MSB nibble of the Initiator ID nexus, as described in bullet k) of the Basic Header Format. The Initiator ID nexus indicates the identity of the Initiator device who created the task request. See Initiator ID description in 10.6.2 for details.

e) Command Set Type

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

10.7.2.1 Basic Header (cont'd)

f) Response

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

g) Status

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

1) SCSI Command Set Status

When the Command Set Type field indicates SCSI Command Set the Status field will contain the standard [SAM] defined SCSI status value. Possible values are listed in the table in 10.7.2.1 h) on the next page. See [SAM] for detailed status code definitions.

UFS devices shall not return Sense Data when a command completes with GOOD status.

UFS devices shall return Sense Data in the Data segment when a command completes with CHECK CONDITION status.

UFS devices may or may not return Sense Data for other status codes. A non-zero value in the Data Segment Length field indicates that such a UPIU contains Sense Data in the Data Segment area.

UFS devices shall support the following status codes:

- GOOD
- CHECK CONDITION
- BUSY
- TASK SET FULL

UFS devices may support the RESERVATION CONFLICT status code.

Other status codes are not applicable to UFS devices.

10.7.2.1 Basic Header (cont'd)

h) Device Information

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

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

Bit[0] and bit[5:2] of the Device Information field are defined, and bit[1] is reserved for HPB (Host Performance 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
bit[5:2]	FAST_RECOVERY_NEEDED	Fast recovery request Device requests host action after listed delay: 0h: No reset requested. 1h: HW reset with no delay. 2h: HW reset after a 1 second delay. 3h: HW reset after a 2 second delay. 4h: HW reset after a 3 second delay. 5h: HW reset after a 4 second delay. 6h: HW reset after a 5 second delay. 7h: HW reset after a 6 second delay. 8h: HW reset after a 7 second delay. 9h: HW reset after a 8 second delay. Ah: HW reset after a 9 second delay. Bh: HW reset after a 10 second delay. Ch: HW reset after a 11 second delay. Dh: HW reset after a 12 second delay. Eh: HW reset after a 13 second delay. Fh: HW reset after a 14 second delay.
Others	Reserved	

The exception sources include: background operations, dynamic capacity, system data pool, etc. See 13.4.12, Exception Events Mechanism, for details.

10.7.2.1 Basic Header (cont'd)

i) Data Segment Length

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

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

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

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

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

j) Residual Transfer Count

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

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

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

Table 10.16 — Flags and Residual Count Relationship

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

10.7.2.1 Basic Header (cont'd)

k) Sense Data Fields

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

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

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

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

l) SCSI Sense Data Fields

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

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

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

10.7.2.1 Basic Header (cont'd)

m) Sense Data Length

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

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

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

n) SCSI Sense Data Length

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

o) Sense Data Format

UFS devices report error information using fixed format sense data as defined in [SPC].

The following fields shall be set as follows when the device reports sense data as part of the data area of a RESPONSE UPIU:

- VALID, RESPONSE CODE, SENSE KEY, INFORMATION, ADDITIONAL SENSE CODE, and ADDITIONAL SENSE CODE QUALIFIER fields shall be set as described in [SPC]
- ADDITIONAL SENSE LENGTH field shall be set to ten
- All other fields shall be set to zero

p) Sense Key

Sense Keys categorize errors as described in [SPC]. Refer to that standard for their encodings and usages, including their relationship with the Additional Sense Code and Additional Sense Code Qualifier fields.

10.7.3 DATA OUT UPIU

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

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

Table 10.17 — DATA OUT UPIU

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

10.7.3.1 Basic Header

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

a) Transaction Type

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

b) Flags

Table 10.18 describes the flags used in DATA OUT UPIU.

Table 10.18 — Flags Definition for DATA OUT UPIU

Flag	Description
Flags.T	A value of '1' in Flags.T indicates that all data in this UPIU shall be retransmitted data resulting from a READY TO TRANSFER UPIU with Flags.T set to '1'.
NOTE The bit assignment of the Flags field is shown in Table 10.6.	

c) IID

This field is the LSB nibble of the Initiator ID nexus, as described in bullet k) of the Basic Header Format. The Initiator ID nexus indicates the identity of the Initiator device who created the task request. See Initiator ID description in 10.6.2 for details.

d) EXT_IID

This field is the MSB nibble of the Initiator ID nexus, as described in bullet k) of the Basic Header Format. The Initiator ID nexus indicates the identity of the Initiator device who created the task request. See Initiator ID description in 10.6.2 for details.

e) Data Segment Length

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

10.7.3.1 Basic Header (cont'd)

f) Data Buffer Offset

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

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

NOTE Out of order sequencing of DATA OUT UPIU packets will only occur if a UFS device supports it (bDataOrdering = 01h or bDataOrdering = 03h) and if this feature is enabled (bOutOfOrderDataEn = 01h or bOutOfOrderDataEn = 03h).

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

g) Data Transfer Count

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

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

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

10.7.3.1 Basic Header (cont'd)

h) Data Segment

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

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

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

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

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

i) Data Out Transfer Example

Figure 10.2 shows an example of data transfer from the Initiator device to the Target device. In particular, the command processing requires the transfer of 577-byte data. The data transfer is done out of order: at the beginning the middle portion of the data, then the last portion and finally the first portion.

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

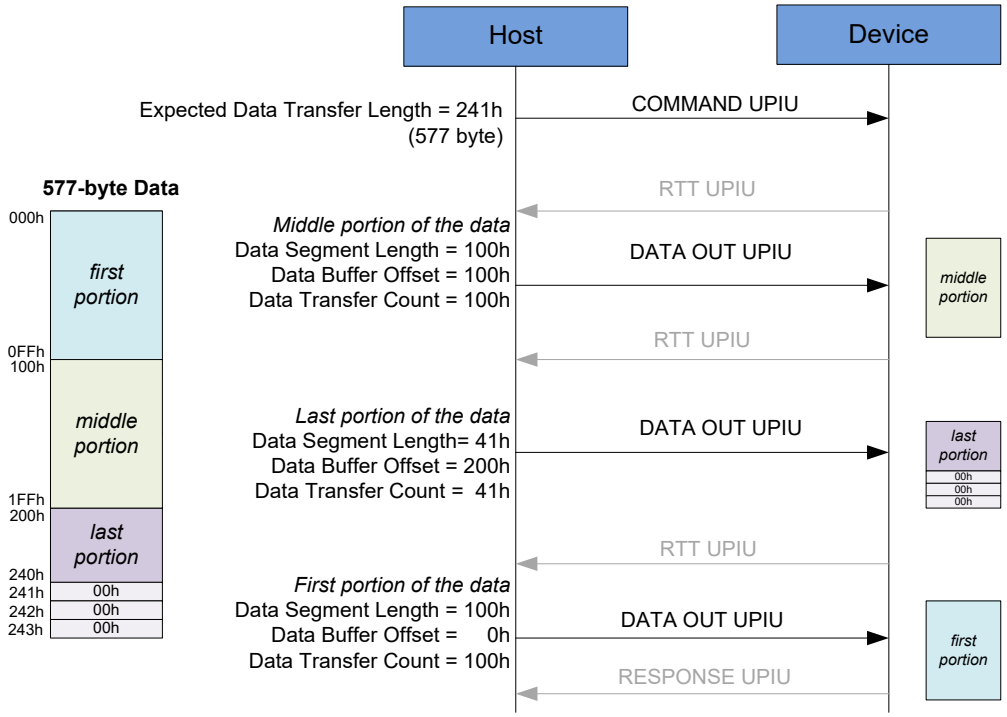


Figure 10.2 — Data Out Transfer Example

10.7.4 DATA IN UPIU

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

Table 10.19 —DATA IN UPIU

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

10.7.4.1 Basic Header

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

a) Transaction Type

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

b) Flags

Table 10.20 describes the flags used in DATA IN UPIU.

Table 10.20 — Flags Definition for DATA IN UPIU

Flag	Description
Flags.T	A value of '1' in Flags.T indicates that all data in this UPIU shall be retransmitted data. See "f) Data Buffer Offset" for further explanation.
NOTE The bit assignment of the Flags field is shown in Table 10.6.	

c) IID

This field is the LSB nibble of the Initiator ID nexus, as described in bullet k) of the Basic Header Format. The Initiator ID nexus indicates the identity of the Initiator device who created the task request. See Initiator ID description in 10.6.2 for details.

d) EXT_IID

This field is the MSB nibble of the Initiator ID nexus, as described in bullet k) of the Basic Header Format. The Initiator ID nexus indicates the identity of the Initiator device who created the task request. See Initiator ID description in 10.6.2 for details.

e) Data Segment Length

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

10.7.4.1 Basic Header (cont'd)

f) Data Buffer Offset

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

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

To retransmit data for a task, the device shall set Data Buffer Offset appropriately and shall set Flags.T. All data in this UPIU shall be retransmitted data.

NOTE: A UFS device may set the Data Buffer Offset to retransmit only if out of order sequencing of DATA IN UPIU packets is enabled (bOutOfOrderDataEn = 01h or bOutOfOrderDataEn = 02h).

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

g) Data Transfer Count

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

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

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

10.7.4.1 Basic Header (cont'd)

h) Data Segment

This is the Data Segment area that contains the data payload. The maximum data payload size that can be transferred within a single DATA IN UPIU packet is 65535 bytes.

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

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

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

i) HintControl

Bit0: This field indicates the validity of the Hint fields provided in the DATA IN UPIU (HintIID, HintLUN, HintTaskTag, Hint Data Buffer Offset, Hint Data Count). When this field is set to 0b, the Hint fields are not valid and expected to be ignored by the host controller. When this field is set to 1b, the Hint fields are valid.
When the feature is not supported or is disabled (bDataOrdering = 00h or bOutOfOrderDataEn = 00h), or the Hint fields (HintTaskTag, etc…) do not refer to a READ(6) / READ (10) / READ (16) / HPB Read command - HintControl shall be set to 0x0 and hint information is not provided.

Other bits: reserved.

j) HintIID [3:0]

This field indicates the IID of DATA IN UPIUs that the device will transfer. This field is valid only when HintControl is set to 1b. HintIID field may be different than IID field.

k) HintEXT_IID [7:4]

This field indicates the EXT_IID of DATA IN UPIUs that the device will transfer. This field is valid only when HintControl is set to 1b. HintEXT_IID field may be different than EXT_IID field.

l) HintLUN

This field indicates the LUN of DATA IN UPIUs that the device will transfer. This field is valid only when HintControl is set to 1b. HintLUN field may be different than LUN field.

m) HintTaskTag

This field indicates the Task Tag of DATA IN UPIUs that the device will transfer. This field is valid only when HintControl is set to 1b. HintTaskTag field may be different than TaskTag field.

n) Hint Data Buffer Offset

This field indicates the Data Buffer Offset in a DATA IN UPIUs that the device will transfer. This field is valid only when HintControl is set to 1b.

10.7.4.1 Basic Header (cont'd)

o) Hint Data Count

This field indicates the number of 4 KB that the Target device is expected to transfer to the Initiator starting from Hint Data Buffer Offset field. The value 0 indicates 4 KB data. The value 1 indicates 8 KB data, and so on.

The device may interleave DATA IN UPIUs pertaining to all hints provided by the device. This field is valid only when HintControl is set to 1b.

p) Data In Transfer Examples

Figure 10.3 shows an example of data transfer from the Target device to the Initiator device. In particular, during the command processing 577-byte data is sent to the Initiator device. The data transfer is done in sequence: at the beginning the first portion, then the middle portion and finally the last portion.

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

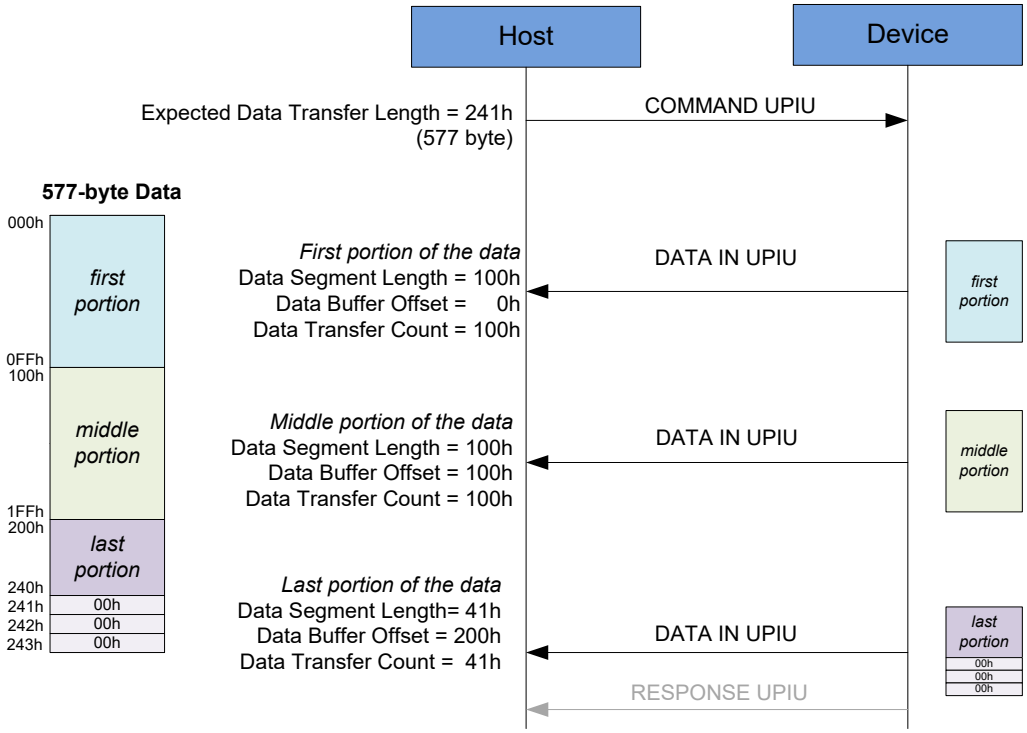


Figure 10.3 — Data In Transfer Example

10.7.4.1 Basic Header (cont'd)

Figure 10.4 shows and example of data transfer for a READ Command when some portion of the data sent using a previous DATA IN UPIU needs to be updated before sending the RESPONSE UPIU. In this example, the device sends a DATA IN UPIU with an appropriate Data Buffer Offset to update that data.

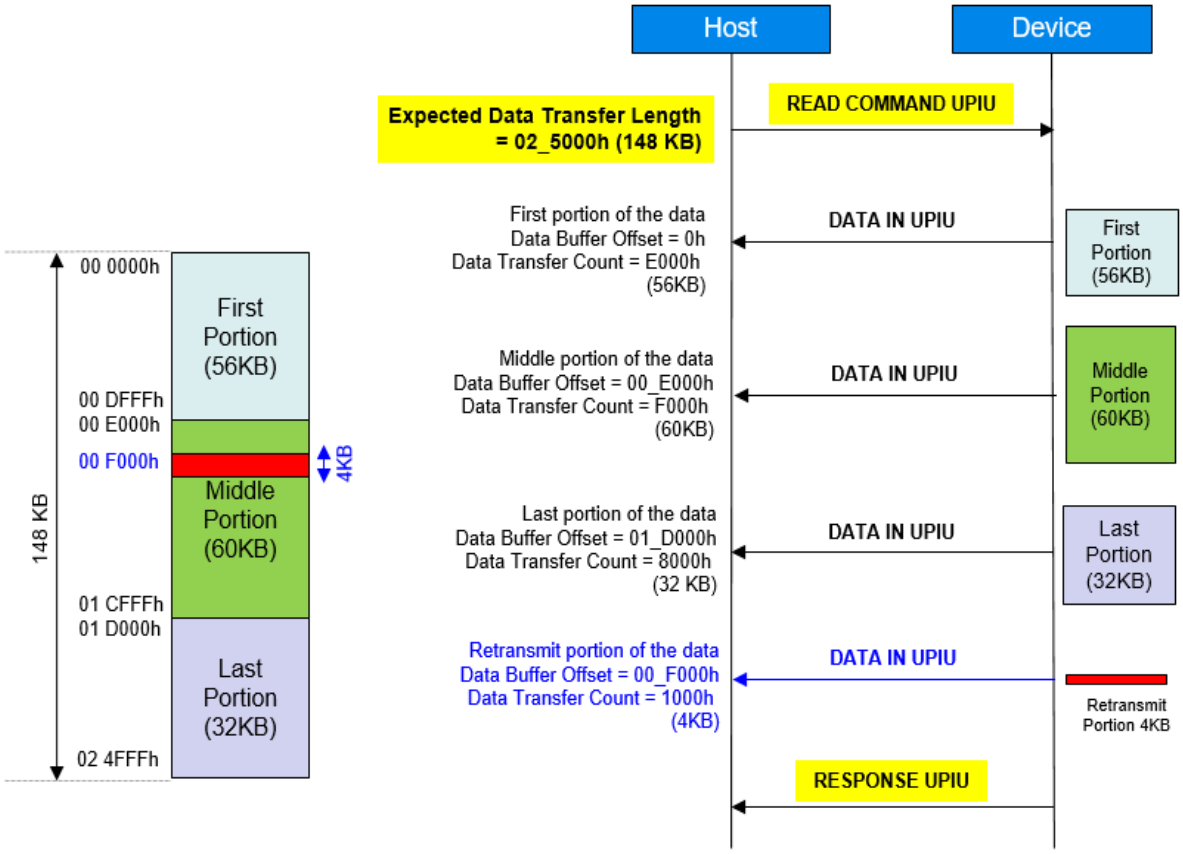


Figure 10.4 — DATA IN UPIU with Retransmission Sequence Example

10.7.5 READY TO TRANSFER UPIU

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

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

See 10.7.13, Data Out Transfer Rules, for further details about Initiator device to Target device data transfer.

Table 10.21 — READY TO TRANSFER UPIU

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

10.7.5.1 Basic Header

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

a) Transaction Type

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

b) Flags

Table 10.22 describes the flags used in READY TO TRANSFER UPIU.

Table 10.22 — Flags Definition for RTT UPIU

Flag	Description
Flags.T	A value of '1' in Flags.T indicates that all requested data shall be retransmitted data. See "f) Data Buffer Offset" for further explanation.
NOTE The bit assignment of the Flags field is shown in Table 10.6.	

c) IID

This field is the LSB nibble of the Initiator ID nexus, as described in bullet k) of the Basic Header Format. The Initiator ID nexus indicates the identity of the Initiator device who created the task request. See Initiator ID description in 10.6.2 for details.

d) EXT_IID

This field is the MSB nibble of the Initiator ID nexus, as described in bullet k) of the Basic Header Format. The Initiator ID nexus indicates the identity of the Initiator device who created the task request. See Initiator ID description in 10.6.2 for details.

e) Data Segment Length

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

10.7.5.1 Basic Header (cont'd)

f) Data Buffer Offset

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

The Data Buffer Offset shall be an integer multiple of four and shall be less than Expected Data Transfer Length.

To request data retransmission for a task, the device shall set Data Buffer Offset appropriately and shall set Flags.T. All data in the resulting DATA OUT UPIU shall be retransmitted data.

NOTE UFS device may set the Data Buffer Offset to retransmit the all or some part of the data that was transferred already by previous READY TO TRANSFER UPIUs and their corresponding DATA OUT UPIUs for the command only if out of order sequencing for DATA OUT UPIUs is enabled (bDataOrdering = 01h or bDataOrdering = 03h).

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

g) Data Transfer Count

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

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

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

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

10.7.5.1 Basic Header (cont'd)

h) HintControl

Bit0: This field indicates the validity of the Hint fields provided in the UPIU (HintIID, HintLUN, HintTaskTag, Hint Data Buffer Offset, Hint Data Count). When this field is set to 0b, the Hint fields are not valid and expected to be ignored by the host controller. When this field is set to 1b, the Hint fields are valid

When the feature is not supported or is disabled (bDataOrdering = 00h or bOutOfOrderDataEn = 00h), or the Hint fields (HintTaskTag, etc…) do not refer to a WRITE(6) / WRITE (10) / WRITE (16) command - HintControl shall be set to 0x0 and hint information is not provided.

The hint only provide RTT information, which is independent with max no. of RTT.

Other bits: reserved.

i) HintIID [3:0]

This field indicates the IID of RTT UPIUs that the device will transfer. This field is valid only when HintControl is set to 1b. HintIID field may be different than IID field.

j) HintEXT_IID [7:4]

This field indicates the EXT_IID of RTT UPIUs that the device will transfer. This field is valid only when HintControl is set to 1b. HintEXT_IID field may be different than EXT_IID field.

k) HintLUN

This field indicates the LUN of RTT UPIUs that the device will transfer. This field is valid only when HintControl is set to 1b. HintLUN field may be different than LUN field.

l) HintTaskTag

This field indicates the Task Tag of RTT UPIUs that the device will transfer. This field is valid only when HintControl is set to 1b. HintTaskTag field may be different than TaskTag field.

m) Hint Data Buffer Offset

This field indicates the Data Buffer Offset in RTT UPIUs that the device will transfer. This field is valid only when HintControl is set to 1b.

n) Hint Data Count

This field indicates the number of 4 KB that the Initiator device is expected to transfer to the Target starting from Hint Data Buffer Offset field. The value 0 indicates 4 KB data. The value 1 indicates 8 KB data, and so on.

The device may interleave RTT UPIUs pertaining to all hints provided by the device.

This field is valid only when HintControl is set to 1b.

10.7.5.1 Basic Header (cont'd)

o) READY TO TRANSFER UPIU Sequence Examples

Figure 10.5 shows an example of READY TO TRANSFER UPIU sequence. In particular, during the command processing the Target device requests the Initiator device to send a total amount of 577-byte data. The data transfer is done out of order (reverse): at the beginning the last portion, then the middle portion and finally the first portion.

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

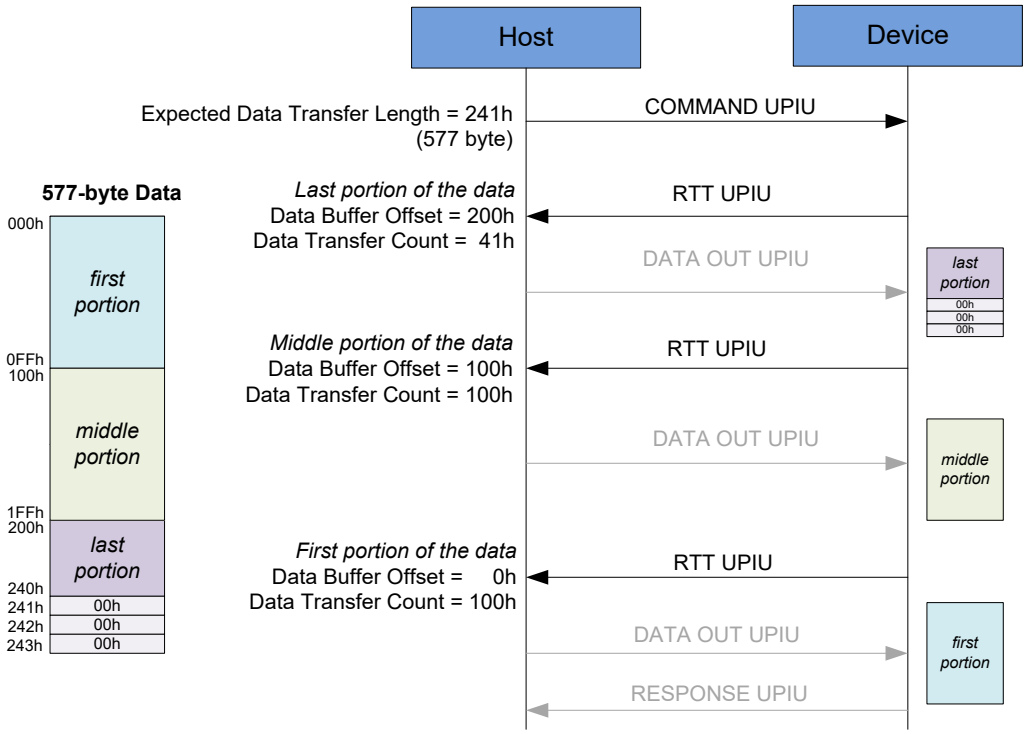


Figure 10.5 — READY TO TRANSFER UPIU Sequence Example

10.7.5.1 Basic Header (cont'd)

Figure 10.6 shows and example of a READY TO TRANSFER UPIU sequence for a WRITE Command when some data sent using a previous DATA OUT UPIU needs to be updated. In this example the device sends an RTT UPIU with an appropriate Data Buffer Offset and Data Transfer Count for the data to be updated.

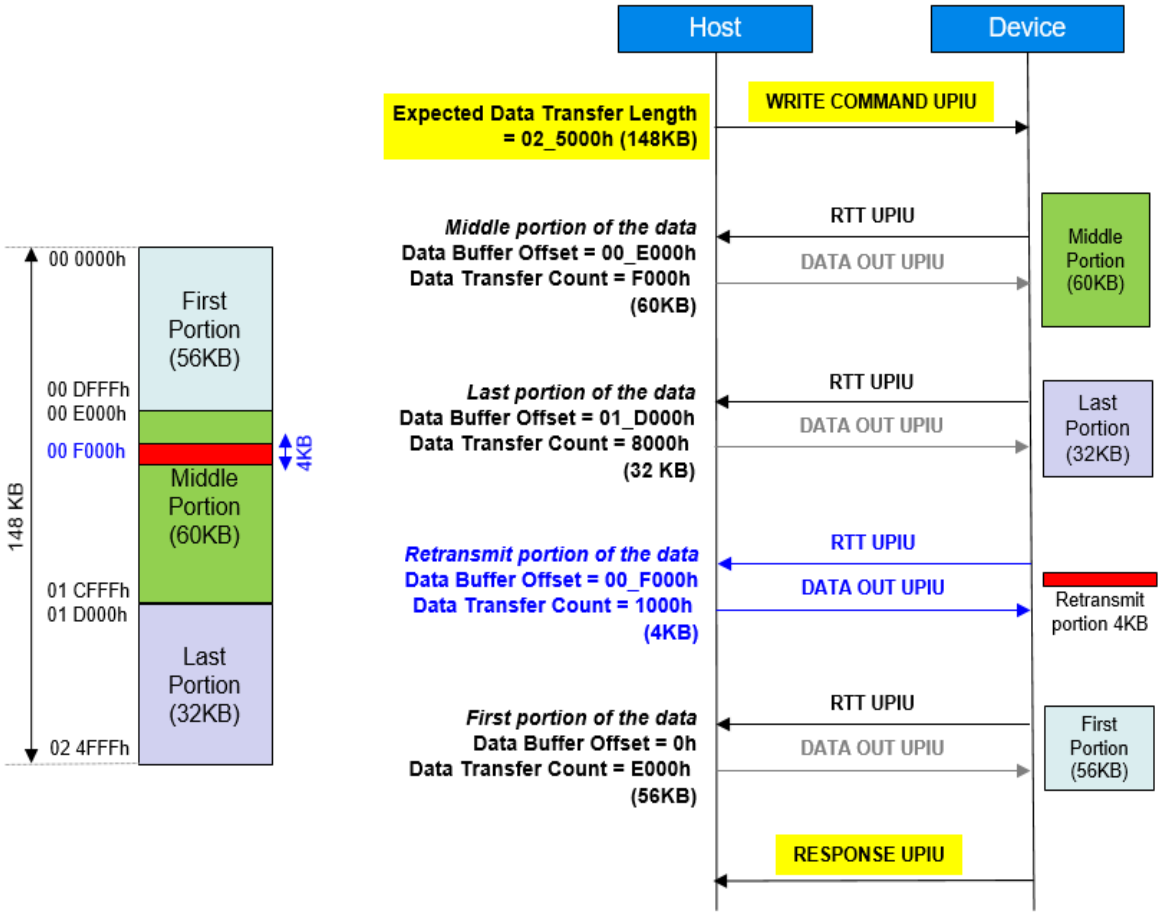


Figure 10.6 — READY TO TRANSFER UPIU with Retransmission Sequence Example

10.7.6 TASK MANAGEMENT REQUEST UPIU

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

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

Table 10.23 — Task Management Request UPIU

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

10.7.6.1 Basic Header

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

a) Transaction Type

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

b) IID

This field is the LSB nibble of the Initiator ID nexus, as described in bullet k) of the Basic Header Format. The Initiator ID nexus indicates the identity of the Initiator device who created the task request. See Initiator ID description in 10.6.2 for details.

c) EXT_IID

This field is the MSB nibble of the Initiator ID nexus, as described in bullet k) of the Basic Header Format. The Initiator ID nexus indicates the identity of the Initiator device who created the task request. See Initiator ID description in 10.6.2 for details.

d) Task Management Function

Table 10.24 defines the Task Management Function encodings for the set of UFS-supported Task Management Functions as defined in [SAM].

Table 10.24 — Task Management Function Values

Function	Value	
Abort Task	01h	
Abort Task Set	02h	
Clear Task Set	04h	
Logical Unit Reset	08h	
Query Task	80h	
Query Task Set	81h	

10.7.6.1 Basic Header (cont'd)

e) Task Management Input Parameters

Table 10.25 — 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 on by the task management function. Other bytes: Reserved
Input Parameter 3	Initiator ID [7:0] Initiator ID of the task/command operated by the task management function. • If bEXTIIDEn is set to 00h, the MSB nibble of this field shall be set to 0000b • If bEXTIIDEn is set to 01h, the MSB nibble of this field shall be set to EXT_IID of the task/command operated on by the task management function The LSB nibble shall be set to the IID of the task/command operated on by the task management function.

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.

10.7.7 TASK MANAGEMENT RESPONSE UPIU

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

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

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

Table 10.26 — Task Management Response UPIU

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

10.7.7.1 Basic Header

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

a) Transaction Type

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

b) IID

This field is the LSB nibble of the Initiator ID nexus, as described in bullet k) of the Basic Header Format. The Initiator ID nexus indicates the identity of the Initiator device who created the task request. See Initiator ID description in 10.6.2 for details.

c) EXT_IID

This field is the MSB nibble of the Initiator ID nexus, as described in bullet k) of the Basic Header Format. The Initiator ID nexus indicates the identity of the Initiator device who created the task request. See Initiator ID description in 10.6.2 for details.

d) Response

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

e) Task Management Output Parameters

Table 10.27 — Task Management Output Parameters

Field	Description
Output Parameter 1	LSB: Task Management Service Response (see Table 10.28) Other bytes: Reserved
Output Parameter 2	Reserved

f) Task Management Service Response

Table 10.28 — Task Management Service Response

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

10.7.8 QUERY REQUEST UPIU

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

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

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

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

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

Table 10.29 — 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) Data Segment Length	11 (LSB)
12	13	14	15
Transaction Specific Fields			
16	17	18	19
Transaction Specific Fields			
20	21	22	23
Transaction Specific Fields			
24	25	26	27
Transaction Specific Fields			
28	29	30	31
Reserved			
Header E2ECRC (omit if HD = 0)			
k Data[0]	k+1 Data[1]	k+2 Data[2]	k+3 Data[3]
…	…	…	…
k+ Length-4 Data[Length - 4]	k+ Length-3 Data[Length - 3]	k+ Length-2 Data[Length - 2]	k+ Length-1 Data[Length - 1]
Data E2ECRC (omit if DD = 0)			

10.7.8.1 Overview

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

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

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

The following is a short overview of the most common data structures that are transferred using the Query Request function. See the related sub-clauses for more detail on these data structures:

a) Descriptors

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

b) Attributes

An Attribute is a single parameter that represents a specific range of numeric values that can be set or read. This value could be a byte or word or floating point number. For example, baud rate or block size would be an attribute. Attribute size can be from 1-bit to 64-bit. Attributes of the same type can be organized in arrays, each element of them identified by an index.

c) Flags

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

10.7.8.2 Query Function

The Query Function field holds the requested query type describing the query function to perform. Common query functions are listed in Table 10.30. Currently, there are two general query functions defined: Read Request and Write Request. Additional Transaction Specific Fields will be used to specify further information needed for the transaction. These fields can describe the specific operation to perform, the target data or information to access, the amount of data to transfer and additional parameters and data.

Table 10.30 — Query Function Field Values

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

a) Standard Read Request

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

b) Standard Write Request

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

10.7.8.3 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 10.31.

Table 10.31 — Transaction Specific Fields

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

a) OPCODE

The opcode indicates the operation to perform. Possible opcode values are listed in Table 10.32.

Table 10.32 — 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	AGGREGATED READ	STANDARD READ REQUEST
09h-EFh	Reserved	Reserved
F0h-FFh	Vendor Specific	Vendor Specific

b) OSF

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

10.7.8.4 Read Descriptor Opcode

The READ DESCRIPTOR OPCODE is used to retrieve a UFS Descriptor from the Target device. A descriptor can be a fixed or variable length. There are up to 256 possible descriptor types. The OSF fields are used to select a particular descriptor and to read a number of descriptor bytes. The OSF fields are defined in Table 10.33.

Table 10.33 — Read Descriptor

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

a) DESCRIPTOR IDN

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

b) INDEX

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

c) SELECTOR

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

d) LENGTH

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

e) Data Segment

The Data Segment area is empty.

10.7.8.5 Write Descriptor Opcode

The WRITE DESCRIPTOR OPCODE is used to write a UFS Descriptor and it is sent from the host to the device. A descriptor can be a fixed or variable length. There are up to 256 possible descriptor types. The OSF fields are used to select a particular descriptor. The OSF fields are defined as listed in Table 10.34.

Table 10.34 — Write Descriptor

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

a) DESCRIPTOR IDN

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

b) INDEX

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

c) SELECTOR

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

d) LENGTH

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

e) Data Segment

The Data Segment area contains the data to be written.

10.7.8.6 Read Attribute Opcode

The READ ATTRIBUTE OPCODE is used to retrieve a UFS Attribute from the Target device. Attribute size can be from 1-bit to 64-bit. There are up to 256 possible Attributes, identified by an identification number, IDN, which ranges from 0 to 255. The OSF fields for this opcode are listed in Table 10.35.

Table 10.35 — 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

a) ATTRIBUTE IDN

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

b) INDEX

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

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

c) SELECTOR

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

d) Data Segment

The Data Segment area is empty.

10.7.8.7 Write Attribute Opcode

The **WRITE ATTRIBUTE OPCODE** is used to write a UFS Attribute to the Target device. Attribute size can be from 1-bit to 64-bit. There are up to 256 possible Attributes, identified by an identification number, IDN, which ranges from 0 to 255. The OSF fields for this opcode are listed in Table 10.36.

Table 10.36 — Write Attribute

Transaction Specific Fields for WRITE ATTRIBUTE OPCODE			
12 04h	13 ATTRIBUTE IDN	14 INDEX	15 SELECTOR
16 (MSB) VALUE [63:56]	17 VALUE [55:48]	18 VALUE [47:40]	19 VALUE [39:32]
20 VALUE [31:24]	21 VALUE [23:16]	22 VALUE [15:8]	23 (LSB) VALUE [7:0]
24	25	26	27
Reserved			

a) ATTRIBUTE IDN

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

b) INDEX

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

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

c) SELECTOR

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

d) VALUE [63:0]

The 64-bit VALUE field contains the data value of the Attribute. The VALUE is a right justified, big Endian value. Unused upper bits shall be set to zero.

e) Data Segment

The Data Segment area is empty.

10.7.8.8 Read Flag Opcode

The READ FLAG OPCODE is used to retrieve a UFS Flag value from the Target device. A Flag is a fixed size single byte value that represents a Boolean value. There can be defined up to 256 possible Flag values. A Flag is identified by its FLAG IDN, an identification number that ranges in value from 0 to 255. The OSF fields for this opcode are listed in Table 10.37.

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

Table 10.37 — Read Flag

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

a) FLAG IDN

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

b) INDEX

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

c) SELECTOR

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

d) Operation

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

e) Data Segment

The Data Segment area is empty.

10.7.8.9 Set Flag

Table 10.38 — Set Flag

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

a) FLAG IDN

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

b) INDEX

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

c) SELECTOR

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

d) Operation

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

e) Data Segment

The Data Segment area is empty.

10.7.8.10 Clear Flag

Table 10.39 — Clear Flag

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

a) FLAG IDN

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

b) INDEX

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

c) SELECTOR

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

d) Operation

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

e) Data Segment

The Data Segment area is empty.

10.7.8.11 Toggle Flag

Table 10.40 — Toggle Flag

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

a) FLAG IDN

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

b) INDEX

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

c) SELECTOR

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

d) Operation

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

e) Data Segment

The Data Segment area is empty.

10.7.8.12 NOP

Table 10.41 defines NOP OPCODE for QUERY REQUEST UPIU.

Table 10.41 — NOP

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

a) Data Segment

The Data Segment area is empty.

10.7.8.13 Aggregated Read Opcode

The AGGREGATED READ OPCODE is used to retrieve an aggregated data packet from the Target device. The aggregated data packet can contain Descriptors, Attributes and Flags. The OSF fields are used to select the aggregation type and to read a number of aggregated data packet bytes. The OSF fields are defined in Table 10.42.

Table 10.42 — Aggregated Read

Transaction Specific Fields for AGGREGATED READ OPCODE			
12	13	14	15
09h	AGGREGATION TYPE	Reserved	
16	17	18 (MSB)	19 (LSB)
Reserved		LENGTH	
20	21	22	23
Reserved			
24	25	26	27
Reserved			

a) AGGREGATION TYPE

The AGGREGATION TYPE field indicates the Descriptors, Attributes and Flags to be retrieved in the aggregated data packet:

xxxxxxx1b: All Flags
xxxxxx1xb: All Attributes
xxxxx1xxb: Device Descriptor
xxxx1xxxb: Unit Descriptors & RPMB Unit Descriptor
xxx1xxxxb: Interconnect Descriptor
xx1xxxxxb: String Descriptors
x1xxxxxxb: Geometry Descriptor & Power Descriptor
1xxxxxxxb: Device Health Descriptor

b) LENGTH

The LENGTH field is used to indicate the number of bytes to read of the aggregated data packet. The aggregated data packet will be returned in a QUERY RESPONSE UPIU packet. This is the length of the aggregated data packet which UFS host requests to read. The response will provide the exact length in the LENGTH field of the QUERY RESPONSE UPIU, where the exact length may be equal to or less than the length which UFS host requests to read.

c) Data Segment

The Data Segment area is empty.

10.7.9 QUERY RESPONSE UPIU

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

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

Table 10.43 — 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) Data Segment Length	11 (LSB)
12	13	14	15
Transaction Specific Fields			
16	17	18	19
Transaction Specific Fields			
20	21	22	23
Transaction Specific Fields			
24	25	26	27
Transaction Specific Fields			
28	29	30	31
Reserved			
Header E2ECRC (omit if HD = 0)			
k Data[0]	k+1 Data[1]	k+2 Data[2]	k+3 Data[3]
…	…	…	…
k+ Length-4 Data[Length - 4]	k+ Length-3 Data[Length - 3]	k+ Length-2 Data[Length - 2]	k+ Length-1 Data[Length - 1]
Data E2ECRC (omit if DD = 0)			

The QUERY RESPONSE UPIU follows the general UPIU format a field defined for query function. The transaction specific fields are defined specifically for each type of operation.

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

10.7.9.1 Overview

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

10.7.9.2 Query Function

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

10.7.9.3 Query Response

The Query Response field indicates the completion code of the action taken in response to the QUERY REQUEST UPIU. Possible values are listed in Table 10.44.

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

Table 10.44 — Query Response Code

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

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

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

10.7.9.4 Device Information

See Device Information field definition in RESPONSE UPIU, 10.7.3.1 Basic Header.

10.7.9.5 Transaction Specific Fields

The transaction specific fields are defined specifically for each type of operation. For the STANDARD READ REQUEST and STANDARD WRITE REQUEST, the operation specific fields are defined as in Table 10.45.

Table 10.45 — Transaction Specific Fields

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

a) OPCODE

The opcode indicates the operation to perform. Possible opcode values are listed in Table 10.32.

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

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

b) OSF

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

10.7.9.6 Read Descriptor Opcode

The READ DESCRIPTOR OPCODE is used to retrieve a UFS DESCRIPTOR from the Target device. A descriptor can be a fixed or variable length. There are up to 256 possible descriptor types. The OSF fields are used to select a particular descriptor and to read a number of descriptor bytes. The OSF fields are defined in Table 10.46.

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

Table 10.46 — Read Descriptor

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

a) DESCRIPTOR IDN

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

b) INDEX

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

c) SELECTOR

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

d) LENGTH

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

e) Data Segment

The Data Segment area contains the descriptor data.

10.7.9.7 Write Descriptor Opcode

The WRITE DESCRIPTOR OPCODE is used to respond to a write descriptor query request. A descriptor can be a fixed or variable length. There are up to 256 possible descriptor types. The OSF fields are used to select a particular descriptor. The OSF fields are defined in Table 10.47.

Table 10.47 — Write Descriptor

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

a) DESCRIPTOR IDN

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

b) INDEX

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

c) SELECTOR

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

d) LENGTH

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

e) Data Segment

The Data Segment area is empty.

10.7.9.8 Read Attribute Opcode

The READ ATTRIBUTE OPCODE is used to retrieve a UFS attribute from the Target device. Attribute size can be from 1-bit to 64-bit. There are up to 256 possible attributes, identified by an identification number, IDN, which ranges from 0 to 255.

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

The attribute data will be returned within the transaction specific fields. The Transaction Specific fields are formatted as indicated in Table 10.48. The first two 32-bit words of those fields will echo the first 32-bit word of the transaction specific fields of the QUERY REQUEST UPIU. The second and third words will contain the Attribute data.

Table 10.48 — Read Attribute Response Data Format

Transaction Specific Fields for READ ATTRIBUTE OPCODE			
12 03h	13 ATTRIBUTE IDN	14 INDEX	15 SELECTOR
16 (MSB) VALUE [63:56]	17 VALUE [55:48]	18 VALUE [47:40]	19 VALUE [39:32]
20 VALUE [31:24]	21 VALUE [23:16]	22 VALUE [15:8]	23 (LSB) VALUE [7:0]
24	25	26 Reserved	27

a) ATTRIBUTE IDN

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

b) INDEX

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

c) SELECTOR

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

d) VALUE [63:0]

The 64-bit VALUE field contains the data value of the ATTRIBUTE. The VALUE is a right justified, big Endian value. Unused upper bits shall be set to zero.

e) Data Segment

The Data Segment area is empty.

10.7.9.9 Write Attribute Opcode

The WRITE ATTRIBUTE OPCODE is used to respond to a write attribute query request. Attribute size can be from 1-bit to 64-bit. There are up to 256 possible attributes, identified by an identification number, IDN, which ranges from 0 to 255. The OSF fields for this opcode are listed in Table 10.49.

Table 10.49 — Write Attribute

Transaction Specific Fields for WRITE ATTRIBUTE OPCODE			
12 04h	13 ATTRIBUTE IDN	14 INDEX	15 SELECTOR
16 (MSB) VALUE [63:56]	17 VALUE [55:48]	18 VALUE [47:40]	19 VALUE [39:32]
20 VALUE [31:24]	21 VALUE [23:16]	22 VALUE [15:8]	23 (LSB) VALUE [7:0]
24	25	26	27
Reserved			

a) ATTRIBUTE IDN

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

b) INDEX

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

c) SELECTOR

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

d) VALUE [63:0]

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

e) Data Segment

The Data Segment area is empty.

10.7.9.9.1 Read Flag Opcode

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

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

The attribute data will be returned within the transaction specific fields. The Transaction Specific fields are formatted as indicated in Table 10.50. The first two 32-bit words of those fields will echo the first two 32-bit words of the transaction specific fields of the QUERY REQUEST UPIU. The third word will contain the FLAG data, a '0' or '1' value.

Table 10.50 — Read Flag Response Data Format

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

a) FLAG IDN

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

b) INDEX

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

c) SELECTOR

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

d) FLAG VALUE

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

e) Data Segment

The Data Segment area is empty.

10.7.9.10 Set Flag

Table 10.51 — 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 Reserved	25	26	27

a) FLAG IDN

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

b) INDEX

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

c) SELECTOR

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

d) FLAG VALUE

The FLAG VALUE field contains the FLAG data: 00h or 01h.
This field is valid only if the Query Response field indicates that the operation has been successfully completed (“Success”).

e) Data Segment

The Data Segment area is empty.

10.7.9.11 Clear Flag

Table 10.52 — 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 Reserved	25	26	27

a) FLAG IDN

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

b) INDEX

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

c) SELECTOR

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

d) FLAG VALUE

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

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

e) Data Segment

The Data Segment area is empty.

10.7.9.12 Toggle Flag

Table 10.53 — 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			

a) FLAG IDN

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

b) INDEX

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

c) SELECTOR

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

d) FLAG VALUE

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

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

e) Data Segment

The Data Segment area is empty.

10.7.9.13 NOP

Table 10.54 defines NOP OPCODE for QUERY RESPONSE UPIU.

Table 10.54 — NOP

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

a) Data Segment

The Data Segment area is empty.

10.7.9.14 Aggregated Read Opcode

The AGGREGATED READ OPCODE is used to retrieve an aggregated data packet from the Target device. The aggregated data packet can contain Descriptors, Attributes and Flags. The OSF fields are used to select the aggregation type and to read a number of aggregated data packet bytes. The OSF fields are defined in Table 10.55.

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

Table 10.55 — Aggregated Read

Transaction Specific Fields for AGGREGATED READ OPCODE			
12 09h	13 AGGREGATION TYPE	14 Reserved	15 Reserved
16 Reserved	17 Reserved	18 (MSB) LENGTH	19 (LSB) LENGTH
20 Reserved	21 Reserved	22 Reserved	23 Reserved
24 Reserved	25 Reserved	26 Reserved	27 Reserved

a) AGGREGATION TYPE

The AGGREGATION TYPE field indicates the Descriptors, Attributes and Flags actually retrieved in the aggregated data packet:

xxxxxxx1b: All Flags
xxxxxx1xb: All Attributes
xxxxx1xxb: Device Descriptor
xxxx1xxxb: Unit Descriptors & RPMB Unit Descriptor
xxx1xxxxb: Interconnect Descriptor
xx1xxxxxb: String Descriptors
x1xxxxxxb: Geometry Descriptor & Power Descriptor
1xxxxxxxb: Device Health Descriptor

b) LENGTH

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

10.7.9.14 Aggregated Read Opcode (cont'd)

c) Data Segment

The Data Segment area contains the aggregated data packet. Figure 10.7 shows an example of aggregated data packet.

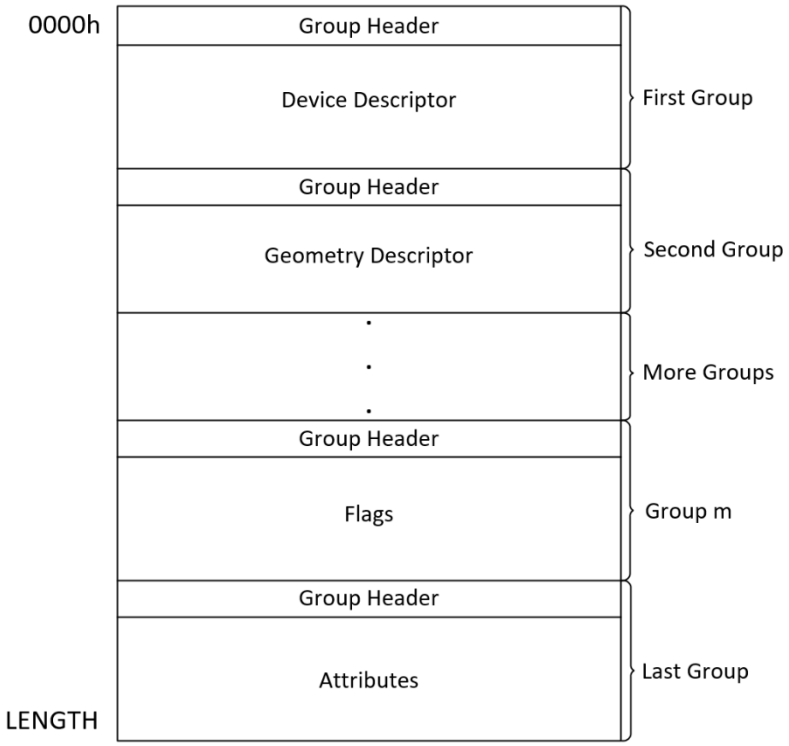


Figure 10.7 — Aggregated Data Packet Example

In the aggregated data packet, Descriptors, Flags and Attributes are organized by groups. Each group starts with a group header, which contains a group type and an offset pointing to the next group. Groups are conjunctive to each other and can come in any order. The group header is defined in Table 10.56.

Table 10.56 — Aggregated Data Packet Group Header Format

Aggregated Data Packet Group Header Format		
Group Type	Reserved	Next Group Offset

10.7.9.14 Aggregated Read Opcode (cont'd)

a) Group Type

The Group Type indicates what type of data is contained in this particular group.

00h: Reserved
01h: Flags
02h: Attributes
03h: Descriptors with unique Descriptor IDNs, i.e., any Descriptors excluding the String Decriptors.
04h: Manufacturer Name String Descriptor
05h: Product Name String Descriptor
06h: OEM ID String Descriptor
07h: Serial Number String Descriptor
08h: Product Revision Level String Descriptor
09h-FFh: Reserved

b) Next Group Offset

The Next Group Offset provides the byte offset from the beginning of the aggregated data packet to the next consecutive group. A value of 0h ends the linked list of groups.

10.7.10 REJECT UPIU

All UPIU packets include the basic header segment and some transaction specific fields. In addition to them, UPIU packets may have: Data Segment, Extra Header Segment, Header E2ECRC, Data E2ECRC. The purpose of the REJECT UPIU is to simplify the software development and the system debug.

Table 10.57 — Reject UPIU

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

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

The Transaction Type is defined in 10.7.2, Basic Header Format, and it is composed of the HD bit, DD bit, and Transaction Code fields.

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

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

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

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

10.7.10.1 Basic Header

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

a) Transaction Type

A type code value of xx11 1111b indicates a Reject UPIU.

b) Flags

The Flags field value shall be equal to zero.

c) LUN

The LUN shall be equal to the LUN value of the rejected UPIU.

d) Task Tag

The Task Tag shall be equal to the Task Tag value of the rejected UPIU.

e) IID

This field is the LSB nibble of the Initiator ID nexus, as described in bullet k) of the Basic Header Format. The Initiator ID nexus indicates the identity of the Initiator device who created the task request. See Initiator ID description in 10.6.2 for details.

The IID shall be equal to the IID value of the rejected UPIU.

f) EXT_IID

This field is the MSB nibble of the Initiator ID nexus, as described in bullet k) of the Basic Header Format. The Initiator ID nexus indicates the identity of the Initiator device who created the task request. See Initiator ID description in 10.6.2 for details.

The EXT_IID shall be equal to the EXT_IID value of the rejected UPIU.

g) Response

The Response field shall be set to 01h (Target Failure) indicating that the Target device was not able to execute the requested operation.

h) Data Segment Length

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

10.7.10.1 Basic Header (cont'd)

i) Basic Header Status

The Basic Header Status field provides information about error detected in the UPIU received by the Initiator device.

Table 10.58 defines the possible values for the Basic Header Status field.

Table 10.58 — Basic Header Status Description

Value	Name
00h	Reserved
01h	Invalid Transaction Type
02h to FFh	Reserved

j) E2E Status

The E2E Status field provides the result of the end-to-end CRC of the rejected UPIU for both Header and Data. E2E Status is reserved if end-to-end CRC is not supported.

Table 10.59 — E2E Status Definition

Bit	Description
Bit 0	0: Header E2ECRC validated or not supported 1: Header E2ECRC error
Bit 1	0: Data E2ECRC validated or not supported 1: Data E2ECRC error
Others	Reserved

10.7.11 NOP OUT UPIU

The Initiator device may use NOP OUT UPIU to check the connection to a device. The Target device will respond to a NOP OUT UPIU sending a NOP IN UPIU back to the Initiator device.

Table 10.60 — 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 Reserved	13	14	15
16 Reserved	17	18	19
20 Reserved	21	22	23
24 Reserved	25	26	27
28 Reserved	29	30	31
32 (MSB) Header E2ECRC (omit if HD = 0)	33	34	35 (LSB)

10.7.11.1 Basic Header

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

a) Task Tag

Task Tag normally is related to I_T_L_Q nexus addressing of SCSI while here it is used in a pure UTP (device level) context.

b) Transaction Type

A type code value of xx00 00000b indicates a NOP OUT UPIU.

c) Flags

The Flags field value shall be equal to zero.

d) Data Segment Length

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

10.7.12 NOP IN UPIU

NOP IN UPIU is the response from the Target device to a NOP OUT UPIU sent by the Initiator device.

Table 10.61 — NOP IN UPIU

NOP IN UPIU			
0 xx10 0000b	1 Flags	2 Reserved	3 Task Tag
4 Reserved	5 Reserved	6 Response (00h)	7 Reserved
8 Total EHS Length (00h)	9 Device Information (00h)	10 (MSB)	11 (LSB) Data Segment Length (0000h)
12	13	14	15 Reserved
16	17	18	19 Reserved
20	21	22	23 Reserved
24	25	26	27 Reserved
28	29	30	31 Reserved
32 (MSB)	33	34	35 (LSB) Header E2ECRC (omit if HD = 0)

10.7.12 1 Basic Header

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

a) Transaction Type

A type code value of xx10 00000b indicates a NOP IN UPIU.

b) Flags

The Flags field value shall be equal to zero.

c) Task Tag

The Task Tag shall be equal to the Task Tag value of the corresponding NOP OUT UPIU.

d) Response

The Response field shall be set to 00h (Target Success) indicating that the Target device was able to respond to the NOP OUT UPIU.

e) Data Segment Length

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

10.7.13 Data Out Transfer Rules

Data out transfer rules related with RTT have been defined for the following reasons:

- To have the same implementation of UFS data transfer mechanisms for data out transactions across various host and device vendors.
- To limit the number of outstanding RTTs sent by the device based on host capability.

RTT requests related to several commands and from any logical units may be sent in any order.

Examples of commands with data out transfer are: MODE SELECT (10), WRITE (6), WRITE (10), WRITE (16), FORMAT UNIT, SECURITY PROTOCOL OUT, etc.

The following rules are applicable for device in generating the RTT and the host in handling the RTT:

- Rule 1 - The host is expected to send only one DATA OUT UPIU for each RTT received from the device.
  - Figure 10.8 shows an example of UTP traffic related to a write command processing: the host sends one and only one DATA OUT UPIU for each READY TO TRANSFER UPIU sent by the device.

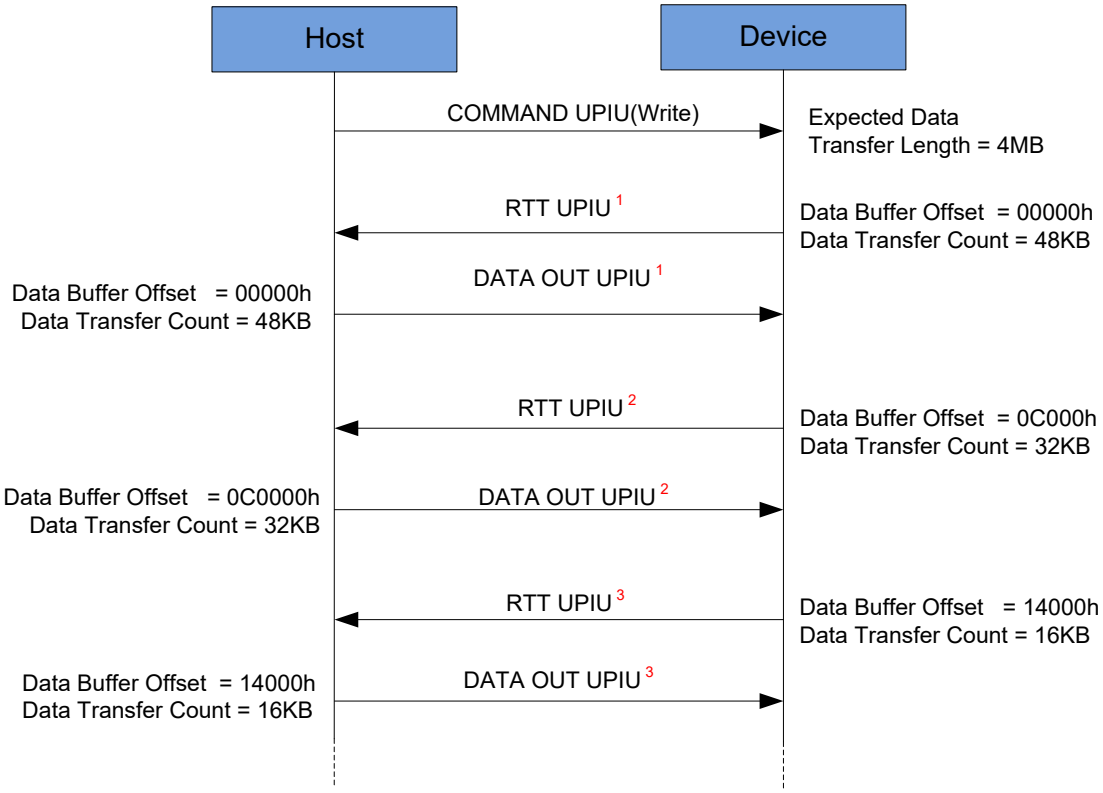


Figure 10.8 — Example for Data Out Transfer Rule 1

10.7.13 Data Out Transfer Rules (cont'd)

- If the Data Buffer Offset field value or the Data Transfer Count field value in the DATA OUT UPIU does not match the corresponding parameters in the RTT request, the device shall terminate the command by sending RESPONSE UPIU with UTP Data Out Mismatch Error flag (Flags.D) set and Status = CHECK CONDITION with SENSE KEY = ABORTED COMMAND. In this case, the device shall wait until it receives all DATA OUT UPIUs related to any outstanding READY TO TRANSFER UPIUs before sending the Response UPIU with UTP Data Out Mismatch Error flag (Flags.D) set and Status = CHECK CONDITION with SENSE KEY = ABORTED COMMAND. Figure 10.9 describes a scenario, where the Data Transfer Count value does not match.

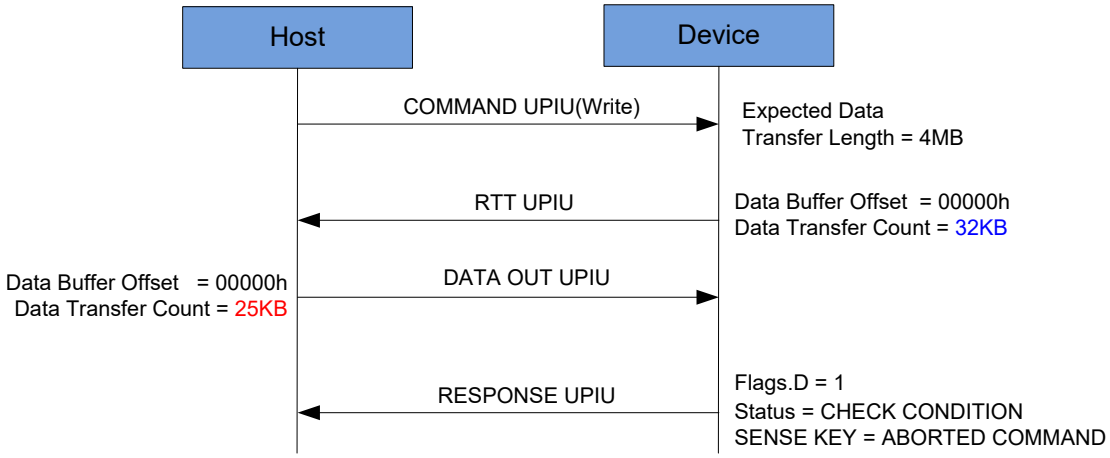


Figure 10.9 — Example for Data Transfer Count Mismatch

- Rule 2 – Device shall not have outstanding RTTs more than specified by host
  - bDeviceRTTCap read-only parameter in Device Descriptor defines the maximum number of outstanding RTTs which can be supported by device. bMaxNumOfRTT read-write attribute limits the number of outstanding RTTs generated by the device. bMaxNumOfRTT is equal to two after device manufacturing, and it may be changed by the host according to its capability to increase performance. In particular, bMaxNumOfRTT value shall not be set to a value greater than bDeviceRTTCap value, and it may be set only when the command queues of all logical units are empty. If the host attempts to write a value higher than what indicated by bDeviceRTTCap, the value shall not be changed and the QUERY RESPONSE UPIU shall have Query Response field set to "Invalid VALUE". If the host attempts to write bMaxNumOfRTT when there is at least one logical unit with command queue not empty, the operation shall fail, and Query Response field in the QUERY RESPONSE UPIU shall be set to FFh ("General failure").

10.7.13 Data Out Transfer Rules (cont'd)

- Figure 10.10 shows an example of UTP traffic related to a write command processing assuming bMaxNumOfRTT = 2. The Target device sends RTT UPIU[1] and RTT UPIU[2], and the Initiator device starts to provide data related to first request. There are two outstanding RTTs (RTT UPIU[1] and RTT UPIU[2]). Therefore the Target device cannot send additional RTT UPIUs. The Target device sends the third data request (RTT UPIU[3]) only after receiving DATA OUT UPIU[1].

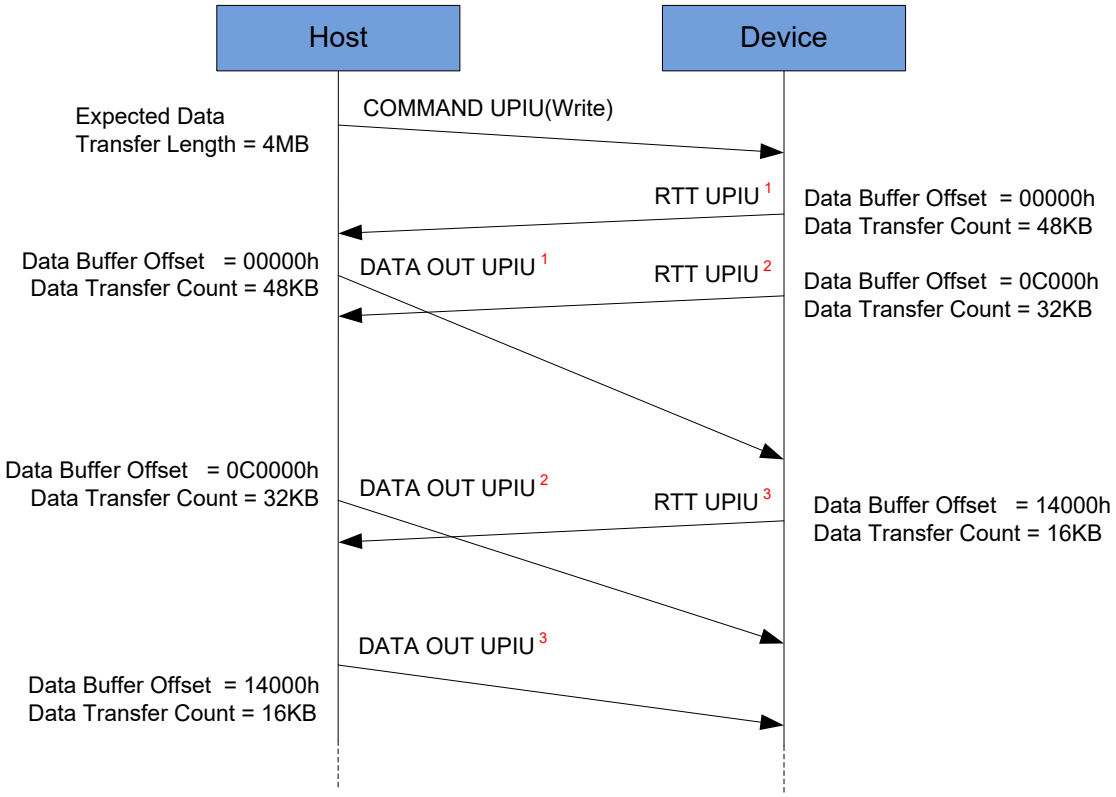


Figure 10.10 — Example for Data Out Transfer Rule 2

- Rule 3 – Across all logical units, DATA OUT UPIUs shall be sent in the same order of RTT UPIUs: data for RTT^{N} shall be transferred before transferring data for RTT^{N+1}.
  - For example, if the host first receives RTT UPIU[1] and then RTT UPIU[2] from the device, it shall send data related to the first request (Data1) prior to data related to the second request (Data2).

10.7.13 Data Out Transfer Rules (cont'd)

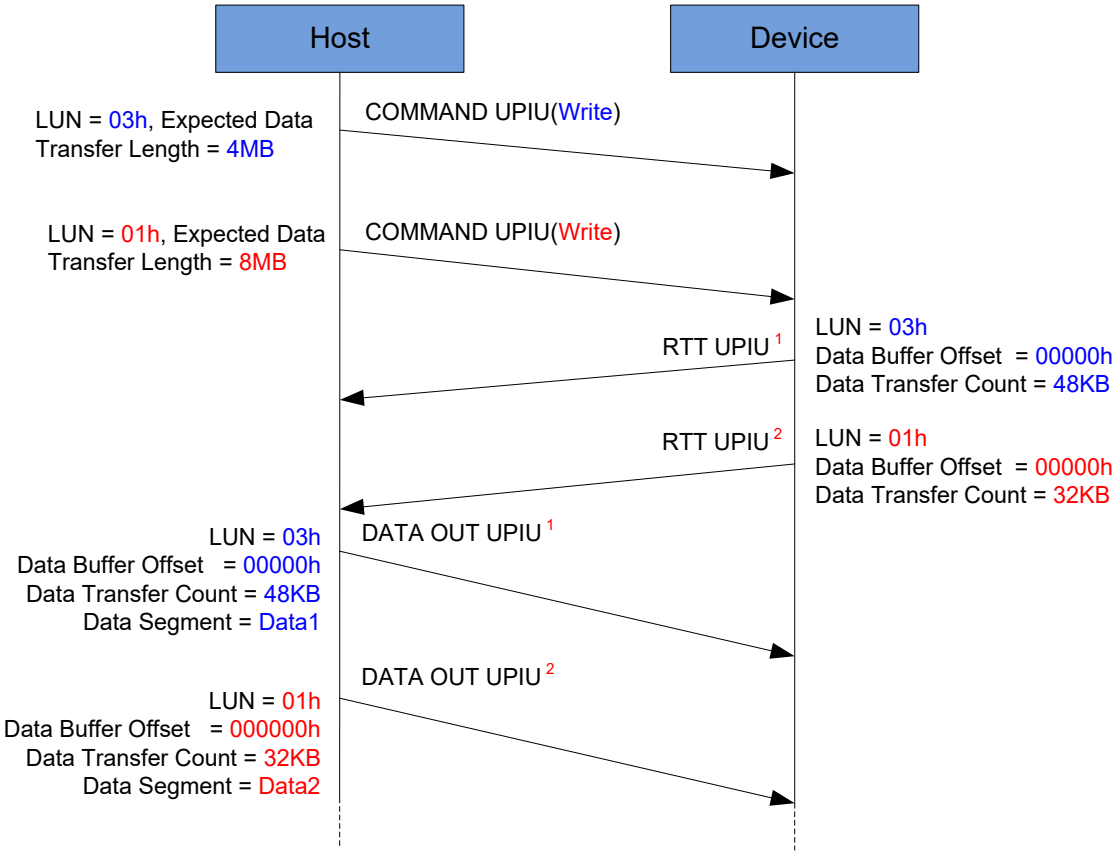


Figure 10.11 — Example for Data Out Transfer Rule 3

- It is recommended for device to determine the sequence of RTTs based on logical unit priority, command priority, internal optimization, etc.

10.7.13.1 Implementation

The following parameters are defined to implement data out transfer rules.

Table 10.62 — Parameters # to Data Out Transfer Rules

bMaxNumOfRTT	Defines the current maximum number of outstanding RTTs that are allowed. This value can be set by the host.
bDeviceRTTCap	Defines the maximum number of outstanding RTTs supported by device
UTP Data Out Mismatch Error Flag(D)	The D flag shall be set to '1' to indicate that a Data Out mismatch error occurred during the command transaction

10.7.14 Overview - Data Out of Order Transfer

Out of order data transfer can improve UFS storage performance. The device & host work together for efficient Out of order data transfer. Out of order data transfer allows the device to control the transfer order for multiple commands sent to same or different LUN(s).

Out of order sequencing may only occur if the UFS device supports it (bDataOrdering = 01h, 02h, or 03h) and if this feature is enabled (bOutOfOrderDataEn = 01h, 02h, or 03h). Host should enable/disable the feature only when no outstanding commands and all queues are empty.

Out of order sequencing allows the device to control the order of data transfer within a command. In each DATA IN UPIU or the RTT UPIU, the value of the Data Buffer Offset field may be set to any valid value within the range of the data transfer for that command.

For example-#1:

If any hint information for a command is not delivered to the host yet, the data may be transferred without relying on hint information. When all hint information is delivered for a command, the data cannot include hint information for the command or may include the hint for another command.

```
Let's assume
    - "DATA #N-M" means that "M-th DATA UPIU for COMMAND-N"
    - "Hint-#N" means that "Hint information for DATA of COMMAND-N"
    - COMMAND #3 is to send 48KB data, COMMAND-#4 is to send 20KB data.

...                                 // Assume that there is no prior hint delivered to host for Data-#3 yet.
DATA #3-1 (4KB) without Hint        // sending 4KB data without relying on hint information
DATA #3-2 (20KB) with Hint-#3(16KB) // sending 20KB data without relying on hint information
                                    //   (Hint information for remainder data(16KB) is sent together.)
DATA #3-3 (8KB) with Hint-#3(8KB)   // sending 8KB data relying on hint information,
                                    //   (Hint information for remainder data(8KB) is sent together.)
DATA #3-4 (4KB) without Hint        // sending 4KB data relying on hint information,
                                    //   (Hint is not included, since all hint information is delivered already.)
DATA #3-5 (12KB) with Hint-#4(16KB) // sending 12KB data relying on hint information,
                                    //   (Hint for data of other Command-#4 may be included.)
...
DATA #4-1 (8KB) with Hint-#4(4KB)   // sending 8KB data relying on hint information
                                    //   (Hint information for remainder data(4KB) is sent together.)
DATA #4-2 (4KB) without Hint
DATA #4-3 (8KB) without Hint
```

Figure 10.12 — Example for Data Out of Order Transfer - 1

10.7.14 Overview - Data Out of Order Transfer (cont'd)

For example-#2:

Each data access size and boundary does not need to same as the size and boundary when each hint was provided. The number of data access may be different from the number of hint information. And, the data may be accessed in different order what hint was given.

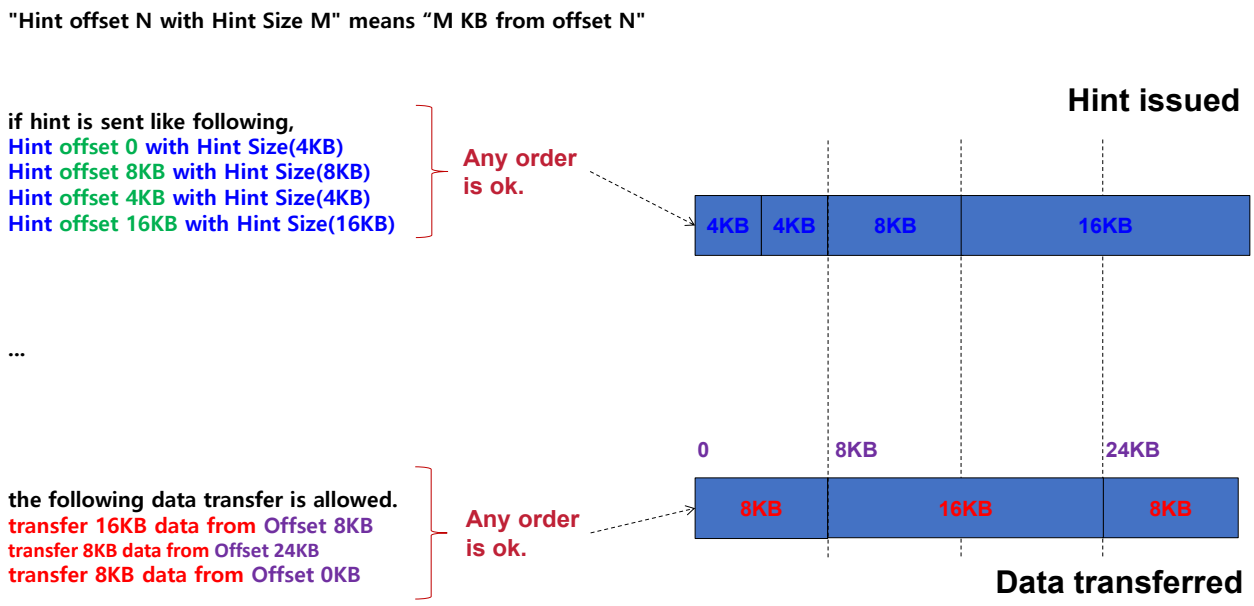


Figure 10.13 — Example for Data Out of Order Transfer - 2

When Out of Order is enabled (bOutOfOrderDataEn = 01h, 02h, or 03h), the device should transfer DATA IN UPIUs or RTT UPIUs for which hints were previously provided.

Once the device start processing the command, the device should start sending hints for that command in the UPIU hint fields, unless providing that hint would exceed wHostHintCacheSize.

When the current DATA IN or RTT UPIU does not match the previous hint information, the host controller is expected to follow the information in the current DATA IN UPIU or RTT UPIU. Previously cached Hint information may be used for later DATA IN UPIU or RTT UPIU.

When the feature is disabled (bOutOfOrderDataEn = 00h), device shall not provide hint information to the host controller and HintControl shall be set to 0x0.

In case a command was aborted or failed (A TMR was issued or command terminated with a CHECK CONDITION), the device and the host may clear all outstanding hint information for all commands from their cache.

10.8 Logical Units

This sub-clause gives more details on the definition of logical unit in the UFS Standard.

10.8.1 UFS SCSI Domain

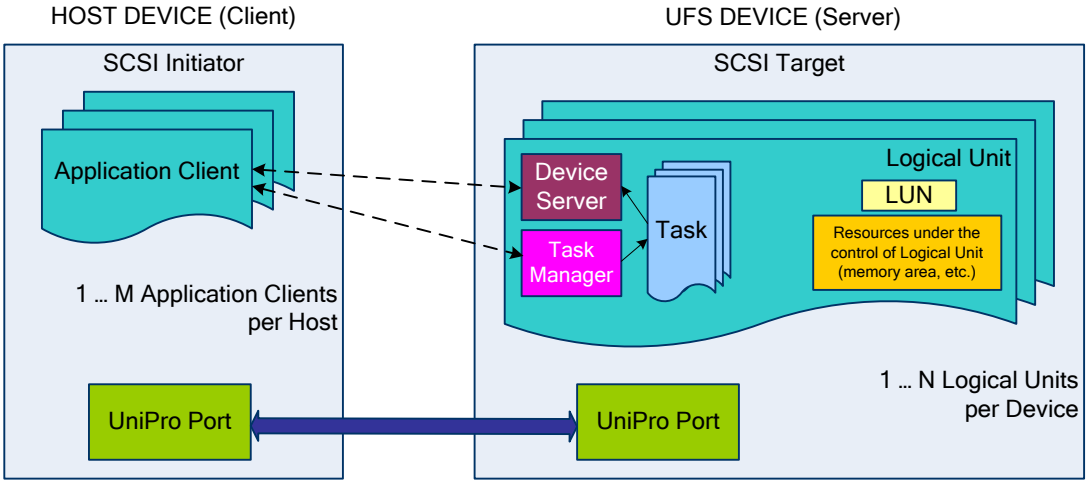


Figure 10.14 — UFS SCSI domain

10.8.2 UFS Logical Unit Definition

A logical unit (LU) is an externally addressable, independent, processing entity that processes SCSI tasks (commands) and performs task management functions.

- Each logical unit is independent of other logical units in a device
- UFS shall support the amount of logical units specified by bMaxNumberLU, in addition to the well known logical units defined in 10.8.5.
- Logical units may be used to store boot code, application code, and mass storage data applications

Commands addressed to logical unit ‘i’ are handled by logical unit ‘i’ exclusively, not visible, handled or processed by logical unit ‘j’

A logical unit contains the following (see [SAM] for definitions):

- DEVICE SERVER
- TASK MANAGER
- TASK SET

10.8.3 Well Known Logical Unit Definition

A UFS device contains well known logical units to support very specific types of commands such as the REPORT LUNS command to allow an application client to issue requests to receive specific information usually relating to the entire device as defined in [SAM].

UFS devices contain additional well known logical units for specific UFS functions, including Boot and RPMB. Each well known logical unit has a well known logical unit number (W-LUN).

10.8.4 Logical Unit Addressing

The 8-bit LUN field in UPIU is used to provide either LUN or W-LUN. In particular, the most significant bit of this field (WLUN_ID) shall be set according to the logical unit type as follows:

- WLUN_ID = 0b for logical unit,
- WLUN_ID = 1b for well known logical unit.

The remaining 7 bits of the LUN field (UNIT_NUMBER_ID) shall be set to either the LUN value or the W-LUN value, depending on the logical unit type.

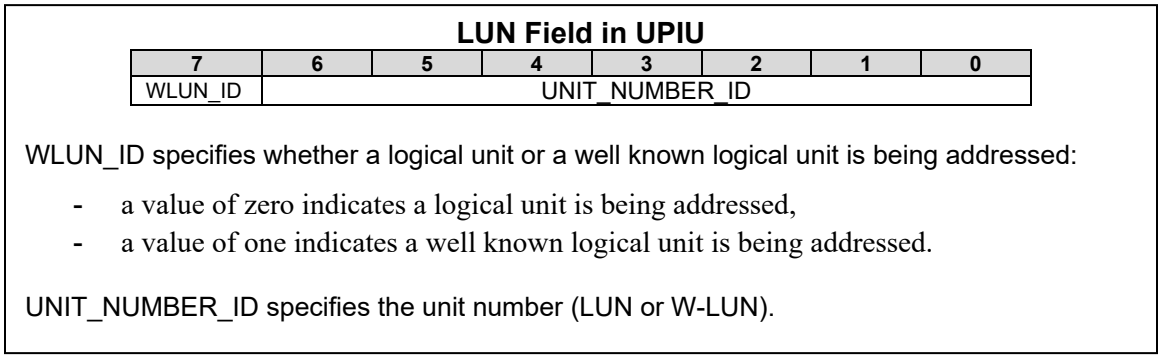


Figure 10.15 — Logical Unit Addressing

Therefore, the encoding of the LUN field in UPIU supports up to 128 LUN's and up to 128 W-LUN's ($0 \leq$ UNIT_NUMBER_ID ≤ 127) .

10.8.5 Well Known Logical Units Defined in UFS

The following well known logical units are defined in this standard for SCSI and UFS specific functions: REPORT LUNS, UFS Device, Boot, RPMB.

The REPORT LUNS well known logical unit is defined in [SPC] and provides the logical unit inventory. The UFS Device well known logical unit provides UFS device level interaction (i.e., Power mode control, Wipe Device). The Boot well known logical unit is a virtual reference to the actual logical unit containing boot code, as designated by the host. The Boot well known logical unit is read at the system startup to access the boot code. The RPMB well known logical unit supports the RPMB function with its own independent processes and memory space as dictated by the RPMB security definition.

Each well known logical unit shall only process the commands listed in Table 10.63. If one of the four well known logical units receives a command that is not listed in the table, then the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID COMMAND OPERATION CODE.

Table 10.63 — Well Known Logical Unit Commands

Well known logical unit	W-LUN	LUN Field in UPIU	Command name
REPORT LUNS	01h	81h	INQUIRY, REQUEST SENSE, TEST UNIT READY, REPORT LUNS
UFS Device	50h	D0h	INQUIRY, REQUEST SENSE, TEST UNIT READY, START STOP UNIT, FORMAT UNIT
Boot	30h	B0h	INQUIRY, REQUEST SENSE, TEST UNIT READY, READ (6), READ (10), READ (16), READ BUFFER
RPMB	44h	C4h	INQUIRY,REQUEST SENSE, TEST UNIT READY, SECURITY IN, SECURITY OUT

NOTE If bBootEnable field in the Device Descriptor is set to zero or if the Boot well known logical is not mapped to an enabled logical unit (see bLUEnable, bBootLunID and bBootLunEn), then the Boot well known logical unit shall terminate TEST UNIT READY, READ (6), READ (10), READ (16), and READ BUFFER commands with CHECK CONDITION status.

10.8.6 Translation of 8-bit UFS LUN to 64-bit SCSI LUN Address

[SAM] describes a 64-bit LUN addressing scheme. The value of C1h in the first 8 bits of the 64-bit address indicates a well known LUN address in the SAM SCSI format. Examples of translation of the 8-bit LUN field value in UPIU to 64-bit SCSI address are shown in Table 10.64.

Table 10.64 — Examples of Logical Unit Representation Format

Logical unit			
Logical Unit Name	**LUN field in UPIU**	**LUN**	**SAM LUN**
Logical Unit 1	01h	01h	00 01 00 00 00 00 00 00h
Logical Unit 6	06h	06h	00 06 00 00 00 00 00 00h
Well known logical unit			
Logical Unit Name	**LUN field in UPIU**	**W-LUN**	**SAM LUN**
Boot	B0h	30h	C1 30 00 00 00 00 00 00h
RPMB	C4h	44h	C1 44 00 00 00 00 00 00h

10.8.7 SCSI Write Command

The UPIU execution of a SCSI write command is composed of the following subsequent phases: command, data, status. In the command phase a write command is sent to the device using COMMAND UPIU.

During the subsequent phase data is delivered to the UFS device using DATA OUT UPIU: the UFS device paces the data delivery by sending a READY TO TRANSFER UPIU when it is ready for the next DATA OUT UPIU. (The UFS device may send new READY TO TRANSFER UPIUs before it receives data for previous request.)

The write command terminates with a RESPONSE UPIU that contains the status.

Figure 10.16 shows an example of UPIU sequence for SCSI write command.

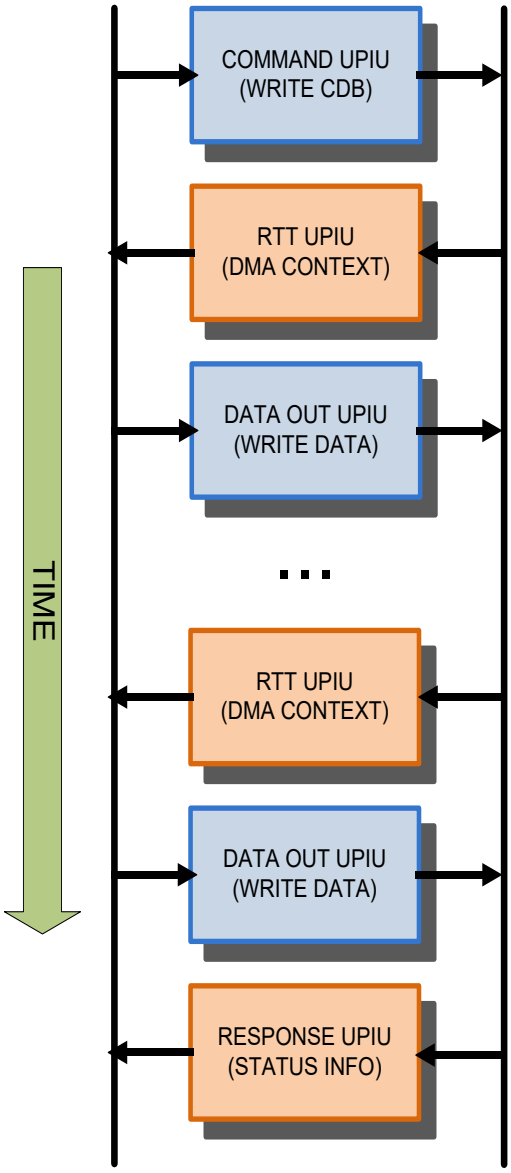


Figure 10.16 — SCSI Write

10.8.8 SCSI Read Command

The UPIU execution of a SCSI read command is composed of the following subsequent phases: command, data, status. In the command phase a read command is sent to the device using COMMAND UPIU. During the subsequent phase the UFS device delivers data to the host using DATA IN UPIU. The read command terminates with a RESPONSE UPIU that contains the status.

Figure 10.17 shows an example of UPIU sequence for SCSI read command.

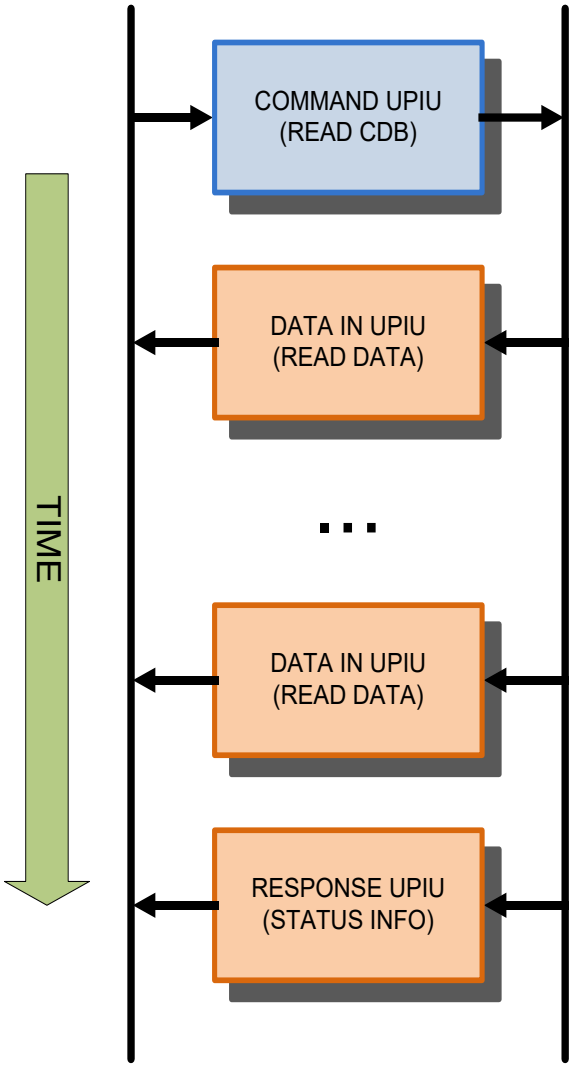


Figure 10.17 — SCSI Read

10.8.9 Unit Attention Condition

Unit Attention Condition (UAC) is a condition which needs to be serviced before the logical unit can process commands (see [SAM]).

Table 10.65 shows all events which shall establish UAC. Power-on, HW Reset, EndPointReset, and Host UniPro Warm Reset shall establish UAC on all LUs including all Well-known LUs (REPORT LUNS, UFS Device, RPMB and BOOT). LU Reset shall establish UAC only for an addressed LU.

Table 10.65 — 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

UAC on each LU shall be cleared before it can be accessed successfully. See Table 10.66 for exceptions.

Except for some specific commands listed in Table 10.66, if a command is sent to a LU with a pending UAC, the command will fail. The device server shall respond with CHECK CONDITION status, Sense Key set to UNIT ATTENTION. Afterwards the device server shall clear UAC on the LU. Table 10.66 shows some exceptional commands on UAC behavior.

Table 10.66 — Commands for Exceptional Behavior on UAC

Command	Status on pending UAC		UAC after command response
REQUEST SENSE	GOOD (1)	-	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 11.3.15).

10.9 Application Layer and Device Manager Transport Protocol Services

10.9.1 UFS Initiator Port and Target Port Attributes

Table 10.67 — UFS Initiator Port and Target Port Attributes

Attribute	Value
Maximum CDB Length	16 bytes
Command Identifier Size	16 bits
Task Attributes Supported	Simple, Head of Queue, Ordered, ACA (not supported)
Maximum Data-In Buffer Size	FFFFh
Maximum Data-Out Buffer Size	FFFFh
Maximum CRN	Not Applicable
Command Priority Supported	Yes (field width = 1 bit)
Maximum Sense Data Length	FFFFh
Status Qualifier Supported	No
Additional Response Information Supported	Yes
Bidirectional Commands Supported	No
Task Management Functions Supported	Abort Task, Abort Task Set, Clear Task Set, Logical Unit Reset, Query Task, Query Task Set

10.9.2 Execute Command Procedure Call Transport Protocol Services

See [SAM] for information on this topic. Figure 10.18 graphically describes UFS command execution without a data phase.

Initiator device	Target device
Send SCSI Command request	SCSI Command Received indication
Command Complete Received confirmation	Send Command Complete response

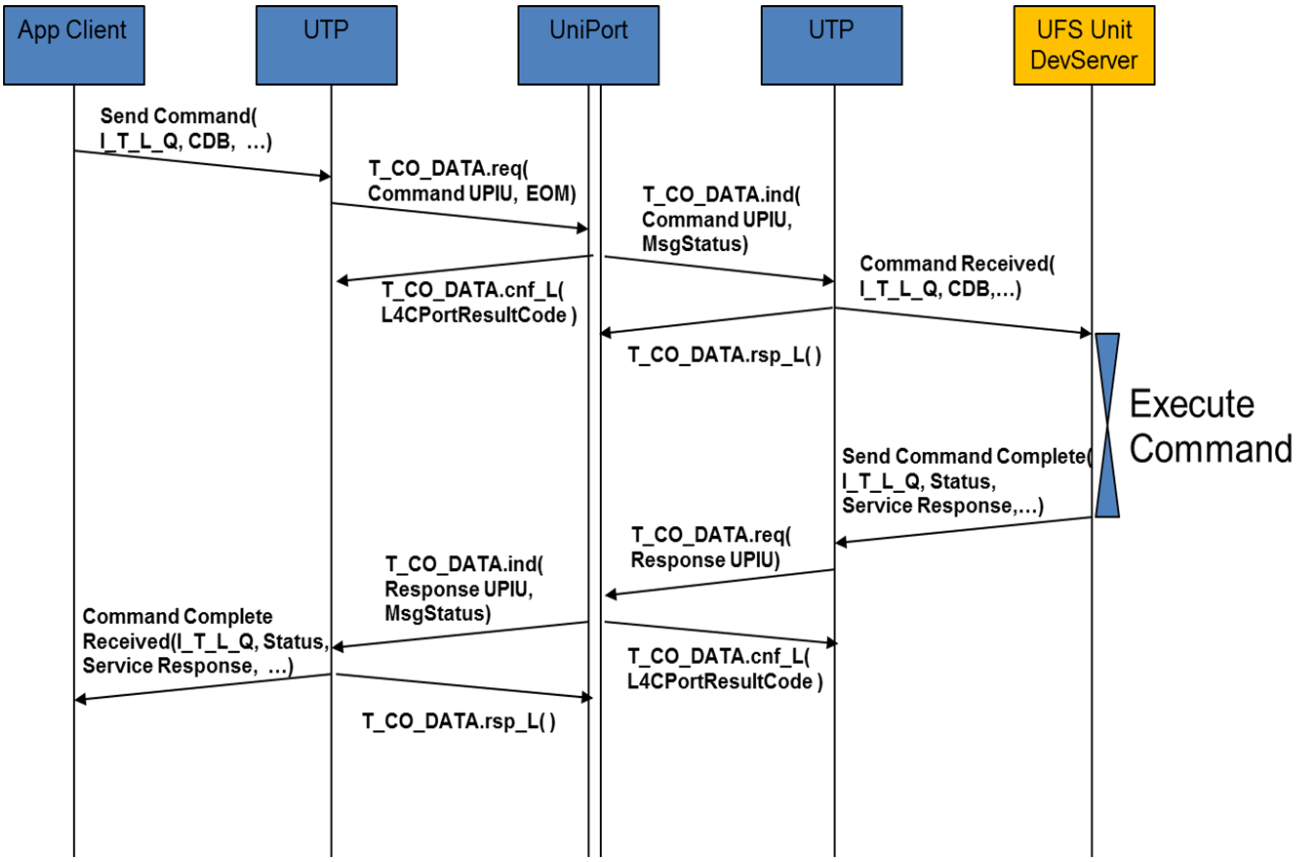


Figure 10.18 — Command without Data Phase

10.9.3 SCSI Command Transport Protocol Service

An application client uses the Send Command transport protocol service request to request a UFS Initiator port transmit a COMMAND UPIU via UniPort. See [SAM] for information on this topic.

10.9.4 SCSI Command Received Transport Protocol

A UFS target port uses the SCSI Command Received transport protocol service indication to notify a task manager that it has received a COMMAND UPIU. See [SAM] for information on this topic.

10.9.5 Send Command Complete Transport Protocol Service

A device server uses the Send Command Complete transport protocol service response to request that a UFS target port transmit a RESPONSE UPIU. See [SAM] for send command complete information.

10.9.6 Command Complete Received Transport Protocol Service

A UFS initiator port uses the Command Complete Received transport protocol service confirmation to notify an application client that it has received a response for its COMMAND UPIU. See [SAM] for command complete received information.

10.9.7 Data Transfer SCSI Transport Protocol Services

See [SAM] for information on this topic.

10.9.7.1 Send Data-In Transport Protocol Service

A device server uses the Send Data-In transport protocol service request to request that a UFS target port sends data. See [SAM] for information on this topic.
A device server shall only call Send Data-In () during a read operation.

A device server shall not call Send Data-In () for a command on a given I_T_L nexus after it has called Send Command Complete () for that command on the I_T_L nexus (e.g., following a RESPONSE UPIU for that command on the I_T_L nexus) or called Task Management Function Executed for a task management function that terminates that command (e.g., an ABORT TASK).

10.9.7.2 Data-In Delivered Transport Protocol Service

This confirmation notifies the device server that the specified data was successfully delivered to the application client buffer, or that a UniPro delivery subsystem error occurred while attempting to deliver the data. See [SAM] for information on this topic.

10.9.7.2 Data-In Delivered Transport Protocol Service (cont'd)

Figure 10.19 and Figure 10.20 graphically describe UFS command execution with a data-in data phase.

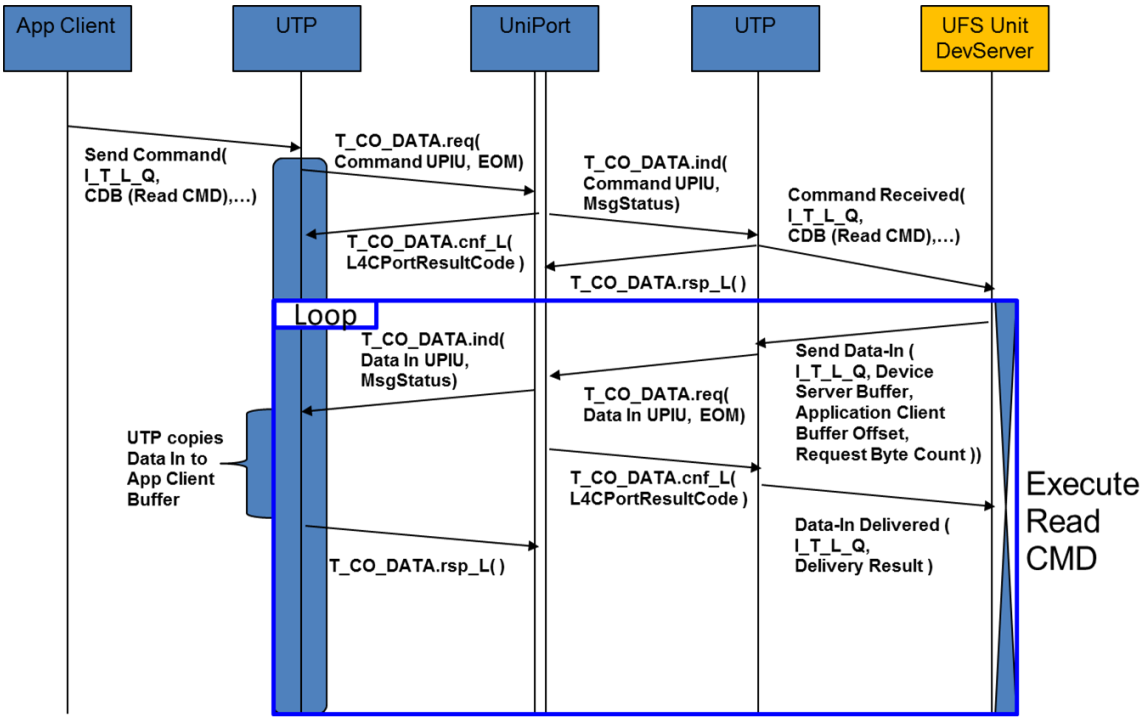


Figure 10.19 — Command + Read Data Phase 1/2

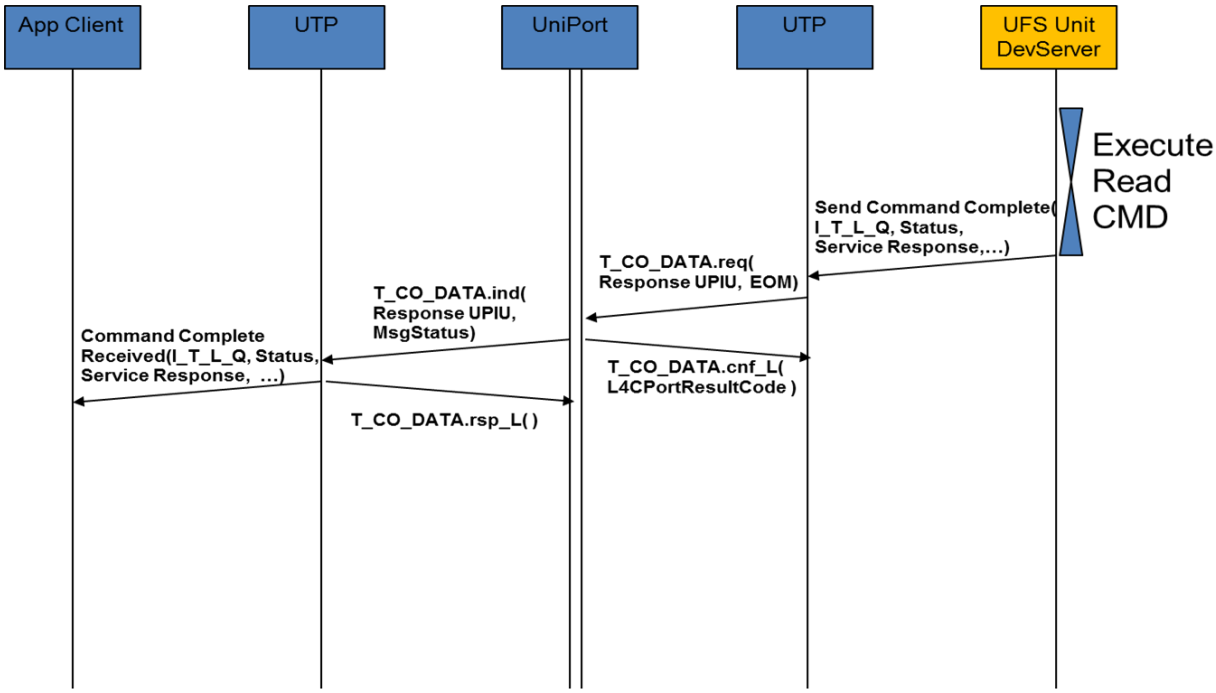


Figure 10.20 — Command + Read Data Phase 2/2

10.9.7.3 Receive Data-Out Transport Protocol Service

A device server uses the Receive Data-Out transport protocol service request to request that a UFS target port receives data. See [SAM] for information on this topic.
A device server shall only call Receive Data-Out () during a write operation.

A device server shall not call Receive Data-Out () for a command on a given I_T_L nexus after a Send Command Complete () has been called for that command on the I_T_L nexus or after a Task Management Function Executed () has been called for a task management function that terminates that command (e.g., an ABORT TASK).

10.9.7.4 Data-Out Received Transport Protocol Service

A UFS target port uses the Data-Out Received transport protocol service indication to notify a device server that it has received data. See [SAM] for information on this topic.

10.9.7.4 Data-Out Received Transport Protocol Service (cont'd)

Figures 10.20 and 10.21 graphically describe UFS command execution with a data-in data phase.

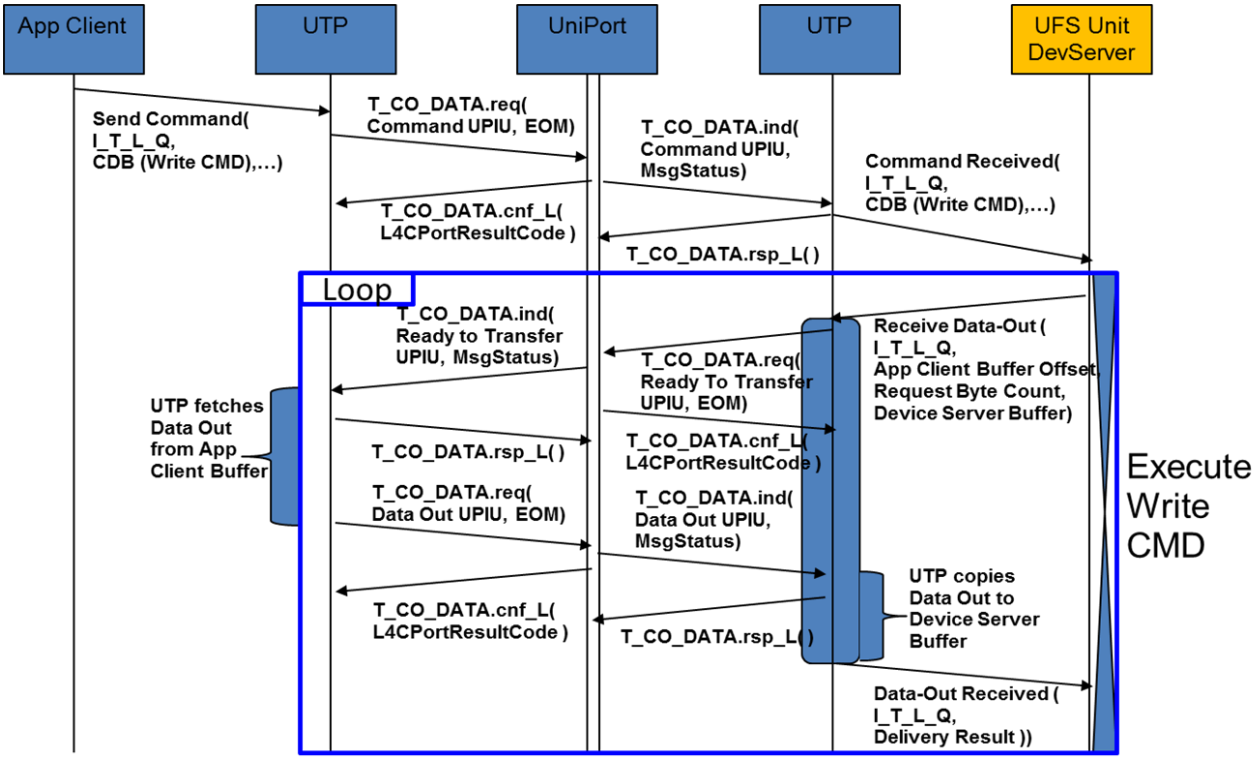


Figure 10.21 — Command + Write Data Phase 1/2

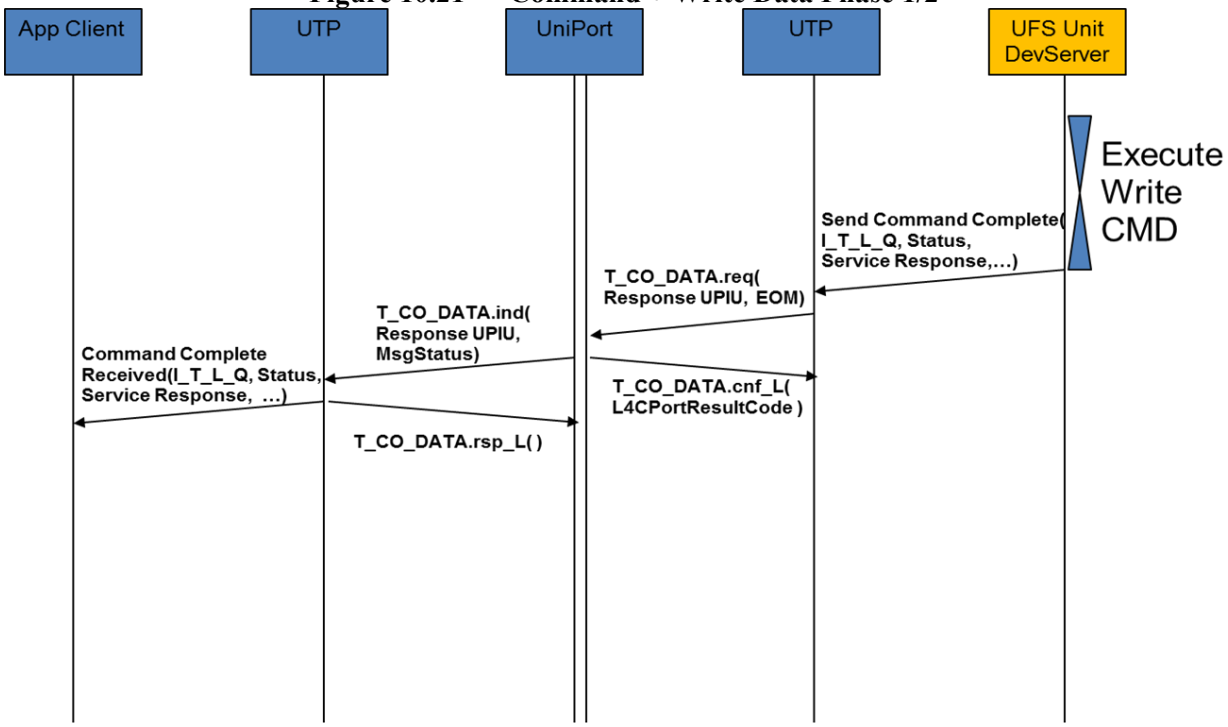


Figure 10.22 — Command + Write Data Phase 2/2

10.9.8 Task Management Function Procedure Calls

See [SAM] for information on this topic. Table 10.24 describes the set of task management functions a UFS device shall support.

10.9.8.1 ABORT TASK

See [SAM] for information on this topic.

If the processing of the task that is requested to be aborted requires Data-Out data transfer, then Target device shall wait until it receives all DATA OUT UPIUs related to any outstanding READY TO TRANSFER UPIUs before sending the task management response.

10.9.8.2 ABORT TASK SET

See [SAM] for information on this topic.

If the processing of one or more tasks in the task set require Data-Out data transfer, then Target device shall wait until it receives all DATA OUT UPIUs related to any outstanding READY TO TRANSFER UPIUs before sending the task management response.

10.9.8.3 CLEAR TASK SET

See [SAM] for information on this topic.

If the processing of one or more tasks in the task set require Data-Out data transfer, then Target device shall wait until it receives all DATA OUT UPIUs related to any outstanding READY TO TRANSFER UPIUs before sending the task management response.

10.9.8.4 LOGICAL UNIT RESET

See [SAM] for information on this topic.

If the processing of one or more tasks in the logical unit requires Data-Out data transfer, then Target device shall wait until it receives all DATA OUT UPIUs related to any outstanding READY TO TRANSFER UPIUs before sending the task management response.

10.9.8.5 QUERY TASK

UFS transport protocols shall support QUERY TASK. See [SAM] for information on this topic.

10.9.8.6 QUERY TASK SET

UFS transport protocols shall support QUERY TASK SET. See [SAM] for information on this topic.

10.9.8.7 Task Management SCSI Transport Protocol Services

See [SAM] for SCSI information on this topic. An application client uses the Send Command transport protocol service request to request a UFS Initiator port transmit a TASK MANAGEMENT REQUEST UPIU via UniPort, as shown in Figure 10.23.

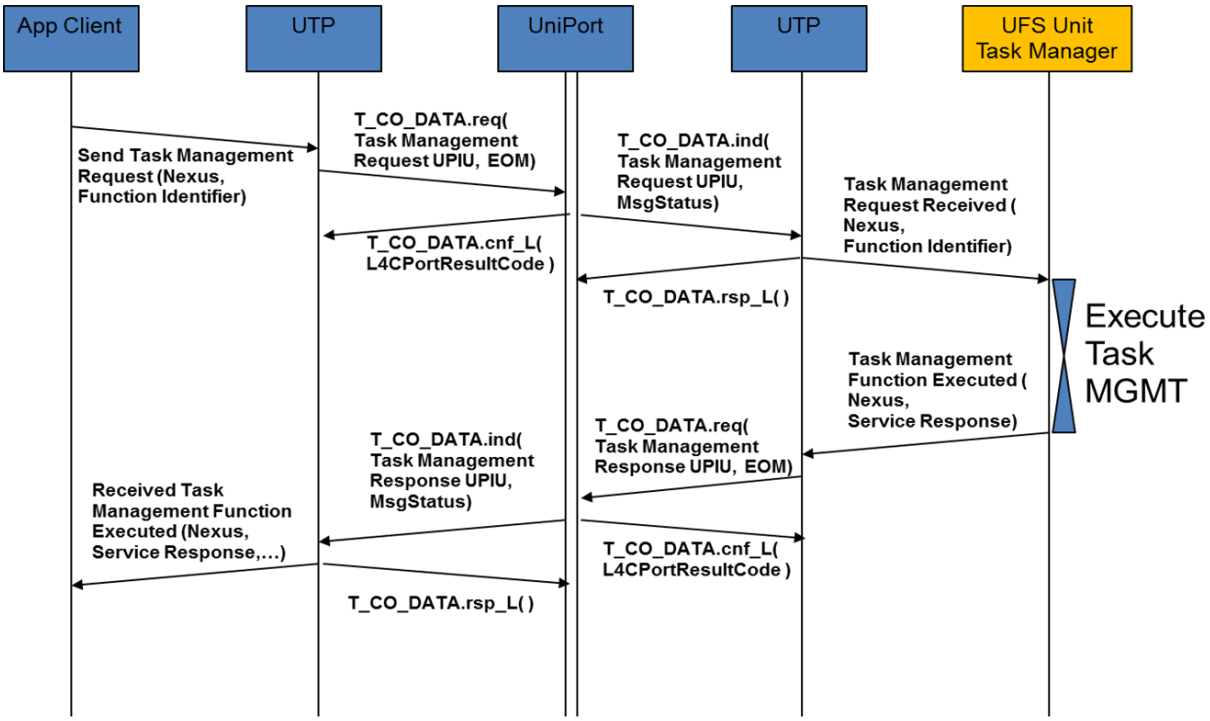


Figure 10.23 — Task Management Function

10.9.9 Query Function Transport Protocol Services

UFS defines Query Function to get/set UFS-specific device-level registers and parameters (not part of SCSI definition).

10.9.9.1 Send Query Request UFS Transport Protocol Service

An application client uses the Send Query Request UFS transport protocol service request to request that a UFS initiator port send a Query Request function.

Send Query Request UFS transport protocol service request:
Send Query Request(I_T Nexus, Operation: Read|Write, Type: UFS|Vendor, Identifier: Descriptor|Attribute|Flag,Length: n, [Index], [Selector], [WriteData]).

Table 10.68 — Send Query Request UFS Transport Protocol Service

Argument	Implementation
Nexus	I_T nexus
Operation	Argument encoding the operation to be performed
Type	Indicates UFS defined or vendor-specific operation
Identifier	Identifier for descriptor type, attribute or flag
Length	Number of bytes to read or write
Index	Index reference for the descriptor, attribute or flag
Selector	Reserved
WriteData	Data to be written in a write query request

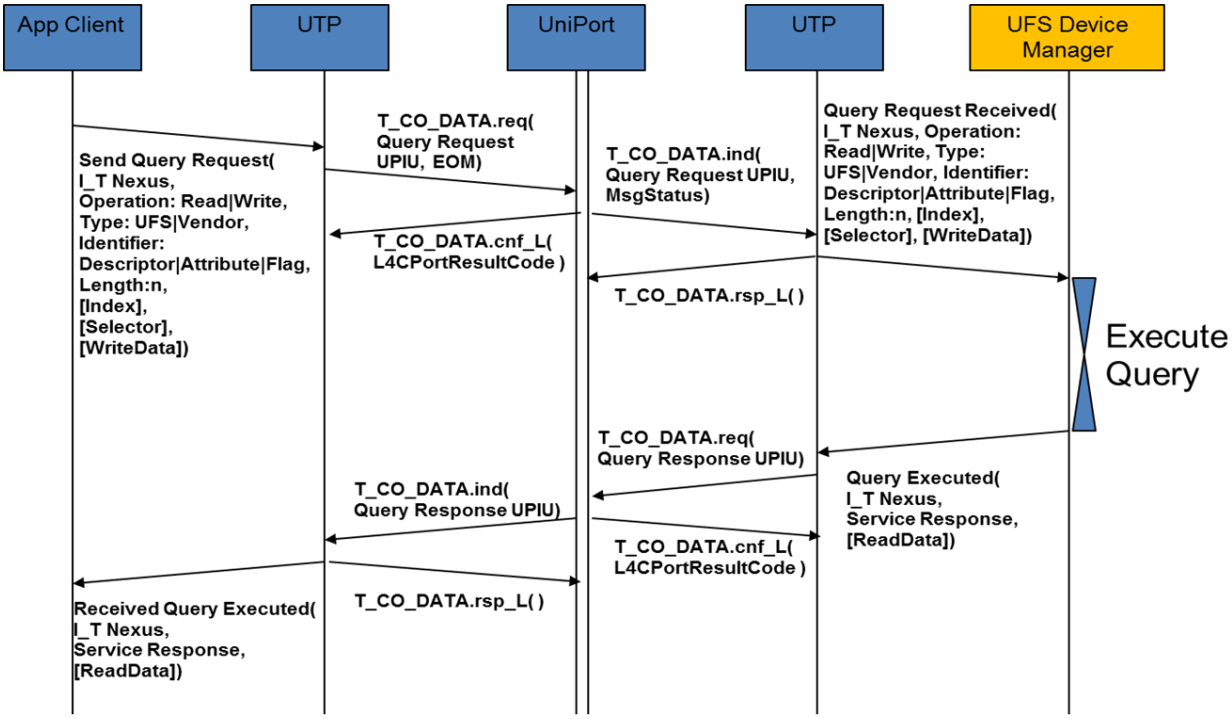


Figure 10.24 — UFS Query Function

10.9.9.2 Query Request Received UFS Transport Protocol Service Indication

A UFS target port uses the Query Request Received UFS transport protocol service indication to notify a UFS device manager that it has received a query function.

Query Request Received UFS transport protocol service indication:
Query Request Received(I_T Nexus, Operation: Read|Write, Type: UFS|Vendor, Identifier:Descriptor|Attribute|Flag,Length:n, [Index], [Selector], [WriteData])

Table 10.69 — Query Request Received UFS Transport Protocol Service Indication

Argument	Implementation
Nexus	I_T nexus
Operation	Argument encoding the operation to be performed
Type	Indicates UFS defined or vendor-specific operation
Identifier	Identifier for descriptor type, attribute or flag
Length	Number of bytes to read or write
Index	Index reference for the descriptor, attribute or flag
Selector	Reserved
WriteData	Data to be written in a write query request

10.9.9.3 Query Function Executed UFS Transport Protocol Service Response

A device manager uses the Query Function Executed UFS transport protocol service response to request that a UFS target port transmit query function executed information.

Query Function Executed UFS transport protocol service response:
Query Executed(I_T Nexus, Service Response,[ReadData])

Table 10.70 — Query Function Executed UFS Transport Protocol Service Response

Argument	Implementation
Nexus	I_T nexus
Service Response	FUNCTION SUCCEEDED
	FUNCTION FAILED
ReadData	Data to be returned in a read query request

10.9.9.4 Received Query Function Executed UFS Transport Protocol Service Confirmation

A UFS initiator port uses the Received Query Function Executed UFS transport protocol service confirmation to notify an application client that it has received task management function executed information.

Received Query Function Executed UFS transport protocol service confirmation:
Received Query Executed(I_T Nexus, Service Response,[ReadData])

Table 10.71 — Received Query Function Executed UFS Transport Protocol Service Confirmation

Argument	Implementation
Nexus	I_T nexus
Service Response	FUNCTION SUCCEEDED
	FUNCTION FAILED
ReadData	Data to be returned in a read query request

11 UFS Application (UAP) Layer – SCSI Commands

11.1 Universal Flash Storage Command Layer (UCL) Introduction

This clause defines the mandatory commands set supported by the UFS device.

Commands may belong to the UFS Native command set or to the UFS SCSI command set.

This version of the standard does not define UFS native commands. These command set may be defined in the future to support specific flash storage or UFS native basic needs.

The UFS SCSI command set (USC) consists of a selection of commands from SCSI Primary Commands [SPC], and SCSI Block Commands [SBC]. Both command types share similar command descriptor block (CDB) format.

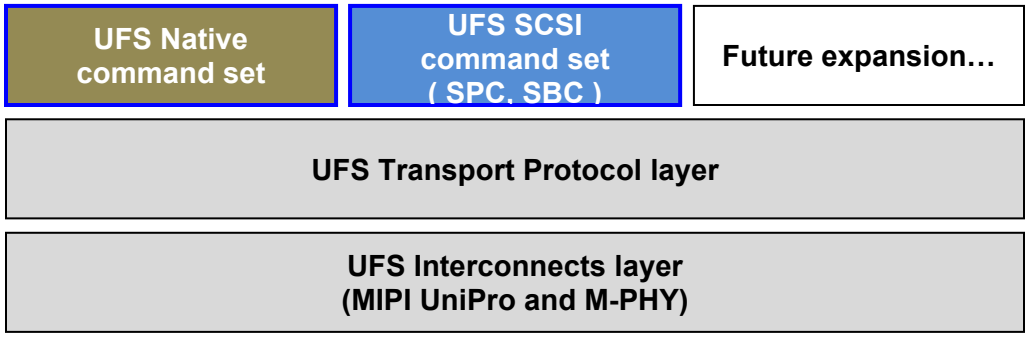


Figure 11.1 — UFS Command Layer

11.1.1 The Command Descriptor Block (CDB)

SCSI command set commands are communicated by sending the SCSI Command Descriptor Block (CDB) to the UFS device. UFS supports only fixed-length CDB formats.

All UFS CDBs shall have an OPERATION CODE field as their first byte and these values are defined by SCSI. Detailed SCSI CDB usage and structure are defined in [SPC].

11.2 Universal Flash Storage Native Commands (UNC)

The UNC are not defined in this version of the standard but may be defined in future versions.

11.3 Universal Flash Storage SCSI Commands

The Basic Universal Flash Storage (UFS) SCSI commands are compatible with [SPC] and [SBC]. Each enabled logical unit (bLUEnable = 01h) shall support the mandatory commands shown in Table 11.1.

Table 11.1 — UFS SCSI Command Set

Command name	Command Support	SCSI Reference	UFS Reference
EXTENDED COPY	O	[SPC]	11.3.27
FINISH ZONE	O	[ZBC]	As in SCSI
FORMAT UNIT	M	[SBC]	11.3.16
GET STREAM STATUS	O	[SBC]	11.3.32
INQUIRY	M	[SPC]	11.3.2
MODE SELECT (10)	M	[SPC]	11.3.3
MODE SENSE (10)	M	[SPC]	11.3.4
POPULATE TOKEN	O	[SBC]	11.3.29
PRE-FETCH (10)	M	[SBC]	11.3.17
PRE-FETCH (16)	O	[SBC]	11.3.18
READ (10)	M	[SBC]	11.3.5
READ (16)	M	[SBC]	11.3.6
READ BUFFER	M	[SPC]	11.3.25
READ CAPACITY (10)	M	[SBC]	11.3.7
READ CAPACITY (16)	M	[SBC]	11.3.8
RECEIVE COPY STATUS	O	[SPC]	As in SCSI
RECEIVE ROD TOKEN INFORMATION	O	[SBC]	11.3.31
REPORT LUNS	M	[SPC]	11.3.11
REPORT ZONES	O	[ZBC]	As in SCSI
REQUEST SENSE	M	[SPC]	11.3.15
RESET WRITE POINTER	O	[ZBC]	As in SCSI
SECURITY PROTOCOL IN (1)	M	[SPC]	11.3.19
SECURITY PROTOCOL OUT (1)	M	[SPC]	11.3.20
SEND DIAGNOSTIC	M	[SPC]	11.3.21
START STOP UNIT	M	[SBC]	11.3.9
SYNCHRONIZE CACHE (10)	M	[SBC]	11.3.22
SYNCHRONIZE CACHE (16)	O	[SBC]	11.3.23
TEST UNIT READY	M	[SPC]	11.3.10
UNMAP	M	[SBC]	11.3.24
VERIFY (10)	M	[SBC]	11.3.12
WRITE (10)	M	[SBC]	11.3.13
WRITE (16)	M	[SBC]	11.3.14
WRITE BUFFER	M	[SPC]	11.3.26
WRITE USING TOKEN	O	[SBC]	11.3.30
M: mandatory, O: optional			
NOTE 1 SECURITY PROTOCOL IN command and SECURITY PROTOCOL OUT command are supported only by the RPMB well known logical unit.			

11.3.1 General information about SCSI commands in UFS

The remaining part of this sub-clause describes the SCSI commands used in UFS devices. Each sub-clause that follows this sub-clause will address an individual command. As the SCSI standards ([SPC] and [SBC] as shown in Table 11.1) fully describe the commands and their CDBs, these paragraphs will focus on UFS usage of these commands, and constraints applicable to UFS devices for these commands. These include, but are not limited to, default parameters for UFS, UPIU requirements, and response and error handling capabilities of UFS devices related to each command.
If a field is not supported by UFS, the sub-clause will so indicate. The device may ignore values in fields that are not supported by UFS.

Some SCSI command names include parentheses with applicable CDB sizes in bytes (e.g., OPERATION CODE (12h)). These names are used as declared in SCSI, and are not modified by UFS.

The following information applies to all SCSI commands used in UFS:

- **CONTROL -** The CONTROL byte is present in several CDB and it is defined in [SAM]. The CONTROL byte is not used in this standard: the CONTROL byte should be set to zero and shall be ignored by UFS devices.
- **Auto Contingent Allegiance (ACA) -** UFS devices shall not support ACA.

11.3.2 INQUIRY Command

See [SPC] for information on this topic.

11.3.2.1 VITAL PRODUCT DATA

See 11.5 for information on UFS device's VPD support requirements.

11.3.2.2 STANDARD INQUIRY DATA

UFS devices shall support the first 36 bytes of standard inquiry data.

The Command CDB shall be sent in a single COMMAND UPIU.

11.3.2.3 Inquiry Command Data Response

- Data returned from an INQUIRY command shall be transferred to the Application Client in a single DATA IN UPIU.
- The Device Server shall transfer the Response Data in the Data Segment area of a DATA IN UPIU.
- An Allocation Length of zero specifies that no data shall be transferred. This condition shall not be considered as an error, and DATA IN UPIU shall not be generated.
- No DATA IN UPIU will be transferred if an error occurs.

11.3.2.4 Inquiry Response Data

A UFS device's standard INQUIRY response data shall be as follows:

- PERIPHERAL DEVICE TYPE shall be 00h, 14h, or 1Eh, dependent on the characteristics of the Logical Unit addressed (see [SPC]).
- VERSION shall be 0Dh.
- RESPONSE DATA FORMAT shall be 02h.
- ADDITIONAL LENGTH shall be 31.
- CMDQUE shall be 1b.
- T10 VENDOR IDENTIFICATION, PRODUCT IDENTIFICATION, and PRODUCT REVISION LEVEL shall follow the rules described in [SPC]. For UFS devices, PRODUCT REVISION LEVEL shall identify the firmware version of the UFS device and shall be uniquely encoded for any firmware modification implemented by the UFS device vendor.
- If copy offloading is supported, 3PC shall be set to 1b.
- All other fields shall be set to 0.

11.3.2.5 Inquiry Command Status Response

- STATUS response will be sent in a single RESPONSE UPIU.

11.3.3 MODE SELECT (10) Command

See [SPC] for information on this topic.UFS devices may support vendor specific mode pages. See 11.4 for details of mode pages for UFS devices. See 11.4.2 for UFS mode page support requirements.

The Command CDB shall be sent in a single COMMAND UPIU.

11.3.3.1 Mode Select Command Parameters

See [SPC] for information on this topic.

For UFS devices:

- PF shall be set to one.
- RTD shall be set to zero.

11.3.3.2 Mode Select Command Data Transfer

The Device Server requests to transfer the mode parameter list from the Application Client data-out buffer by issuing one or more READY TO TRANSFER UPIUs (RTT).

The mode parameter list is delivered in one or more segments sending DATA OUT UPIU packets, as indicated in the RTT requests.

See 11.4.1.2 for details about the mode parameter list.

11.3.3.3 Mode Select Command Status Response

STATUS response shall be sent in a single RESPONSE UPIU.

11.3.4 MODE SENSE (10) Command

See [SPC] for information on this topic.See 11.4 for details of mode pages for UFS devices. See 11.4.2 for UFS mode page support requirements.

The Command CDB shall be sent in a single COMMAND UPIU.

11.3.4.1 Mode Sense Command Parameters

See [SPC] for information on this topic.

For UFS devices:

- LLBAA shall be set to zero.
- DBD shall be set to one.

11.3.4.2 Page Control Function

See [SPC] for information on this topic.

11.3.4.3 Mode Sense Command Data Transfer

Data will be transferred from the Device Server to the Application Client via a series of DATA IN UPIUs

- The data transferred from the Device Server will be contained within the Data Segment of the DATA IN UPIU.

Zero or an incomplete number of DATA IN UPIUs will be transferred if an error occurs before the entire data transfer is complete.

See 11.4.1.2 for details about the mode parameter list.

11.3.4.4 Mode Sense Command Status Response

STATUS response will be sent in a single RESPONSE UPIU.

11.3.5 READ (10) Command

See [SBC] for information on this topic.

The Command CDB shall be sent in a single COMMAND UPIU.

11.3.5.1 Read (10) Command Parameters

- The RDPROTECT field is set to zero for UFS.
- GROUP NUMBER: For zoned logical units the GROUP NUMBER shall be ignored. For conventional logical units the GROUP NUMBER notifies the Target device that the data is linked to a ContextID or to an IO advice hint. To use GROUP NUMBER as an IO advice hint, the device shall declare such usage in the IO Advice Hints Grouping mode page as declared in [SBC]. UFS GROUP NUMBER utilization for read operations is as follows:

GROUP NUMBER Value	Function
0	Default, no Context ID is associated with the read operation.
1 to 15	Context ID. (XXXX from 0001b to 1111b - Context ID value)
16 and greater	Reserved

In case the GROUP NUMBER is set to a reserved value, the operation shall fail and a status response of CHECK CONDITION will be returned along with the sense key set to ILLEGAL REQUEST.

11.3.5.2 Read (10) Command Data Transfer

- The Device Server will read the specified logical block(s) from the medium and transfer them to the Application Client in a series of DATA IN UPIUs.
- The data segment of each DATA IN UPIU shall contain an integer number of logical blocks.
- Zero or an incomplete number of DATA IN UPIUs could be transferred if a read error occurs before the entire data transfer is complete.

11.3.5.3 Read (10) Command Status Response

- Status response will be sent in a single RESPONSE UPIU.

11.3.6 READ (16) Command

See [SBC] for information on this topic. The Command CDB shall be sent in a single COMMAND UPIU.

113.6.1 Read (16) Command Parameters

- See Read (10) Command.

11.3.6.2 Read (16) Command Data Transfer

- The Device Server will read the specified logical block(s) from the medium and transfer them to the Application Client in a series of DATA IN UPIUs.
- The data segment of each DATA IN UPIU shall contain an integer number of logical blocks.
- Zero or an incomplete number of DATA IN UPIUs could be transferred if a read error occurs before the entire data transfer is complete.

11.3.6.3 Read (16) Command Status Response

- Status response will be sent in a single RESPONSE UPIU.

11.3.7 READ CAPACITY (10) Command

See [SBC] for information on this topic. The Command CDB shall be sent in a single COMMAND UPIU.

For UFS devices:

- PMI shall be zero.
- Logical Block Address shall be zero.

Read Capacity (10) Data Response:

- Data returned from a READ CAPACITY (10) command will be transferred to the Application Client in a single DATA IN UPIU.
- The Device Server will transfer READ CAPACITY (10) parameter data (see SBC) in the Data Segment area of a DATA IN UPIU.
- No DATA IN UPIU will be transferred if an error occurs.

11.3.7.1 Read Capacity (10) Parameter Data

For UFS devices, the LOGICAL BLOCK LENGTH IN BYTES shall be at minimum 4096.

11.3.7.2 Read Capacity (10) Status Response

- STATUS response will be sent in a single RESPONSE UPIU.

11.3.8 READ CAPACITY (16) Command

See [SBC] for information on this topic. The Command CDB shall be sent in a single COMMAND UPIU.

See READ CAPACITY (10) for parameter requirements.

11.3.8.1 Read Capacity (16) Data Response

- Data returned from a READ CAPACITY (16) command will be transferred to the Application Client in a single DATA IN UPIU.
- The Device Server will transfer READ CAPACITY (16) parameter data (see SBC) in the Data Segment area of a DATA IN UPIU.
- No DATA IN UPIU will be transferred if an error occurs.

11.3.8.2 Read Capacity (16) Parameter Data

For UFS devices:

- The LOGICAL BLOCK LENGTH IN BYTES field shall be at minimum 4096.P_TYPE, PROT_EN shall be zero.
- LBPME shall be set to zero if the bProvisioningType parameter in the UFS Configuration Descriptor is set to 00h. LBPME shall be set to ‘1’ if bProvisioningType is set to 02h or 03h.
- LBPRZ shall be set according to the bProvisioningType parameter in the UFS Configuration Descriptor.
- If Zoned logical units are supported, then the device shall support the [ZBC] extension to READ CAPACITY (16) (see 13.4.23, Zoned Logical Units).

11.3.8.3 Read Capacity (16) Status Response

- STATUS response will be sent in a single RESPONSE UPIU.

11.3.9 START STOP UNIT Command

- See [SBC] for information on this topic.

The Command CDB shall be sent in a single COMMAND UPIU.

11.3.9.1 START STOP UNIT Parameters

See 7.4 for definitions of UFS power modes and associated POWER CONDITIONS parameters for use with START STOP UNIT on UFS devices.

For UFS devices:

- Power Condition Modifier shall be zero.
- LOEJ shall be zero.

11.3.9.2 START STOP UNIT Data Response

- The START STOP UNIT command does not have a data phase.
- No DATA IN or DATA OUT UPIUs are transferred.

11.3.9.3 Start STOP UNIT STATUS Response

- STATUS response will be sent in a single RESPONSE UPIU.

11.3.10 TEST UNIT READY Command

The Command CDB shall be sent in a single COMMAND UPIU.

11.3.10.1 TEST UNIT READY Data Response

- The TEST UNIT READY command does not have a data response.
- No DATA IN or DATA OUT UPIUs are transferred.

11.3.10.2 TEST UNIT READY STATUS Response

- STATUS response will be sent in a single RESPONSE UPIU.

11.3.11 REPORT LUNS Command

The Command CDB shall be sent in a single COMMAND UPIU

11.3.11.1 Report LUNS Command Parameters

UFS devices only support SELECT REPORT values of 00h, 01h, and 02h.

11.3.11.2 Report LUNS Data Response

- Data returned from a REPORT LUNS command will be transferred to the Application Client in one or more DATA IN UPIUs.
- Most likely one DATA IN UPIU.
- UFS uses two [SAM] formats to populate the REPORT LUNS parameter data:
  - Peripheral device addressing format with the BUS IDENTIFIER field set to zero.
  - Well known logical unit extended addressing format.
- For both formats, the LUN or W-LUN field contains the value of the UNIT_NUMBER_ID from the UFS LUN format, as shown in 11.3.11.4.

11.3.11.3 Report LUNS Status Response

- Status response will be sent in a single RESPONSE UPIU.

11.3.11.4 UFS LUN Format

Table 11.2 — Format of LUN field in UPIU

7	6	5	4	3	2	1	0
WLUN_ID	UNIT_NUMBER_ID						

The UFS 8-bit LUN field in UPIU supports two types of LUN addressing:

- If WLUN_ID bit = ‘0’ then the UNIT_NUMBER_ID field addresses a standard logical unit (LUN).
- If WLUN_ID bit = ‘1’ then the UNIT_NUMBER_ID field addresses a well known logical unit (W-LUN).

Up to 128 LUN’s and up to 128 W-LUNs.

- $0 \leq$ UNIT_NUMBER_ID ≤ 127.

The following table defines the logical unit number for UFS well known logical units (WLUN_ID bit set to ‘1’).

Table 11.3 — Well Known Logical Unit Numbers

Well known logical unit	WLUN_ID	UNIT_NUMBER_ID	LUN Field in UPIU
REPORT LUNS	1b	01h	81h
UFS Device	1b	50h	D0h
RPMB	1b	44h	C4h
BOOT	1b	30h	B0h

11.3.12 VERIFY (10) Command

See [SBC] for information on this topic.The Command CDB shall be sent in a single COMMAND UPIU.

11.3.12.1 Verify Command Parameters

For UFS devices, the following parameters shall be zero:

- VRPROTECT
- DPO
- BYTCHK
- GROUP NUMBER
- Reserved and Obsolete

11.3.12.2 Verify Command Data Transfer

- The VERIFY command does not have a data response.
- No DATA IN or DATA OUT UPIUs are transferred.

11.3.12.3 Verify Command Status Response

- STATUS response will be sent in a single RESPONSE UPIU.

11.3.13 WRITE (10) Command

See [SBC] for information on this topic. The Command CDB shall be sent in a single COMMAND UPIU.

11.3.13.1 Write (10) Command Parameters

- For UFS, the WRPROTECT field shall be zero.
- GROUP NUMBER: For zoned logical units, GROUP NUMBER is a permanent stream identifier. For conventional logical units, GROUP NUMBER notifies the Target device that the data has System Data characteristics or linked to a ContextID or to an IO advice hint. To use GROUP NUMBER as an IO advice hint, the device shall declare such usage in the IO Advice Hints Grouping mode page as declared in [SBC]. UFS GROUP NUMBER utilization for write operations is as follows:

GROUP NUMBER Value	Function
0	Default, no Context ID or System Data characteristics is associated with the write operation.
1 to 15	Context ID. (XXXX from 0001b to 1111b - Context ID value)
16	IO advice hint indicating data has System Data characteristics
17	Indicates that this WRITE should be written to the WriteBooster Buffer
18 to 23	Reserved
24	IO advice hint indicating pinned data in Pinned Partial Flush Mode of WriteBooster Buffer
25 and greater	Reserved

In case the GROUP NUMBER is set to a reserved value, then the operation shall fail and a status response of CHECK CONDITION will be returned along the sense key set to ILLEGAL REQUEST.

11.3.13.2 Write(10) Command Data Transfer

The Device Server requests to transfer the specified logical block(s) from the Application Client data-out buffer by issuing a series of READY TO TRANSFER UPIUs (RTT).

The data is delivered in one or more segments sending DATA OUT UPIU packets, as indicated in the RTT requests. The data contained in DATA OUT UPIU is written.

The number of bytes requested and the Data Buffer Offset field in each RTT shall both be integer multiples of the Logical Block Size (bLogicalBlockSize).

The data segment of each DATA OUT UPIU shall contain an integer number of logical blocks.

Zero or an incomplete number of segments may be requested, if an error occurs before the entire data transfer is complete.

11.3.13.3 Write (10) Command Status Response

- STATUS response will be sent in a single RESPONSE UPIU.

11.3.14 WRITE (16) Command

See [SBC] for information on this topic. The Command CDB shall be sent in a single COMMAND UPIU.

11.3.14.1 Write (16) Command Parameters

See Write (10) Command.

11.3.14.2 Write (16) Command Data Transfer

See Write (10) Command.

11.3.14.3 Write (16) Command Status Response

- STATUS response will be sent in a single RESPONSE UPIU.

11.3.15 REQUEST SENSE Command

See [SPC] for more information on this topic.UFS devices will return a fixed format data record of 18 bytes of sense data as described in 10.7.2. The Command CDB shall be sent in a single COMMAND UPIU.

UFS devices are not required to support descriptor format sense data.

11.3.15.1 Request Sense Data Response

- Data returned from a REQUEST SENSE command will be transferred to the Application Client in a single DATA IN UPIU.
- The Device Server will transfer up to 18 bytes of Response Data in the Data Segment area of a DATA IN UPIU.
    - Return 18 bytes if Allocation Length in CDB ≥ 18.
    - Return Allocation Length bytes if Allocation Length in CDB < 18.
    - An Allocation Length of zero specifies that no data shall be transferred. This condition shall not be considered as an error, and DATA IN UPIU shall not be generated.
- Data will be returned in the indicated Sense Data Format.

11.3.15.2 Sense Data

See 10.7.2.

11.3.15.3 Sense Key

See 10.7.2.

11.3.15.4 Request Sense Status Response

- STATUS response will be sent in a single RESPONSE UPIU.

11.3.16 FORMAT UNIT Command

See [SBC] for more information on this topic.

A FORMAT UNIT command sent to the Device well known logical unit requests the device format all enabled logical units except the RPMB well known logical unit (see 12.2.3.4).

If the medium is write-protected, then the command shall be terminated with CHECK CONDITION status with the sense key set to DATA PROTECT.

Following a successful format operation all LBAs:

a) shall be mapped on a fully provisioned logical unit (bProvisioningType set to 00h).
b) shall be unmapped on a thin provisioned logical unit (bProvisioningType set to 02h or 03h).

For an unwritten LBA in a formatted logical unit specified by a read operation, the device server shall send user data with all bits set to zero to the data in buffer.

The Command CDB shall be sent in a single COMMAND UPIU.

11.3.16.1 Format Unit Command Parameters

For a UFS device, LONGLIST and CMPLST fields may be set to either one or zero. All other fields shall be set to zero.

11.3.16.2 Format Unit Command Data Transfer

- If needed, the Device Server requests to transfer the FORMAT UNIT parameter list from the Application Client data-out buffer by issuing a series of READY TO TRANSFER UPIUs (RTT).
- The FORMAT UNIT parameter list is delivered in one or more segments sending DATA OUT UPIU packets, as indicated in the RTT requests.
- Zero or an incomplete number of segments may be requested if an error occurs before the entire data transfer is complete.

11.3.16.3 Format Unit Command Status Response

- STATUS response will be sent in a single RESPONSE UPIU.

11.3.17 PRE-FETCH (10) Command

See [SBC] for more information on this topic. The Command CDB shall be sent in a single COMMAND UPIU.

11.3.17.1 PRE-FETCH (10) Command Parameters

For UFS devices, GROUP NUMBER and Obsolete fields shall be set to zero.

11.3.17.2 PRE-FETCH Command Data Transfer

The PRE-FETCH command does not have a data transfer phase.

- No DATA IN or DATA OUT UPIUs are transferred.

11.3.17.3 PRE-FETCH Command Status Response

- STATUS response will be sent in a single RESPONSE UPIU.

11.3.18 PRE-FETCH (16) Command

See PRE-FETCH (10) command and [SBC] for details.

11.3.19 SECURITY PROTOCOL IN Command

See [SPC] for more information on this topic.

11.3.19.1 SECURITY PROTOCOL IN Command Parameter

- SCSI has reserved SECURITY PROTOCOL = ECh for JEDEC Universal Flash Storage (see [SPC]).
- UFS devices shall support the following SECURITY PROTOCOL value:
  - ECh: JEDEC UFS
- Support of other SECURITY PROTOCOL values is device specific.

11.3.19.2 SECURITY PROTOCOL IN Command Data Transfer

- The Device Server transfers security protocol data to the Application Client using one or more DATA IN UPIUs.

11.3.19.3 SECURITY PROTOCOL IN Command Status Response

- Status response shall be sent in a single RESPONSE UPIU.

11.3.20 SECURITY PROTOCOL OUT Command

See [SPC] for more information on this topic.

11.3.20.1 SECURITY PROTOCOL OUT Command Parameter

- SCSI has reserved SECURITY PROTOCOL = ECh for JEDEC Universal Flash Storage (see [SPC]).
- UFS devices shall support the following SECURITY PROTOCOL value:
  - ECh: JEDEC UFS
- Support of other SECURITY PROTOCOL values is device specific.

11.3.20.2 SECURITY PROTOCOL OUT Command Data Transfer

The Device Server requests to transfer data from the Application Client data-out buffer by issuing a series of READY TO TRANSFER UPIUs (RTT).

The data is delivered in one or more segments sending DATA OUT UPIU packets, as indicated in the RTT requests.

Zero or an incomplete number of segments may be requested, if an error occurs before the entire data transfer is complete.

11.3.20.3 SECURITY PROTOCOL OUT Command Status Response

- Status response shall be sent in a single RESPONSE UPIU.

11.3.21 SEND DIAGNOSTIC Command

See [SPC] for more information on this topic. The Command CDB shall be sent in a single COMMAND UPIU.

11.3.21.1 Send Diagnostic Parameters

Parameters for the SEND DIAGNOSTIC command are as described in [SPC].

11.3.21.2 Send Diagnostic Command Data Transfer

The SEND DIAGNOSTIC command will transfer out the number of bytes specified by the Parameter List Length. If that value is zero then no data out transfer will occur. The Device Server will request the transfer of the specified bytes from the Application Client by issuing a series READY TO TRANSFER UPIU (RTT). RTT will be followed by DATA OUT UPIU containing the number of bytes to transfer.

11.3.21.3 Send Diagnostic Command Status Response

- STATUS response will be sent in a single RESPONSE UPIU.

11.3.22 SYNCHRONIZE CACHE (10) Command

See [SBC] for more information on this topic.

11.3.22.1 Synchronize Cache Command Parameters

For UFS devices, the GROUP NUMBER and Obsolete fields shall be set to zero.

11.3.22.2 Synchronize Cache Command Data Transfer

The SYNCHRONIZE CACHE command does not have a data transfer phase.

- No DATA IN or DATA OUT UPIUs are transferred.

11.3.22.3 Synchronize Cache Command Status Response

- STATUS response will be sent in a single RESPONSE UPIU.

11.3.23 SYNCHRONIZE CACHE (16) Command

See SYNCHRONIZE CACHE (10) command and [SBC] for details.

11.3.24 UNMAP Command

See [SBC] for more information on this topic.

UFS supports UNMAP only for logical units with thin provisioning enabled (see bProvisioningType in 14.1.5.5).

In UFS, a thin provisioned logical unit shall have sufficient physical memory resources to support the logical block address space when the device is configured by the user. Mapped State and De-Allocated State are mandatory in a UFS thin provisioned logical unit.
For UFS devices:

- GROUP NUMBER shall be set to zero
- ANCHOR shall be set to zero. If the device receives an UNMAP command with ANCHOR set to one, 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

11.3.24.1 UNMAP parameter list transfer

- The Device Server requests to transfer the UNMAP parameter list from the Application Client data-out buffer by issuing a series of READY TO TRANSFER UPIUs (RTT).
- The UNMAP parameter list is delivered in one or more segments sending DATA OUT UPIU packets, as indicated in the RTT requests.
- Zero or an incomplete number of segments may be requested, if an error occurs before the entire data transfer is complete.

11.3.24.2 UNMAP Command Status Response

- STATUS response will be sent in a single RESPONSE UPIU.

11.3.25 READ BUFFER Command

See [SPC] for more information on this topic, which is under the READ BUFFER(10) command sub-clause of that standard. The Command CDB shall be sent in a single COMMAND UPIU.

For UFS devices:

- The MODE SPECIFIC field is unused and shall be set to zero.
- The following MODEs shall be supported:
  - Data mode
  - Error history mode
- Data will be transferred from the Device Server to the Application Client via a series of DATA IN UPIUs.
  - The data transferred from the Device Server will be contained within the Data Segment of the DATA IN UPIU.
- Zero or an incomplete number of DATA IN UPIUs will be transferred if an error occurs before the entire data transfer is complete.

11.3.25.1 Error History Mode

UFS devices support a subset of the error history mode functionality as described in [SPC]. The subset has the following restrictions:

- Error history snapshots are not supported. The returned error history may be real time contents or may be the contents captured at a vendor specific point in time.
- The following error history BUFFER IDs are supported:
  - Return error history directory.
  - Return error history.
  - Clear error history (for UFS this is equivalent to clear error history I_T nexus).
- The retrieved error history directory shall have EHS_RETRIEVED, EHS_SOURCE, and CLR_SUP fields set to zero.
- Error history directory entries shall support only SUPPORTED BUFFER ID and MAXIMUM AVAILABLE LENGTH fields. All other directory entry fields shall be zero.

11.3.25.2 Read Buffer Command Status Response

- Status response will be sent in a single RESPONSE UPIU.

11.3.26 WRITE BUFFER Command

See [SPC] for more information on this topic.

- The Command CDB shall be sent in a single COMMAND UPIU.
- UFS devices shall support the following MODE:
  - Data

11.3.26.1 Field Firmware Update

UFS Field Firmware Update (FFU) is based on microcode download definition in [SPC].

[SPC] describes multiple operation modes for microcode download which are selected using the MODE field in the WRITE BUFFER command. UFS supports only the MODE field value 0Eh: “Download microcode with offsets, save, and defer active”.

The deferred microcode shall be activated and no longer considered deferred when a power on or a hard reset occurs. Note that in UFS, START STOP UNIT command, FORMAT UNIT command or WRITE BUFFER command (MODE = 0Fh) will not activate the microcode.

UFS FFU uses the following mechanism:

1) Host delivers the microcode using one or more WRITE BUFFER commands through any logical unit which supports the WRITE BUFFER command. The host specifies: MODE = 0Eh, BUFFER OFFSET, which should be aligned to 4 Kbyte, BUFFER ID = 00h, and the PARAMETER LIST LENGTH field indicating the number of bytes to be transferred. All WRITE BUFFER commands should be sent to the same logical unit with task attribute set to simple or ordered. In the sequence of WRITE BUFFER commands used to deliver the microcode, the BUFFER OFFSET values should be in increasing order and it should start from zero.
2) bFFUTimeout indicates the maximum time in which the device may handle the WRITE BUFFER command. Within this time access to the device is limited or not possible.
3) Following a successful delivery of the microcode, the host activates the new firmware using a hardware reset or a power cycle. The UFS device shall use new firmware upon hard reset or power up. Host should be aware that the first initialization flow after a successful delivery of the microcode may be longer than usual.
4) After device initialization, the host should read bDeviceFFUStatus attribute and verify that the new firmware was updated successfully.

Other modes of WRITE BUFFER command are not supported by the UFS device for FFU process.

11.3.26.2 Write Buffer Command Data Transfer

The Device Server requests to transfer the buffer data from the Application Client data-out buffer by issuing a series of READY TO TRANSFER UPIUs (RTT).

The buffer data is delivered in one or more segments sending DATA OUT UPIU packets, as indicated in the RTT requests.

Zero or an incomplete number of segments may be requested, if an error occurs before the entire data transfer is complete.

The received data is written to the location specified by the BUFFER OFFSET field within the buffer specified by the BUFFER ID field.

11.3.26.3 Write Buffer Command Status Response

- STATUS response will be sent in a single RESPONSE UPIU.

11.3.27 EXTENDED COPY Command

If the EXTENDED COPY command is supported, it shall support copying within a single logical unit and also from one logical unit to another logical unit. Support for held data is not required. Support for aborting an EXTENDED COPY operation is not required either.

Regarding the EXTENDED COPY parameter list fields:

- The PARAMETER LIST FORMAT field shall have the value 01h.
- The STR (sequential striped) field may be ignored by UFS devices.
- The PRIORITY field shall be supported. The UFS device may choose to use or ignore this information for command scheduling..
- The G_SENSE bit shall be supported. This bit indicates whether or not sense data is included with the GOOD status, e.g. the number of segment descriptors that have been processed. The G_SENSE bit setting overrides the “no Sense Data on GOOD status” rule described in the RESPONSE UPIU description (see 10.7.2.1).
- Support for CSCD descriptor type E4h (identification descriptor) is mandatory. Support for other descriptor types is optional.
- Support for segment descriptor type 02h (block device to block device) is required. Support for other segment descriptor types is optional.
- Support for inline data is optional.

A Copy Source Copy Destination (CSCD) descriptor specifies either the source or the destination of a copy operation. Regarding the CSCD descriptors:

- The DESCRIPTOR TYPE CODE field shall have the value E4h (Identification Descriptor).
- The LU ID TYPE field shall be ignored. This is required by SPC for identification descriptors.
- The PERIPHERAL DEVICE TYPE field shall have the value 0 (direct access block device) or 14h (zoned block device).
- The RELATIVE INITIATOR PORT IDENTIFIER field shall have the value 0.
- CODE SET, ASSOCIATION, DESIGNATOR TYPE, DESIGNATOR LENGTH and DESIGNATOR fields shall match one of the descriptors reported in the Device Identification VPD page.

A segment descriptor refers to a source CSCD, destination CSCD and includes a number of blocks, source LBA and destination LBA. Regarding the segment descriptors:

- Support for DESCRIPTOR TYPE CODE 02h is mandatory.
- Support for the FCO (Fast Copy Only) bit is optional.
- Support for DC (Destination Count) 0 and 1 is mandatory. The destination count (DC) bit specifies whether the BLOCK DEVICE NUMBER OF BLOCKS field refers to the source or destination device. A DC bit set to zero specifies that the BLOCK DEVICE NUMBER OF BLOCKS field refers to the source device. A DC bit set to one specifies that the BLOCK DEVICE NUMBER OF BLOCKS field refers to the destination device.
- Support for CSCD descriptor ID values ≥ C000h is optional.
- Support for PAD = 0 and CAT = 0 is mandatory. Support for other PAD, CAT combinations is optional. These bits control how to handle residual source and destination data. Such residuals can only occur if source and destination have different block sizes.

11.3.27 EXTENDED COPY Command (cont'd)

If any field has a value that is not supported then the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID COMMAND OPERATION CODE.

11.3.28 POPULATE TOKEN Command

Regarding the POPULATE TOKEN and the POPULATE TOKEN parameter list fields:

- All possible values of the LIST IDENTIFIER field shall be supported.
- The IMMED field shall be zero. This means that a status is reported to the initiator after the POPULATE TOKEN operation has finished.
- The INACTIVITY TIMEOUT field shall be supported.
- Support for other ROD TYPE field values than 0001 0000h (access upon reference) is optional.

If any field has a value that is not supported then the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID COMMAND OPERATION CODE.

11.3.29 WRITE USING TOKEN Command

Regarding the WRITE USING TOKEN and the WRITE USING TOKEN parameter list fields:

- The LIST IDENTIFIER field shall be supported.
- The DEL_TKN field shall be supported.
- The OFFSET INTO ROD field shall be zero.
- The ROD TOKEN field (512 bytes) shall be supported. This field specifies the copy operation source. The block device range descriptors in the parameters of this command specify the copy operation destination.

If any field has a value that is not supported then the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID COMMAND OPERATION CODE.

11.3.30 RECEIVE ROD TOKEN INFORMATION Command

The RECEIVE ROD TOKEN INFORMATION command implementation shall be as specified in the ANSI SPC and SBC standards. This command can be used to retrieve the ROD token created by the POPULATE TOKEN command. This command can also be used to retrieve status information about a WRITE USING TOKEN command. The LIST IDENTIFIER field in the RECEIVE ROD TOKEN INFORMATION command identifies the POPULATE TOKEN or WRITE USING TOKEN command to retrieve information about.

If any field has a value that is not supported then the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID COMMAND OPERATION CODE.

11.3.31 GET STREAM STATUS Command

Supporting the GET STREAM STATUS command is mandatory for zoned logical units only. Additionally, the PERM bit must be set in the GET STREAM STATUS response for zoned logical units.

11.4 Mode Pages

This sub-clause describes the mode pages used with MODE SELECT command and MODE SENSE command. Subpages are identical to mode pages except that they include a SUBPAGE CODE field that further differentiates the mode page contents.

11.4.1 Mode Page Overview

11.4.1.1 Mode Page/Subpage Codes

See [SPC] for information on this topic. UFS devices are not required to support subpages.

11.4.1.2 Mode Parameter List Format

See [SPC] for information on this topic. UFS does not implement the Block Descriptor field.

11.4.1.3 Mode Parameter Header

See [SPC] for information on this topic. For UFS devices:

- See [SBC] for device type command standard requirements.
- The WP bit shall be set to one when the entire logical unit is write-protected by any method.
- LONGLBA shall be set to zeroBLOCK DESCRIPTOR LENGTH shall be set to zero.

11.4.2 UFS Supported Pages

Table 11.4 shows the mode pages supported by UFS devices. This standard does not define any additional subpages.

Table 11.4 — UFS Supported Pages

PAGE NAME	Source
CONTROL	[SPC]
READ-WRITE ERROR RECOVERY	[SBC]
CACHING	[SBC]
IO Advice Hints Grouping	[SBC]
ALL PAGES	[SPC]
ALL SUBPAGES	[SPC]

UFS devices may support vendor specific mode pages.

11.4.2.1 Control Mode Page

See [SPC] for information on this topic.

A UFS device's Control mode page shall have default values as follows:

- The QUEUE ALGORITHM MODIFIER field has a default value of one.
- Default values for PS, BUSY TIMEOUT PERIOD, and EXTENDED SELF-TEST COMPLETION TIME fields are device specific.
- All other fields have a default value of zero.

A UFS device's Control mode page fields shall be changeable as follows:

- SWP shall be changeable.
- TST and BUSY TIMEOUT PERIOD shall not be changeable.
- All other fields are changeable as documented in the device's vendor datasheet.

11.4.2.2 Read-Write Error Recovery Mode Page

See [SBC] for information on this topic.

A UFS device's Read-Write Error Recovery mode page shall have default values as follows:

- The AWRE field has a default value of one.
- Default values for PS, READ RETRY COUNT, and RECOVERY TIME LIMIT fields are device specific.
- All other fields have a default value of zero.

A UFS device's vendor datasheet describes which Read-Write Error Recovery mode page fields are changeable.

11.4.2.3 Caching Mode Page

See [SBC] for information on this topic.

A UFS device's Caching Mode Page mode page shall have default values as follows:

- The WCE field has a default value of one.
- All other fields have a default value of zero.

A UFS device's Caching Mode Page mode page fields shall be changeable as follows:

- WCE and RCD shall be changeable.
- All other fields are changeable as documented in the device's vendor datasheet.

11.4.2.4 IO Advice Hints Grouping Mode Page

For zoned logical units, IO ADVICE HINTS MODE shall be 0 and ST_ENBLE shall be 1 in group descriptors that represent permanent streams.

11.5 Vital Product Data Parameters

See [SPC] for information on this topic.

A UFS device shall support the following VPD pages:

- Supported VPD Pages.
- Extended INQUIRY Data:
  - GROUP_SUP shall be 1 in the Extended INQUIRY Data VPD page for usages as described in [SPC] and [SBC]. For UFS's ContextID usage, GROUP_SUP is set to zero.
- Mode Page Policy:
  - Mode page policy descriptors shall have their MODE PAGE POLICY field set to zero.

If the device supports the zoned logical units feature (see 13.4.23), the device shall support the following additional VPD pages:

- Block Limits.
- Block Limits Extension:
  - If the device supports the zoned logical units feature (see 13.4.23), the RSCS bit shall be set to 1.
- Block Device Characteristics.
- Logical Block Provisioning.
- Zoned Block Device Characteristics (for zoned logical units only, see [ZBC]):
  - URSWRZ shall be set to 1 for this VPD page.

If the EXTENDED COPY command is supported, the third party copy VPD page shall be supported and the following descriptors shall be present in this VPD page:

- Block Device ROD Limits (mandatory only if the POPULATE TOKEN command and the WRITE USING TOKEN command are supported).
- Supported commands.
- Parameter data.
- General copy operations.

Support for all other VPD pages is as defined by [SPC].

12 UFS Security

This sub-clause summarizes UFS device security features and the implementation details. These features include: Secure mode operation, data and register protection, RPMB and reset.

12.1 UFS Security Feature Support Requirements

The security features defined in this standard are mandatory for all devices.

The following security features are defined: replay protected memory block (RPMB), secure mode and different types of logical unit write protection.

12.2 Secure Mode

12.2.1 Description

UFS devices will be used to store user's personal and/or corporate data information. The UFS device provides a way to remove the data permanently from the device when requested, ensuring that it cannot be retrieved using reverse engineering on the memory device.

The UFS device shall support a secure and insecure mode of operation. In the secure mode all operations that result in the removal or retiring of information on the device will purge this information in a secure manner, as outlined in 12.2.2.1, Secure Removal.

The secure mode is applied at the logical unit level, so different logical unit may have different secure modes.

12.2.2 Requirements

12.2.2.1 Secure Removal

The way in which data is removed securely from the device is dependent on the type of memory technology that is used to implement the UFS device. Three common methods that apply to most memory types implemented at the time of this spec are:

1) The device controller shall issue an erase operation to the addressed location.
2) The device controller shall overwrite the addressed locations with a single character and erase the device.
3) The device controller shall overwrite the addressed locations with a character, its complement, then a random character.

UFS devices shall support at least one secure removal method.

12.2.2.2 Erase Operation

Erase is an operation that moves data from the mapped address space to the unmapped address space. Logical blocks where erase was applied will be set to the erased value of zero. This operation places no requirement on what the device is required to do with the data in the unmapped address space. After an erase is executed, software on the host should not be able to retrieve the erased logical block data.

The minimum data range that an erase operates on is the logical block.

12.2.2.3 Discard Operation

Discard is a non-secure variant of the erase functionality. The distinction between discard and erase is the device behavior where the device is not required to guarantee that host would not retrieve the original data from one or more LBAs that were marked for discard when a read operation is directed to the LBAs.

12.2.2.4 Purge Operation

The Purge operation shall be performed on physical blocks that are not being used to store logical block data (e.g.,physical blocks previously used to store logical block data). When the operation is executed it results in removing all the data from such physical blocks. Note that data of LBA that were discarded (bProvisioningType = 02h) may not be removed. This is done in accordance with the bSecureRemovalType parameter value of the Device Descriptor. This mode allows the host system to protect against die level attacks.

12.2.2.4.1 RPMB Purge Overview

RPMB Purge is a variant of the Purge operation, targeting an RPMB region where overwritten physical copies are made un-recoverable by the device. This allows the host system to overwrite logical blocks in an RPMB region, send RPMB Purge Enable Request message, and receive acknowledgement that the device has erased the overwritten physical copies of those logical blocks in accordance with bSecureRemovalType. The benefit of RPMB Purge is that the purge targets an RPMB region's physical blocks, based on how the device manages physical blocks in RPMB region. Therefore, an RPMB Purge to these regions may be treated as an independent region or as an entire RPMB LU. This may allow the device to minimize processing time without incurring the penalty of erasing all the device's physical blocks. Note: the write counter should only increment for the region of the RPMB purge request, not for other regions. However, the life time of the RPMB regions shall be the RPMB LU life time as highlighted in "bRPMBLifeTimeEst" status.

While the RPMB Purge is in progress, any other operation received by the device may be delayed or rejected by the device. When rejected, device shall return a good status with the result code (000Ch/008Ch), "Rejected, a RPMB purge operation in progress".

RPMB Purge operation is a mandatory feature. Note: since RPMB purge will impact the remaining life of RPMB LU, the host should use RPMB Purge judiciously.

12.2.3 Implementation

12.2.3.1 Erase

The erase functionality is implemented using the UNMAP command and it is enabled if the bProvisioningType parameter in the Unit Descriptor is set to 03h (TPRZ = 1).

The device behavior shall comply with the UNMAP definition in [SBC] when the TPRZ bit in the READ CAPACITY(16) parameter data is set to one. As defined in [SBC],

- The UNMAP command causes a mapped LBA to transition from mapped state to deallocated state if an unmap operation completes without error.
- Since the TPRZ bit is set to one if the erase functionality is enabled, a READ command specifying a deallocated LBA shall return zero.
- The device server may maintain a deallocated LBA in deallocated state until a write operation specifying that LBA is completed without error.
- Or, the device server may transition a deallocated LBA from deallocated state to mapped state at any time (autonomous state transition). For UFS, if TPRZ bit is set to one and an autonomous transition to the mapped state occurs, the LBA shall be mapped to a physical block(s) containing data with all bits set to zero.

LBAs to be erased may be aligned to multiples of the dEraseBlockSize parameter value, where it is possible, to minimize performance impact. dEraseBlockSize is a parameter included in the Unit Descriptor.

12.2.3.2 Discard

The discard functionality is implemented using the UNMAP command and it is enabled if the bProvisioningType parameter in the Unit Descriptor is set to 02h (TPRZ = 0). The device behavior shall comply with the UNMAP definition in [SBC] when the TPRZ bit in the READ CAPACITY(16) parameter data is set to zero.

As defined in [SBC],

- The UNMAP command causes a mapped LBA to transition from mapped state to deallocated state if an unmap operation completes without error.
- Since the TPRZ bit is set to zero if the discard functionality is enabled, a READ command specifying a deallocated LBA may return any data.
- The device server may maintain a deallocated LBA in deallocated state until a write operation specifying that LBA is completed without error.
- Or, the device server may transition a deallocated LBA from deallocated state to mapped state at any time (autonomous state transition). For UFS, if TPRZ bit is set to zero and an autonomous transition to the mapped state occurs, the LBA shall be mapped to a physical block(s) containing any data including the original data before UNMAP operation.

LBAs to be discarded may align to multiples of the dEraseBlockSize where possible to minimize performance impact. dEraseBlockSize is a parameter included in the Unit Descriptor.

12.2.3.3 Purge Operation

The purge operation is implemented via Query Functions with Attributes and Flags. In particular, the fPurgeEnable flag allows to enable or disable the execution of a purge operation, and the bPurgeStatus attribute provides information about the operation status.

- fPurgeEnable flag
  - Write only volatile flag, set to zero after power on or reset.
  - Purge operation is enabled when this flag is equal to one, otherwise it is disabled.
  - This flag can only be set when the command queue of all logical units are empty.
  - This flag is automatically cleared by the UFS device when the operation completes or an error condition occurs.
  - This flag can be cleared by the host to interrupt an ongoing purge operation.
- bPurgeStatus attribute
  - Read only attribute.
  - This attribute can be set to one of the following values:
    - 00h: Idle (purge operation disabled).
    - 01h: Purge operation in progress.
    - 02h: Purge operation stopped prematurely by the host.
    - 03h: Purge operation completed successfully.
    - 04h: Purge operation failed due to logical unit queue not empty
    - 05h: Purge operation general failure.

    Other values are reserved and shall not be set.
  - bPurgeStatus is set to 00h (Idle) after power on or reset.
  - When the host enables the purge operation setting fPurgeEnable flag to one, and if all logical unit command queue are empty, the bPurgeStatus will be set to 01h to indicate that the purge operation is in progress. The bPurgeStatus shall be set to 03h if the operation is completed successfully, or to 05h if a failure occurred.
  - The host should send a query request to set fPurgeEnable flag to one only if command queues are empty. A query request to set fPurgeEnable flag which is processed when device command queues are not empty may fail. If the request fails,Query Response field in the QUERY RESPONSE UPIU shall be set to FFh (“General Failure”), the purge operation shall not start, and the bPurgeStatus shall be set to 04h.
  - If an ongoing purge operation is interrupted by the host setting the fPurgeEnable flag to zero, the bPurgeStatus shall be set to 02h.
  - When the bPurgeStatus is equal to the values 02h, 03h, 04h or 05h, the bPurgeStatus shall be automatically cleared to 00h (Idle) the first time that it is read. The bPurgeStatus values of 00h and 01h shall not be modified as a result of a read.

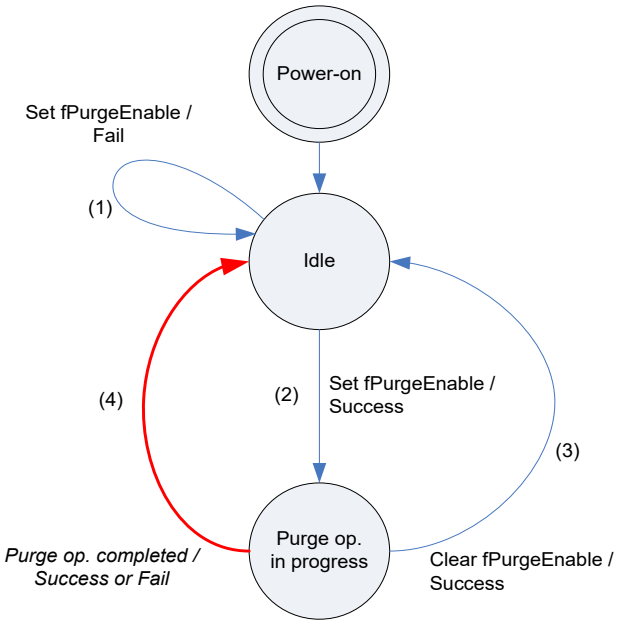
12.2.3.3 Purge operation (cont'd)

- If a purge operation is in progress (bPurgeStatus = 01h) commands sent to any logical units or to the RPMB well known logical unit will fail. The device shall return the sense key "NOT READY" to show that the command failed because a purge operation was in progress. Descriptors, attributes and flags may be read when a purge operation is in progress, while only fPurgeEnable flag may be written. A query request to write descriptors, attributes or flags (except fPurgeEnable) shall be terminated with Query Response field set to "General Failure".
- If the host needs to execute a command urgently when a purge operation is in progress, it may interrupt the purge operation. In particular, before issuing any command, the host sets fPurgeEnable flag to zero, waits until the device interrupts the operation, and then sets the bPurgeStatus attribute to 02h (purge operation stopped prematurely).
- If a power failure occurs the fPurgeEnable flag and bPurgeStatus attribute shall be reset to zero. In this case the device will not indicate that operation failed.

Figure 12.1 shows the Purge operation state machine. There are two states: "Idle" and "Purge Op. in progress". After power on, the purge operation state is "Idle", and the purge operation is disabled.

To enable the execution of a purge operation, the host sets fPurgeEnable flag to one sending a QUERY REQUEST UPIU. If the setting is executed successfully, the state will transition to "Purge Op. in progress" and the purge operation will start (bPurgeStatus = 01h). If there is a least one logical unit with command queue not empty, the setting of fPurgeEnable flag shall fail, the purge operation shall not start, the state shall remain "Idle", and bPurgeStatus shall be set to 04h.

When the purge operation is completed, the state will transition automatically to "Idle", and bPurgeStatus shall be set to 03h if the operation is completed successfully, or 05h in case of failure.



State	bPurgeStatus
Idle	00h, 02h, 03h, 04h, 05h
Purge op. in progress	01h

Transition	Description
1	Set fPurgeEnable and LU command queues not empty
2	Set fPurgeEnable and LU command queues empty
3	Clear fPurgeEnable
4	Automatic transition when the purge operation completes successfully or with failure

NOTE 1 On each transition the input event (triggering the state transition) and the output of the transition itself are mentioned.

Figure 12.1 — Purge Operation State Machine

The host may interrupt an ongoing purge operation clearing the fPurgeEnable flag, when the operation has been interrupted the state will transition to "Idle" and bPurgeStatus will be set to 02h.

12.2.3.4 Wipe Device

The wipe device operation is fulfilled issuing the FORMAT UNIT command to all enabled logical units. If the logical unit is write protected using one of the methods described in 12.3, Device Data Protection, or if the SWP bit in Control Mode Page is one, then the FORMAT UNIT command shall fail and the content of the medium shall not be altered.

A FORMAT UNIT command sent to the Device well known logical unit requests the device format all enabled logical units except the RPMB well known logical unit. If any logical unit is write protected when the FORMAT UNIT command is issued to Device well known logical unit, the FORMAT UNIT command shall fail and the content of the medium shall not be altered.

The fields of the FORMAT UNIT command should be set as described in the following:

- The Format data (FMTDATA) bit shall be set to zero to specify that no parameter list will be provided.
- The DEFECT LIST FORMAT shall be set to 000b.
- The format protection information (FMTPINFO) shall be set to 00b.
- The vendor specific byte shall be set to 00h.

The UFS device shall ignore CMPLST and LONGLIST bits since FMTDATA is set to zero.

12.2.3.5 bProvisioningType Parameter

Logical units can be configured in secure mode using bProvisioningType parameter of the Unit Descriptor. This parameter allows to enable thin provisioning and define TPRZ bit value in the READ CAPACITY parameter data.

The secure mode is enabled if thin provisioning is enabled and TPRZ bit is equal to one. In this mode all operations shall be performed using the mode defined by bSecureRemovalType parameter in the Device Descriptor. Only one type of removal type can be defined for an entire device.

bProvisioningType parameter can be set to the following values:

- 00h: to disable thin provisioning
- 02h: to enable thin provisioning and set TPRZ to zero
- 03h: to enable thin provisioning and set TPRZ to one

TPRZ bit value of zero indicates the device is in normal mode.

As all other Unit Descriptor configurable parameters, the bProvisioningType value is set writing the Configuration Descriptor. (see 14.1.5.3)

12.2.3.6 bSecureRemovalType Parameter

The bSecureRemovalType parameter within the Device Descriptor defines how information is removed from the physical memory during a Purge operation. This parameter may be set during system integration writing the Configuration Descriptors. bSecureRemovalType values are defined as follows:

- Value of '03h' will result in the information being removed using a vendor defined mechanism.
- Value of '02h' will result in all information being removed by overwriting the addressed locations with a character, its complement, then a random character.
- Value of '01h' will result in all information being removed by overwriting the addressed locations with a single character followed by an erase.
- Value of '00h' will result in all information being removed by an erase of the physical memory (default).
- Other values are reserved for future use and shall not be set.

  Device manufacturers are only required to support the mechanism required by their memory array technology.

  For additional information please refer to the http://www.killdisk.com/dod.htm or to the following documents for more details:

  - DoD 5220.22-M (http://www.dtic.mil/whs/directives/corres/pdf/522022m.pdf) and
  - NIST SP 800-88 (http://csrc.nist.gov/publications/nistpubs/800-88/NISTSP800-88_rev1.pdf)

12.3 Device Data Protection

12.3.1 Description and Requirements

UFS device data content can be protected at the logical unit level. The following protection modes shall be available:

- Permanent write protection (permanent: once enabled it cannot be reversed)
- Power on write protection (write protection can be cleared with a power cycle or a hardware reset event)
- Secure write protection (write protection can only be configured and enabled/disabled using secure authenticated methods)

These modes of write protections are not implemented in the RPMB well known logical unit. There shall also be a method to read the protection mode that is currently enabled for a logical unit.

12.3.2 Implementation

The protection mode can be defined at logical unit level configuring the bLUWriteProtect parameter of the Unit Descriptor. The write protection modes are encoded as in the following:

00h: Logical unit not write protected (or secure write protected if one or more secure write protect entry is set)

01h: Logical unit power on write protected

02h: Logical unit permanently write protected

A power on write protected logical unit (bLUWriteProtect = 01h) can be written only if fPowerOnWPEn flag is equal to zero. The fPowerOnWPEn flag is set to zero after a power cycle or hardware reset event, once it is set to one it cannot be toggled or cleared by the host.

The fPermanentWPEn flag shall be set to one to enable the write protection of all permanently write protected logical unit (bLUWriteProtect = 02h); logical unit can be written if fPermanentWPEn flag is equal to 0b. The fPermanentWPEn flag is write once: it cannot be toggled or cleared once it is set. The fPermanentWPEn flag shall be zero after device manufacturing.

LBA areas within logical units may be write protected using the secure write protection mode.

Secure write protect areas are configured setting the Secure Write Protect Configuration Block, and they may only be created in logical units configured as “not write protected” (bLUWriteProtect = 00h).

One logical unit may have up to four secure write protect areas. However the total number of secure write protect areas in a device shall not be more than bNumSecureWPArea.

A secure write protect area can be written only if write protection is disabled in the related Secure Write Protect Entry (WPF flag = 0b). (see 12.4.3.1)

It is recommended to set fPowerOnWPEn flag, fPermanentWPEn flag or WPF flag when all command queues are empty, and wait device query response before enqueuing write command.

If an LBA is write protected, then otherwise valid commands that request unmap, format or alteration of the medium related to that LBA shall be rejected with CHECK CONDITION status with the sense key set to DATA PROTECT.

12.4 RPMB

12.4.1 Introduction

A signed access to a Replay Protected Memory Block is provided. This function provides means for the system to store data to the specific memory area in an authenticated and replay protected manner. This is provided by first programming authentication key information to the UFS device memory (shared secret).

As the system cannot be authenticated yet in this phase the authentication key programming have to take in a secure environment like in an OEM production. Further on the authentication key is utilized to sign the read and write accesses made to the replay protected memory area with a Message Authentication Code (MAC).

Usage of random number generation and count register are providing additional protection against replay of messages where messages could be recorded and played back later by an attacker.

There are two RPMB modes; Normal RPMB mode and Advanced RPMB mode using EHS.

The RPMB mode can be configured by setting Bit4 of bRPMBRegionEnable parameter of the RPMB descriptor in the configuration stage. If the device receives an RPMB operation request of a different mode than the configured RPMB mode, the device shall respond with ILLEGAL REQUEST.

12.4.2 RPMB Well Known Logical Unit Description

The RPMB is contained in a unique well known logical unit whose size is defined in the RPMB Unit Descriptor. RPMB well known logical unit size shall be a multiple of 128 Kbytes, therefore its minimum size is 128 Kbytes. The contents of the RPMB well known logical unit can only be read or written via successfully authenticated read and write accesses. The data may be overwritten by the host but can never be erased.

All accesses to the RPMB will reference the specific RPMB well known logical unit number (W-LUN).

RPMB well known logical unit may be configured into multiple RPMB regions where each RPMB region has its own dedicated authentication key, write counter, result register, and logical address which starts from zero. Refer to 13.2.3 to see how to configure RPMB well known logical unit into multiple RPMB regions.

Each RPMB region can process a single RPMB authenticated operation at any given point in time where a single authenticated RPMB operation corresponds to whole Figure 12.5, Authentication Key Programming Flow, Figure 12.6, Read Counter Value Flow, or Figure Figure 12.10, Authenticated Secure Write Protect Configuration Block Read Flow, and so on as listed in Table 12.8, Request Message Types. For example, the Initiator 3 may start an authenticated operation in RPMB Region 0 and Initiator 4 in RPMB Region 1. The RPMB Region 0 should be accessed by a new authenticated operation request after completing the authenticated operation started by Initiator 3; and RPMB Region 1 after completing the request from the Initiator 4.

12.4.3 Requirements

12.4.3.1 RPMB Resources

- Authentication Key
  - Type: Write once, not erasable or readable
  - Size: 32 bytes
  - Description: Authentication key register which is used to authenticate accesses when MAC is calculated. Each RPMB region has a dedicated authentication key.
- Write Counter
  - Type: Read only
  - Size: 4 bytes
  - Description: Counter value for the total amount of successful authenticated data write requests made by the host. The initial value of this register after production is 0000 0000h. The value will be incremented by one automatically by the UFS device along with each successful programming access. The value cannot be reset. After the counter has reached the maximum value of FFFF FFFFh, it will not be incremented anymore (overflow prevention). Each RPMB region has dedicated write counter.
- Result Register
  - Type: Read only
  - Size: 2 bytes
  - Description: This register provides the result of an authenticated operation. Each RPMB region has dedicated result register.
- RPMB Data Area
  - Type: Readable and writable
  - Size: Multiples of 128 Kbytes defined in RPMB Unit Descriptor
    - 128 Kbytes minimum, 16 Mbytes maximum.
    - Each RPMB region size is defined as bRPMBRegion0Size – bRPMBRegion3Size in the RPMB Unit Descriptor.
  - Description: Data which can only be read and written via successfully authenticated read/write access. This data may be overwritten by the host but can never be erased.
- Secure Write Protect Configuration Block for Normal RPMB
  - Type: Readable and writable
  - Size : 256 Bytes
  - 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.

12.4.3.1 RPMB Resources (cont'd)

- Secure Write Protect Configuration Block for Advanced RPMB
  - Type: Readable and writable
  - Size : 4 Kbytes
  - 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.

Table 12.1 — Secure Write Protect Configuration Block for Normal RPMB

Bit / Byte (1)	7	6	5	4	3	2	1	0
0 (228)	LUN							
1 (229)	DATA LENGTH							
2 (230)	Reserved							
…								
15 (243)								
16 (244)	Secure Write Protect Entry 0							
…								
31 (259)								
32 (260)	Secure Write Protect Entry 1							
…								
47 (275)								
48 (276)	Secure Write Protect Entry 2							
…								
63 (291)								
64 (292)	Secure Write Protect Entry 3							
…								
79 (307)								
80 (308)	Reserved							
…								
255 (483)								

NOTE 1 Values in parentheses indicate the byte number in the RPMB Message Data Frame.

12.4.3.1 RPMB Resources (cont'd)

Table 12.2 — Secure Write Protect Configuration Block for Advanced RPMB

Bit / Byte (1)	7	6	5	4	3	2	1	0
0	LUN							
1	DATA LENGTH							
2	Reserved							
…								
15								
16	Secure Write Protect Entry 0							
…								
31								
32	Secure Write Protect Entry 1							
…								
47								
48	Secure Write Protect Entry 2							
…								
63								
64	Secure Write Protect Entry 3							
…								
79								
80	Reserved							
…								
4095								

12.4.3.1 RPMB Resources (cont'd)

a) LUN

The LUN field indicates the logical unit to which secure write protection shall apply. Valid values are from 0 to the number of LU specified by bMaxNumberLU.

b) DATA LENGTH

The DATA LENGTH field specifies the length in bytes of the Secure Write Protect Entries (0 for no entry, 16 for one entry, 32 for two entries, 48 for three entries and 64 for four entries). In a write request, the device shall ignore the bytes from DATA LENGTH + 16 to 255 in Normal RPMB mode, from DATA LENGTH + 16 to 4095 in Adv. RPMB mode and set these bytes of the Secure Write Protect Configuration Block to zero.

c) Secure Write Protect Entry 0 to Entry 3

The Secure Write Protect Configuration Block may contain only the Entry 0, the Entries 0 and 1, the Entries 0, 1 and 2, or all four Entries. If the Secure Write Protect Configuration Block does not contain any entry (DATA LENGTH = 00h), all entries in the specified logical unit will be removed.
Table 12.3 defines the structure of Secure Write Protect Entry.

Table 12.3 — Secure Write Protect Entry

Byte \ Bit	7	6	5	4	3	2	1	0
0	Reserved					WPT		WPF
1	Reserved							
2	Reserved							
3	Reserved							
4	(MSB)							
…				LOGICAL BLOCK ADDRESS				
11								(LSB)
12	(MSB)							
…				NUMBER OF LOGICAL BLOCKS				
15								(LSB)

12.4.3.1 RPMB Resources (cont'd)

d) WPT (Write Protect Type)

The write protect type field (WPT) specifies how WPF bit may be modified.

Table 12.4 — Write Protect Type Field

Code	Definition
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

e) WPF (Write Protect Flag)

0b : Secure Write Protection is disabled.

1b : Secure Write Protection is enabled.

A WPF set to one specifies that the logical unit shall inhibit alteration of the medium for LBA within the range indicated by LOGICAL BLOCK ADDRESS field and NUMBER OF LOGICAL BLOCKS field. Commands requiring writes to the medium shall be terminated with CHECK CONDITION status, with the sense key set to DATA PROTECT, and the additional sense code set to WRITE PROTECTED.

Logical units that contain cache shall write all cached logical blocks to the medium (e.g., as they would do in response to a SYNCHRONIZE CACHE command with the LOGICAL BLOCK ADDRESS field and the NUMBER OF LOGICAL BLOCKS field set to the values indicated in the Secure Write Protect Entry) prior to enabling the write protection.

A WPF bit set to zero specifies that the logical unit may allow writing to the medium, depending on other write inhibit mechanisms implemented by the logical unit.

WPF shall be set to zero after device manufacturing.

f) LOGICAL BLOCK ADDRESS

This field specifies the LBA of the first logical block of the Secure Write Protect area.

12.4.3.1 RPMB Resources (cont'd)

g) NUMBER OF LOGICAL BLOCKS

This field specifies the number of contiguous logical blocks that belong to the Secure Write Protect area.

If the NUMBER OF LOGICAL BLOCKS field is set to zero, then the secure write protection shall apply to the entire logical unit. In that case, only Entry-0 needs to be configured to enable secure write protection for the entire logical unit.

- RPMB Purge Response Packet Format
  - The RPMB Purge Response Packet is contained in the Data field of the RPMB Purge Status Response message. Refer to the Data field in Figure 12.12, RPMB Purge Status Read Flow.

Table 12.5 — RPMB Purge Response Packet Format (Legacy RPMB Mode)

Bit / Byte	7	6	5	4	3	2	1	0
0 (228)		STATUS						
1 (229)		bRPMBLifeTimeEst						
2 (230)		RESERVED						
3 (231)								
4 (232)								
….								
255 (483)								

Table 12.6 — RPMB Purge Response Packet Format (Advanced RPMB Mode)

Bit / Byte	7	6	5	4	3	2	1	0
0	STATUS							
1	bRPMBLifeTimeEst							
2	RESERVED							
3								
4								
….								
4095								

12.4.3.1 RPMB Resources (cont'd)

RPMB Purge Status Read Response includes:

a) STATUS

00h : RPMB Purge not initiated (reset value)

01h : RPMB Purge in progress

02h : RPMB Purge successfully completed. The device will send this status on the next **RPMB Purge Status Read Request**, and this status will change to 00, RPMB Purge not initiated.

03h : RPMB Purge General Failure. The device will send this status on the next **RPMB Purge Status Read Request**, and this status will change to 00, RPMB Purge not initiated.

b) bRPMBLifeTimeEst

This field provides an indication of the RPMB LU life time based on the amount of performed program/erase cycles of the RPMB LU. Note that whenever the RPMB purge operation is executed, multiple program/erase cycles may be performed depending on the fragmentation of the valid RPMB data scattered in the RPMB LU. The calculation method is vendor specific.

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

12.4.3.2 Algorithm and Key for MAC Calculation

The message authentication code (MAC) is calculated using HMAC SHA-256 as defined in [HMAC-SHA]. The HMAC SHA-256 calculation takes as input a key and a message. The resulting MAC is 256 bits (32 bytes), which are embedded in the data frame as part of the request or response.

The key used for the MAC calculation is always the 256-bit Authentication Key stored in the target RPMB region. The message used as input to the MAC calculation is the concatenation of the fields in the RPMB packet.

12.4.3.3 MAC Calculation for Advanced RPMB

The key used for the MAC calculation is the 256-bit Authentication Key stored in the device.

If RPMB Message includes data in a DATA IN UPIU or a DATA OUT UPIU, the concatenation of the data transferred in each DATA IN/OUT UPIU in the order in which it is sent is input into the MAC calculation. Then the concatenation of the fields in the Advanced RPMB Meta Information from byte 0 to byte 27 is input into the MAC calculation. After this, four 00h bytes are input into the MAC calculation.

12.4.3.4 RPMB Message Components

Each RPMB message includes specific components. These components are displayed in Table 12.7.

Table 12.7 — RPMB Message Components

Component Name	Length	Request	Response	Description
Request Message Type	2 bytes	Yes	No	This component indicates the request message type. See 12.4.3.5.
Response Message Type	2 bytes	No	Yes	This component indicates the response message type. See 12.4.3.6.
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 12.4.3.7.
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.
Advanced RPMB Data	4096 bytes	Yes	Yes	Data to be written or read by signed access in Advanced RPMB operation.
Block Count	2 bytes	Yes	Yes	Number of 256-byte logical blocks requested to be read or programmed.
Advanced RPMB Block Count	2 bytes	Yes	Yes	Number of 4 Kbytes (Advanced RPMB data) requested to be read or programmed.

12.4.3.5 Request Message Types

The host sends messages to the device to initate RPMB operations.

Table 12.8 lists RPMB's Requst Message Types and their codes.

Table 12.8 — 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 (Normal RPMB Mode only)
0006h	Secure Write Protect Configuration Block write request
0007h	Secure Write Protect Configuration Block read request
0008h	RPMB Purge Enable Request
0009h	RPMB Purge Status Read Request
0010h	Authenticated Vendor Specific Command Request
0011h	Authenticated Vendor Specific Command Status Read Request
Others	Reserved

12.4.3.6 Response Message Types

The device responds with messages to the host during RPMB operations.

Table 12.9 lists RPMB's Response Message Types and their codes.

Table 12.9 — Response Message Types

Code	Response Message Types
0100h	Authentication Key programming response
0200h	Write Counter read response
0300h	Authenticated data write response
0400h	Authenticated data read response
0500h	Reserved
0600h	Secure Write Protect Configuration Block write response
0700h	Secure Write Protect Configuration Block read response
0800h	RPMB Purge Enable Response
0900h	RPMB Purge Status Read Response
1000h	Authenticated Vendor Specific Command Response (Advanced RPMB Mode only)
100h	Authenticated Vendor Specific Command Status Response
thers	Reserved

12.4.3.7 RPMB Operation Result

- Result component of an RPMB message is composed of two bytes. The most significant byte is reserved and shall be set to zero.
- Bit 7 of Result field shall indicate if the write counter has expired (i.e., reached its maximum value) or not.
  - Value of one will represent an expired write counter
  - Value of zero will represent a valid write counter
- The other bits shall indicate the operation status.
  - Operation Okay (00h)
    - General Failure (01h)
  - Authentication Failure (02h)
    - MAC comparison not matching, MAC calculation failure
  - Counter Failure (03h)
    - Counters not matching in comparison, counter incrementing failure
  - Address Failure (04h)
    - Address out of range, wrong address alignment
  - Write Failure (05h)
    - Data, Counter or Result write failure
  - Read Failure (06h)
    - Data, Counter or Result read failure
  - Authentication key not yet programmed (07h)
    - This value is the only valid result until the Authentication Key has been programmed in the target RPMB region, after which it can never occur again
  - Secure Write Protect Configuration Block access failure (08h).
    - Secure Write Protect Configuration read or write failure.
  - Invalid Secure Write Protect Block Configuration parameter (09h).
    - Invalid LUN (or logical unit not enabled), DATA LENGTH, LOGICAL BLOCK ADDRESS, NUMBER OF LOGICAL BLOCKS, or overlapped areas.
  - Secure Write Protection not applicable (0Ah).
    - Logical unit configured with other write protection modes (permanent or power-on)
  - Unrecognized/Unsupported Request Type (0Bh)
    - Request type unrecognized or unsupported
  - Rejected, RPMB purge operation in progress (0Ch)
    - While an RPMB Purge is in progress, authenticated Read or Write is rejected.

Table 12.10 — Result Data Structure

Bit[15:8]	Bit[7]	Bit[6:0]
Reserved	Write Counter Status	Operation Status

12.4.3.7 RPMB Operation Result (cont'd)

Table 12.11 — Result Code Definition

Code	Description
0000h (0080h)	Operation OK
0001h (0081h)	General failure
0002h (0082h)	Authentication failure • MAC comparison not matching, MAC calculation failure
0003h (0083h)	Counter failure • Counters not matching in comparison, counter incrementing failure
0004h (0084h)	Address failure • Address out of range, wrong address alignment
0005h (0085h)	Write failure • Data / Counter / Result write failure
0006h (0086h)	Read failure • Data / Counter / Result read failure
0007h	Authentication Key not yet programmed. • This value is the only valid Result value until the Authentication Key has been programmed. Once the key is programmed, this value will no longer be used.
0008h (0088h)	Secure Write Protect Configuration Block access failure • Secure Write Protect Configuration read or write failure
0009h (0089h)	Invalid Secure Write Protect Block Configuration parameter • Invalid LUN or logical unit not enabled, DATA LENGTH, LOGICAL BLOCK ADDRESS, NUMBER OF LOGICAL BLOCKS, or overlapped areas
000Ah (008Ah)	Secure Write Protection not applicable • Logical unit configured with other write protection modes (permanent or power-on)
000Bh (008Bh)	Unrecognized/Unsupported Request Type • Request type unrecognized or unsupported
000Ch (008Ch)	Rejected, RPMB purge operation in progress • While an RPMB Purge is in progress, authenticated Read or Write is rejected.
NOTE The values in parentheses are valid when Write Counter has expired.	

12.4.4 Implementation in Normal RPMB Mode

12.4.4.1 RPMB Message

An RPMB Message may be composed of one or more RPMB Message Data Frames.

RPMB Message Data Frame size is 512 bytes and it is organized as shown in Table 12.12.

Table 12.12 — RPMB Message Data Frame

Byte \ Bit	7	6	5	4	3	2	1	0
0	(MSB)							
195				Stuff Bytes				(LSB)
196	(MSB)							
227				Key / MAC				(LSB)
228				Data [255]				
483				Data [0]				
484	(MSB)							
499				Nonce				(LSB)
500	(MSB)							
503				Write Counter				(LSB)
504	(MSB)							
505				Address				(LSB)
506	(MSB)							
507				Block Count				(LSB)
508	(MSB)							
509				Result				(LSB)
510	(MSB)							
511				Request Message Type / Response Message Type				(LSB)

12.4.4.2 MAC Calculation

The key used for the MAC calculation is always the 256 bit Authentication Key stored in the device. Input to the MAC calculation is the concatenation of the fields in the RPMB Message Data Frames from byte 228 to byte 511 (stuff bytes and the MAC are excluded).

If RPMB Message is composed by several RPMB Message Data Frames then the input message to MAC is the concatenation of bytes [228:511] of each data frame in the order in which the data frames are sent. The MAC is valid only in the last data frame.

12.4.4.3 RPMB Message Data Frame Delivery

The RPMB messages are delivered using SCSI security protocol commands:

- SECURITY PROTOCOL IN is used to send request messages to the device
- SECURITY PROTOCOL OUT is used to request to the device the sending of response messages.

12.4.5 Implementation in Advanced RPMB Mode

12.4.5.1 Advanced RPMB Message

An Advanced RPMB Message is composed of an Advanced RPMB Meta Information and a MAC/KEY in the EHS field in COMMAND UPIU and RESPONSE UPIU. Advanced RPMB Data is delivered through the Data Segment in DATA IN UPIU and DATA OUT UPIU. Advanced RPMB Data size shall be a multiple of 4 Kbytes.

Advanced RPMB Message size is 60 bytes and organized in EHS field as shown in Figure 12.2. Since the bLength indicates the total size of the EHS Header and EHS Data in 32 Byte units, the value of the bLength for Advanced RPMB is 02h. The Advanced RPMB Meta Information has fields to contain the Request/Response Message Type, Nonce, Write Counter, Address or LUN, Block Count, and Result. See Table 12.13 for details. Depending on the value of the Request/Response Message Type, the validity of the MAC/KEY is determined.

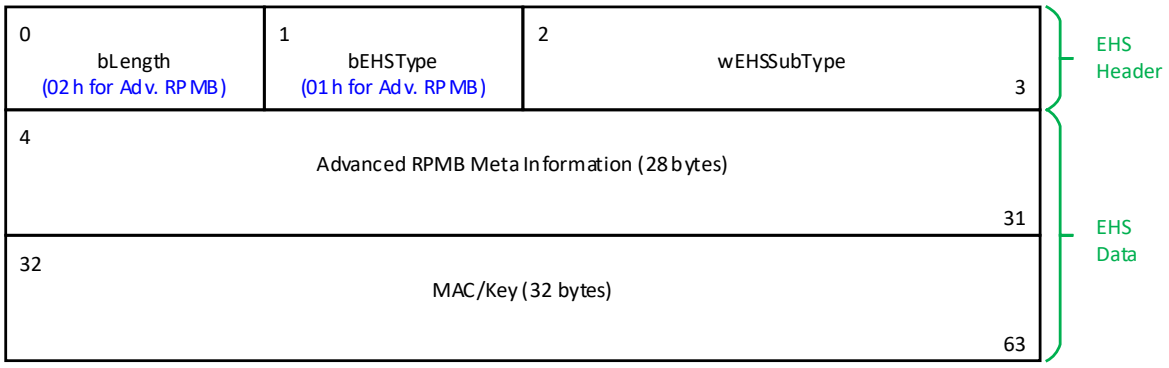


Figure 12.2 — Advanced RPMB Message Structure in EHS Field

12.4.5.1 Advanced RPMB Message (cont'd)

Table 12.13 shows the detailed fields of Advanced RPMB Meta Information.

Table 12.13 — Advanced RPMB Meta Information

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)	Request Message Type / Response Message Type						
1								(LSB)
2	(MSB)	Nonce						
17								(LSB)
18	(MSB)	Write Counter						
21								(LSB)
22	(MSB)	Address / LUN (1)						
23								(LSB)
24	(MSB)	Block Count						
25								(LSB)
26	(MSB)	Result						
27								(LSB)

NOTE 1 LUN is used in case of Authenticated Secure Write Protect Configuration Block Read and Authenticated Secure Write Protect Configuration Block Write RPMB request. Address is used in case of Authenticated Data Read and Authenticated Data Write request.

12.4.6 SECURITY PROTOCOL IN/OUT Commands

SECURITY PROTOCOL IN command and SECURITY PROTOCOL OUT command defined in [SPC] are used to encapsulate and deliver data packets of any security protocol between host and device without interpreting, dis-assembling or re-assembly the data packets for delivery.

The SECURITY PROTOCOL IN command and SECURITY PROTOCOL OUT command contain a SECURITY PROTOCOL field. A unique security protocol ID is assigned by T10 for JEDEC UFS application.

- SECURITY PROTOCOL = ECh (JEDEC Universal Flash Storage).

12.4.6.1 CDB Format of SECURITY PROTOCOL IN/OUT Commands

See 11.3.19 and 11.3.20 for UFS descriptions of SECURITY PROTOCOL IN and SECURITY PROTOCOL OUT commands. See [SPC] for the CDB format.

The RPMB well known logical unit shall support the following SECURITY PROTOCOL field values:

00h: Security protocol information.

ECh: JEDEC Universal Flash Storage (security protocol ID assigned for JEDEC UFS application in [SPC]).

Other values are invalid.

SECURITY PROTOCOL IN/OUT commands shall consider the unique Security Protocol ID assigned to JEDEC UFS as the only valid Security Protocol ID.

When the SECURITY PROTOCOL field is set to ECh (i.e., the JEDEC Universal Flash Storage),

- INC_512 bit shall be set to zero to specify that the ALLOCATION LENGTH or the TRANSFER LENGTH field expresses the number of bytes to be transferred.
- If the ALLOCATION LENGTH field in a SECURITY PROTOCOL IN command is not equal to an integer multiple of 512 in Normal RPMB mode or an integer multiple of 4K in Advanced RPMB mode, then the command shall be terminated with CHECK CONDITION status.
- If the TRANSFER LENGTH field in a SECURITY PROTOCOL OUT command is not equal to an integer multiple of 512 in Normal RPMB mode or an integer multiple of 4K in Advanced RPMB mode, then the command shall be terminated with CHECK CONDITION status.
- The SECURITY PROTOCOL SPECIFIC field specifies the RPMB Protocol ID.

The RPMB Protocol ID indicates the RPMB region as defined in Table 12.14.

Table 12.14 — 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

If the SECURITY PROTOCOL SPECIFIC field is set to an invalid value or the corresponding RPMB region is not enabled, then SECURITY PROTOCOL IN/OUT command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB. In this case, Total EHS length shall be ‘0h’ and EHS field shall not be included in RESPONSE UPIU in Advanced RPMB mode.

Secure Write Protect Configuration Block write, Secure Write Protect Configuration Block read, Authenticated Vendor Specific Command, and Authenticated Vendor Specific Command Status Read requests are supported only by RPMB region 0.

12.4.6.1 CDB Format of SECURITY PROTOCOL IN/OUT Commands (cont'd)

As required by [SPC], the SECURITY PROTOCOL value of 00h (security protocol information) shall be supported if the SECURITY PROTOCOL IN command is supported by the device. The security protocol information security protocol (i.e., the SECURITY PROTOCOL field set to 00h in a SECURITY PROTOCOL IN command) is used to transfer security protocol related information from the logical unit.

When the SECURITY PROTOCOL field is set to 00h in a SECURITY PROTOCOL IN command, the two bytes SECURITY PROTOCOL SPECIFIC field shall contain a numeric value as defined in Table 12.15.

Table 12.15 — 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	

According to [SPC], if the SECURITY PROTOCOL field is set to 00h and the SECURITY PROTOCOL SPECIFIC field is set to 0000h in a SECURITY PROTOCOL IN command, the parameter data shall have the format shown in the [SPC]-referenced "Security Features for SCSI Commands" standard, found under the "Supported security protocols SECURITY PROTOCOL IN parameter data" header.

12.4.6.2 Certificate Data Description

If the SECURITY PROTOCOL field is set to 00h and the SECURITY PROTOCOL SPECIFIC field is set to 0001h in a SECURITY PROTOCOL IN command, the parameter data shall have the format shown in the [SPC]-referenced "Security Features for SCSI Commands" standard, found under "Certificate data SECURITY PROTOCOL IN parameter data".

Because the UFS device server does not have a certificate to transfer, the CERTIFICATE LENGTH field shall be set to 0000h.

12.4.7 RPMB Operations

12.4.7.1 Request Type Message Delivery

- An RPMB region can process only one RPMB operation at any time.
  - An initiator sends a request type message to RPMB well known logical unit to request the execution of an operation.
- To deliver a request type message, the initiator sends a SECURITY PROTOCOL OUT command with SECURITY PROTOCOL field is set to ECh (i.e., the JEDEC Universal Flash Storage) and indicating the target RPMB region in the SECURITY PROTOCOL SPECIFIC field.
- For an authenticated data write request, the data to be written into the RPMB data area is included in the request message. The maximum data size in a single Authenticated Data Write request is equal to bRPMB_ReadWriteSize × 256 bytes in Normal RPMB and bRPMB_ReadWriteSize × 4 Kbytes in Advanced RPMB; multiple Authenticated Data Write operations should be executed if the desired data size exceeds this value. The bRPMB_ReadWriteSize value shall be changed to the appropriate value based on RPMB configuration.

12.4.7.1 Request Type Message Delivery (cont'd)

- For SECURITY PROTOCOL OUT command, the Flags.W in the COMMAND UPIU is set to one since data is transferred from the host to the device.
- Table 12.16 defines the Expected Data Transfer Length field value in the COMMAND UPIU for the various cases.

Table 12.16 — Expected Data Transfer Length Value for Request Type Messages

RPMB Message	Value	
	Normal RPMB	Adv. RPMB
Authentication Key programming request	512	0
Result read request	512	N/A
Write Counter read request	512	N/A
Authenticated data read request	512	N/A
Secure Write Protect Configuration Block write request	512	4096
Secure Write Protect Configuration Block read request	512	N/A
Authenticated data write request	512 × Block Count	4096 × Advanced RPMB Block Count
RPMB Purge Enable request	512	0
RPMB Purge Status Read request	512	N/A
Authenticated Vendor Specific Command Request	512 × Block Count	4096 × Advanced RPMB Block Count
Authenticated Vendor Specific Command Status Read Request	512 × Block Count	4096 × Advanced RPMB Block Count
NOTE In Normal RPMB mode, Block Count is equal to the data size divided by 256.		

- The device indicates to the host that it is ready to receive the request type message sending READY TO TRANSFER UPIU. If the Expected Data Transfer Length is 512 byte, then Data Buffer Offset field shall be set to a value of zero and Data Transfer Count field shall be set to a value of 512.
- The number of bytes requested in a single READY TO TRANSFER UPIU shall not be greater than the value indicated by bMaxDataOutSize attribute. A single READY TO TRANSFER UPIU may request the transfer of one or more RPMB Messages.

12.4.7.1 Request Type Message Delivery (cont'd)

- In response to each READY TO TRANSFER UPIU, the host delivers the requested portion of the message sending DATA OUT UPIU. See 10.7.13 for details about data transfer.
- To complete the SECURITY PROTOCOL OUT command, the device returns a RESPONSE UPIU with the status.
- Figure 12.3 depicts a request type message delivery. The application client loads the RPMB Message in the Data Out Buffer and indicates the target RPMB Region in SECURITY PROTOCOL SPECIFIC field.

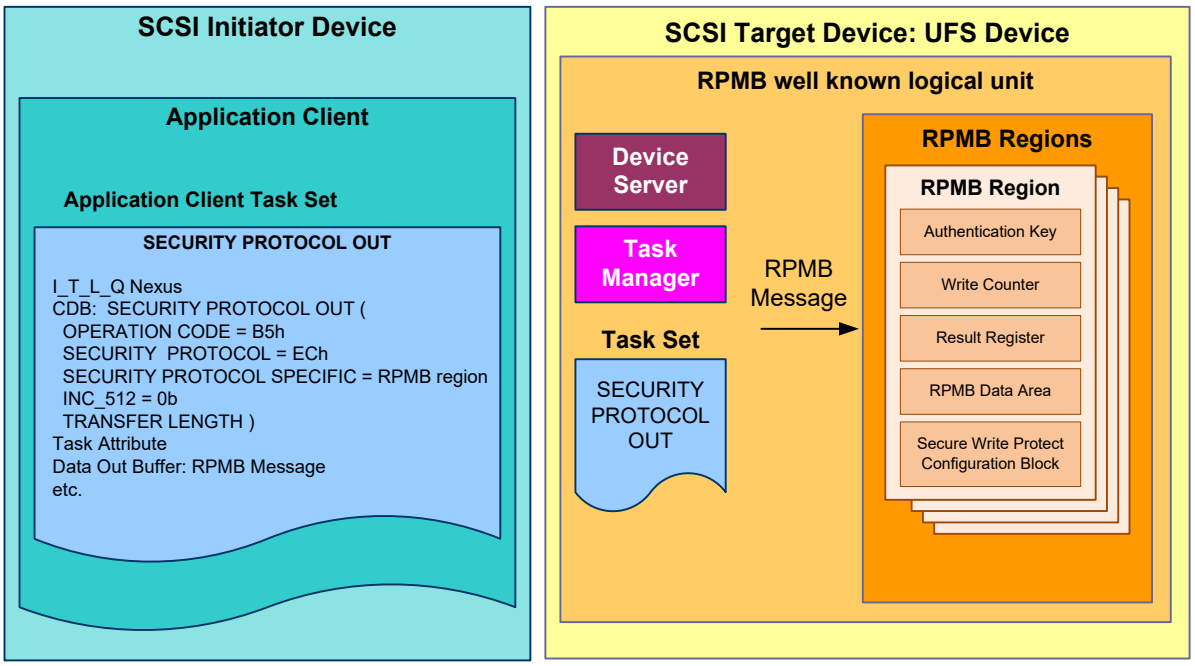


Figure 12.3 — Request Type Message Delivery

12.4.7.2 Response Type Message Delivery

- A initiator requests the RPMB well known logical unit to send a response type message to retrieve the result of a previous operation, to retrieve the Write Counter, to retrieve data from the RPMB data area, or to retrieve the contents of a Secure Write Protect Configuration Block.
- To request the delivery of a response type message, the host sends a SECURITY PROTOCOL IN command with SECURITY PROTOCOL field is set to ECh (i.e., the JEDEC Universal Flash Storage) and indicating the RPMB region in the SECURITY PROTOCOL SPECIFIC field.
- For an Authenticated Data Read request, the data read from the RPMB data area is included in the response message. The maximum data size in a single Authenticated Data Read request is equal to bRPMB_ReadWriteSize × 256 bytes in Normal RPMB and bRPMB_ReadWriteSize × 4K bytes in Advanced RPMB; multiple Authenticated Data Read operations should be executed if the desired data size exceeds this value. The bRPMB_ReadWriteSize value shall be changed to the appropriate value based on RPMB configuration.
- For SECURITY PROTOCOL IN command, the Flags.R in the COMMAND UPIU is set to one since data is transferred from the device to the host.
- Table 12.17 defines the Expected Data Transfer Length field value in the COMMAND UPIU for the various cases.

Table 12.17 — Expected Data Transfer Length Value for Response Type Messages

RPMB Message	Value	
	Normal RPMB	Adv. RPMB
Authentication Key programming response	512	N/A
Authenticated data write response	512	N/A
Secure Write Protect Configuration Block write response	512	N/A
Write Counter read response	512	0
Secure Write Protect Configuration Block read response	512	4096
Authenticated data read response	512 × Block Count	4096 × Advanced RPMB Block Count
RPMB Purge Enable response	512	N/A
RPMB Purge Status Read response	512	4096
Authenticated Vendor Specific Command Response (Advanced RPMB Mode only)	N/A	N/A
Authenticated Vendor Specific Command Status Response	512 × Block Count	4096 × Advanced RPMB Block Count
NOTE In Normal RPMB mode, Block Count is equal to the data size divided by 256.		

- The device returns the result or data requested in the RPMB message. The RPMB message is delivered by sending one or more DATA IN UPIU in the data phase. A single DATA IN UPIU may deliver one or more RPMB Messages.
- The data size in DATA IN UPIU shall not exceed the value indicated by bMaxDataInSize attribute.
- To complete the SECURITY PROTOCOL IN, the device sends a RESPONSE UPIU with the status.

12.4.7.2 Response Type Message Delivery (cont'd)

- Figure 12.4 depicts a response type message delivery. An application client requests a RPMB Region to transfer the RPMB Message in the Data In Buffer specifying the RPMB Region ID in SECURITY PROTOCOL SPECIFIC field of the CDB.

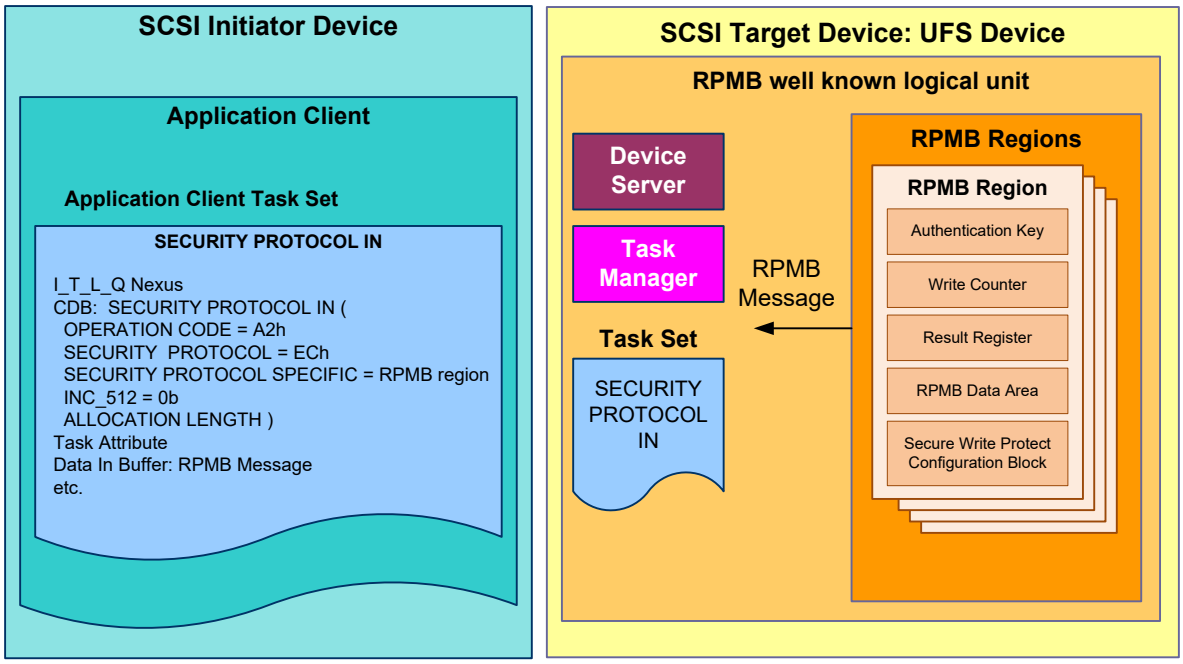


Figure 12.4 — Response Type Message Delivery

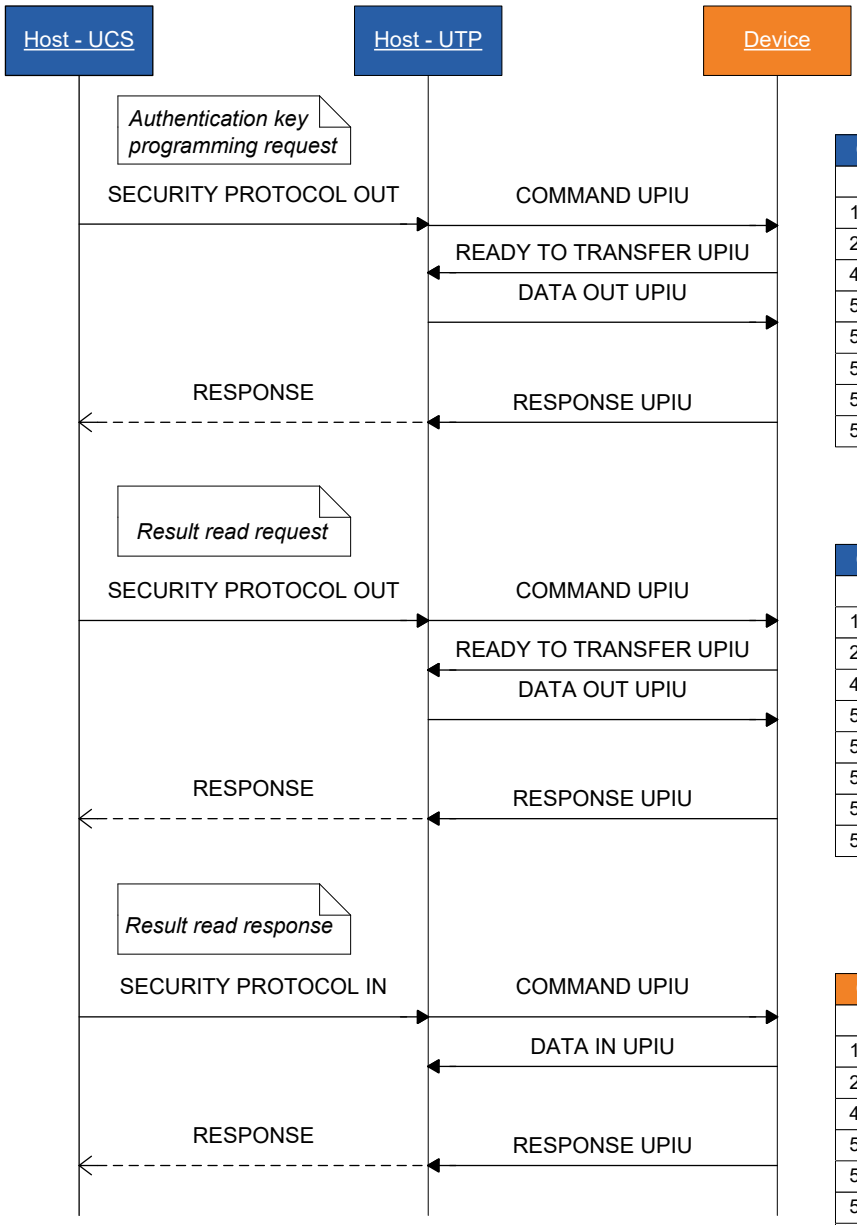
12.4.7.3 RPMB Operations in Normal RPMB Mode

12.4.7.3.1 Authentication Key Programming

- The Authentication Key programming is initiated by a SECURITY PROTOCOL OUT command
  - 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 = 0001h and the Authentication Key.
  - The device returns GOOD status in status response when Authentication Key programming is completed.
- The Authentication Key programming verification process starts by issuing a SECURITY PROTOCOL OUT command
  - An initiator sends a 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 contains the Request Message Type = 0005h (Result read request). Note that any request other than the Result read request from any initiator will overwrite the Result register of the RPMB Region.
  - The device returns GOOD status in status response when the operation result is ready for retrieval.
- An initiator retrieves the operation result by issuing a SECURITY PROTOCOL IN command.
  - The SECURITY PROTOCOL field is set to ECh and the SECURITY PROTOCOL SPECIFIC field indicates the RPMB region.
  - Device returns the RPMB data frame containing the Response Message Type = 0100h and the Result code.
  - If programming of Authentication Key failed then returned result is “Write failure” (0005h). If some other error occured during Authentication Key programming then returned result is “General failure” (0001h).

Access to RPMB data area is not possible before the Authentication Key is programmed in the corresponding RPMB region. The state of the device can be checked by trying to write/read data to/from the RPMB data area: if the Authentication Key is not programmed then the Result field in the response message will be set to “Authentication Key not yet programmed” (0007h).

12.4.7.3.1 Authentication Key Programming (cont'd)



Authentication Key Programming Request

Offset	Field Name	Value
0 :195	Stuff bytes	0...00h
196:227	MAC / Key	*Key*
228:483	Data	0...00h
484:499	Nonce	0...00h
500:503	Write counter	0...00h
504:505	Address	0000h
506:507	Block Count	0000h
508:509	Result	0000h
510:511	Req./Resp.	0001h

Result Read Request

Offset	Field Name	Value
0 :195	Stuff bytes	0...00h
196:227	MAC / Key	0...00h
228:483	Data	0...00h
484:499	Nonce	0...00h
500:503	Write counter	0...00h
504:505	Address	0000h
506:507	Block Count	0000h
508:509	Result	0000h
510:511	Req./Resp.	0005h

Result Read Response

Offset	Field Name	Value
0 :195	Stuff bytes	0...00h
196:227	MAC / Key	0...00h
228:483	Data	0...00h
484:499	Nonce	0...00h
500:503	Write counter	0000 0000h
504:505	Address	0000h
506:507	Block Count	0000h
508:509	Result	*Result code*
510:511	Req./Resp.	0100h

Figure 12.5 — Authentication Key Programming Flow

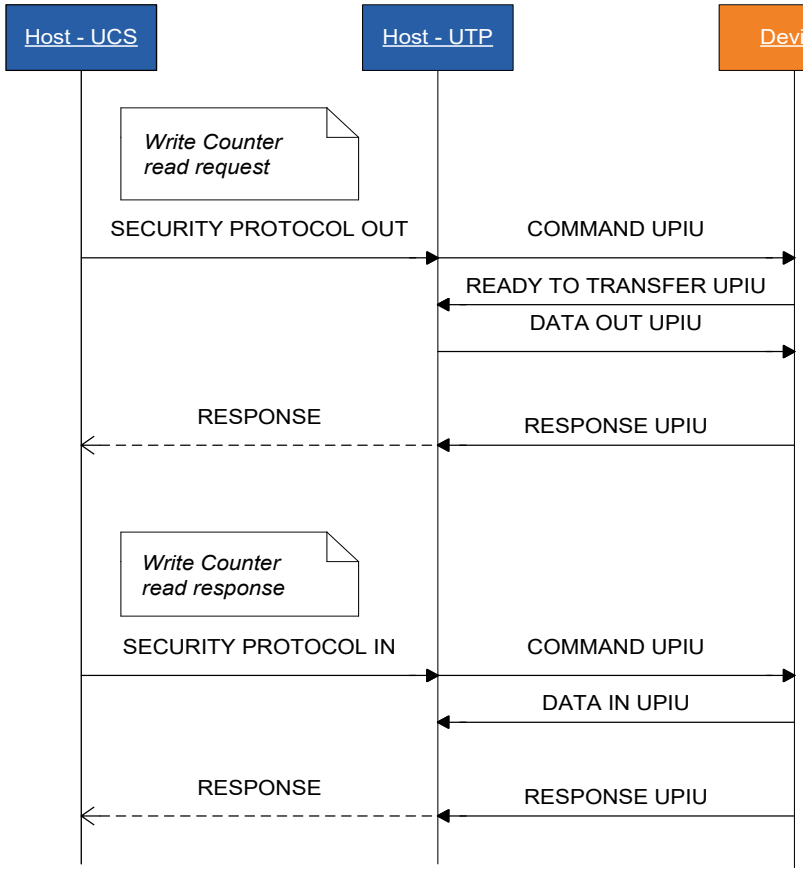
12.4.7.3.2 Read Counter Value

- The Read Counter Value sequence is initiated by a SECURITY PROTOCOL OUT command.
  - 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.
- When a GOOD status in the status response is received, the write counter value is retrieved sending a SECURITY PROTOCOL IN command.
  - 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.
  - The device returns a RPMB data frame with Response Message Type = 0200h, a copy of the Nonce received in the request, the Write Counter value, the MAC and the Result.

If reading of the counter value fails then returned result is “Read failure” (0006h/0086h).

If some other error occurs then Result is “General failure” (0001h/0081h).

If counter has expired also bit 7 is set to 1 in returned results.



Write Counter Read Request

Offset	Field Name	Value
0 :195	Stuff bytes	0...00h
196:227	MAC / Key	0...00h
228:483	Data	0...00h
484:499	Nonce	*Nonce from the host*
500:503	Write counter	0...00h
504:505	Address	0000h
506:507	Block Count	0000h
508:509	Result	0000h
510:511	Req./Resp.	0002h

Write Counter Read Response

Offset	Field Name	Value
0 :195	Stuff bytes	0...00h
196:227	MAC / Key	MAC from the device
228:483	Data	0...00h
484:499	Nonce	*Copy of the Nonce*
500:503	Write counter	*Counter Value*
504:505	Address	0000h
506:507	Block Count	0000h
508:509	Result	*Result code*
510:511	Req./Resp.	0200h

Figure 12.6 — Read Counter Value Flow

12.4.7.3.3 Authenticated Data Write

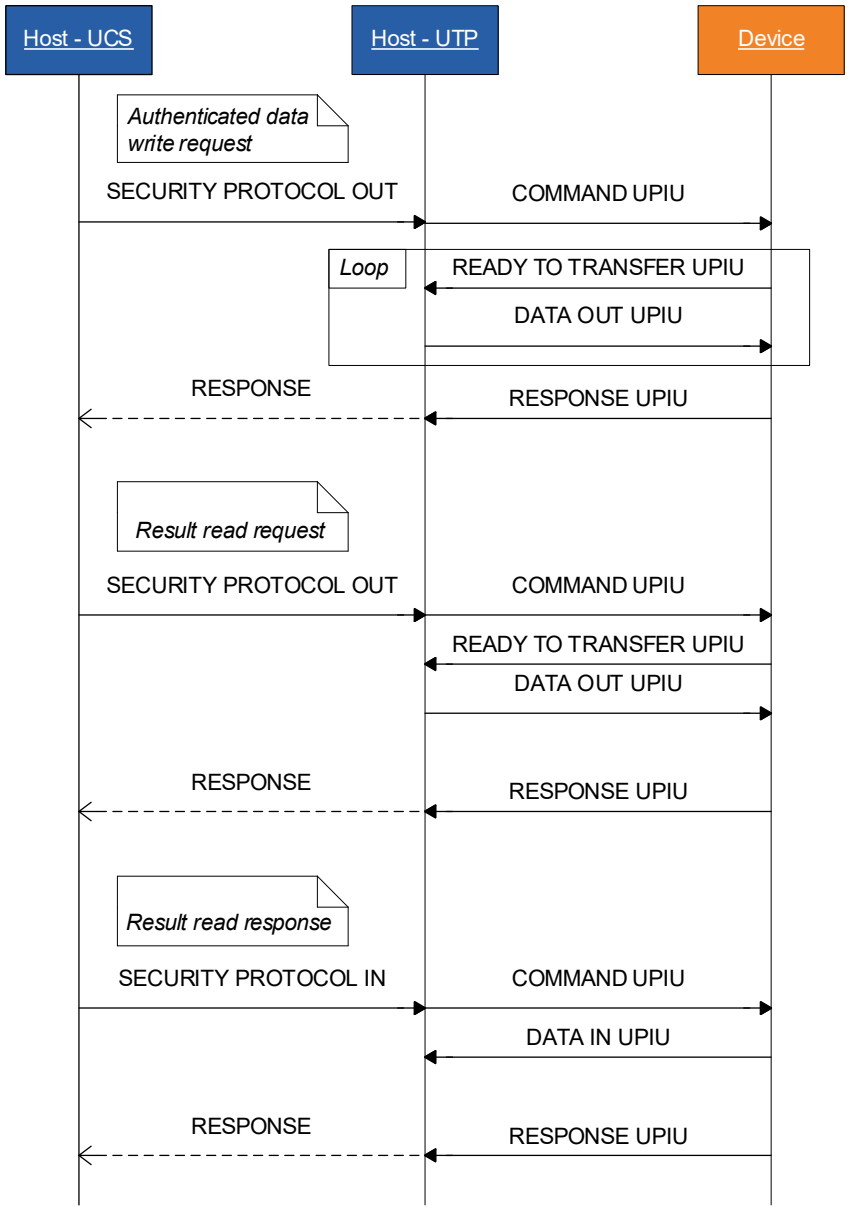
- The Authenticated Data Write sequence is initiated by a SECURITY PROTOCOL OUT command.
  - An initiator sends the SECURITY PROTOCOL OUT command with SECURITY PROTOCOL field set to ECh and indicating the RPMB region in the SECURITY PROTOCOL SPECIFIC field. The RPMB message is composed of one or more RPMB message data frames, each of which includes: Request Message Type = 0003h, Block Count, Address, Write Counter, Nonce, Data and MAC.
  - When the device receives the RPMB message, it first checks whether the write counter has expired. If the write counter is expired then the device sets the Result to “Write failure, write counter expired” (0085h). No data is written to the RPMB data area.
  - 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.
  - 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.
  - 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).
  - 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 MACs are different, then the device sets the Result to ”Authentication failure” (0002h). No data is written to the RPMB data area.
  - If the MAC in the request and the calculated MAC are equal then the device compares the write counter in the request with the write counter stored in the device. If the two counters are different then the device sets the Result to “Counter failure” (0003h). No data is written to the RPMB data area.
  - If the MAC and write counter comparisons are successful then the write request is considered to be authenticated. The data is written to the address indicated in the request.
  - The write counter is incremented by one if the write operation is successfully executed.
  - If write fails then returned result is “Write failure” (0005h).
  - If some other error occurs during the write procedure then returned result is “General failure” (0001h).
  - In an authenticated data write request with Block Count greater than one
    - the MAC is included only in the last RPMB message data frame. The MAC field is zero in all previous data frames. The device behavior is undefined if a MAC field is non-zero in any but the last RPMB message data frame.
    - In each data frame, the write counter indicates the current counter value, the address is the start address of the full access (not address of the individual logical block) and the block count is the total count of the blocks (not the block numbers).

12.4.7.3.3 Authenticated Data Write (cont'd)

  - When the authenticated data write operation is completed, the device may return GOOD status in response to the SECURITY PROTOCOL OUT command regardless of whether the Authenticated data write was successful or not.

- The authenticated data write verification process starts by issuing a SECURITY PROTOCOL OUT command.
  - An initiator sends a 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 contains the Request Message Type = 0005h (Result read request). Note that any request other than the Result read request from any initiator will overwrite the Result register of the RPMB Region.
  - The device returns GOOD status when the operation result is ready for retrieval.
- An initiator retrieves the operation result by issuing a SECURITY PROTOCOL IN command.
  - The SECURITY PROTOCOL field is set to ECh and the SECURITY PROTOCOL SPECIFIC field indicates the RPMB region.
- Device returns the RPMB data frame containing the Response Message Type = 0300h, the counter value (incremented if the write operation is successfully executed), a copy of the Nonce received in the request, the address received in the Authenticated data write request, the MAC and result of the authenticated data write operation.

12.4.7.3.3 Authenticated Data Write (cont'd)



Authenticated Data Write Request

Offset	Field Name	Value
0 :195	Stuff bytes	0...00h
196:227	MAC / Key	*MAC from the host*
228:483	Data	*Data*
484:499	Nonce	*Nonce from the Host*
500:503	Write counter	*Current Counter value*
504:505	Address	Address
506:507	Block Count	Number of 256B blocks
508:509	Result	0000h
510:511	Req./Resp.	0003h

Result Read Request

Offset	Field Name	Value
0 :195	Stuff bytes	0...00h
196:227	MAC / Key	0...00h
228:483	Data	0...00h
484:499	Nonce	0...00h
500:503	Write counter	0...00h
504:505	Address	0000h
506:507	Block Count	0000h
508:509	Result	0000h
510:511	Req./Resp.	0005h

Result Read Response

Offset	Field Name	Value
0 :195	Stuff bytes	0...00h
196:227	MAC / Key	*MAC from the device*
228:483	Data	0...00h
484:499	Nonce	*Copy of the Nonce*
500:503	Write counter	*New Counter value* (1)
504:505	Address	Address
506:507	Block Count	0000h
508:509	Result	Result code
510:511	Req./Resp.	0300h

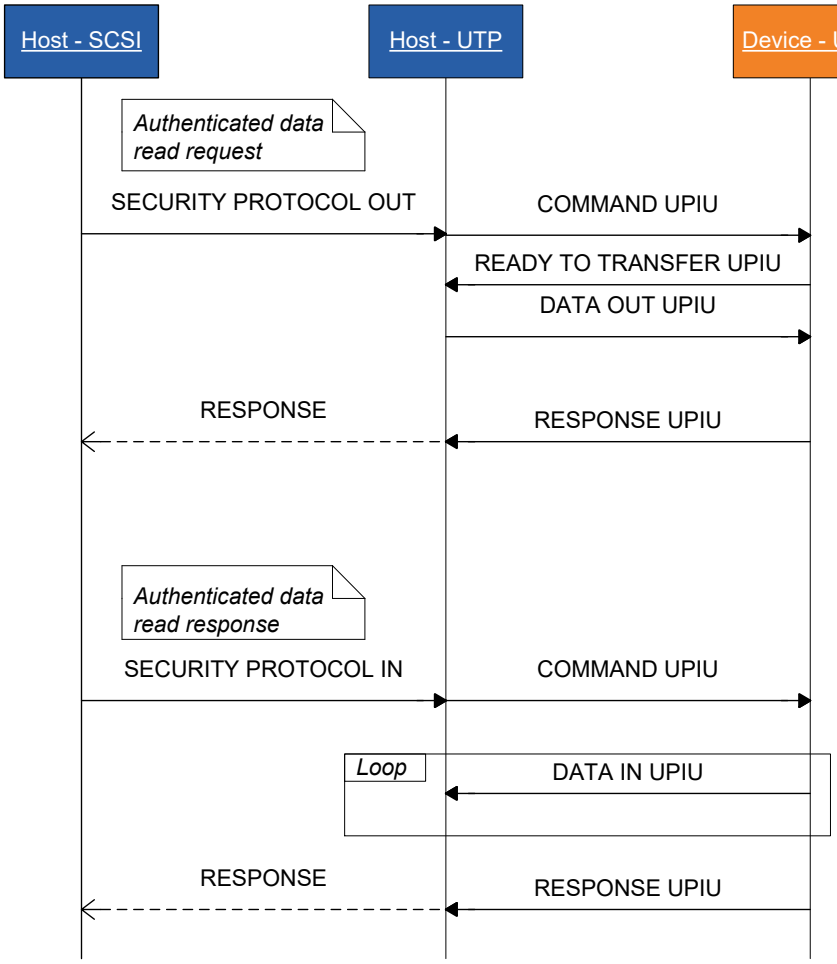
Note 1 The Write Counter is incremented only if the operation was successfully completed.

Figure 12.7 — Authenticated Data Write Flow

12.4.7.3.4 Authenticated Data Read

- The Authenticated Data Read sequence is initiated by a SECURITY PROTOCOL OUT command.
  - 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.
  - 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.
  - 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.
  - If the Block Count indicates a value greater than bRPMB_ReadWriteSize, then the Authenticated Data Read operation fails and the Result is set to "General failure" (0001h).
  - 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).
- If the SECURITY PROTOCOL OUT command completes with GOOD status, data can be retrieved sending a SECURITY PROTOCOL IN command.
  - 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.
  - 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.
  - In an authenticated data read response with Block Count greater than one,
    - the MAC is included only in the last RPMB message data frame. The MAC field is zero in all previous data frames.
    - In each data frame, the Nonce contains a copy of the received nonce, the address is the start address of the full access (not address of the individual logical block) and the block count is the total count of the blocks (not the sequence number of blocks).
- When the authenticated data read operation is completed, the device may return GOOD status in response to the SECURITY PROTOCOL IN command regardless of whether the Authenticated data read was successful or not.
- If data fetch from addressed location inside device fails then returned result is "Read failure" (0006h/0086h). If some other error occurs during the read procedure then returned result is "General failure" (0001h/0081h).

12.4.7.3.4 Authenticated Data Read (cont'd)



Authenticated Data Read Request

Offset	Field Name	Value
0 :195	Stuff bytes	0...00h
196:227	MAC / Key	0...00h
228:483	Data	0...00h
484:499	Nonce	*Nonce from the host*
500:503	Write counter	0...00h
504:505	Address	Address
506:507	Block Count	Number of 256B blocks
508:509	Result	0000h
510:511	Req./Resp.	0004h

Authenticated Data Read Response

Offset	Field Name	Value
0 :195	Stuff bytes	0...00h
196:227	MAC / Key	*MAC from the device*
228:483	Data	*Data*
484:499	Nonce	*Copy of the Nonce*
500:503	Write counter	0...00h
504:505	Address	Address
506:507	Block Count	Number of 256B blocks
508:509	Result	Result code
510:511	Req./Resp.	0400h

Figure 12.8 — Authenticated Data Read Flow

12.4.7.3.5 Authenticated Secure Write Protect Configuration Block Write

- Authenticated Secure Write Protect Configuration Block write operation is supported by RPMB region 0 only. If Authenticated Secure Write Protect Configuration Block write operation is issued to the RPMB region other than RPMB region 0, then returned result is “General failure” (0001h/0081h).

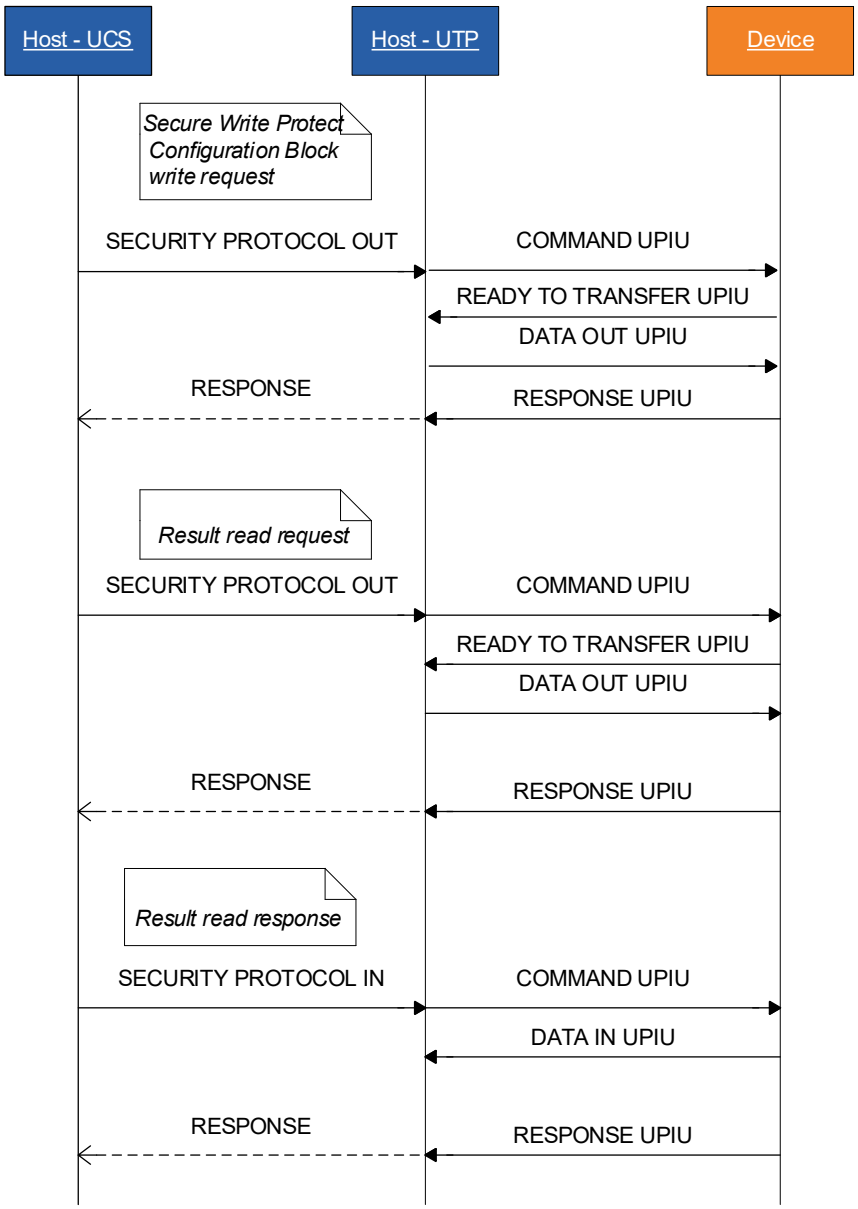
- The Authenticated Secure Write Protect Configuration Block write sequence is initiated by a SECURITY PROTOCOL OUT command.
  - An initiator sends the SECURITY PROTOCOL OUT command with SECURITY PROTOCOL field set to ECh and indicating the RPMB region 0 in the SECURITY PROTOCOL SPECIFIC field.
  - If the INC_512 bit and TRANSFER LENGTH field are not set to zero and 512 respectively, then the command shall be terminated with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.
  - The SECURITY PROTOCOL OUT command delivers a single RPMB message data frame which contains the Secure Write Protect Configuration Block in the Data field. The Secure Write Protect Configuration Block is specific of the logical unit indicated by the LUN field (byte 228 of the data frame).
  - The other fields of RPMB data frame are set as specified in the following: Request Message Type = 0006h, Result = 0000h, Block Count = 0001h, Address = 0000h, Write Counter = current counter value, Nonce from the host, and MAC = (see 12.4.4.2).
  - When the device receives the RPMB message data frame, it first checks whether the write counter has expired. If the write counter is expired then the device sets the result to “Write failure, write counter expired” (0085h). The Secure Write Protect Configuration Block is not updated.
  - If the write counter was not expired, then the device calculates the MAC of request type, block count, write counter, address and data, and compares this with the MAC in the request. If the two MACs are different, then the device sets the result to “Authentication failure” (0002h). The Secure Write Protect Configuration Block is not updated.
  - If the MAC in the request and the calculated MAC are equal, then the device compares the write counter in the request with the write counter stored in the device. If the two counters are different then the device sets the result to “Counter failure” (0003h). The Secure Write Protect Configuration Block is not updated.
  - If the MAC and write counter comparisons are successful then the write request is considered to be authenticated.
  - If the LUN field indicates a logical unit with bLUWriteProtect set to a value different from zero, then the device sets the result to “Secure Write Protection not applicable” (000Ah). The Secure Write Protect Configuration Block is not updated.

12.4.7.3.5 Authenticated Secure Write Protect Configuration Block Write (cont'd)

- The device sets the result to "Invalid Secure Write Protect Block Configuration parameter" (0009h) and it does not update the Secure Write Protect Configuration Block if one or more of the following conditions occurs.
  - the LUN field is invalid: greater than the value specified by bMaxNumberLU or if the logical unit is not enabled (bLUEnable = 00h).
  - the DATA LENGTH is set to a value different from the following ones: 0, 16, 32, 48, 64.
  - the LOGICAL BLOCK ADDRESS in a Secure Write Protect entry exceeds the logical unit capacity,
  - the LOGICAL BLOCK ADDRESS plus the NUMBER OF LOGICAL BLOCKS in a Secure Write Protect entry exceeds the logical unit capacity,
  - two or more Secure Write Protect entries specify overlapping areas,
  - with this request, the number of Secure Write Protect areas set in the entire device is increased to a value greater than what indicated by bNumSecureWPArea.
- If some other error occurs during the write procedure then returned result is "Secure Write Protect Configuration Block access failure" (0008h). The Secure Write Protect Configuration Block is not updated.
- If no error occurred, then the Secure Write Protect Configuration Block is updated overwriting the former configuration and the write counter is incremented by one.

- The device may return GOOD status in response to the SECURITY PROTOCOL OUT command regardless of whether the Authenticated Secure Write Protect Configuration Block Write was successful or not.
- The successfulness of the programming of the data can be checked by retrieving the result register of the RPMB.
- The verification process starts by issuing a SECURITY PROTOCOL OUT command.
  - An initiator sends a SECURITY PROTOCOL OUT command with SECURITY PROTOCOL field set to ECh and indicating the RPMB region 0 in the SECURITY PROTOCOL SPECIFIC field. The RPMB data frame contains the Request Message Type = 0005h (Result read request). Note that any request other than the Result read request from any initiator will overwrite the Result register of the RPMB Region.
  - The device returns "Good" status when the operation result is ready for retrieval.
- An initiator retrieves the operation result by issuing a SECURITY PROTOCOL IN command.
  - The SECURITY PROTOCOL field is set to ECh and the SECURITY PROTOCOL SPECIFIC field indicates the RPMB region.
- Device returns the RPMB data frame containing the Response Message Type = 0600h, the incremented counter value, a copy of the Nonce received in the request, the MAC and result of the Authenticated Secure Write Protect Configuration Block write operation.

12.4.7.3.5 Authenticated Secure Write Protect Configuration Block Write (cont'd)



Secure Write Protect Configuration Block write request

Offset	Field Name	Value
0 :195	Stuff bytes	0...00h
196:227	MAC / Key	*MAC from the host*
228:483	Data	*LUN, DATA LENGTH, Secure Write Protect Entries*
484:499	Nonce	*Nonce from the host*
500:503	Write counter	*Current Counter value*
504:505	Address	0000h
506:507	Block Count	0001h
508:509	Result	0000h
510:511	Req./Resp.	0006h

Result Read Request

Offset	Field Name	Value
0 :195	Stuff bytes	0...00h
196:227	MAC / Key	0...00h
228:483	Data	0...00h
484:499	Nonce	0...00h
500:503	Write counter	0...00h
504:505	Address	0000h
506:507	Block Count	0000h
508:509	Result	0000h
510:511	Req./Resp.	0005h

Result Read Response

Offset	Field Name	Value
0 :195	Stuff bytes	0...00h
196:227	MAC / Key	*MAC from the device*
228:483	Data	0...00h
484:499	Nonce	*Copy of the Nonce*
500:503	Write counter	*New Counter value* (1)
504:505	Address	0000h
506:507	Block Count	0000h
508:509	Result	*Result code*
510:511	Req./Resp.	0600h

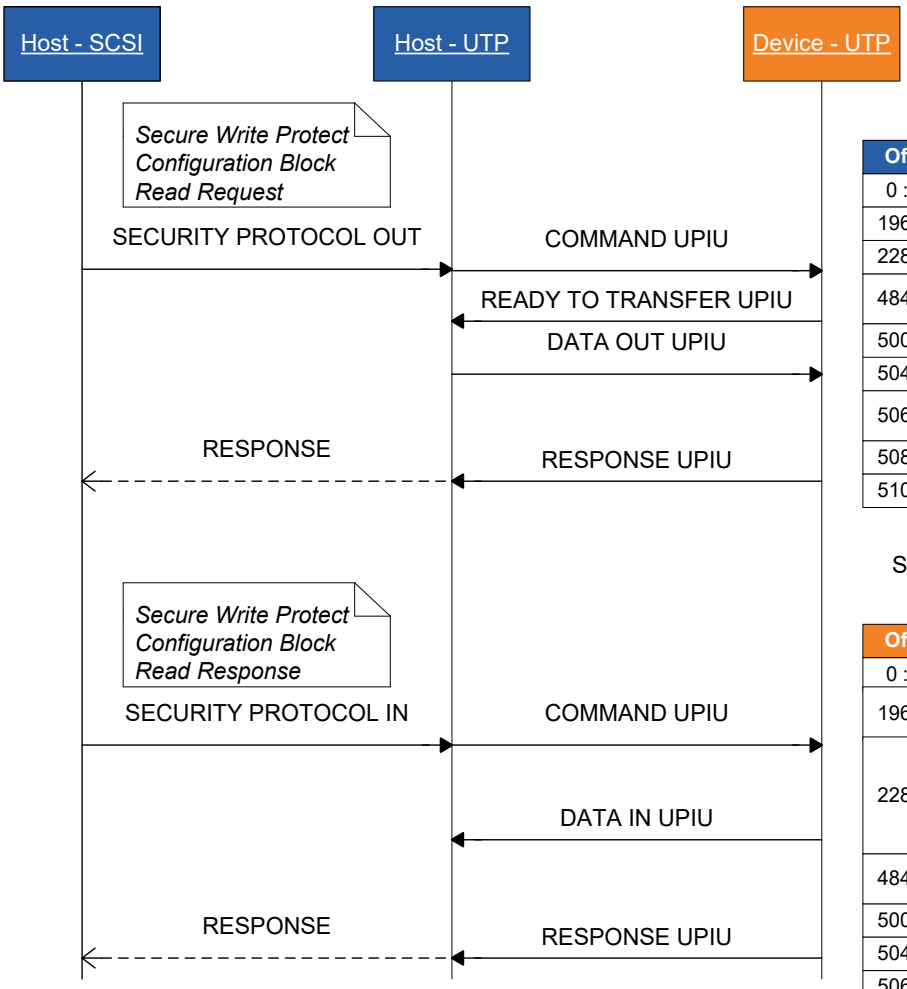
NOTE 1 The Write Counter is incremented only if the operation was successfully completed.

Figure 12.9 —Authenticated Secure Write Protect Configuration Block Write Flow

12.4.7.3.6 Authenticated Secure Write Protect Configuration Block Read

- Authenticated Secure Write Protect Configuration Block read operation is supported by RPMB region 0 only. If Authenticated Secure Write Protect Configuration Block read operation is issued to the RPMB region other than RPMB region 0, then returned result is “General failure” (0001h/0081h).
- The Authenticated Secure Write Protect Configuration Block Read sequence is initiated by a SECURITY PROTOCOL OUT command.
    - An initiator sends the SECURITY PROTOCOL OUT command with SECURITY PROTOCOL field set to ECh and indicating the RPMB region 0 in the SECURITY PROTOCOL SPECIFIC field.
    - If the INC_512 bit and TRANSFER LENGTH field are not set to zero and 512 respectively, then the command shall be terminated with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.
    - The SECURITY PROTOCOL OUT command delivers a single RPMB message data frame which contains in the Data field a LUN value (byte 228 of the data frame). The Secure Write Protect Configuration Block that will be returned is specific of the logical unit indicated by the LUN field.
    - The RPMB data frame delivered from the host to the device includes the Request Message Type = 0007h, Block Count = 0001h, Address = 0000h, the Data and Nonce. In this request, the LUN is the only relevant byte of the Data field, all other bytes shall be considered reserved and they shall be ignored.
    - If the LUN field is not valid, then the device sets the result to “Invalid Secure Write Protect Block Configuration parameter” (0009h/0089h). The LUN field is invalid if it is greater than the value specified by bMaxNumberLU or if the logical unit is not enabled (bLUEnable = 00h).
    - If the LUN field indicates a logical unit with bLUWriteProtect set to a value different from zero, then the device sets the result to “Secure Write Protection not applicable” (000Ah/008Ah).
    - After successful fetch of the Secure Write Protect Configuration Block, the MAC is calculated from response type, nonce, address, data and result. If the MAC calculation fails then returned result is “Authentication failure” (0002h/0082h).
- If the SECURITY PROTOCOL OUT command completes with GOOD status, then the Secure Write Protect Configuration Block can be retrieved sending a SECURITY PROTOCOL IN command.
    - An initiator sends the SECURITY PROTOCOL IN command with SECURITY PROTOCOL field set to ECh, indicating the RPMB region 0 in the SECURITY PROTOCOL SPECIFIC field, and in which INC_512 bit and ALLOCATION LENGTH field indicate 512 bytes.
- The device returns a RPMB data frame with Response Message Type = 0700h, the block count, a copy of the nonce received in the request, the contents of the Secure Write Protect Configuration Block in the Data field, the MAC and the result
- If data fetch from addressed location inside device fails or some other error occurs during the read procedure then returned result is “Secure Write Protect Configuration Block access failure” (0008h/0088h).

12.4.7.3.6 Authenticated Secure Write Protect Configuration Block Read (cont'd)



Secure Write Protect Configuration Block Read Request

Offset	Field Name	Value
0 :195	Stuff bytes	0...00h
196:227	MAC / Key	0...00h
228:483	Data	*LUN, 0...00h*
484:499	Nonce	*Nonce from the host*
500:503	Write counter	0...00h
504:505	Address	0...00h
506:507	Block Count	0001h
508:509	Result	0000h
510:511	Req./Resp.	0007h

Secure Write Protect Configuration Block Read Response

Offset	Field Name	Value
0 :195	Stuff bytes	0...00h
196:227	MAC / Key	*MAC from the device*
228:483	Data	*LUN, DATA LENGTH, Secure Write Protect Entries*
484:499	Nonce	*Copy of the Nonce*
500:503	Write counter	0...00h
504:505	Address	0...00h
506:507	Block Count	0001h
508:509	Result	Result code
510:511	Req./Resp.	0700h

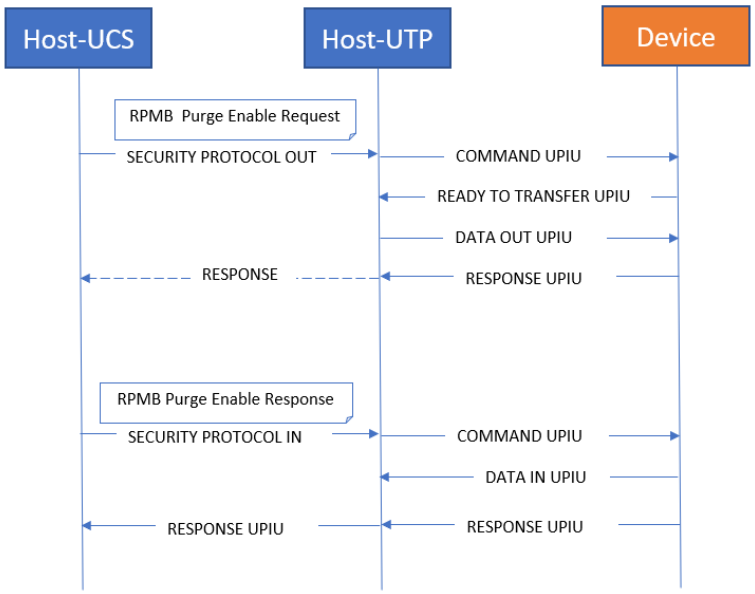
Figure 12.10 — Authenticated Secure Write Protect Configuration Block Read Flow

12.4.7.3.7 RPMB Purge Enable

The RPMB Purge Enable Request sequence is initiated by a SECURITY PROTOCOL OUT command. The sequence is completed by a SECURITY PROTOCOL IN command. This pair of commands is the RPMB Purge Enable Request operation.

- 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 = 0008h (**RPMB Purge Enable Request**), Write Counter, Nonce, and MAC.
  - When the device receives the Request message (RPMB Purge Enable Request), 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) and the RPMB Purge is not initiated.
  - If the Write Counter was not expired, then the device calculates the MAC of request and compares this with the MAC in the request. If the two MACs are different, then the device sets the Result to "Authentication failure" (0002h) and RPMB Purge operation is not initiated.
  - If the Write Counter and MAC comparisons are successful, then the RPMB Purge Enable Request is considered to be authenticated and the RPMB Purge operation on the target RPMB Region is initiated.
  - When the device is ready to start purge the RPMB, the device can return a GOOD status in response to the SECURITY PROTOCOL OUT command, regardless of whether the RPMB purge operation was successful or not. The success or failure of RPMB purge can be known by the operation of RPMB Purge Status Read.
- When a GOOD status in the status response is received, the RPMB operation result is retrieved by sending a SECURITY PROTOCOL IN command.
  - The 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.
  - The device returns an RPMB data frame with the Request Message Type = 0800h (**RPMB Purge Enable Response**), Write Counter, Nonce, and MAC.
  - On the successful completion of the RPMB Purge Enable Request Message, the device shall update the write counter in the RPMB Purge Read Enable Response.
- If a GOOD status is not returned, a general failure error should be reported.

12.4.7.3.7 RPMB Purge Enable (cont'd)



RPMB Purge Enable Request

Offset	Field Name	Value
0:195	Stuff bytes	00h
196:227	MAC / Key	MAC From Host
228:483	Data	00..00h
484:499	Nonce	Nonce Value
500:503	Write Counter	Current Counter value
504:505	Address	0000h
506:507	Block Count	0001h
508:509	Result	0000h
510:511	Req./Resp.	0008h

RPMB Purge Enable Response

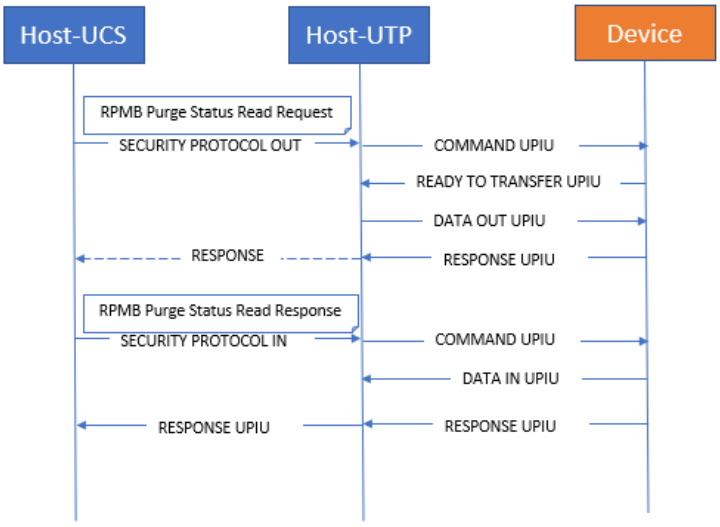
Offset	Field Name	Value
0:195	Stuff bytes	00h
196:227	MAC / Key	MAC From Device
228:483	Data	00..00h
484:499	Nonce	Copy of the Nonce
500:503	Write Counter	New Counter value (if purge enable successfully)
504:505	Address	0000h
506:507	Block Count	0001h
508:509	Result	Result Code
510:511	Req./Resp.	0800h

Figure 12.11 — RPMB Purge Enable Flow

12.4.7.3.8 RPMB Purge Status Read

RPMB Purge Status Read Request sequence is initiated by a SECURITY PROTOCOL OUT command. The sequence is completed by a SECURITY PROTOCOL IN command. This pair of commands is the RPMB Purge Status Read Request operation.

- An initiator sends a 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 contains the Request Message Type = 0009h (**RPMB Purge Status Read Request**) and Nonce.
  - Note that any request other than the RPMB Purge Status Read Request from any initiator will overwrite the Result register of the RPMB Region.
  - The device returns GOOD status when the operation result is ready for retrieval.
- When a GOOD status in the status response is received, the RPMB purge status is retrieved by sending a SECURITY PROTOCOL IN command.
  - 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.
  - The device returns a RPMB data frame with Response Message Type = 0900h (**RPMB Purge Status Read Response**), a copy of the Nonce received in the request, the MAC and the Result.
  - Supported purge status value are as follows, reported in the RPMB Purge Response Packet:
    - 00 - RPMB Purge not initiated (reset value)
    - 01 - RPMB Purge in progress
    - 02 - RPMB Purge successfully completed
    - 03 - RPMB Purge general failure
- If a GOOD status is not returned, a general failure error should be reported.



RPMB Purge Status Read Request

Offset	Field Name	Value
0:195	Stuff bytes	00h
196:227	MAC / Key	00..00h
228:483	Data	00..00h
484:499	Nonce	Nonce Value
500:503	Write Counter	00..00h
504:505	Address	0000h
506:507	Block Count	0001h
508:509	Result	0000h
510:511	Req./Resp.	0009h

RPMB Purge Status Read Response

Offset	Field Name	Value
0:195	Stuff bytes	00h
196:227	MAC / Key	MAC From Device
228:483	Data	RPMB Purge Response
484:499	Nonce	Copy of the Nonce
500:503	Write Counter	00..00h
504:505	Address	0000h
506:507	Block Count	0001h
508:509	Result	Result Code
510:511	Req./Resp.	0900h

Figure 12.12 — RPMB Purge Status Read Flow

12.4.7.3.9 Authenticated Vendor Specific Command Request

Authenticated Vendor Specific Command allows the vendor command & response to be tunneled to the device via RPMB authentication mechanism.

The Authenticated Vendor Specific Command Request sequence is initiated by a SECURITY PROTOCOL OUT command.

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 = 0010h **(Authenticated Vendor Specific Command Request)**, Block Count, Write Counter, Nonce, data and MAC.

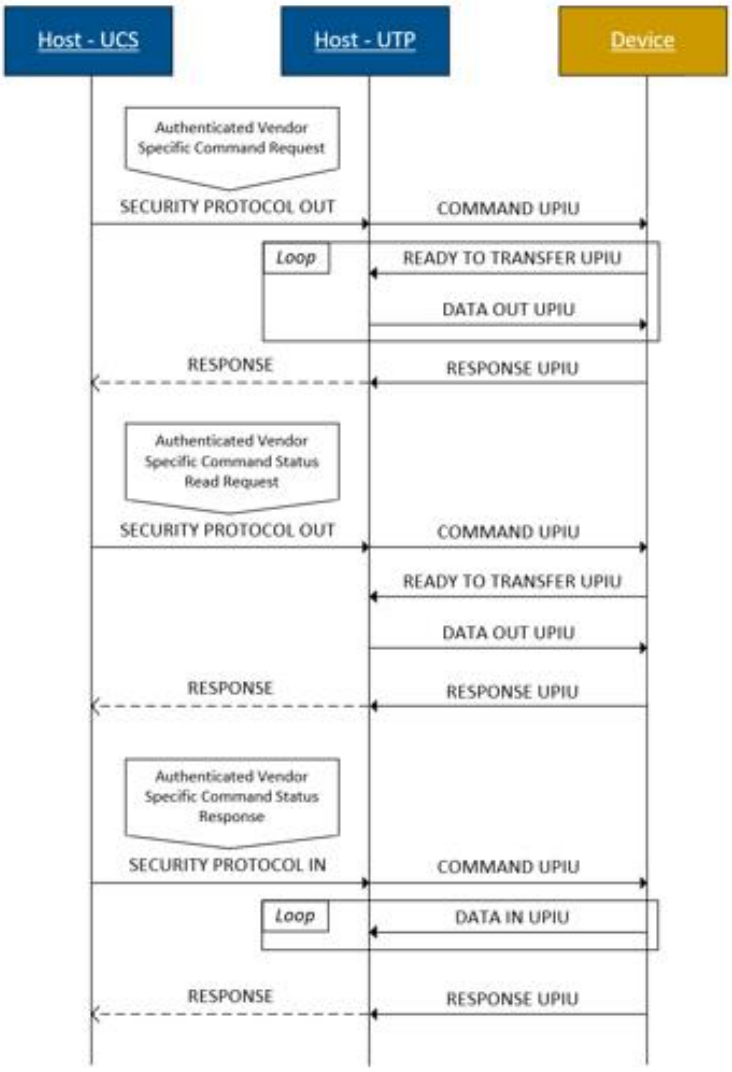
- When the device receives the Request message (Authenticated Vendor Specific Command Request), it first checks whether the write counter has expired. If the write counter is expired then the device sets the Result to "Write failure, write counter expired" (0085h) and the Vendor Specific Command operation is not initiated.
- If the Write Counter was not expired, then the device calculates the MAC of request and compares this with the MAC in the request. If the two MACs are different, then the device sets the Result to "Authentication failure" (0002h) and Vendor Specific Command operation is not initiated.
- If the Write Counter and MAC comparisons are successful, then the Vendor Specific Command operation Request is considered to be authenticated and the operation is initiated.
- When the device is ready to start the vendor specific command, the device can return a GOOD status in response to the SECURITY PROTOCOL OUT command, regardless of whether the vendor specific command was successful or not. The success or failure of vendor specific command request can be known by the operation of Authenticated Vendor Specific Command Response Read Request.
- On the successful completion of the Vendor Specific Command Request Message, the device shall increment the write counter.

When a GOOD status in the status response is received, the result of the vendor specific command request is retrieved by sending a SECURITY PROTOCOL OUT command.

If a GOOD status is not returned, a general failure error should be reported.

- In an Authenticated Vendor Specific Command Request with Block Count greater than one,
  - the MAC is included only in the last RPMB message data frame. The MAC field is zero in all previous data frames. The device behavior is undefined if a MAC field is non-zero in any but the last RPMB message data frame.
  - In each data frame, the write counter indicates the current counter value, the block count is the total count of the blocks (not the block numbers).

12.4.7.3.9 Authenticated Vendor Specific Command Request (cont'd)



Authenticated Vendor Specific Command Request

Offset	Field Name	Value
0:195	Stuff bytes	0..00h
196:227	MAC/Key	Mac from the host
228:483	Data	Vendor Specific Command + Data
484:499	Nonce	Nonce from the host
500:503	Write counter	Current Counter value
504:505	Address	0000h
506:507	Block Count	Number of 256B blocks
508:509	Result	0000h
510:511	Req./Resp.	0010h

Authenticated Vendor Specific Command Status Read Request

Offset	Field Name	Value
0:195	Stuff bytes	0..00h
196:227	MAC/Key	0..00h
228:483	Data	0..00h
484:499	Nonce	0..00h
500:503	Write counter	0..00h
504:505	Address	0000h
506:507	Block Count	Number of 256B blocks
508:509	Result	0000h
510:511	Req./Resp.	0011h

Authenticated Vendor Specific Command Status Response

Offset	Field Name	Value
0:195	Stuff bytes	0..00h
196:227	MAC/Key	Mac from the device
228:483	Data	Vendor Specific Response + Data
484:499	Nonce	Copy of the Nonce
500:503	Write counter	New Counter value
504:505	Address	0000h
506:507	Block Count	Number of 256B blocks
508:509	Result	Status
510:511	Req./Resp.	1100h

Figure 12.13 — Authenticated Vendor Specific Command Request Flow

12.4.7.3.10 Authenticated Vendor Specific Command Status Read Request

Authenticated Vendor Specific Command Status Read Request sequence is initiated by a SECURITY PROTOCOL OUT command. The sequence is completed by a SECURITY PROTOCOL IN command. This pair of commands is the Authenticated Vendor Specific Command Status Read Operation.

An initiator sends a 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 contains the Request Message Type = 0011h (**Authenticated Vendor Specific Command Read Request**), Block Countand Nonce.

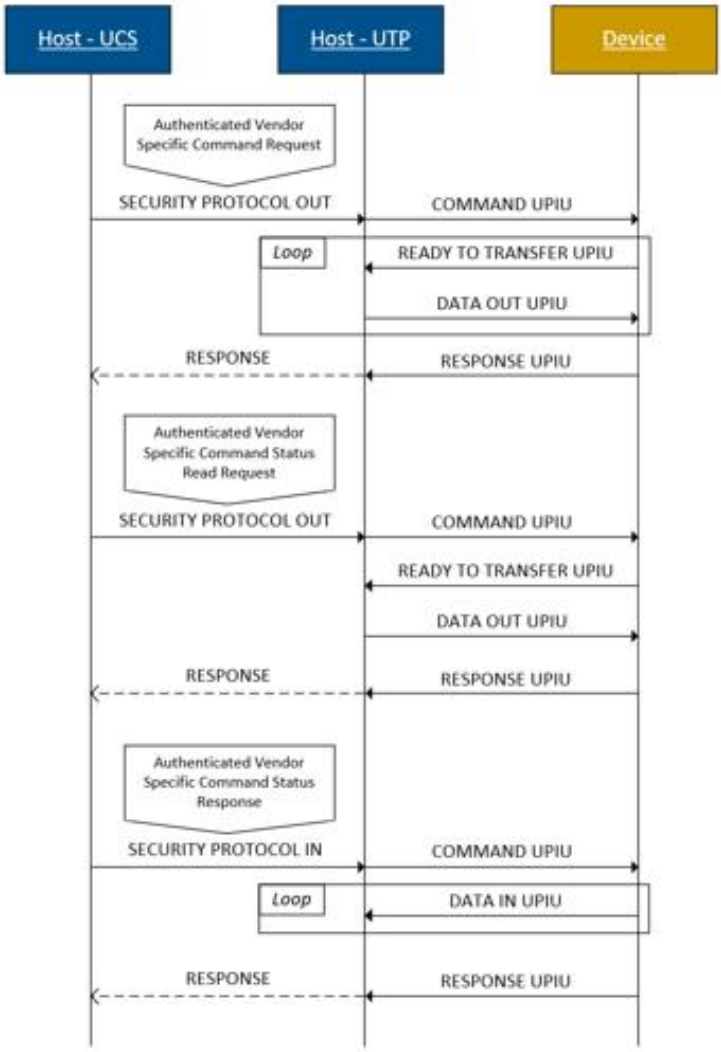
- Note that any request other than the Authenticated Vendor Specific Command Response Read Request from any initiator will overwrite the Result register of the RPMB Region.
- The device returns GOOD status when the operation result is ready for retrieval.

When a GOOD status in the status response is received, the Authenticated Vendor Specific Command status is retrieved by sending a SECURITY PROTOCOL IN command.

- 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.
- The device returns a RPMB data frame with Response Message Type = 1100h (**Authenticated Vendor Specific Command Status Response**), the Block Count, a copy of the Nonce received in the request, the data, the MAC and the Result.
- In an Authenticated Vendor Specific Command Status Response with Block Count greater than one,
  - the MAC is included only in the last RPMB message data frame. The MAC field is zero in all previous data frames.
  - In each data frame, the Nonce contains a copy of the received nonce, the block count is the total count of the blocks (not the sequence number of blocks).
- Supported Vendor Command status value are as follows, reported in the Authenticated Vendor Specific Command Response Packet's Result field:
  - 00 - Vendor Specific Command not initiated (reset value)
  - 01 - Vendor Specific Command in progress
  - 02 - Vendor Specific Command completed
  - 03 - Vendor Specific Command general failure

If a GOOD status is not returned, a general failure error should be reported.

12.4.7.3.10 Authenticated Vendor Specific Command Status Read Request (cont'd)



Authenticated Vendor Specific Command Status Read Request

Offset	Field Name	Value
0:195	Stuff bytes	0..00h
196:227	MAC/Key	0..00h
228:483	Data	0..00h
484:499	Nonce	0..00h
500:503	Write counter	0..00h
504:505	Address	0000h
506:507	Block Count	Number of 256B blocks
508:509	Result	0000h
510:511	Req./Resp.	0011h

Authenticated Vendor Specific Command Status Response

Offset	Field Name	Value
0:195	Stuff bytes	0..00h
196:227	MAC/Key	Mac from the device
228:483	Data	Vendor Specific Response + Data
484:499	Nonce	Copy of the Nonce
500:503	Write counter	New Counter value
504:505	Address	0000h
506:507	Block Count	Number of 256B blocks
508:509	Result	Status
510:511	Req./Resp.	1100h

Figure 12.14 — Authenticated Vendor Specific Command Read Request Flow

12.4.7.4 RPMB Operations in Advanced RPMB Mode

In Advanced RPMB, EHS shall be used. Therefore, "Total EHS Length" shall be "2h", "bLength" and "bEHSType" in "EHS Entry" shall be respectively "02h" and "01h". Otherwise, the command shall be terminated with CHECK CONDITION status. See 12.4.5.1 for the details of error handling.

12.4.7.4.1 Authentication Key Programming

- The Authentication Key programming is initiated by a SECURITY PROTOCOL OUT command.
  - 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 = 0001h and the Authentication Key.
  - If "Transfer Length" is not "0h", then the command shall be terminated with CHECK CONDITION status. See 12.4.5.1 for the details of error handling.
  - The device returns GOOD status in status response when Authentication Key programming is completed.
  - If programming of Authentication Key failed then returned result is "Write failure" (0005h). If some other error occurred during Authentication Key programming then returned result is "General failure" (0001h).

Access to RPMB data area is not possible before the Authentication Key is programmed in the corresponding RPMB region. The state of the device can be checked by trying to write/read data to/from the RPMB data area: if the Authentication Key is not programmed then the Result field in the response message will be set to "Authentication Key not yet programmed" (0007h).

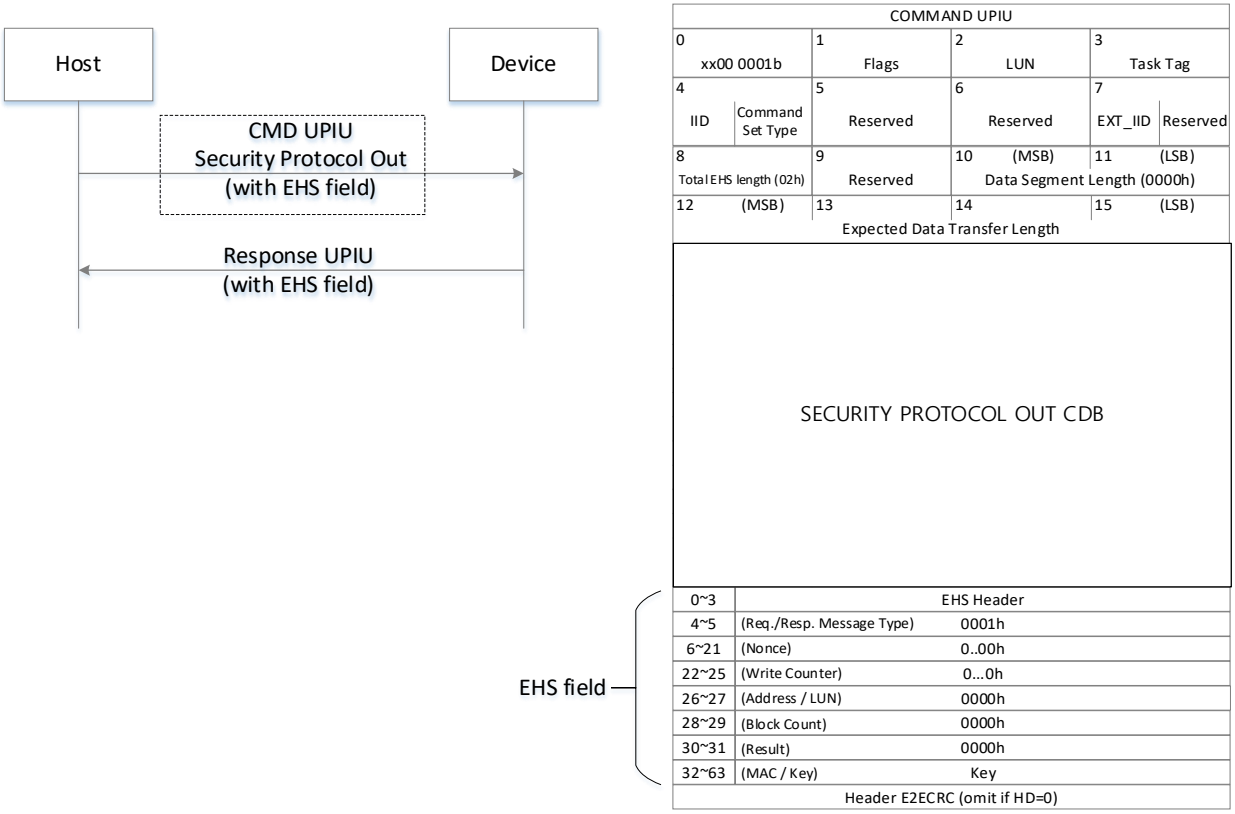
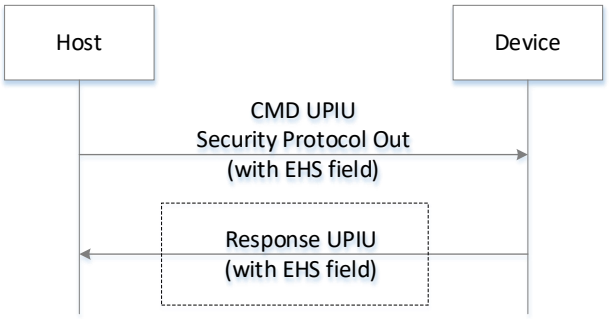


Figure 12.15 — Authentication Key Programming Flow (in Advanced RPMB Mode)

12.4.7.4.1 Authentication Key Programming (cont'd)



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

EHS field(2)

0~3	EHS Header	
4~5	(Req./Resp. Message Type)	0100h
6~21	(Nonce)	0..00h
22~25	(Write Counter)	0..00h
26~27	(Address / LUN)	0000h
28~29	(Block Count)	0000h
30~31	(Result)	Result Code
32~63	(MAC / Key)	0...00h

Header E2ECRC (omit if HD=0)			
K (MSB) Sense Data Length	K+1 (LSB)	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 and EHS field is not included
K=96 if HD=0 and EHS field is included
Note 2 EHS field shall not be sent when SPI/SPO command terminated with CHECK CONDITION status. In this case, the Total EHS length is 0h.

Figure 12.15 - Authentication Key Programming Flow (in Advanced RPMB Mode) (cont'd)

12.4.7.4.2 Read Counter Value

- The Read Counter Value sequence is initiated by a SECURITY PROTOCOL OUT IN command.
  - An initiator sends the SECURITY PROTOCOL OUT IN 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.
  - If "Allocation Length" is not "0h", then the command shall be terminated with CHECK CONDITION status. See 12.4.5.1 for the details of error handling.
  - The device returns a RPMB data frame with Response Message Type = 0200h, a copy of the Nonce received in the request, the Write Counter value, the MAC and the Result.

If reading of the counter value fails then returned result is "Read failure" (0006h/0086h).

If some other error occurs then Result is "General failure" (0001h/0081h).

If counter has expired also bit 7 is set to 1 in returned results.

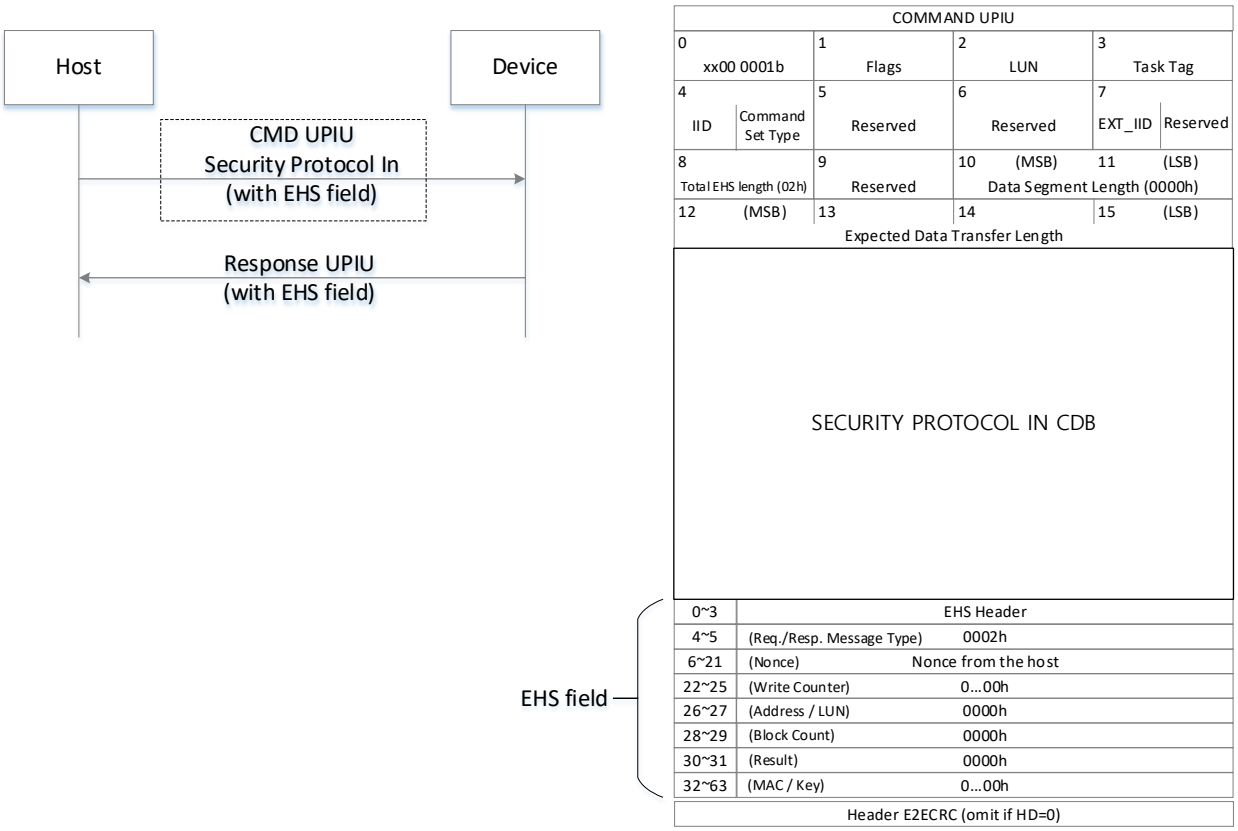
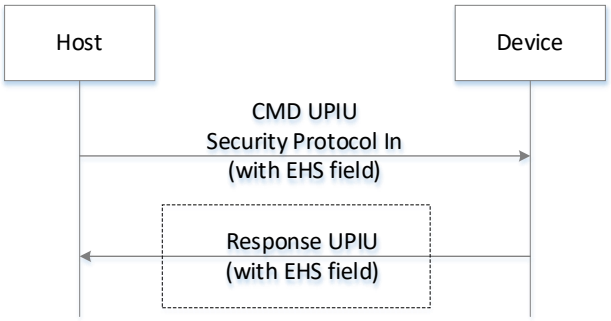


Figure 12.16— Read Counter Value Flow (in Advanced RPMB Mode)

12.4.7.4.2 Read Counter Value (cont'd)



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

EHS field(2)

0~3	EHS Header	
4~5	(Req./Resp. Message Type)	0200h
6~21	(Nonce)	Copy of the Nonce
22~25	(Write Counter)	Counter Value
26~27	(Address / LUN)	0000h
28~29	(Block Count)	0000h
30~31	(Result)	Result Code
32~63	(MAC / Key)	MAC from the Device

Header E2ECRC (omit if HD=0)			
K (MSB)	K+1 (LSB)	K+2	K+3
Sense Data Length		Sense Data[0]	Sense Data[1]
...	...	...	...
K+16	K+17	K+18	K+19
Sense Data[14]	Sense Data[15]	Sense Data[16]	Sense Data[17]
Data E2ECRC (omit if DD=0)			

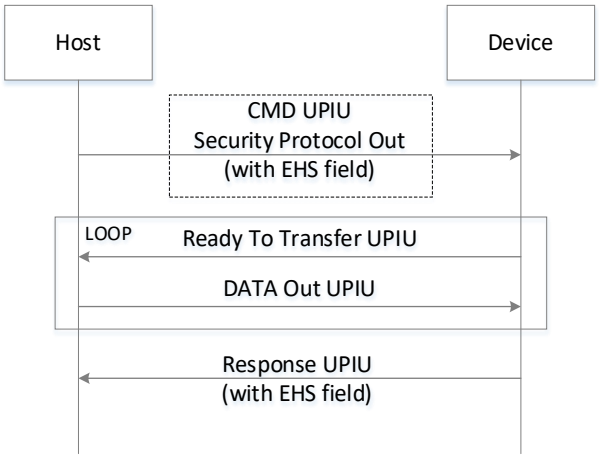
Note 1 K=32 if HD=0 and EHS field is not included
K=96 if HD=0 and EHS field is included
Note 2 EHS field shall not be sent when SPI/SPO command terminated with CHECK CONDITION status. In this case, the Total EHS length is 0h.

Figure 12.16 — Read Counter Value Flow (in Advanced RPMB Mode) (cont'd)

12.4.7.4.3 Authenticated Data Write

- The Authenticated Data Write sequence is initiated by a SECURITY PROTOCOL OUT command.
  - 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 Request Message Type = 0003h, Advanced RPMB Block Count, Address, Write Counter, Nonce, Data and MAC.
  - When the device receives the RPMB message, it first checks whether the write counter has expired. If the write counter is expired then the device sets the Result to “Write failure, write counter expired” (0085h). No data is written to the RPMB data area.
  - 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.
  - If the Address value plus the Advanced RPMB 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.
  - If the Advanced RPMB 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).
  - If the write counter was not expired then the device calculates the MAC of request type, Advanced RPMB Block Count, write counter, address and data, and compares this with the MAC in the request. If the two MACs are different, then the device sets the Result to ”Authentication failure” (0002h). No data is written to the RPMB data area.
  - If the MAC in the request and 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.
  - If the MAC and write counter comparisons are successful then the write request is considered to be authenticated. The data is written to the address indicated in the request.
  - The write counter is incremented by one if the write operation is successfully executed.
  - If write fails then returned result is “Write failure” (0005h).
  - If some other error occurs during the write procedure then returned result is “General failure” (0001h).
  - In an authenticated data write request with Advanced RPMB Block Count greater than one
    - the MAC is included in RESPONSE UPIU.
    - When the authenticated data write operation is completed, the device may return GOOD status in response to the SECURITY PROTOCOL OUT command regardless of whether the Authenticated Data Write was successful or not.
- Device returns the RPMB data frame containing the Response Message Type = 0300h, the counter value (incremented if the write operation is successfully executed), a copy of the Nonce received in the request, the address received in the Authenticated data write request, the MAC and result of the authenticated data write operation.

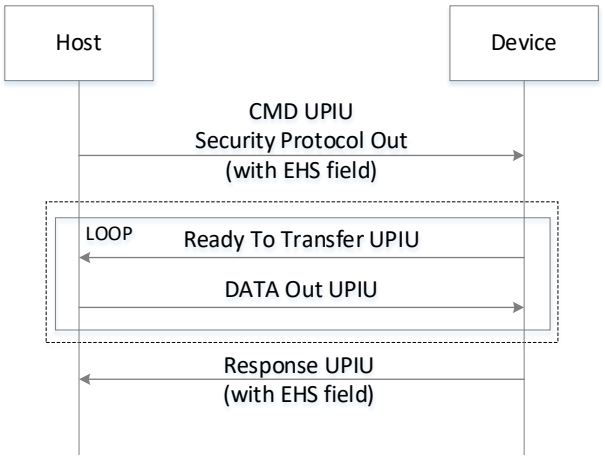
12.4.7.4.3 Authenticated Data Write (cont'd)



COMMAND UPIU							
0		1	2	3			
xx00 0001b		Flags	LUN	Task Tag			
4		5	6	7			
IID	Command Set Type	Reserved	Reserved	EXT_IID	Reserved		
8		9	10 (MSB)	11 (LSB)			
Total EHS length (02h)		Reserved	Data Segment Length (0000h)				
12 (MSB)		13	14	15 (LSB)			
Expected Data Transfer Length							
SECURITY PROTOCOL OUT CDB							

EHS field

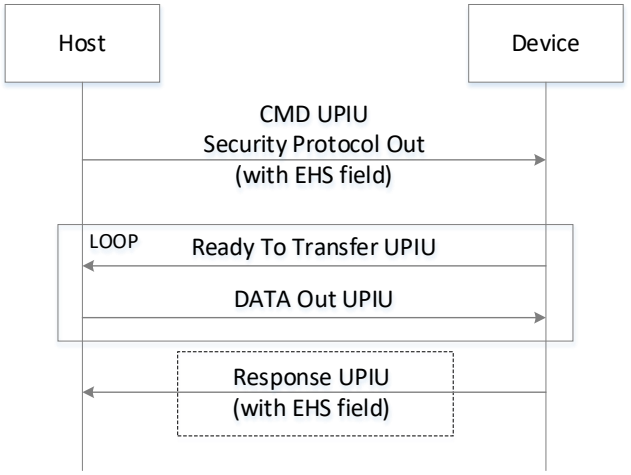
0~3	EHS Header	
4~5	(Req./Resp. Message Type)	0003h
6~21	(Nonce)	Nonce from the host
22~25	(Write Counter)	Current counter Value
26~27	(Address / LUN)	Address
28~29	(Block Count)	Advanced RPMB Block Count
30~31	(Result)	0000h
32~63	(MAC / Key)	Mac from the host
Header E2ECRC (omit if HD=0)		



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

Figure 12.17 — Authenticated Data Write Flow (in Advanced RPMB Mode)

12.4.7.4.3 Authenticated Data Write (cont'd)



EHS field(2)

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

0~3	EHS Header
4~5	(Req./Resp. Message Type) 0300h
6~21	(Nonce) Copy of Nonce
22~25	(Write Counter) New Counter Value(3)
26~27	(Address / LUN) Address
28~29	(Block Count) 0000h
30~31	(Result) Result Code
32~63	(MAC / Key) MAC from the device

Header E2ECRC (omit if HD=0)			
K (MSB) Sense Data Length	K+1 (LSB)	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 and EHS field is not included
K=96 if HD=0 and EHS field is included
Note 2 EHS field shall not be sent when SPI/SPO command terminated with CHECK CONDITION status. In this case, the Total EHS length is 0h.
Note 3 The Write Counter is incremented only if the Enable operation was successfully completed

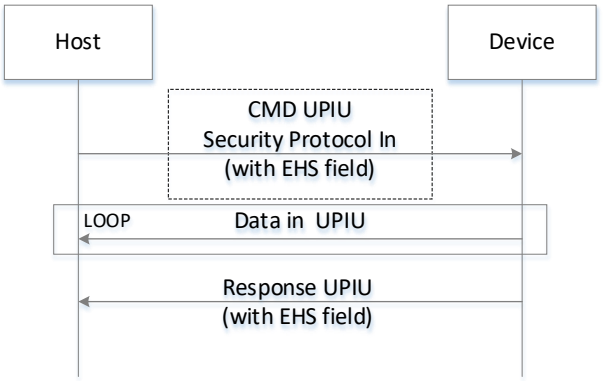
Figure 12.17 — Authenticated Data Write Flow (in Advanced RPMB Mode) (cont'd)

12.4.7.4.4 Authenticated Data Read

The Authenticated Data Read sequence is initiated by a SECURITY PROTOCOL IN command.

- 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. The RPMB data frame includes the Request Message Type = 0004h, the nonce, the data address, and the Advanced RPMB Block Count.
  - 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.
  - If the Address value plus the Advanced RPMB 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.
  - If the Block Count indicates a value greater than bRPMB_ReadWriteSize, then the Authenticated Data Read operation fails and the Result is set to “General failure” (0001h).
  - 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).
  - In an authenticated data read response with Advanced RPMB Block Count greater than one,
    - the MAC is included only in the RESPONSE UPIU
    - When the authenticated data read operation is completed, the device may return GOOD status in response to the SECURITY PROTOCOL IN command regardless of whether the Authenticated Data Read was successful or not.
- If data fetch from addressed location inside device fails then returned result is “Read failure” (0006h/0086h). If some other error occurs during the read procedure then returned result is “General failure” (0001h/0081h).

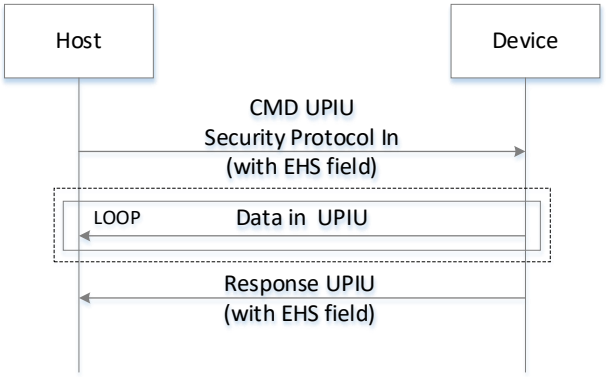
12.4.7.4.4 Authenticated Data Read (cont'd)



COMMAND UPIU							
0		1	2	3			
xx00 0001b		Flags	LUN	Task Tag			
4		5	6	7			
IID	Command Set Type	Reserved	Reserved	EXT_IID	Reserved		
8		9	10 (MSB)	11 (LSB)			
Total EHS length (02h)		Reserved	Data Segment Length (0000h)				
12 (MSB)		13	14	15 (LSB)			
Expected Data Transfer Length							
SECURITY PROTOCOL IN CDB							

EHS field

0~3	EHS Header
4~5	(Req./Resp. Message Type) 0004h
6~21	(Nonce) Nonce from the host
22~25	(Write Counter) 0…00h
26~27	(Address / LUN) Address
28~29	(Block Count) Advanced RPMB Block Count
30~31	(Result) 0000h
32~63	(MAC / Key) 0…00h
Header E2ECRC (omit if HD=0)	

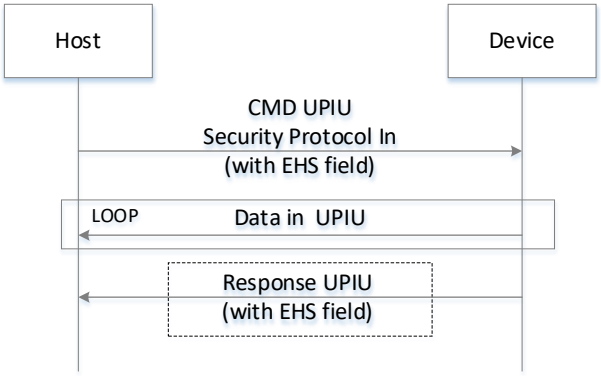


DATA IN UPIU							
0		1	2	3			
xx10 0010b		Flags	LUN	Task Tag			
4		5		6	7		
IID	Reserved	EXT_IID	Reserved	Response	Reserved		
8		9	10 (MSB)	11 (LSB)			
Total EHS length (00h)		Reserved	Data Segment Length				
12 (MSB)		13	14	15 (LSB)			
Data Buffer Offset							
16		17	18	19			
Data transfer Count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

Note 1 K=32 if HD=0

Figure 12.18 — Authenticated Data Read Flow (in Advanced RPMB Mode)

12.4.7.4.4 Authenticated Data Read (cont'd)



EHS field(2)

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

0~3	EHS Header
4~5	(Req./Resp. Message Type) 0400h
6~21	(Nonce) Copy of the Nonce
22~25	(Write Counter) 0…00h
26~27	(Address / LUN) Address
28~29	(Block Count) Advanced RPMB Block Count
30~31	(Result) Result Code
32~63	(MAC / Key) MAC from the device

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 and EHS field is not included
K=96 if HD=0 and EHS field is included
Note 2 EHS field shall not be sent when SPI/SPO command terminated with CHECK CONDITION status. In this case, the Total EHS length is 0h.

Figure 12.18 — Authenticated Data Read Flow (in Advanced RPMB Mode) (cont'd)

12.4.7.4.5 Authenticated Secure Write Protect Configuration Block Write

- Authenticated Secure Write Protect Configuration Block write operation is supported by RPMB region 0 only. If Authenticated Secure Write Protect Configuration Block write operation is issued to the RPMB region other than RPMB region 0, then returned result is “General failure” (0001h/0081h).
  - The Authenticated Secure Write Protect Configuration Block write sequence is initiated by a SECURITY PROTOCOL OUT command.
  - An initiator sends the SECURITY PROTOCOL OUT command with SECURITY PROTOCOL field set to ECh and indicating the RPMB region 0 in the SECURITY PROTOCOL SPECIFIC field.
  - If the INC_512 bit and TRANSFER LENGTH field are not set to zero and 4096 respectively, then the command shall be terminated with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.
  - The SECURITY PROTOCOL OUT command delivers a single RPMB message data frame which contains the Secure Write Protect Configuration Block in the Data field. The Secure Write Protect Configuration Block is specific of the logical unit indicated by the LUN field
  - The other fields of RPMB data frame are set as specified in the following: Request Message Type = 0006h, Result = 0000h, Advanced RPMB Block Count = 0001h, Address = LUN, Write Counter = current counter value, Nonce from the host, and MAC.
  - When the device receives the RPMB message data frame, it first checks whether the write counter has expired. If the write counter is expired then the device sets the result to “Write failure, write counter expired” (0085h). The Secure Write Protect Configuration Block is not updated.
  - If the write counter was not expired, then the device calculates the MAC of request type, Advanced RPMB Block Count, write counter, address and data, and compares this with the MAC in the request. If the two MACs are different, then the device sets the result to “Authentication failure” (0002h). The Secure Write Protect Configuration Block is not updated.
  - If the MAC in the request and the calculated MAC are equal, then the device compares the write counter in the request with the write counter stored in the device. If the two counters are different then the device sets the result to “Counter failure” (0003h). The Secure Write Protect Configuration Block is not updated.
  - If the MAC and write counter comparisons are successful then the write request is considered to be authenticated.
  - If the LUN field indicates a logical unit with bLUWriteProtect set to a value different from zero, then the device sets the result to “Secure Write Protection not applicable” (000Ah). The Secure Write Protect Configuration Block is not updated.
  - The device sets the result to “Invalid Secure Write Protect Block Configuration parameter” (0009h) and it does not update the Secure Write Protect Configuration Block if one or more of the following conditions occurs.
    - The LUN field is invalid: greater than the value specified by bMaxNumberLU or if the logical unit is not enabled (bLUEnable = 00h)), or if the LUN value differs from the LUN value in the Advanced RPMB Meta Information in EHS.
    - The DATA LENGTH is set to a value different from the following ones: 0, 16, 32, 48, 64.
    - The LOGICAL BLOCK ADDRESS in a Secure Write Protect entry exceeds the logical unit capacity.

12.4.7.4.5 Authenticated Secure Write Protect Configuration Block Write (cont'd)

- The LOGICAL BLOCK ADDRESS plus the NUMBER OF LOGICAL BLOCKS in a Secure Write Protect entry exceeds the logical unit capacity.
- Two or more Secure Write Protect entries specify overlapping areas.
- With this request, the number of Secure Write Protect areas set in the entire device is increased to a value greater than what indicated by bNumSecureWPArea.
  - If some other error occurs during the write procedure then returned result is "Secure Write Protect Configuration Block access failure" (0008h). The Secure Write Protect Configuration Block is not updated.
  - If no error occurred, then the Secure Write Protect Configuration Block is updated overwriting the former configuration and the write counter is incremented by one.

- The device may return GOOD status in response to the SECURITY PROTOCOL OUT command regardless of whether the Authenticated Secure Write Protect Configuration Block Write was successful or not.
- Device returns the RPMB data frame containing the Response Message Type = 0600h, the incremented counter value, a copy of the Nonce received in the request, the MAC and result of the Authenticated Secure Write Protect Configuration Block write operation.

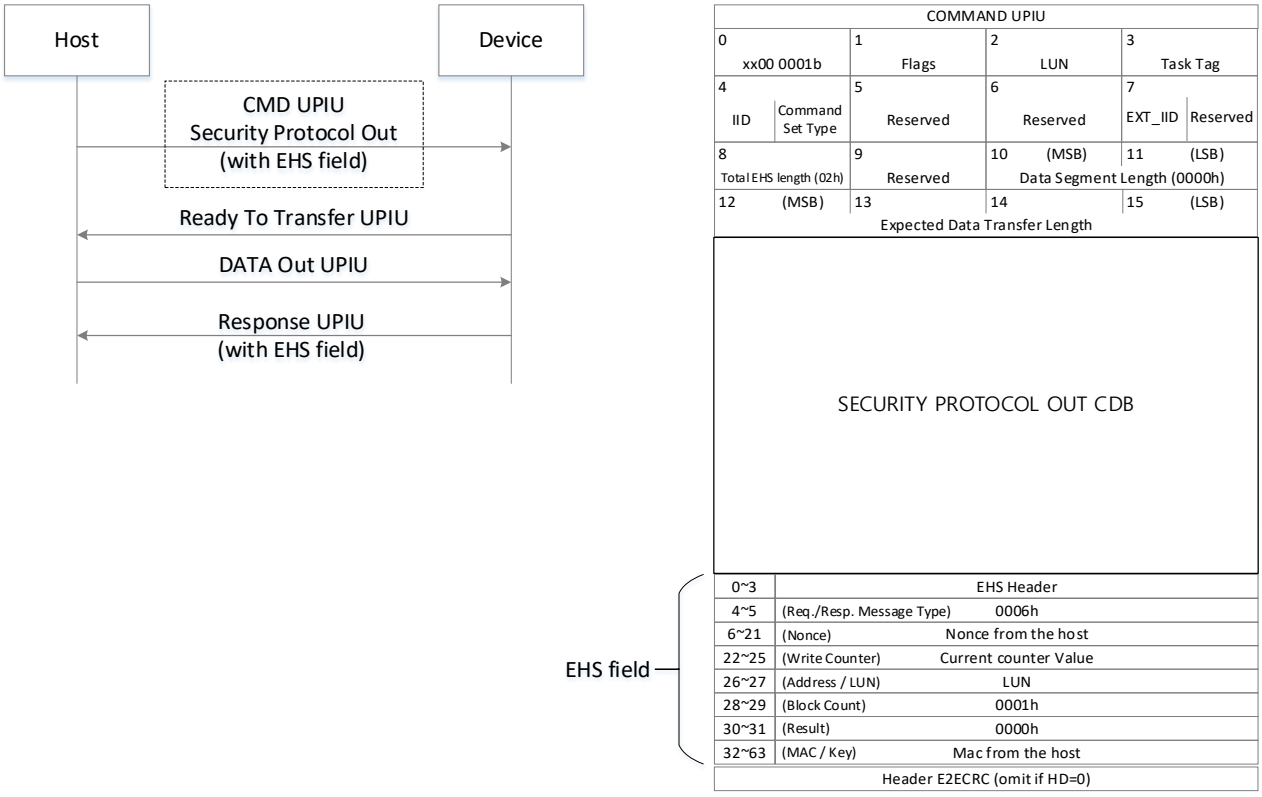
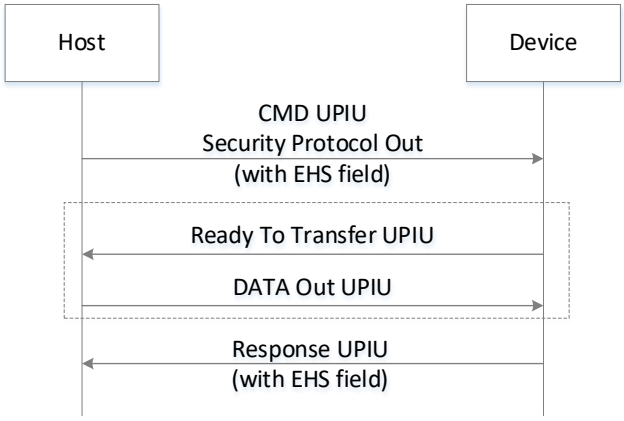


Figure 12.19 — Authenticated Secure Write Protect Configuration Block Write Flow (in Advanced RPMB Mode)

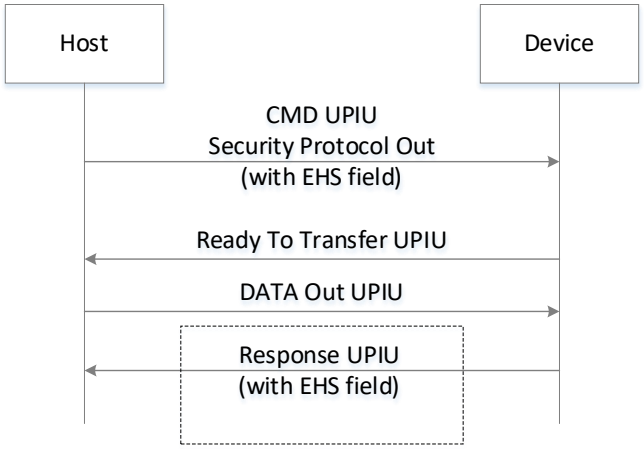
12.4.7.4.5 Authenticated Secure Write Protect Configuration Block Write (cont'd)



DATA OUT UPIU			
0 xx00 0010b	1 Flags	2 LUN	3 Task Tag
4 IID / Reserved	5 Reserved	6 Reserved	7 EXT_IID / Reserved
8 Total EHS length (00h)	9 Reserved	10 (MSB) / 11 (LSB) Data Segment Length	
12 (MSB) / 13	14 / 15 (LSB) Data Buffer Offset		
16 / 17	18 / 19 Data transfer Count		
20 / 21	22 / 23 Reserved		
24 / 25	26 / 27 Reserved		
28 / 29	30 / 31 Reserved		
Header E2ECRC (omit if HD=0)			
K LUN	K+1 DATA LENGTH	K+2 Reserved	K+3 Reserved
K+16 / ...	... / ...	... / ...	K+31 Secure Write Protect Entry 0
...	...	...	...
K+64 / ...	... / ...	... / ...	K+79 Secure Write Protect Entry 3
K+80 / ...	... / ...	... / ...	... Reserved ("0" padding)
Data E2ECRC (omit if DD=0)			
Note 1 K=32 if HD=0			

Figure 12.19 — Authenticated Secure Write Protect Configuration Block Write Flow (in Advanced RPMB Mode) (cont'd)

12.4.7.4.5 Authenticated Secure Write Protect Configuration Block Write (cont'd)



EHS field(2)

RESPONSE UPIU			
0 Xx00 0001b	1 Flags	2 LUN	3 Task Tag
4 IID / Command Set Type	5 EXT_IID / Reserved	6 Response	7 Status
8 Total EHS length (2)	9 Device Information	10 (MSB) / 11 (LSB) Data Segment Length	
12 (MSB) / 13 / 14 / 15 (LSB) Residual Transfer Count			
16 / 17 / 18 / 19 Reserved			
20 / 21 / 22 / 23 Reserved			
24 / 25 / 26 / 27 Reserved			
28 / 29 / 30 / 31 Reserved			

EHS Header	
0~3	
4~5	(Req./Resp. Message Type) 0600h
6~21	(Nonce) Copy of Nonce
22~25	(Write Counter) New Counter Value(3)
26~27	(Address / LUN) LUN
28~29	(Block Count) 0000h
30~31	(Result) Result Code
32~63	(MAC / Key) MAC from the device

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)			

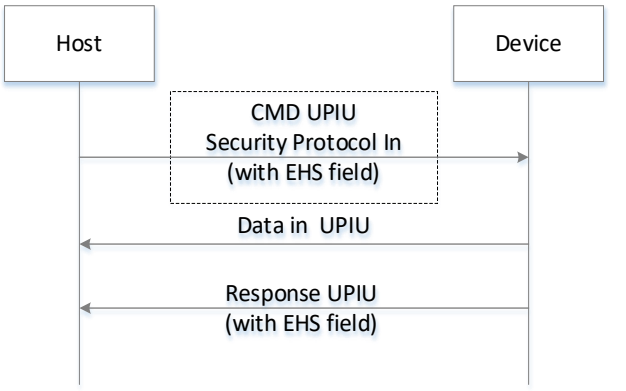
Note 1 K=32 if HD=0 and EHS field is not included
K=96 if HD=0 and EHS field is included
Note 2 EHS field shall not be sent when SPI/SPO command terminated with CHECK CONDITION status. In this case, the Total EHS length is 0h.
Note 3 The Write Counter is incremented only if the Enable operation was successfully completed

Figure 12.19 — Authenticated Secure Write Protect Configuration Block Write Flow (in Advanced RPMB Mode) (cont'd)

12.4.7.4.6 Authenticated Secure Write Protect Configuration Block Read

- Authenticated Secure Write Protect Configuration Block read operation is supported by RPMB region 0 only. If Authenticated Secure Write Protect Configuration Block read operation is issued to the RPMB region other than RPMB region 0, then returned result is “General failure” (0001h/0081h).
- The Authenticated Secure Write Protect Configuration Block Read sequence is initiated by a SECURITY PROTOCOL IN command.
  - An initiator sends the SECURITY PROTOCOL IN command with SECURITY PROTOCOL field set to ECh and indicating the RPMB region 0 in the SECURITY PROTOCOL SPECIFIC field.
  - If the INC_512 bit and TRANSFER LENGTH field are not set to zero and 4096 respectively, then the command shall be terminated with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.
  - The SECURITY PROTOCOL IN command delivers a single RPMB message data frame which contains the LUN field. The Secure Write Protect Configuration Block that will be returned is specific of the logical unit indicated by the LUN field.
  - The RPMB data frame delivered from the host to the device includes the Request Message Type = 0007h, Advanced RPMB Block Count = 0001h, Address = LUN, the Data and Nonce.
  - If the LUN field is not valid, then the device sets the result to “Invalid Secure Write Protect Block Configuration parameter” (0009h/0089h). The LUN field is invalid if it is greater than the value specified by bMaxNumberLU or if the logical unit is not enabled (bLUEnable = 00h).
  - If the LUN field indicates a logical unit with bLUWriteProtect set to a value different from zero, then the device sets the result to “Secure Write Protection not applicable” (000Ah/008Ah).
  - After successful fetch of the Secure Write Protect Configuration Block, the MAC is calculated from response type, nonce, address, data and result. If the MAC calculation fails then returned result is “Authentication failure” (0002h/0082h).
- The device returns a RPMB data frame with Response Message Type = 0700h, the Advanced RPMB Block Count, a copy of the nonce received in the request, the contents of the Secure Write Protect Configuration Block in the Data field, the MAC and the result
- If data fetch from addressed location inside device fails or some other error occurs during the read procedure then returned result is “Secure Write Protect Configuration Block access failure” (0008h/0088h).

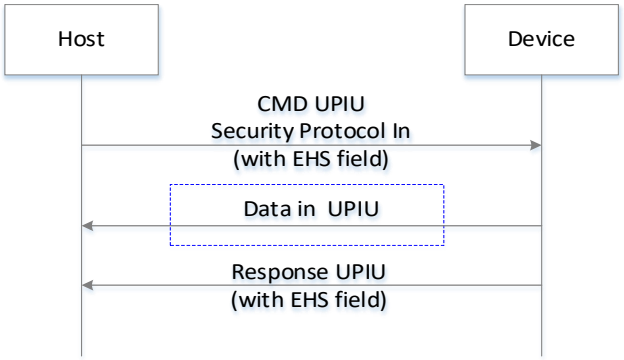
12.4.7.4.6 Authenticated Secure Write Protect Configuration Block Read Flow (cont'd)



COMMAND UPIU							
0		1	2		3		
Xx00 0001b		Flags	LUN		Task Tag		
4		5	6		7		
IID	Command Set Type	Reserved	Reserved		EXT_IID	Reserved	
8		9	10	(MSB)	11	(LSB)	
Total EHS length (02h)		Reserved	Data Segment Length (0000h)				
12	(MSB)	13	14		15	(LSB)	
Expected Data Transfer Length							
SECURITY PROTOCOL IN CDB							

EHS field

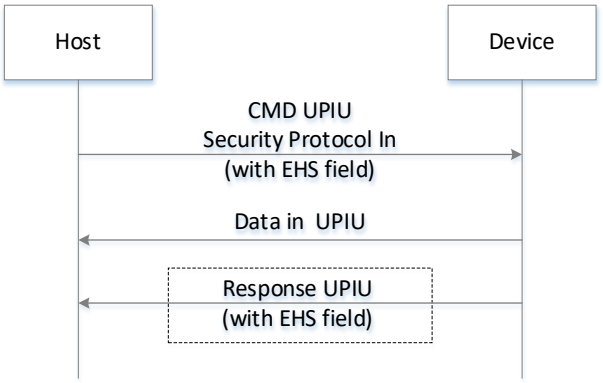
0~3	EHS Header	
4~5	(Req./Resp. Message Type)	0007h
6~21	(Nonce)	Nonce from the host
22~25	(Write Counter)	0...00h
26~27	(Address / LUN)	LUN
28~29	(Block Count)	0001h
30~31	(Result)	0000h
32~63	(MAC / Key)	0...00h
Header E2ECRC (omit if HD=0)		



DATA IN UPIU							
0		1	2		3		
xx10 0010b		Flags	LUN		Task Tag		
4		5			6	7	
IID	Reserved	EXT_IID	Reserved		Reserved	Reserved	
8		9	10	(MSB)	11	(LSB)	
Total EHS length (00h)		Reserved	Data Segment Length				
12	(MSB)	13	14		15	(LSB)	
Data Buffer Offset							
16		17	18		19		
Data transfer Count							
20		21	22		23		
Reserved							
24		25	26		27		
Reserved							
28		29	30		31		
Reserved							
Header E2ECRC (omit if HD=0)							
K		K+1	K+2		K+3		
LUN		DATA LENGTH	Reserved		Reserved		
K+16		...	...		K+31		
Secure Write Protect Entry 0							
...		...	...		...		
K+64		...	...		K+79		
Secure Write Protect Entry 3							
K+80		...	...		...		
Reserved ("0" padding)							
Data E2ECRC (omit if DD=0)							
Note 1 K=32 if HD=0							

Figure 12.20 — Authenticated Secure Write Protect Configuration Block Read Flow (in Advanced RPMB Mode)

12.4.7.4.6 Authenticated Secure Write Protect Configuration Block Read Flow (cont'd)



EHS field(2)

RESPONSE UPIU			
0 Xx00 0001b	1 Flags	2 LUN	3 Task Tag
4 IID / Command Set Type	5 EXT_IID / Reserved	6 Response	7 Status
8 Total EHS length (2)	9 Device Information	10 (MSB) / 11 (LSB) Data Segment Length	
12 (MSB) / 13 / 14 / 15 (LSB) Residual Transfer Count			
16 / 17 / 18 / 19 Reserved			
20 / 21 / 22 / 23 Reserved			
24 / 25 / 26 / 27 Reserved			
28 / 29 / 30 / 31 Reserved			

EHS Header		
0~3		
4~5	(Req./Resp. Message Type)	0700h
6~21	(Nonce)	Copy of the Nonce
22~25	(Write Counter)	0...00h
26~27	(Address / LUN)	LUN
28~29	(Block Count)	0001h
30~31	(Result)	Result Code
32~63	(MAC / Key)	MAC from the device

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 and EHS field is not included
K=96 if HD=0 and EHS field is included
Note 2 EHS field shall not be sent when SPI/SPO command terminated with CHECK CONDITION status. In this case, the Total EHS length is 0h.

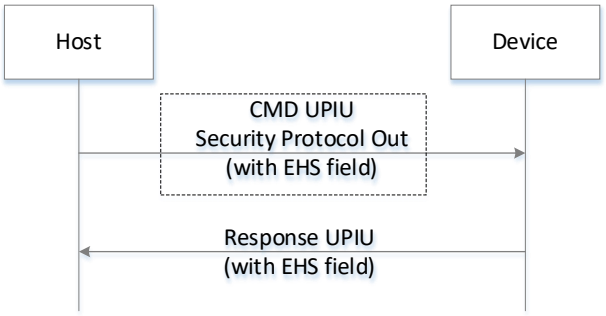
Figure 12.20 — Authenticated Secure Write Protect Configuration Block Read Flow (in Advanced RPMB Mode) (cont'd)

12.4.7.4.7 RPMB Purge Enable

The RPMB Purge operation is initiated by a SECURITY PROTOCOL OUT command.

- 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 = 0008h (**RPMB Purge Enable Request**), Write Counter, Nonce, and MAC.
    - When the device receives the Request message (RPMB Purge Enable Request), 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) and the RPMB Purge is not initiated.
    - If the Write Counter was not expired, then the device calculates the MAC of request and compares this with the MAC in the request. If the two MACs are different, then the device sets the Result to "Authentication failure" (0002h) and RPMB Purge operation is not initiated.
    - If the Write Counter and MAC comparisons are successful, then the RPMB Purge Enable Request is considered to be authenticated and the RPMB Purge operation on the target RPMB Region is initiated.
    - When the device is ready to start purge the RPMB, the device can return a GOOD status in response to the SECURITY PROTOCOL OUT command, regardless of whether the RPMB purge operation was successful or not. The success or failure of RPMB purge can be known by the operation of RPMB Purge Status Read.
    - On the successful completion of the RPMB Purge Enable Request Message, the device shall update the write counter in the RPMB Purge Enable Response.
- If a GOOD status is not returned, a general failure error should be reported.

12.4.7.4.7 RPMB Purge Enable (cont'd)



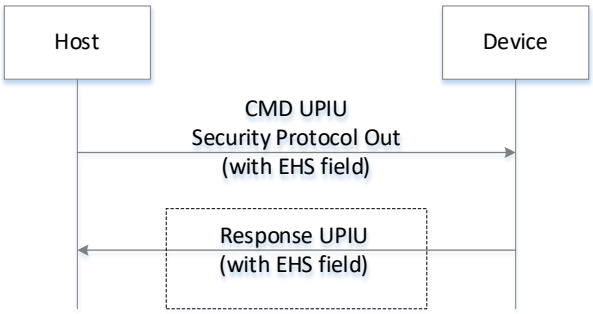
COMMAND UPIU							
0		1	2	3			
xx00 0001b		Flags	LUN	Task Tag			
4		5	6	7			
IID	Command Set Type	Reserved	Reserved	EXT_IID	Reserved		
8		9	10 (MSB)	11 (LSB)			
Total EHS length (02h)		Reserved	Data Segment Length (0000h)				
12 (MSB)		13	14	15 (LSB)			
Expected Data Transfer Length							
SECURITY PROTOCOL OUT CDB							

EHS field

0~3	EHS Header	
4~5	(Req./Resp. Message Type)	0008h
6~21	(Nonce)	Nonce from Host
22~25	(Write Counter)	Current counter Value
26~27	(Address / LUN)	0000h
28~29	(Block Count)	0000h
30~31	(Result)	0000h
32~63	(MAC / Key)	Mac from the host
Header E2ECRC (omit if HD=0)		

Figure 12.21 — RPMB Purge Enable Flow (in Advanced RPMB Mode)

12.4.7.4.7 RPMB Purge Enable Flow (cont'd)



EHS field(2)

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

EHS field(2)		
0~3	EHS Header	
4~5	(Req./Resp. Message Type)	0800h
6~21	(Nonce)	Copy of Nonce
22~25	(Write Counter)	New Counter Value(3)
26~27	(Address / LUN)	0000h
28~29	(Block Count)	0000h
30~31	(Result)	Result Code
32~63	(MAC / Key)	MAC from the device

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 and EHS field is not included
K=96 if HD=0 and EHS field is included
Note 2 EHS field shall not be sent when SPI/SPO command terminated with CHECK CONDITION status. In this case, the Total EHS length is 0h.
Note 3 The Write Counter is incremented only if the Enable operation was successfully completed

Figure 12.21 — RPMB Purge Enable Flow (in Advanced RPMB Mode) (cont'd)

12.4.7.4.8 RPMB Purge Status Read

RPMB Purge polling starts by issuing a SECURITY PROTOCOL IN command.

- An initiator sends a SECURITY PROTOCOL IN command with SECURITY PROTOCOL field set to ECh and indicating the RPMB region in the SECURITY PROTOCOL SPECIFIC field. The RPMB data frame contains the Request Message Type = 0009h (**RPMB Purge Status Read Request**) and Nonce.
- The device returns purge status by issuing a DATA IN UPIU with an RPMB Purge Response Packet for Advanced RPMB.
  - Supported purge status value are as follows:
    - 00 – RPMB Purge not initiated (reset value)
    - 01 – RPMB Purge in progress
    - 02 – RPMB Purge successfully completed
    - 03 – RPMB Purge general failure
- The device returns a RPMB data frame with Response Message Type = 0900h (**RPMB Purge Status Read Response**), a copy of the Nonce received in the request, the MAC and the Result.
- If a GOOD status is not returned, a general failure error should be reported.

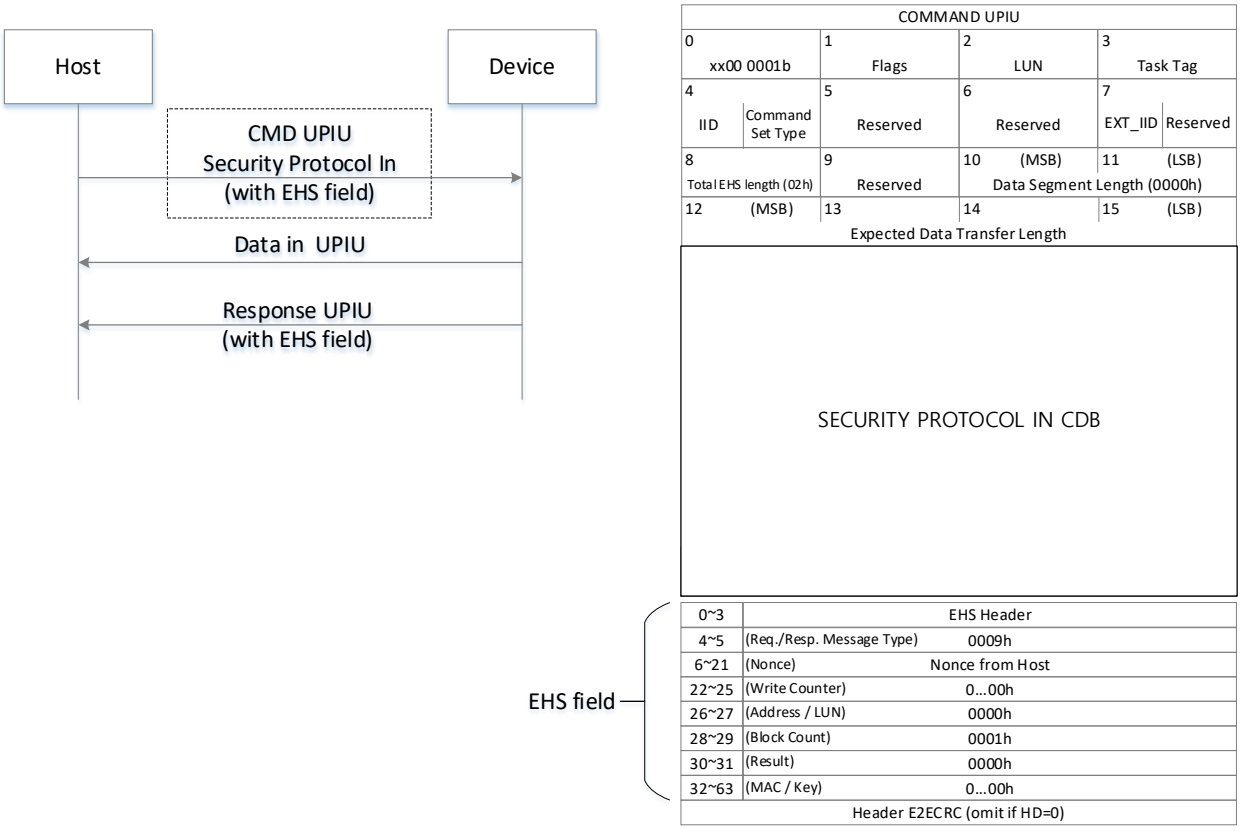
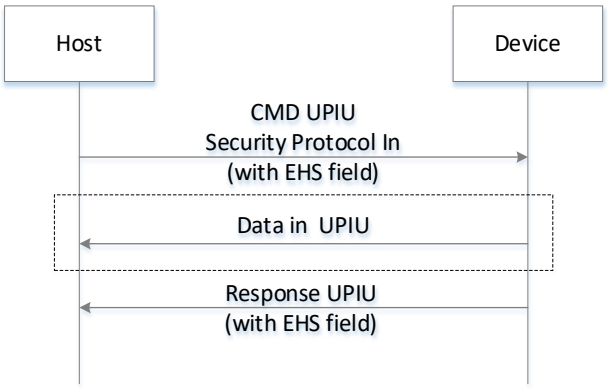


Figure 12.22 — RPMB Purge Status Read Flow (in Advanced RPMB Mode)

12.4.7.4.8 RPMB Purge Status Read (cont'd)

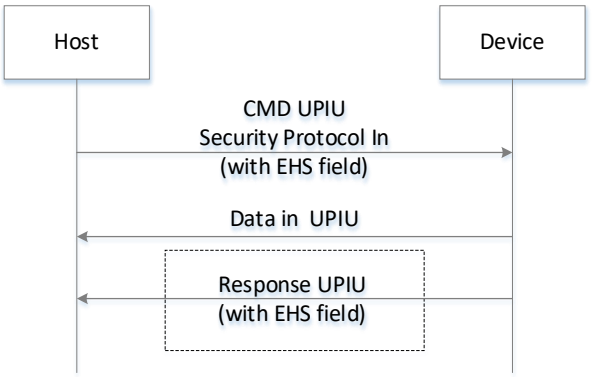


DATA IN UPIU			
0 xx10 0010b	1 Flags	2 LUN	3 Task Tag
4 IID / Reserved	5 EXT_IID / Reserved	6 Response	7 Reserved
8 Total EHS length (00h)	9 Reserved	10 (MSB) Data Segment Length	11 (LSB)
12 (MSB)	13 Data Buffer Offset	14	15 (LSB)
16	17 Data transfer Count	18	19
20	21 Reserved	22	23
24	25 Reserved	26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

Note 1 K=32 if HD=0

Figure 12.22 — RPMB Purge Status Read Flow (in Advanced RPMB Mode) (cont'd)

12.4.7.4.8 RPMB Purge Status Read (cont'd)



EHS field(2)

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

0~3	EHS Header	
4~5	(Req./Resp. Message Type)	0900h
6~21	(Nonce)	Copy of Nonce
22~25	(Write Counter)	0…00h
26~27	(Address / LUN)	0000h
28~29	(Block Count)	0001h
30~31	(Result)	Result Code
32~63	(MAC / Key)	MAC from the device

Header E2ECRC (omit if HD=0)			
K (MSB)	K+1 (LSB)	K+2	K+3
Sense Data Length		Sense Data[0]	Sense Data[1]
…	…	…	…
K+16	K+17	K+18	K+19
Sense Data[14]	Sense Data[15]	Sense Data[16]	Sense Data[17]
Data E2ECRC (omit if DD=0)			

Note 1 K=32 if HD=0 and EHS field is not included
K=96 if HD=0 and EHS field is included
Note 2 EHS field shall not be sent when SPI/SPO command terminated with CHECK CONDITION status. In this case, the Total EHS length is 0h.

Figure 12.22 — RPMB Purge Status Read Flow (in Advanced RPMB Mode) (cont'd)

12.4.7.4.9 Authenticated Vendor Specific Command Request

The Authenticated Vendor Specific Command operation is initiated by a SECURITY PROTOCOL OUT command. 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 includes the Request Message Type = 0010h (**Authenticated Vendor Specific Command Request**), Advanced RPMB Block Count, Write Counter, Nonce, and MAC.

- When the device receives the Request message (Authenticated Vendor Specific Command Request), 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) and the RPMB Purge is not initiated.
- If the Write Counter was not expired, then the device calculates the MAC of request and compares this with the MAC in the request. If the two MACs are different, then the device sets the Result to "Authentication failure" (0002h) and vendor specific command operation is not initiated.
- If the Write Counter and MAC comparisons are successful, then the Vendor Specific Command Request is considered to be authenticated and the vendor specific command operation is initiated.
- When the device is ready to start the vendor specific command, the device can return a GOOD status in response to the SECURITY PROTOCOL OUT command, regardless of whether the vendor specific command operation was successful or not. The success or failure of vendor specific command can be known by the operation of vendor specific command Status Read.
- On the successful completion of the Vendor Specific Command Request Message, the device shall increment the write counter.

If a GOOD status is not return, a general failure error should be reported.

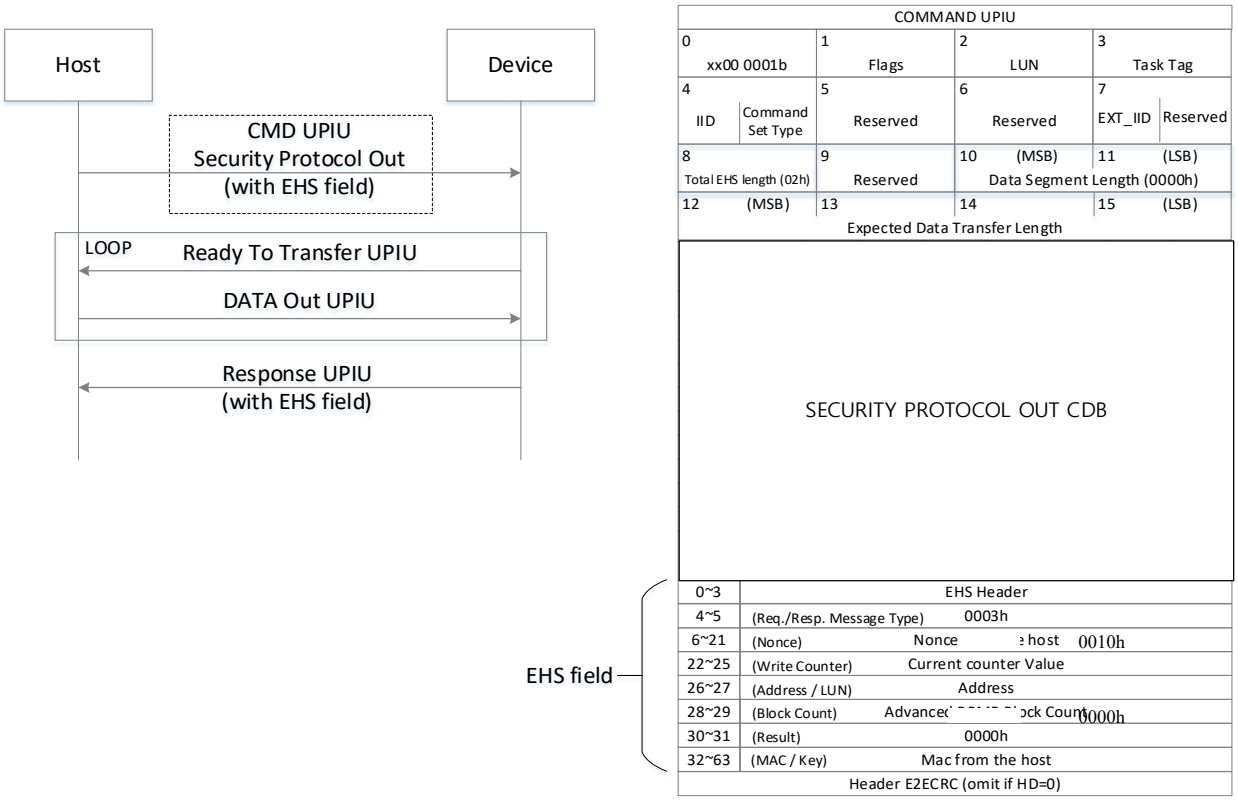
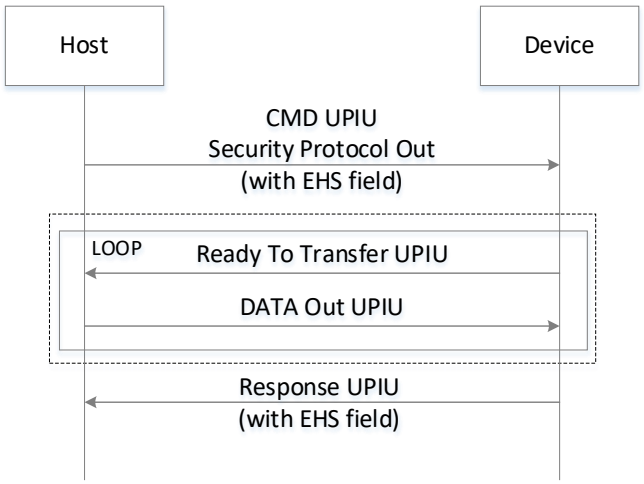


Figure 12.23 — Authenticated Vendor Specific Command Request Flow (in Advanced RPMB Mode)

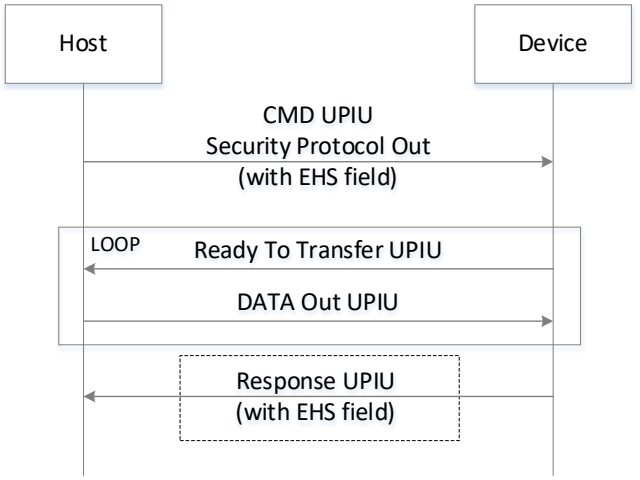
12.4.7.4.9 Authenticated Vendor Specific Command Request (cont'd)



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

Figure 12.23 — Authenticated Vendor Specific Command Request Flow (in Advanced RPMB Mode) (cont'd)

12.4.7.4.9 Authenticated Vendor Specific Command Request (cont'd)



EHS field[2]

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

0~3	EHS Header	
4~5	(Req./Resp. Message Type)	1000h
6~21	(Nonce)	Copy of Nonce
22~25	(Write Counter)	New Counter Value[3]
26~27	(Address / LUN)	0000h
28~29	(Block Count)	0000h
30~31	(Result)	Result Code
32~63	(MAC / Key)	MAC from the device

Header E2ECRC (omit if HD=0)			
K (MSB)	K+1 (LSB)	K+2	K+3
Sense Data Length		Sense Data[0]	Sense Data[1]
...	...	...	...
K+16	K+17	K+18	K+19
Sense Data[14]	Sense Data[15]	Sense Data[16]	Sense Data[17]
Data E2ECRC (omit if DD=0)			

Note 1 K=32 if HD=0 and EHS field is not included
K=96 if HD=0 and EHS field is included
Note 2 EHS field shall not be sent when SPI/SPO command terminated with CHECK CONDITION status. In this case, the Total EHS length is 0h.

Figure 12.23 — Authenticated Vendor Specific Command Request Flow (in Advanced RPMB Mode) (cont'd)

12.4.7.4.10 Authenticated Vendor Specific Command Status Read Request

Authenticated Vendor Specific Command Status Read sequence starts by issuing a SECURITY PROTOCOL IN command.

An initiator sends a SECURITY PROTOCOL IN command with SECURITY PROTOCOL field set to ECh and indicating the RPMB region in the SECURITY PROTOCOL SPECIFIC field. The RPMB message contains the Request Message Type = 0011h (**Authenticated Vendor Specific Command Status Read Request**), Nonce and the Advanced RPMB Block Count.

The device returns vendor specific command status by issuing one or more DATA IN UPIUs for Advanced RPMB.

The device returns a RPMB data frame with Response Message Type = 1100h (**Authenticated Vendor Specific Command Status Response**), a copy of the Nonce received in the request, the MAC and the Result.

- Supported vendor command status value are as follows, reported in the Authenticated Vendor Specific Command Status Response UPIUs Result field:
  - 00 – Vendor Specific Command not initiated (reset value)
  - 01 – Vendor Specific Command in progress
  - 02 – Vendor Specific Command completed
  - 03 – Vendor Specific Command general failure

If a GOOD status is not returned, a general failure error should be reported.

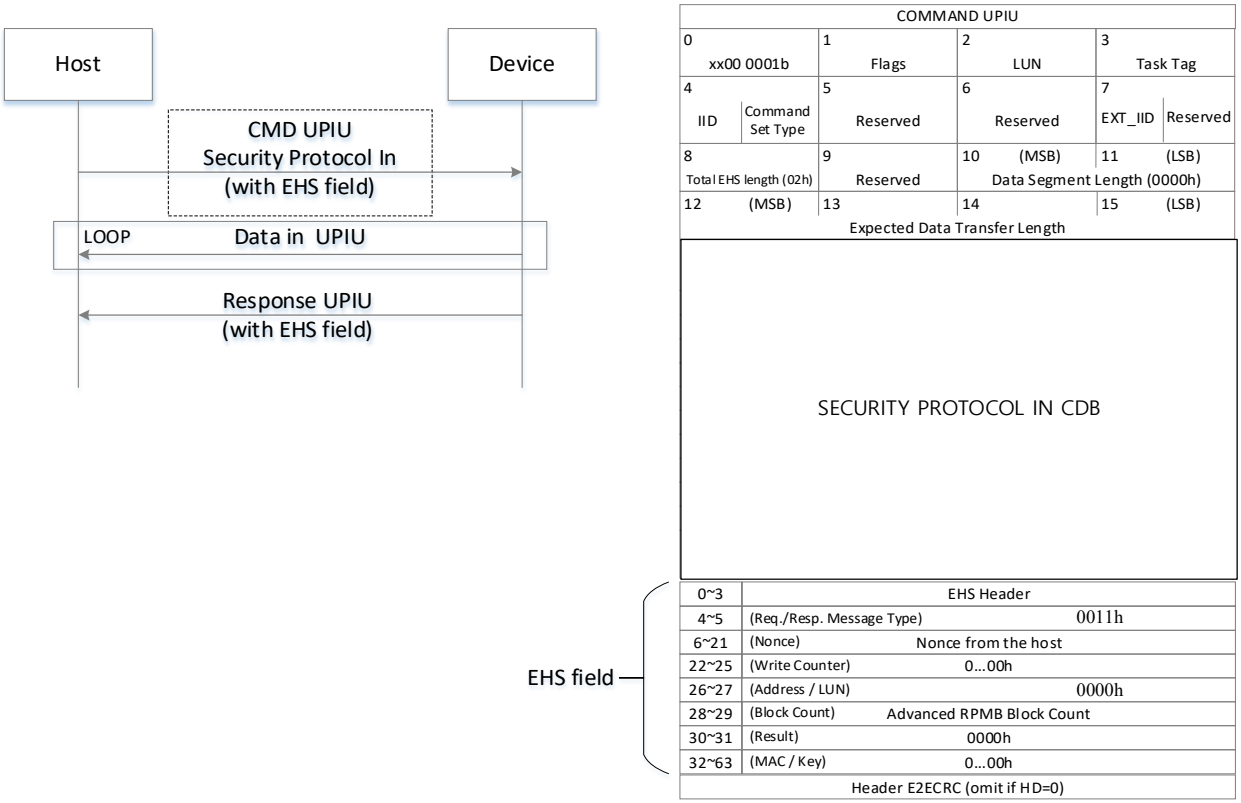
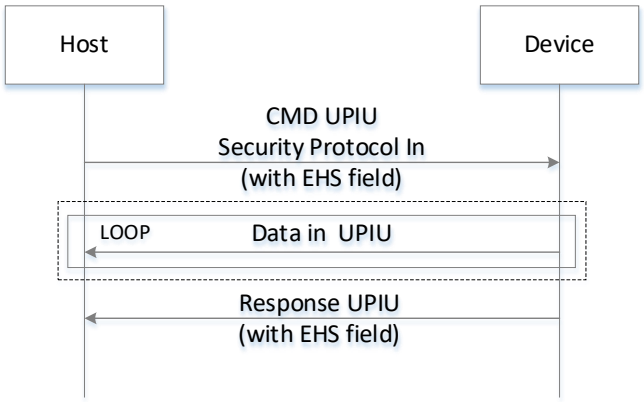


Figure 12.24 — Authenticated Vendor Specific Command Status Read Request Flow (in Advanced RPMB Mode)

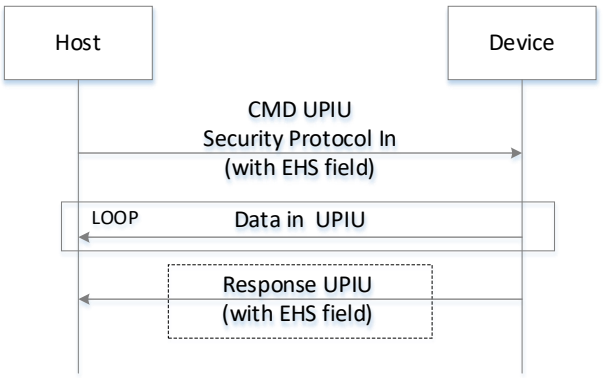
12.4.7.4.10 Authenticated Vendor Specific Command Status Read Request (cont'd)



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

Figure 12.24 — Authenticated Vendor Specific Command Status Read Request Flow (in Advanced RPMB Mode) (cont'd)

12.4.7.4.10 Authenticated Vendor Specific Command Status Read Request (cont'd)



EHS field[2]

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

0~3	EHS Header	
4~5	(Req./Resp. Message Type)	1100h
6~21	(Nonce)	Copy of the Nonce
22~25	(Write Counter)	0...00h
26~27	(Address / LUN)	0000h
28~29	(Block Count)	Advanced RPMB Block Count
30~31	(Result)	Status
32~63	(MAC / Key)	MAC from the device

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 and EHS field is not included
K=96 if HD=0 and EHS field is included
Note 2 EHS field shall not be sent when SPI/SPO command terminated with CHECK CONDITION status. In this case, the Total EHS length is 0h.

Figure 12.24 — Authenticated Vendor Specific Command Status Read Request Flow (in Advanced RPMB Mode) (cont'd)

12.5 Malware Protection

The UFS device will also have the option to protect boot, bus configuration settings and other important device configuration settings so that once they are set they cannot be modified. The implementation of the protection of these parameters is defined within the spec where the parameter is defined.

13 UFS Functional Descriptions

13.1 UFS Boot

13.1.1 Introduction

Some computing systems can have the need to download the system boot loader from an external non-volatile source. This task can be accomplished through an internal boot ROM contained in the host SOC whose code when executed determines a minimal initialization of the system to start the boot code transfer. Several features of the boot functionality can be configured in order to be adapted to different system requirements.

Moreover specific features to ensure boot data integrity and no corruption of boot code are defined.

13.1.2 Boot Configuration

During boot operation the UFS host controller retrieves the system boot code stored in the UFS device. In this version of the standard, the boot mechanism is defined for a point-to-point topology (see Figure 13.1).

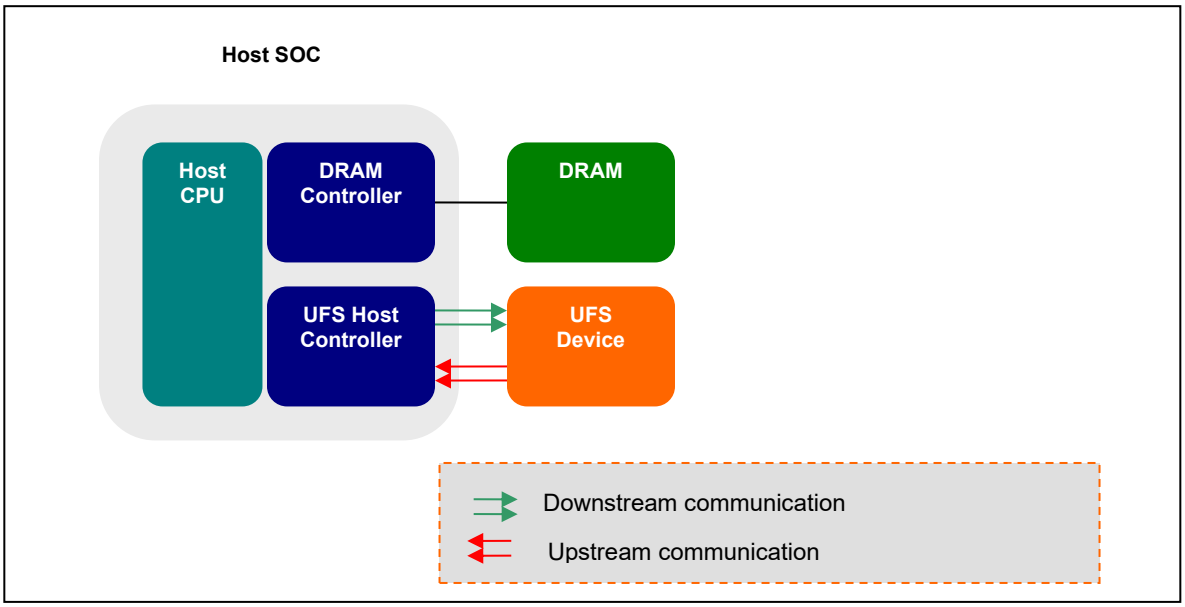


Figure 13.1 — UFS System Diagram

Two logical units (Boot LU A, Boot LU B) can be used to store the boot code, but only one of them will be active during the boot process. Any logical unit can be configured as “Boot LU A” or “Boot LU B”. No more than one logical unit may be configured as “Boot LU A”, no more than one logical unit may be configured as “Boot LU B”. The logical unit active during boot is mapped onto the Boot well known logical unit (W-LUN = 30h) for read access. In this way, when the host updates the boot code, a fix logical unit number is kept when the active logical unit is swapped from A to B or vice versa.

13.1.2 Boot Configuration (cont'd)

Several configurable fields of the Device Descriptor and the Unit Descriptors determine the device behavior during boot. Device Descriptor and Unit Descriptors are configured by writing the Configuration Descriptor.

For a UFS bootable device, the boot feature is enabled if bBootEnable field in the Device Descriptor is set to 01h or 02h.

The characteristics of the logical units used during boot are configured setting the corresponding fields of the Configuration Descriptor. (see 14.1.5.3)

The number of allocation units (dNumAllocUnits) field configures the logical unit size, and the boot logical unit ID (bBootLunID) field allows to designate the logical unit as being "Boot LU A" or "Boot LU B".

The logical unit active during the boot shall be configured by writing the bBootLunEn attribute, as described in Table 13.1.

Table 13.1 — bBootLunEn Attribute

bBootLunEn	Description
00h	Boot LU A = disabled Boot LU B = disabled
01h	Boot LU A = enabled Boot LU B = disable
02h	Boot LU A = disable Boot LU B = enabled
Others	Reserved

The host should not attempt to set bBootLunEn to 'Reserved' values, and UFS device shall generate an error in case of an attempt to set 'Reserved' values and not execute the request.

When bBootLunEn attribute is 00h the boot feature is disabled, the device behaves as if bBootEnable would be equal to zero.

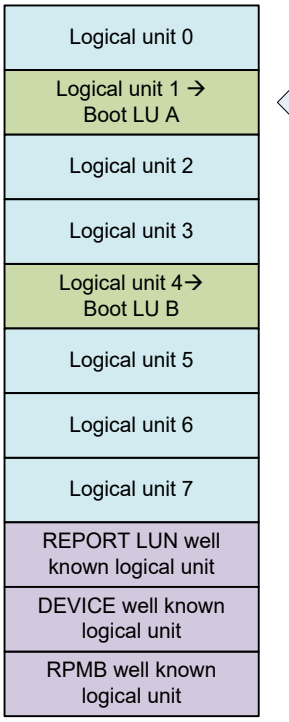
The boot feature is critical for the initialization of the platform and it is important that the boot feature will not be disabled unintentionaly.

When bBootEnable is provisioned to 02h, the device is in permanent-bootable configuration in which a Query Request to change bBootLunEn value from 01h or 02h to 00h shall fail and and the Query Response field shall be set to "General failure (FFh). In this configuration, the device still enable transitions of bBootLunEn between 01h and 02h, and vice versa.

The active boot logical unit will be mapped onto the Boot well known boot logical unit (W-LUN = 30h) once the bBootLunEn has been properly configured.

Figure 13.2 shows an example of a UFS device having eight logical units: LU 1 and LU 4 are configured, respectively, as "Boot LU A" and "Boot LU B". In particular, LU 1 is the active one (bBootLunEn = 01h).

13.1.2 Boot Configuration (cont'd)



bBootLunID in Unit Descriptors		
LU	Value	Description
0	00h	Disabled
1	01h	Boot LU A
2	00h	Disabled
3	00h	Disabled
4	02h	Boot LU B
5	00h	Disabled
6	00h	Disabled
7	00h	Disabled

bBootLunEn = 01h. "Boot LU A" is the active logical unit for boot. Therefore, the logical unit 1 is mapped onto the Boot well known logical unit (W-LUN=30h, LUN field in UPIU = B0h).

Figure 13.2 — Example of UFS Device Memory Organization for Boot

13.1.3 Initialization and Boot Code Download Process

The initialization and boot code download process is made up of the following phases: partial initialization, boot transfer and initialization completion.

13.1.3.1 Partial Initialization

The partial initialization phase starts after power on, or hardware reset, or EndPointReset and involves the entire UFS stack. At the end of this phase, the UniPro boot sequence shall be completed, and the UTP layer shall be capable of accessing Device Descriptor (if the bDescrAccessEn field of the Device Descriptor is '01h') and exchanging UPIU for READ command and TEST UNIT READY command. If the bDescrAccessEn field is '00h' descriptors will be accessible only after the initialization completion phase.

Each single layer in the UFS protocol stack executes the initialization process on both UFS host and UFS device sides.

a) Physical Layer (M-PHY)

After reset events, the physical layer will move from DISABLED state to HIBERN8 state.

b) Link Layer (UniPro)

On host and device side UniPro boot sequence takes place:

1) The UniPro stack is reset using the DME_RESET.req primitive.
2) Wait until the reset completion is indicated by the DME_RESET.cnf_L primitive.
3) The UniPro stack is enabled using the DME_ENABLE.req primitive.
4) Wait until the enable completion is indicated by the DME_ENABLE.cnf_L primitive.
5) The UniPro Link StartUp sequence is initiated using the DME_LINKSTARTUP.req primitive. The UniPro Link Startup consists of a series of multiphase handshakes to establish initial link communication in both directions between UFS host and device.
6) Wait until the link startup completion is indicated by the DME_LINKSTARTUP.cnf_L primitive.

c) UFS Transport Layer (UTP)

At the end of the UFS Interconnect Layer initialization on both host and device side, the host is expected to send a NOP OUT UPIU to verify that the device UTP Layer is ready.

For some implementations, the device UTP layer may not be initialized yet, therefore the device may not respond promptly to NOP OUT UPIU sending NOP IN UPIU.

The host waits until it receives the NOP IN UPIU from the device. When the NOP IN UPIU is received, the host is acknowledged that the UTP layer on the device is ready to execute UTP transactions.

d) Link Configuration

The host may configure the Link Attributes (i.e., Gear, HS Series, PWM Mode in Rx and Tx) by using DME primitives at UniPro level.

13.1.3.1 Partial Initialization (cont'd)

e) Device Descriptor Reading

The UFS host may optionally discover relevant device info for the boot process by accessing the Device Descriptor (i.e., Device Class/Subclass, Boot Enable, Boot LUs size, etc.). The UFS host is allowed to access the Device Descriptor only if the bDescrAccessEn is '01h', otherwise this descriptor can be accessed only after the device has fully completed its initialization.

13.1.3.2 Boot Transfer

The following steps can be executed only if bBootEnable field is set.

Boot code download

At first, the UFS host issues a TEST UNIT READY command to the Boot well known logical unit to verify if the latter can be accessed. If the command succeeds, the UFS host reads the Boot well known logical unit by issuing SCSI READ commands and the UFS device will start to send the boot code on the Upstream Link. During this phase, only the Boot well known logical unit is accessible: this logical unit shall accept read commands, while other logical units may not be ready.

13.1.3.3 Initialization Completion

After the host has completed the boot code download from the Boot well known logical unit, the initialization process proceeds as described in the following. The host sets the fDeviceInit flag to "01h" to communicate to the UFS device that it can complete its initialization. The device shall reset the fDeviceInit flag when the initialization is complete. The host polls the fDeviceInit flag to check the completion of the process. When the fDeviceInit is reset, the device is ready to accept any command.

13.1.3.3 Initialization Completion (cont'd)

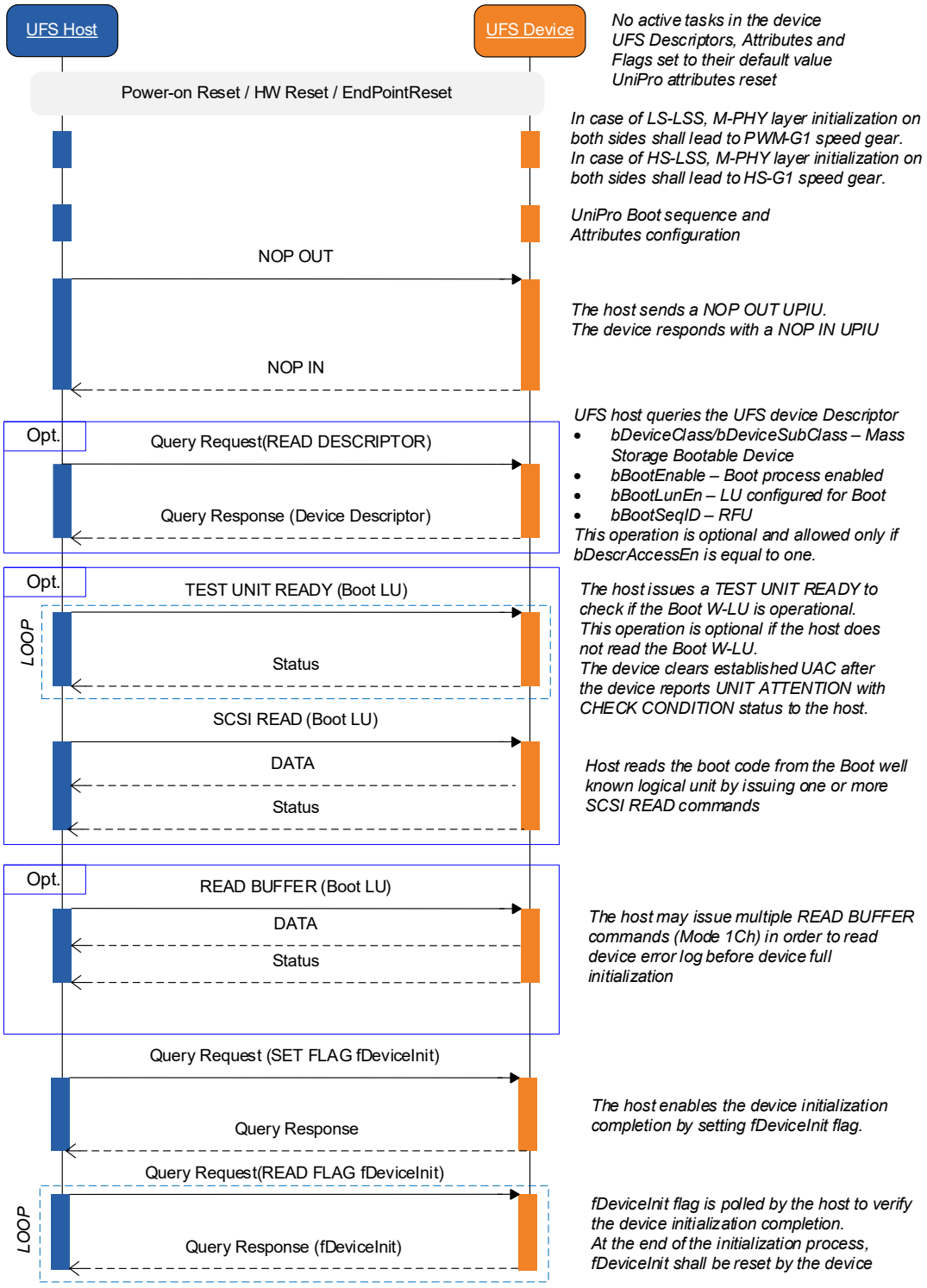


Figure 13.3 — Device Initialization and Boot Procedure Sequence Diagram

13.1.3.3 Initialization Completion (cont'd)

Table 13.2 — Valid UPIUs and SCSI Commands for Each Initialization Phase

Phase	Event	Valid UPIU	Valid SCSI command
Before Initialization	Power On Reset / HW Reset / EndPointReset	None	None
UIC Layer Initialization Phase	M-PHY layer initialization UniPro Boot sequence and UIC Layer Attributes configuration	None	None
UTP Layer Initialization Phase	Receive a single NOP OUT UPIU Send NOP IN UPIU for response to NOP OUT UPIU	NOP OUT UPIU NOP IN UPIU	None
Boot W-LU Ready Phase (Optional)	Read Device Descriptor (Optional) (1)	QUERY REQUEST UPIU (READ DESCRIPTOR Device Descriptor)	None
	Boot Transfer (Optional) (2)	COMMAND UPIU for Boot W-LU	INQUIRY, REQUEST SENSE, TEST UNIT READY, READ (6), READ (10), READ (16) (3), READ BUFFER
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' or '02h', at least one logical unit is configured for boot (bBootLunEn) and bBootLunID selects the desired boot logical unit.
NOTE3 READ (16) command support is optional.

13.1.4 Initialization Process without Boot Code Download

If the boot process is not enabled on the UFS device, or it is not supported by the device class, or the host does not need to transfer the boot code, the host executes the initialization process as described in 13.1.3, omitting the boot transfer phase.

13.1.5 Boot Logical Unit Operations

The Boot well known logical unit is read only, therefore the boot code can be stored only writing the boot logical units (A or B).

Boot logical units are written to store the boot code during the system manufacturing phase and they may be also updated during the system lifecycle. These logical units can be read to verify their content.

13.1.5 Boot Logical Unit Operations (cont'd)

Therefore the following operations are permitted on the Boot logical units:

1. boot code write – for boot code upload/update
2. boot code read – to verify the content programmed
3. boot code removal – to remove the content of the Boot logical unit

These operations can be executed regardless the bBootEnable field value in the Device Descriptor.

Boot logical units (A or B) can be write protected using the methods described in 12.3, Device Data Protection.

13.1.6 Configurability

The boot process is configurable through several parameters in the Configuration Descriptor (see 14.1.5.3) to adapt it to different usage models and system features.

The following parameters refer to boot capabilities.

- Device Descriptor parameters:
  - bBootEnable (Boot Enable)
  - bDescrAccessEn (Descriptor Access Enable)
  - bInitPowerMode (Initial Power Mode)
  - bInitActiveICCLevel (Initial Active ICC Level)
- Unit Descriptor parameters for Boot LU A and Boot LU B:
  - bLUEnable (Logical Unit Enable)
  - bBootLunID (Boot LUN ID)
  - bLUWriteProtect (Logical Unit Write Protect)
  - bMemoryType (Memory Type)
  - dNumAllocUnits (Number of Allocation Units)
  - bDataReliability (Data Reliability)
  - bLogicalBlockSize (Logical Block Size)
  - bProvisioningType (Provisioning Type)

NOTE These parameters are non volatile and they may be programmed during the system manufacturing phase.

In addition to the parameters mentioned, the following attributes are relevant for device initialization and boot

- bBootLunEn (Boot LUN Enable)
- bRefClkFreq (Reference Clock Frequency value)

13.1.7 Security

13.1.7.1 Boot Area Protection

Boot areas might be protected in order to avoid boot code alteration by a third party: the write protection mechanism for the boot logical units can be defined configuring the corresponding bLUWriteProtect parameter of the Unit Descriptor.

In particular, the boot logical units may be permanently write protected or power-on write protected. In case of power-on write protection, the boot logical units can be written only when the fPowerOnWPEn flag is equal to zero.

13.2 Logical Unit Management

13.2.1 Introduction

The functionality aims to provide a mechanism to let an external application define and use a virtual memory organization which could easily fit different usage models in a versatile way.

Besides segmenting the available addressable space, the mechanism introduces the possibility of differentiating each logical unit through dedicated functionalities and features.

This sub-clause describes the procedure to configure the UFS device in terms of: number of logical units, logical unit size, logical unit memory type, etc. Security features can be configured as described in 12, UFS Security.

A UFS device can be organized in different logical units. Each one represents an autonomous computing entity with independent logical address ranges and singularly accessible.

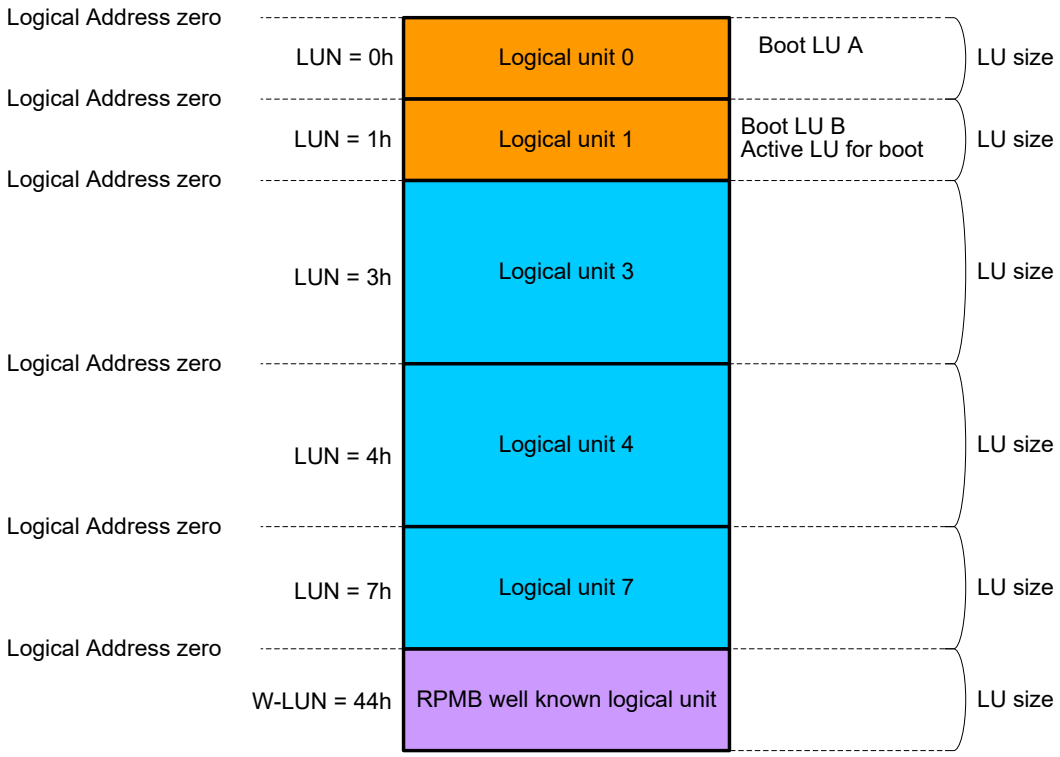
Moreover, each logical unit can be defined for a specified use and with peculiar attributes (i.e., memory type) in order to be adapted to different UFS host usage models and operating systems requirements.

13.2.2 Logical Unit Features

UFS device address space is organized in several memory areas configurable by the user. In particular, such memory areas are denoted as logical units and characterized by the fact that they have independent logical addressable spaces starting from the logical address zero.

In addition to the logical units, the UFS device supports the following well known logical units for specific purposes: UFS Device, REPORT LUNS, Boot and RPMB. Logical units are addressed by the LUN (logical unit number), while well known logical unit are addressed by the W-LUN (well known logical unit number).

13.2.2 Logical Unit Features (cont'd)



NOTE 1 The figure shows an example of device configuration in which LU 0 and LU 1 are used as boot logical units, and the logical units 3, 4 and 7 for code and data storage. LU 1 is the boot active logical unit and it may be accessed in read using the W-LUN = 30h (LUN field in UPIU = B0h).

Figure 13.4 — Example of UFS Device Memory Organization

Each logical unit will have a physical implementation on the non-volatile storage media

In particular, the UFS device shall support:

- The number of logical units specified by bMaxNumberLU. Each of them configurable as boot logical units with a maximum of two.
- One RPMB well known logical unit (W-LUN = 44h, LUN field in UPIU = C4h). RPMB well known logical unit may be further configured into up to four separate RPMB regions (RPMB region 0 - RPMB region 3).

Two logical units can be configured as boot logical unit, with only one of them active and readable through the Boot well known logical unit (W-LUN = 30h) for the execution of the system boot (see 13.2). The RPMB well known logical unit is accessed by authenticated operations by a well defined security algorithm (see 12.4). The other logical units will be used to fulfill other use cases.

13.2.2 Logical Unit Features (cont'd)

Common features of each logical unit are:

- Independent logical addressable spaces (starting from logical address zero up to the logical unit size).
- Configurable logical unit size.

The size of each logical unit is determined by the number of allocation units assigned to it: dNumAllocUnits parameter value of the Configuration Descriptor. The dNumAllocUnits is expressed in terms of allocation unit size (bAllocationUnitSize).

Moreover each logical unit is characterized by the memory type parameter which can be configured. Examples of memory types to differentiate logical unit properties are the following ones:

- Default type – regular memory characteristic.
- System Code type – a logical unit that is rarely updated (e.g., system files or binary code executable files or the host operating system image and other system data structures).
- Non-Persistent type – a logical unit that is used for temporary information (e.g., swap file extend the host virtual memory space).
- Enhanced Memory type – vendor specific attribute.

The definition of the Enhanced Memory type is left open in order to accomplish different needs and vendor specific implementations.

Mechanisms of write protection can be configured for each logical unit. Write protection feature types are:

- Permanent write protection (permanent read only).
- Power on write protection (write protection can be cleared with a power cycle or a hardware reset event).

Write protection is not available in the RPMB well known logical unit.

13.2.3 Logical Unit Configuration

The user shall configure the logical units of the UFS device according to the following rules:

- Maximum number of logical units is specified by bMaxNumberLU.
- One or two logical units can be configured as boot logical units.

When a UFS device is shipped only the following well known logical units will be available: UFS Device, REPORT LUNS and the RPMB. All other logical units shall be configured before they can be accessed.

The RPMB well known logical unit shall be configured by the device manufacturer before shipping the device. Only RPMB region 0 shall be enabled and bRPMBRegion0Size shall be set to the same size as the total RPMB well known logical unit size before shipping the device.

Logical units and RPMB regions may be configured writing the Configuration Descriptors. (see 14.1.3)

RPMB regions may be configured multiple times until the Configuration Descriptor is locked by setting bConfigDescrLock attribute value to 01h. RPMB key may be programmed any time when the target RPMB region is enabled, and may be programmed independent to whether the value of bConfigDescrLock attribute is set or not. When RPMB regions are re-configured with different configuration setting, the data which was previously written to any RPMB region shall be erased while RPMB key and RPMB write counter are maintained. To keep the compatibility to the previous version of the standard, RPMB region 0 is always enabled independent of configuration value of bRPMBRegionEnable.

The configuration of each logical unit and the configuration of RPMB regions may be retrieved by reading the corresponding Unit Descriptor.

It is recommended to execute logical unit configuration and RPMB region configuration during the system manufacturing phase.

For a zoned logical unit (see 13.4.23, Zoned Logical Units) the size of a zoned logical unit shall be a multiple of the zone size. A UFS device shall reject attempts to configure a zoned logical unit with a size that is not a multiple of the zone size.

13.2.3 Logical Unit Configuration (cont'd)

Table 13.3 summarizes the configurable parameters per logical unit. See 14.1.5, Descriptor Definitions, for details about these parameters.

Table 13.3 — Logical Unit Configurable Parameters

Configurable parameters		Logical Unit
Name	**Description**	
bLUEnable	Logical Unit Enable	LU 0, …, Maximum LU specified by bMaxNumberLU
bBootLunID	Boot LUN ID	LU 0, …, Maximum LU specified by bMaxNumberLU
bLUWriteProtect	Logical Unit Write Protect	LU 0, …, Maximum LU specified by bMaxNumberLU
bMemoryType	Memory Type	LU 0, …, Maximum LU specified by bMaxNumberLU
dNumAllocUnits	Number of allocation units assigned to the logical unit. The value shall be calculated considering the capacity adjustment factor of the selected memory type.	LU 0, …, Maximum LU specified by bMaxNumberLU
bDataReliability	Data Reliability	LU 0, …, Maximum LU specified by bMaxNumberLU
bLogicalBlockSize	Logical Block Size	LU 0, …, Maximum LU specified by bMaxNumberLU
bProvisioningType	Provisioning Type	LU 0, …, Maximum LU specified by bMaxNumberLU
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

13.2.3 Logical Unit Configuration (cont'd)

The following Geometry Descriptor parameters provide relevant information for configuring the logical units:

- qTotalRawDeviceCapacity (total raw device density in unit of 512 bytes)
- dSegmentSize
- bAllocationUnitSize (Allocation Unit Size, value expressed in number of segments)
- wSupportedMemoryTypes
- Maximum number of allocation unit for each memory type (dSystemCodeMaxNAllocU, dNonPersistMaxNAllocU, etc.)
- Capacity Adjustment Factor for each memory type (wSystemCodeCapAdjFac, wNonPersistCapAdjFac, etc.)
- bMinAddrBlockSize (this parameter indicates a value equal or greater than 4 Kbyte)
- bOptimalReadBlockSize and bOptimalWriteBlockSize
- bMaxInBufferSize
- bMaxOutBufferSize

To enable the access to a logical unit, the user shall configure Unit Descriptor parameters as described in the following:

- bLUEnable
  bLUEnable shall be set to 01h to enable the logical unit. If bLUEnable is equal to 00h the logical unit is disabled and all Unit Descriptor parameters are don't care.
- bMemoryType
  bMemoryType shall be set to value corresponding to the desired memory type. The wSupportedMemoryTypes parameter in the Geometry Descriptor indicates which memory types are supported by the device.
- bLogicalBlockSize
  bLogicalBlockSize value shall adhere to the following rules:
  - $2^{bLogicalBlockSize} \geq bMinAddrBlockSize \times 512$,
  - $2^{bLogicalBlockSize} \leq bMaxInBufferSize \times 512$,
  - $2^{bLogicalBlockSize} \leq bMaxOutBufferSize \times 512$.

13.2.3 Logical Unit Configuration (cont'd)

To optimize the device performance, it is recommended to configure the logical block size (bLogicalBlockSize) to represent the value indicated by dOptimalLogicalBlockSize for the specific logical unit memory type.

Supported bLogicalBlockSize values are device specific, refer to the vendor datasheet for further information.

- dNumAllocUnits
  dNumAllocUnits determines the size of the logical unit. If LUCapacity is the desired logical unit size expressed in bytes, the dNumAllocUnits value shall be calculated using the following equation:
  If (bCapAdjFacRepresentation = = 0h) then

$$\text{dNumAllocUnits} = \text{CEILING}\left(\frac{\text{LUCapacity} \times \text{CapacityAdjFactor}}{\text{bAllocationUnitSize} \times \text{dSegmentSize} \times 512}\right)$$

  Else

$$\text{dNumAllocUnits} = \text{CEILING}\left(\frac{\text{LUCapacity} \times \text{MSB(CapacityAdjFactor)}}{\text{bAllocationUnitSize} \times \text{dSegmentSize} \times 512 \text{ x LSB(CapacityAdjFactor)}}\right)$$

  where:
  - CapacityAdjFactor = Capacity Adjustment Factor of the particular memory type
    The Capacity Adjustment Factor value for Normal memory type is one.
    The following example shows dNumAllocUnits calculation for two logical units (LU 1 and LU 4) with the characteristics:
  - LU 1: 12 Gbyte, Normal memory type
  - LU 4: 32Mbyte, Enhanced memory type 1

  Assuming that the medium of the UFS device is composed by NAND flash memories which support 2 bit-per-cell and 1 bit-per-cell operation modes. The 2 bit-per-cell operation mode may be associated with the Normal memory type, while the 1 bit-per-cell operation mode may be associated with the Enhanced memory type 1.

  The Capacity Adjustment Factor for the Enhanced memory type 1 will be equal to 2.

  If:
  - dSegmentSize = 1024
  - bAllocationUnitSize = 8

  Then dNumAllocUnits for LU 1 and LU 4 are:

$$\text{dNumAllocUnits LU 1} = \text{CEILING}\left(\frac{12 \text{ Gbyte} \times 1}{8 \times 1024 \times 512 \text{ byte}}\right) = \text{CEILING}\left(\frac{12 \text{ Gbyte}}{4 \text{ Mbyte}}\right) = 3072$$

$$\text{dNumAllocUnits LU 4} = \text{CEILING}\left(\frac{32 \text{ Mbyte} \times 2}{8 \times 1024 \times 512 \text{ byte}}\right) = \text{CEILING}\left(\frac{64\text{Mbyte}}{4 \text{ Mbyte}}\right) = 16$$

  The logical unit capacity can be retrieved by either reading the qLogicalBlockCount parameter in the Unit Descriptor or issuing the READ CAPACITY command.

13.2.3 Logical Unit Configuration (cont'd)

In particular, the relations between the parameters returned by READ CAPACITY (RETURNED LOGICAL BLOCK ADDRESS and LOGICAL BLOCK LENGTH IN BYTES), and bLogicalBlockSize and qLogicalBlockCount parameters in Unit Descriptors are:

- RETURNED LOGICAL BLOCK ADDRESS = qLogicalBlockCount – 1,
- LOGICAL BLOCK LENGTH IN BYTES = $2^{bLogicalBlockSize}$

- bBootLunID
  bBootLunID shall be set as described in the following:
  00h: if the logical unit is not a boot logical unit,
  01h: to configure the logical unit as "Boot LU A",
  02h: to configure the logical unit as "Boot LU B",

  NOTE The 01h value and 02h value shall be assigned to no more than one logical unit.

- bLUWriteProtect
  bLUWriteProtect shall be set as described in the following:
  00h: if the logical unit is not write protected,
  01h: to configure power on write protection,
  02h: to configure permanent write protection.

- bDataReliability
  bDataReliability shall be set to configure the device behavior when a power failure occurs during a write operation to the logical unit:
  00h: logical unit is not protected. Logical unit's entire data may be lost as a result of a power failure during a write operation,
  01h: logical unit is protected. Logical unit's data is protected against power failure.

- bProvisioningType
  bProvisioningType shall be set to configure the logical unit provisioning type:
  00h: to disable thin provisioning,
  02h: to enable thin provisioning with TPRZ = 0,
  03h: to enable thin provisioning with TPRZ = 1.

RPMB well known logical unit may be divided in multiple RPMB regions configuring the parameters described in the following.

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.

13.2.3 Logical Unit Configuration (cont'd)

- bRPMBRegionEnable
  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.
  - Bit-0: Don't care. RPMB region 0 is always enabled independent of this bit value.
  - Bit-1: Set to 1 to enable RPMB region 1
  - Bit-2: Set to 1 to enable RPMB region 2
  - Bit-3: Set to 1 to enable RPMB region 3
  - Bit-4: Set to 1 to enable Advanced RPMB Mode. (Set to 0 to enable Normal RPMB Mode.)
  - Bit-5: Set to 1 to enable RPMB Purge Operation
  - Bit-6 to Bit-7: Reserved
- bRPMBRegion0Size
  bRPMBRegion0Size configures the size of RPMB region 0 in 128 KB unit (00h: 0 KB, 01h: 128 KB, … , 80h: 16384 KB). The size is not directly configurable, instead it is determined by the following formula.

  In Normal RPMB mode:
  bRPMBRegion0Size = qLogicalBlockCount / 512 - bRPMBRegion1Size (if enabled) – bRPMBRegion2Size (if enabled) – bRPMBRegion3Size (if enabled)

  In Advanced RPMB mode:
  bRPMBRegion0Size = qLogicalBlockCount / 32 - bRPMBRegion1Size (if enabled) – bRPMBRegion2Size (if enabled) – bRPMBRegion3Size (if enabled)
  qLogicalBlockCount's value shall be changed appropriately based on RPMB configuration.
- bRPMBRegion1Size
  bRPMBRegion1Size configures the size of RPMB region 1 in 128 KB unit (00h: 0 KB, 01h: 128 KB, … , 80h: 16384 KB) if RPMB region 1 is enabled.
- bRPMBRegion2Size
  bRPMBRegion2Size configures the size of RPMB region 2 in 128 KB unit (00h: 0 KB, 01h: 128 KB, … , 80h: 16384 KB) if RPMB region 2 is enabled.
- bRPMBRegion3Size
  bRPMBRegion3Size configures the size of RPMB region 3 in 128 KB unit (00h: 0 KB, 01h: 128 KB, … , 80h: 16384 KB) if RPMB region 3 is enabled.

13.3 Logical Block Provisioning

13.3.1 Overview

Logical Block Provisioning is the concept that describes the relationship between the logical block address space and the physical memory resources that supports the logical address space.

A logical unit is either a conventional logical unit or a zoned logical unit (see [ZBC]). A conventional logical unit in a UFS device shall be either a Full Provisioned LU or a Thin Provisioned LU.

13.3.2 Full Provisioning

Every LBA in a fully provisioned logical unit is mapped. UFSs “fully provisioned” is equivalent to “resource provisioned” as defined in [SBC].

A logical unit that is fully provisioned shall provide enough LBA mapping resources to contain all logical blocks for the logical unit’s capacity as reported by the device server in response to a READ CAPACITY command.

The device server shall not cause any LBA on a fully provisioned logical unit to become unmapped.

A fully provisioned logical unit does not support logical block provisioning management – i.e., does not support UNMAP command.

13.3.3 Thin Provisioning

In thin provisioning there is no requirement that the available physical memory resources match the size of the logical address space.

A thin provisioned logical unit is not required to provide LBA mapping resources sufficient to contain all logical blocks for the logical unit’s capacity as reported by the device server in response to a READ CAPACITY command.

In UFS device, a thin provisioned logical unit shall have sufficient physical memory resources for all addressable logical blocks when the logical unit is configured by writing the Configuration Descriptor: the number of LBAs reported in READ CAPACITY shall not exceed the number of physical memory blocks available.

Logical address to physical resource allocation is managed by Logical Block Provisioning Management. Every LBA in a thin provisioned logical unit shall be either mapped or deallocated.

In UFS device, a thin provisioned logical unit shall support the Mapped and Deallocated states in the Logical Block Provisioning State Machine. The anchored state defined in [SBC] is not supported. An unmapped LBA is a deallocated LBA.

UFS device shall support Thin Provisioned logical unit including: Logical Block Provisioning Management, UNMAP command, erased, discard and purge functionalities as described in 12, UFS Security.

13.4 Host Device Interaction

13.4.1 Overview

This sub- clause describes several UFS features conceived to improve performance and/or reliability. Some of them are related directly to commands present in the logical unit queues (e.g., inter-LU priority, system data tag, context management), others are related to the device state (e.g., background operations, dynamic device capacity) or focused on reliability (e.g., data reliability, real-time clock information).

13.4.2 Applicable Devices

All the features described 13.4 should be implemented on UFS devices. The extent to which the features are implemented will be up to the device manufacturer. It is expected that poor implementations will result in lower device performance or reliability and higher max power consumption in the low power modes.

13.4.3 Command Queue: Inter-LU Priority

The specific details of how commands interact in a queue of a specific logical unit are handled in other clauses. This sub-clause outlines how the queues within a device interact with each other. There can be many different implementations of a UFS device. For example, it may be implemented as a single or multithreaded processor. In order to make the UFS spec implementation agnostic this sub-clause outlines a parameter that allow the host to communicate to the device its priorities so that the device can take this into account when executing commands from the host.

In the implementation case where several queues are serviced from a single execution unit, it is necessary for the host to either designate all queues as having the same priority or designate a single Queue as having a higher priority. A parameter value shall be defined to allow the host to designate a queue as having high or no priority. In the case where a queue is designated as having a higher priority, whenever a command enters the queue with the high priority it will be executed as soon as possible resulting in commands from other queues being stalled.

One example of where a host may want to take advantage of this feature is when the host has allocated one logical unit to be a code execution unit and another unit to be a mass storage unit. In this example the execution of code takes priority over mass storage transfers, so the host would set the parameter bit associated with the code logical unit to be high priority and leave the remaining queues as lower priority. The logical unit supports two level of queue priorities:

1. High priority – This is used for logical unit with high priority requests. Any commands sent to this logical unit will have higher priority than commands sent to logical units with lower priority. For example, when servicing demand-paging applications, read commands would have this priority.
2. No priority – This is used for all regular logical units which do not belong to priority #1

In addition to the execution of commands explicitly issued by the host, the device may execute background operations for device house-keeping. In general those operations have a much lower priority than the commands sent by the host and are implemented using a specific method described in 13.4.4, Background Operations Mode.

13.4.3.1 Implementation

The bHighPriorityLUN parameter in the Device Descriptor shall be set to configure which logical unit has the command queue with the higher priority.

bHighPriorityLUN shall be set to:

- 7Fh: if all logical units have the same priority,
- LUN of the higher priority logical unit.

Name	Description	Valid Values	Default(1)
bHighPriorityLUN	bHighPriorityLUN defines the high priority logical unit. If this parameter value is 7Fh all logical units have the same priority.	0 to the number of LU specified by bMaxNumberLU, and 7Fh	7Fh
NOTE 1 The column "Default" defines the parameter value set by the device manufacturer.			

13.4.4 Background Operations Mode

13.4.4.1 Introduction

A managed device requires time to execute management tasks. The background operations mode grants the device time to execute the commands associated with these flash management tasks. Flash management operations may include, but are not limited to, wear leveling, bad block management, wipe and garbage collection. The operations completed during the background operations period are determined by the device manufacturer and are not covered by the UFS spec.

13.4.4.2 Purpose

Performance Improvement
The intent of this mode is to improve a device response to host commands by allowing the device to postpone device management activities that occur as a result of host initiated operations to periods when the host is not using the device.

Systems that will use a UFS device tend to have peak periods of activity, in which the best response possible is needed from the UFS device. These peak periods are followed by idle periods, where the host can allow the device to do device management operations. Allowing better communication between the host and the device on when these idle periods will occur allows the UFS device to perform more optimally in the system.

The device will still be permitted to do device management when the host initiates operations, however the downside of doing this may be poorer device performance. This mode will just give device manufacturers the option to delay device management operations to improve performance. It is recommended that devices postpone as many tasks as possible to take full advantage of the possible performance improvements associated with this feature.

13.4.4.2 Purpose (cont'd)

Host Power Management
This mode will also allow the host to control when the device uses power to perform management activities. The host will have more knowledge about the power consumed by the system and can make the appropriate tradeoffs about when to use system power and when to conserve it.

An example of where host control over power consumed by the device could be an advantage would be the case where the system has very little battery power and the UFS device has a lot of unused memory.

In this case the host may not wish for the device to perform clean up operations but conserve the power for more critical system functions.

Allowing the host to communicate with the device on when activities can be performed will allow better system power management, which can be controlled by the host.

13.4.4.3 Background Operations Status

The device signals to the host that the device has a need for background operations using the Exception Event mechanism, and in particular the URGENT_BKOPS bit in wExceptionEventStatus

- '0': No immediate need to execute background operation (corresponds to no operations or non-critical)
- '1': Immediate need to execute background operation (corresponds to performance being impacted or critical)

When the host detects a request for executing background operations (URGENT_BKOPS set to one) it may read the bBackgroundOpStatus attribute to find out the need level, as follows:

00h : No operations required

01h : Operations outstanding (non-critical)

02h : Operations outstanding (performance being impacted)

03h : Operations outstanding (critical)

The URGENT_BKOPS bit is set to zero if the bBackgroundOpStatus is set to zero or one, otherwise it is set to one.

It is expected that the host will respond as soon as possible when the status changes to service, since if the background operations are not properly managed, then the device could fail to operate in an optimal way.

If device status is operations outstanding (critical), commands other than mode sense and mode select may be terminated with CHECK CONDITION status. The host should put in all possible measures to ensure the device never reaches this state since it means the device is no longer able to operate.

The point at which the device enters each of these states is up to the manufacturer and is not defined in this standard.

13.4.4.4 Operation Initiation

There is no explicit command to start the background operations. A mode enable bit indicates whether the device is allowed to execute background operations. If the background operations enable bit is set and the device is in Active power mode or Idle power mode, then the device is allowed to execute any internal operations.

When the device receives a command which requires a data transfer and if all command queues are empty, then the device shall start the data transfer sending DATA IN UPIU or RTT UPIU within the time declared in bBackgroundOpsTermLat. The host can minimize the device response time by disabling background operations mode during critical performance times.

In the case where the background operations status is “operations outstanding (critical)”, the aforementioned latency limit does not apply.

13.4.4.5 Power Failure

It is the device’s responsibility to ensure that the data in the device is not corrupted if a power failure occurs during a background operation.

13.4.4.6 Implementation

13.4.4.6.1 Background Operations Enable

fBackgroundOpsEn is the Flag to be used to enable or disable the execution of background operations. This Flag is defined as follows:

- 0 = Device is not permitted to run background operations
- 1 = Device is permitted to run background operations.

The default value of this Flag is one: background operations permitted. The device shall terminate ongoing background operations when this Flag is cleared by the host. For more details **see** Table 14.26**.**

13.4.4.6.2 Background Operations Status

bBackgroundOpStatus is an attribute defined as follows:

00h = not required

01h = required, not critical

02h = required, performance impact

03h = critical.

For more details, see 14.2, Flags.

13.4.4.6.3 Background Operations Termination Latency

bBackgroundOpsTermLat defines the maximum latency for starting data transmission when background operations are ongoing. The termination latency limit applies to two cases:

- When the device receives a COMMAND UPIU with a transfer request. The device shall start the data transfer and send a DATA IN UPIU or a RTT UPIU within the latency limit.
- When the device receives QUERY REQUEST UPIU clearing fBackgroundOpsEn Flag. The device is expected to terminate background operations within the latency limit.

The termination limit does not apply in the case where the background operations status is “operations outstanding (critical)”.

bBackgroundOpsTermLat is a parameter of the Device Descriptor and its granularity is 10msec. It is expected that transitions between link states (e.g., HIBERNATE to ACTIVE) or temporary congestion on the link occur in shorter timescales and are therefore negligible in comparison to the background operations timescale.

13.4.5 Power Off Notification

A UFS host will notify the device when it is going to power the device off by requesting the device to move to UFS-PowerDown power mode. This will give the device time to cleanly complete any ongoing operations. The device will respond to the host when it is ready for power off, meaning that the device entered the UFS-PowerDown power mode. Host can then power off the device without the risk of data loss.

13.4.6 Dynamic Device Capacity

Common storage devices assume a fixed capacity. This presents a problem as the device ages and gets closer to its end of life. When some blocks of the device become too old to be used reliably, spare blocks are reallocated to replace them. A device contains some spare blocks for this purpose, as well as for some housekeeping operations. However, when all spare blocks are consumed, the device can no longer meet its fixed capacity definition and it stops being functional (some become read-only, some stop responding completely).

A simple solution to enable the host to continue operation at the end of the device lifetime, would be to allow the capacity to be dynamic. If the device is allowed to reduce its reported capacity, it can reallocate blocks that were used to store data as new spare blocks to compensate for aging. To support this, the device needs to report to the host how much more spare blocks it needs for each logical unit, and then the host needs to relinquish some used blocks to let the device reallocate them as spares.

A device may implement a spare blocks resource management policy either per logical unit, allocating a fixed amount of spare blocks per logical unit, or per memory type, allocating a fixed amount of spare blocks for all logical units with the same memory type (bMemoryType).

bDynamicCapacityResourcePolicy parameter in the Geometry Descriptor indicates which spare blocks resource management policy is implemented.

13.4.6.1 Implementation

13.4.6.1.1 Initial Device Requirements

- Only logical units that support thin provisioning and logical block provisioning management functions can be involved in the dynamic device capacity process. The host can discover if thin provisioning is enabled and if a logical unit supports logical block provisioning management functions at UTP level, reading the bProvisioningType parameter in the Unit Descriptor of each logical unit, or at SCSI level through the TPE bit in the READ CAPACITY (16) parameter data.
  - bProvisioningType shall be either 02h or 03h.
  - TPE bit shall be set to one.
- The Unit Descriptor of each logical unit includes the following two parameters: qLogicalBlockCount and qPhyMemResourceCount.
  - The qLogicalBlockCount is equal to the total number of addressable logical blocks in the logical unit. Its value is established when the logical unit is configured and never changes during the device life time. In particular, the qLogicalBlockCount value shall be equal to RETURNED LOGICAL BLOCK ADDRESS + 1 (RETURNED LOGICAL BLOCK ADDRESS is a field included in the Read Capacity Parameter Data and corresponds to the last addressable block on medium under control of logical unit).
  - The qPhyMemResourceCount is equal to the total physical memory resources available in the logical unit, expressed in $2^{bLogicalBlockSize}$ unit. Its value decreases with the execution of dynamic capacity process.
- UFS requires that there shall be sufficient resource in the physical memory resources pool to support the logical addressable memory space reported in READ CAPACITY when the device is first configured. Therefore, qPhyMemResourceCount shall be equal to qLogicalBlockCount in the Unit Descriptor initially.
- Dynamic device capacity feature does not involve the following logical units: RPMB well known logical unit, power-on write protected logical units (independently of fPowerOnWPEn flag value), permanently write protected logical units (independently of fPermanentWPEn flag value), or logical units configured as boot logical unit (Boot LUN ID = 01h or 02h, independently of bBootLunEn value).

13.4.6.1.2 Dynamic Capacity Procedural Flow

1) When the physical memory resources necessary for proper operation in a logical unit has been drawn down, the device may request to the host to remove some resources from the physical memory resources pool serving the logical address space of the logical units. This is achieved setting to one the DYNCAP_NEEDED bit in the wExceptionEventStatus attribute. If DYNCAP_EVENT_EN bit in wExceptionEventControl attribute is one, then the EVENT_ALERT bit of Device Information field included in the RESPONSE UPIU will be set to one.
2) Device shall indicate the amount of physical memory to be removed from the resource pool in dDynCapNeeded attribute. Each element of the dDynCapNeeded[LUN] shall be equal to the amount of bytes to be removed divided by the optimal write block size (bOptimalWriteBlockSize is a parameter in the Geometry Descriptor). Therefore, the host can calculate the amount of physical memory to be removed from the resource pool for each logical unit multiplying the dDynCapNeeded[LUN] attribute by bOptimalWriteBlockSize parameter. UFS device shall not request to remove physical memory from the resource pool of logical units which are write protected or configured as boot logical unit (dDynCapNeeded[LUN] shall be zero).

13.4.6.1.2 Dynamic Capacity Procedural Flow (cont'd)

3) The host ensures that all outstanding tasks in the device queues have been completed or aborted.

4) When it is the case that the device spare blocks resource management policy is per memory type (bDynamicCapacityResourcePolicy = 01h), then the host should ensure that the amount of LBAs in the deallocated state in all logical units with the same memory type (bMemoryType) is equal to or greater than the amount of requested physical memory resources from all logical units with the same memory type (bMemoryType).

   For example, assuming that bDynamicCapacityResourcePolicy = 01h and the device asks to remove logical blocks from the resource pool of a single logical unit, the host may remove blocks from one or more logical units having that particular memory type as long as the total amount of logical blocks in a deallocated state results be greater than or equal to the total amount of logical blocks specified by the device.

   However, when it is the case that the device spare blocks resource management policy is per logical unit (bDynamicCapacityResourcePolicy = 00h), then the host ensures that the amount of LBAs in the deallocated state is equal or greater than the amount of requested physical memory resources for each logical unit.

   The host deallocates LBAs sending one or more UNMAP commands.

   a) The range(s) of LBA in the deallocate state shall be aligned to a bOptimalWriteBlockSize boundary, and their size shall be an integer multiple of bOptimalWriteBlockSize .
   b) The LBA range(s) shall be equal or greater than the memory resources amount that has been requested by the device for each logical unit.
   c) The LBA range(s) does not need to be at the end of the logical address space.

5) The host sets fPhyResourceRemoval flag to one and triggers an EndPointReset or a device hardware reset to initiate the dynamic capacity operation.

6) The host may either execute the complete boot process as described in 13.1, UFS Boot, or may skip reading the boot logical unit and set fDeviceInit flag to one to start the device initialization. During this phase the device will execute the dynamic capacity operation. The host waits until the device clears fDeviceInit flag. The device initialization may take a time longer than normal initialization due to internal processes necessary to re-arrange physical memory resources. If a power loss occurs, the dynamic capacity operation will proceed at the next power up during the device initialization.

13.4.6.1.3 Dynamic Capacity Procedural Flow (cont'd)

7) When the dynamic capacity operation is completed, the device will clear fDeviceInit flag and the fPhyResourceRemoval flag. If the operation is completed successfully, the DYNCAP_NEEDED bit is cleared too, therefore the EVENT_ALERT bit in Device Information will return to zero, if no other exception events are active. The qPhyMemResourceCount parameter in the Unit Descriptors is updated with a new value reflecting the amount of physical memory resources remaining in the resource pool.
   NOTE The qLogicalBlockCount parameter in the Unit Descriptors and the RETURNED LOGICAL BLOCK ADDRESS parameter in Read Capacity Parameter Data will stay the same as initially configured. Therefore, the updated qPhyMemResourceCount value will be less than those two parameters after dynamic capacity operation.

8) If the dynamic capacity operation does not succeed, for example because LBA range(s) does not meet one or more of the requirements described in point 4), the DYNCAP_NEEDED bit shall remain set to one. Therefore, the EVENT_ALERT bit in the Device Information field of the RESPONSE UPIU will remain set to one to notify the host that further dynamic capacity operation is needed.

   NOTE The dDynCapNeeded[LUN] attribute value may have been updated.

9) To account for the reduction of the physical memory resources pool after dynamic capacity operation, the host maintains a range(s) of LBAs in deallocated state that are aligned in address and size to integer multiples of bOptimalWriteBlockSize, with the total equal to or greater than qLogicalBlockCount minus qPhyMemResourceCount. Otherwise, a write error will result when the host attempts to write (map) more LBAs than the available physical memory resources.

   NOTE The host may change the LBA range(s) that are in deallocate state during the use of the device.

13.4.6.1.3 Dynamic Device Capacity Notification

The dynamic device capacity is one of the exception events that may set the EVENT_ALERT bit of the Device Information field included in the RESPONSE UPIU.

To enable the setting of this bit, the DYNCAP_EVENT_EN bit in the wExceptionEventControl attribute shall be set to one.

When the host detects that the EVENT_ALERT bit is set to one, it should read the wExceptionEventStatus attribute to discover if the source of this event is a request for reducing the device capacity. In particular, if DYNCAP_NEEDED bit is set to one, the host should process the request as described in this standard.

The DYNCAP_EVENT_EN bit shall be set to zero if the host is not capable or does not intend to use the dynamic device capacity feature. Otherwise, a dynamic capacity request event will set the EVENT_ALERT bit to one, masking out notification of other exception events.

13.4.6.1.4 Error Handling

- Success/failure to unmap (release) LBAs is as defined in UNMAP command.
- If the host ignores the dynamic device capacity notification and continues to write to the device without unmapping LBAs to free up physical memory, the device may become non-writeable over time and cause a WRITE command to fail. In which case, the device server shall return the following error condition in response to the WRITE command.
  - The device server shall terminate the command requesting the operation with CHECK CONDITION status with the sense key set to DATA PROTECT and the additional sense code set to SPACE ALLOCATION FAILED WRITE PROTECT.
- When the qPhyMemResourceCount is less than qLogicalBlockCount in Unit Descriptors, it indicates that the physical memory resources pool is smaller than the logical addressable memory space (LBAs) in the logical unit. It is the host's responsibility to keep track of the amount of physical memory available. If the host attempts to write more data than the available physical memory resources and the device is unable to complete the write operation successfully, the device shall return the following error condition.
  - The device server shall terminate the command requesting the operation with CHECK CONDITION status with the sense key set to DATA PROTECT and the additional sense code set to SPACE ALLOCATION FAILED WRITE PROTECT.

13.4.6.1.5 Physical Memory Resource State Machine

Figure 13.5 shows the state machine for the physical memory resources. In addition to the "Mapped" state and the "Deallocated" state, which are defined in logical block provisioning state machine too, there is the "Removed from the Resource Pool" state.

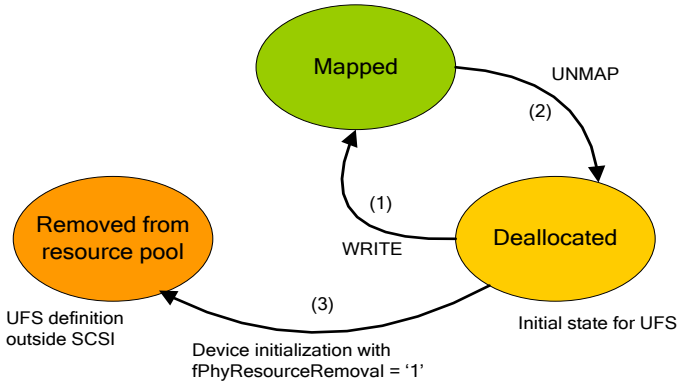


Figure 13.5 — Physical Memory Resource State Machine

1) Write operation: physical memory resource from the resource pool is mapped to LBA containing valid data.
2) UNMAP operation for erase/discard:
   a) Physical memory resource is unmapped (deallocated) from LBA and returned to the resource pool.
   b) Residual data in unmapped physical memory resource is not valid.
3) Device re-initialization with fPhyResourceRemoval flag previously set to one causes some physical memory resources to be removed from the resource pool servicing the logical address space. After conversion the qPhyMemResourceCount is updated by UFS device to indicate the amount of physical memory resources remaining in the resource pool of each logical unit.

13.4.7 Data Reliability

13.4.7.1 Description

The UFS host has the ability to define the level of data reliability during normal operation and power failure per logical unit.

There are two components to the data reliability that are defined for UFS. The first component deals with the data currently being written and the second deals with the data that has previously been stored in the medium.

The first component, also known as Reliable Write, means that if the device loses power during a write operation to the medium (when executing a SCSI write command, a host initiated flush of data to the medium, etc.), the data in the range affected by the operation will be either the old data or the new written data when the device recovers from power failure.

Keeping either the old or new data is important for the file system to recover after power loss. The file system may keep its structures segmented and signed with CRC or equivalent mechanism. The device insures that a large enough granularity of data is authentic either with an old or new copy, to make sure the file system can validate or invalidate these segments.

The resolution for the old and new data shall be aligned to the logical block size. Figure 13.6 shows some of the possible scenarios that the host will see when it recovers from power failure during a 32 Kbyte write operation in a logical unit with 4 Kbyte logical block size.

For RPMB well known logical unit, the data reliability granularity shall be equal to bRPMB_ReadWriteSize × 256 bytes in Normal RPMB & bRPMB_ReadWriteSize X 4K bytes in Advanced RPMB. The bRPMB_ReadWriteSize value shall be changed to the appropriate value based on RPMB configuration.

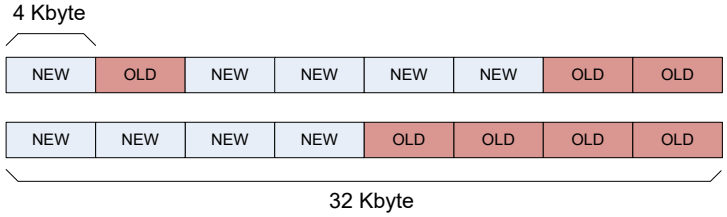


Figure 13.6 — Example of Data Status After a Power Failure During Reliable Write Operation

The second component, also known as Data Reliability, allows the host to define the level of protection to be applied to existing data on the device. In some technologies that will be used to implement UFS devices, the device has the option to potentially sacrifice some of the existing data on a device during a power failure in order to provide better write performance. Depending on the application for the end device the host can select per logical unit whether the data in that logical unit shall be protected during power failure, which may have a performance impact, or to select device performance with the risk of losing data during a power failure.

Data Reliability feature for each logical unit can be set when the device is configured during system integration.

13.4.7.1 Description (cont'd)

In particular, if Data Reliability is enabled, the logical unit will execute Reliable Write operation and the data already stored in the medium will not be corrupted by a power failure occurred during the execution of a write operation to the medium.

The data reliability feature will be configurable only for logical units and not for well known logical units.

The RPMB well known logical unit will automatically select data reliability.

13.4.7.2 Implementation

bDataReliability parameter in the Unit Descriptor shall be used by the host to configure the logical unit data reliability.
bDataReliability values are defined as in the following:

00h: Data reliability disabled
Corruption of the existing data in the medium of the specific logical unit may occur if a power loss happens during device activity like writing of data to medium.

01h: Data reliability enabled
The existing data stored in the medium of the specific logical unit shall not be corrupted if a power loss occurs, and the memory range that was accessed by the interrupted write command shall contain the old data or new data (or a mixture of old and new data as explained in 13.4.7.1) once power is restored.

Table 13.4 — Parameter for Controlling Logical Unit Data Reliability

Parameter Name	Description
bDataReliability	bDataReliability defines the device behavior when a power failure occurs when writing data to the medium 00h: Data reliability disabled 01h: Data reliability enabled Others: Reserved

13.4.8 Real-Time Clock Information

Providing Real Time Clock (RTC) information to a storage device could be useful for the device internal maintenance operations (execution of RTC internal operations is not affected by fBackgroundOpsEn flag value).

Host may provide either absolute time if available, or relative time information. This feature provides a mechanism for the host to update either absolute or relative time.

The host sends RTC information when a wPeriodicRTCUpdate has passed since the last RTC information update. In case the device is not powered up or asleep when the period has expired, the host updates RTC information when the device wakes.

NOTE When a device is configured with wPeriodicRTCUpdate set to 'Undefined', a periodic update is not possible and the host may update RTC information according to vendor recommendations.

13.4.8 Real-Time Clock Information (cont'd)

It is recommended to provide RTC information after transitioning from UFS-Sleep or UFS-PowerDown power mode to Active power mode.

Updating RTC information is done by writing to dSecondsPassed attribute.

While the device is busy handling an RTC update event and the related background operation, the device may keep the fBusyRTC flag is set to one.

The device may perform operations in the background as a result of receiving RTC update. In order to allow optimized efficiency of these background operations, it is recommended that the host refrains from sending commands, other than Query Request to the device, and keep the device powered and in Active power mode as long as fBusyRTC flag is set to one. The device may consume active power during that time. Therefore, it would be advisable to update RTC in times where the system is usually idle and has no specific power limitations, e.g., at night time when battery is being charged.

13.4.9 Timestamp Information

Providing Timestamp information (qTimestamp) to a storage device could be useful for synchronizing device Error Log (retrieved through READ BUFFER command) and host error log.

This feature provides a mechanism for the host to provide a time tag (timestamp) to the device; the device may store this information in non-volatile memory and utilize for time tagging in the device error log when an event occurs.

In case the device is not in Active state, device may not know how much time elapsed since it was last active. Hence, it is recommended that the host set the device qTimestamp upon device power-on Reset / HW reset or when switching to Active state (using Start Stop Unit command).

13.4.10 Context Management

To better differentiate between large sequential operations and small random operations and to improve multitasking support, contexts can be associated with groups of read or write commands. Associating a group of commands with a shared context allows the device to optimize data handling.

A context can be seen as an active session, configured for a specific read/write pattern (e.g., sequential in some granularity). Multiple read or write commands are associated with this context to create some logical association between them, to allow device performance optimization. For example, a large sequential write pattern may have better performance by allowing the device to improve internal locality and reduce some overhead (e.g., if some large unit of data is allowed to be affected by a power failure as a whole while it is being written, all of the commands that fill this unit can work faster because they can reduce the overhead usually required to protect each write individually in case of a power failure). Furthermore, handling of multiple concurrent contexts allows the device to better recognize each of the write patterns when they are all mixed together.

The maximum number of contexts the device can support is reported in the bMaxContexIDNumber field of the Geometry Descriptor. When the host configures the device, it divides this number across the created LUs and writes the number of contexts to be supported in each LU to wContextCapabilities field in the Configuration Descriptor.

To use a context, an available Context ID shall be picked. Then, it shall be initialized by writing the configuration attribute of the relevant LU (wContextConf). Then, data can be read/written associated to the context by specifying the Context ID in GROUP NUMBER field of the CDB of the read/write command. When the context is no longer used, the configuration attribute (wContextConf) should be written as all ‘00’ by the host to close the context. A context shall be closed prior to re-configuring it for another configuration/use.

No ContextID shall be open after power cycle.

13.4.10.1 Context Configuration

Before any context can be used it shall be configured.

Configuration is done by setting the context configuration attribute of the relevant LU (setting wContextConf with INDEX equal to LUN and SELECTOR equal to ContextID, SELECTOR ‘0’ is reserved.). Then, all read commands or write commands that are associated with this ContextID shall be sent using GROUP NUMBER set to ContextID.

When the context is no longer needed, it should be closed by writing a zero byte to the configuration attribute.

13.4.10.1 Context Configuration (cont'd)

To configure a specific ContextID for a specific LU, the following fields shall be written to the context configuration attribute of the specific context needed:

- Activation and direction mode (read-only, write-only or read/write)
  The direction indicates if all following accesses to this context would be either read-only, write-only or both read/write. Writing a non-zero direction to this field is the 'activation code' for the context. A zero in this field means a closed context which can no longer be addressed by its ID until re-configured.
- Large Unit context flag
  This indicates if the context is following Large Unit rules or not
  - If Large Unit context flag is set, then the Large Unit multiplier is used to specify the larger unit size.
- Reliability mode
  Controls how data written to a context should respond to interruptions.

13.4.10.2 Activation and Direction Mode

A non-zero context can be configured as a read-only context, a write-only context or a read/write context.

Any read command associated with any context to an address that is part of an active write-only context is not allowed, and may either fail the command or return dummy data.

Any write command associated with any context to an address that is part of an active read-only context is not allowed, and may either fail the command, ignore the data written or cause unexpected data to return in the read commands.

A context that is configured as read/write context may be read or written, as long as the writing follows the context rules.

NOTE read/write context may have reduced performance compared to read-only or write-only contexts.

13.4.10.3 Large-Unit Mode

The Large Unit is the smallest unit that can be used for large sequential read/write operations, in order to reduce internal overhead and improve performance.

Accessing a Large Unit (both read and write) shall:

- Use a ContextID configured to operate in Large Units.
- Always access a full Large Unit, in order and from beginning to end.
  - Multiple read/write commands with TRANSFER LENGTH smaller than the Large Unit size may be used to read or write the Large Unit. Read/write commands may be interleaved with other accesses and commands, as long as the specific Large Unit is being accessed with its own separate Context ID.
  - A Large Unit that is being written shall not be modified outside the scope of the context (e.g., no other writes from other contexts to the address range of the Large Unit shall be used, no erases/trims to that range, etc.).

13.4.10.3 Large Unit Mode (cont'd)

- Different Large Units belonging to the same context may be located in non consecutive addresses on the media, as long as alignment is kept (a Large Unit shall be accessed in order from beginning to end, but only within the range of the specific Large Unit – the next Large Unit can be non-consecutive and even in a lower address).

- When writing a Large Unit context, data shall always be aligned and in multiples of bOptimalWriteBlockSize.

When writing a Large Unit context, the last Large Unit before closing the context may be partially written, as long as it is written from the beginning, in order and up to a specific point where it is closed. The rest of the Large Unit may be padded by the device to the end of the Large Unit with random data.

13.4.10.4 Reliability Mode

In case a write command to a Context ID is interrupted, the device behaves as if all the writes to the context from its configuration were written in one large write command.

In case of a power failure or software reset before closing an active context – even if not in the middle of a write command to the specific context (even if not in the middle of any command) – is considered as if the event occurred during writing the entire context.

A context behavior is determined as part of its configuration and is applied to all writes to this context until it is closed. Interruption during any of the writes may cause some of the data not to be fully programmed on the device. Still no partial Logical Block (of bLogicalBlockSize size) shall exist – any Logical Block written as part of the context shall contain either the new data written or its old data before the context was configured. The scope of data that may be affected by the interruption depends on the mode configured:

- For non-Large Unit contexts:
  - MODE0 – Normal mode – Any data written to the context from the time it was configured may be affected.
  - MODE1 – Non-Large Unit, reliable mode – Only data written by a specifically interrupted write command may be affected. Any previously completed write to the context shall not be changed because of any interruption.
- For Large Unit contexts:

  NOTE In the following cases, the unit N refers to the current Large Unit which is being written when interruption occurs, unit N-1 refers to the last Large Unit of the context that was written completely before the current one and unit N-2 and earlier are Large Units that were completed before the N-1 unit.

  - MODE0 – Normal mode – Any unit may be affected: Any data written to the context from the time it was configured may be affected.
  - MODE1 – Large Unit, unit-by-unit mode – Unit N may be affected, units N-1 and earlier are not: Any data written to a Large Unit context may affect the entire specific Large Unit accessed. Any previously completed Large Units in the context shall not be changed because of any interruption.
  - MODE2 – Large Unit, one-unit-tail mode – Unit N and N-1 may be affected, units N-2 and earlier are not: Any data written to a Large Unit context may affect the entire specific Large Unit accessed and the entire completed Large Unit that was accessed before the current one. Any other completed Large Units in the context shall not be changed because of any interruption.

13.4.10.4 Reliability Mode (cont'd)

In case the host sends a Task Management Request to abort a write command to a non-zero context or the write command fails with an error, the write may still be interrupted like any context-less write. In case these scenarios are interrupting a write to a Large Unit context, the device shall always stop writing on a bOptimalWriteBlockSize boundary.

13.4.10.5 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.

13.4.11 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.

13.4.11 System Data Tag Mechanism (cont'd)

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 14.1.5.4 for details).

When a host tags system data during a write operation, an entire storage area of bSysDataTagUnitSize size is handled by the device as system data area even if the size of the data being written is less than bSysDataTagUnitSize. In addition, any command (Write, Unmap, etc.) which updates a system data unit with data not tagged as System Data will change the entire system data unit storage characteristics to regular data. Therefore, it is recommended to handle system data in full units of bSysDataTagUnitSize size.

System Data areas are available only in Normal memory type logical units.

13.4.12 Exception Events Mechanism

The Exception Events Mechanism is used by the device to report occurrence of certain events to the host. It consists of three components EVENT_ALERT bit, the wExceptionEventStatus attribute and wExceptionEventControl attribute:

- A bit in wExceptionEventStatus attribute is assigned to each exception event. The device shall set the wExceptionEventStatus bits to one when the corresponding exception events are active, otherwise they shall be set to zero.
- A bit in wExceptionEventControl attribute is assigned to each exception event.
  EVENT_ALERT bit shall be set if there is at least one wExceptionEventStatus bit and wExceptionEventControl bit pair set to one.
  The setting of an wExceptionEventStatus bit to one will not force the EVENT_ALERT bit to one if the corresponding bit in the wExceptionEventControl is zero.
- The EVENT_ALERT is a bit in the Device Information field of the RESPONSE UPIU which is the logical OR of all bits in the wExceptionEventStatus masked by the bits of the wExceptionEventControl.
  The EVENT_ALERT bit is set to one when at least one bit in the wExceptionEventStatus is set and the corresponding wExceptionEventControl bit is one.
  The EVENT_ALERT is set to zero if all exception events that are enabled in the wExceptionEventControl are not active.

13.4.12 Exception Events Mechanism (cont'd)

The bits in the wExceptionEventStatus associated with those exception events are described as follows:

- DYNCAP_NEEDED – the device requests a Dynamic Capacity operation (see 13.4.6). This bit is cleared once a Dynamic Capacity operation has completed successfully, releasing the entire capacity that the device had requested to release.
- SYSPOOL_EXHAUSTED – the device ran out of resources to treat further host data as System Data (see 13.4.11). This bit is cleared once the host has turned enough memory areas that were previously handled as System data areas, to non-system data areas.
- URGENT_BKOPS – the device requests host attention for the level of need in Background Operations (see 13.4.4, Background Operations Mode). This bit is cleared once bBackgroundOpStatus returns to 00h or 01h.
- TOO_HIGH_TEMP – the device requests that the host takes action to reduce the device's Tcase temperature.
- TOO_LOW_TEMP – the device requests that the host takes action to increase the device's Tcase temperature.
- PERFORMANCE_THROTTLING – the device is operating at reduced performance. The host may read bThrottlingStatus to determine the cause of reduced performance.
- WRITEBOOSTER_FLUSH_NEEDED – Buffer for WriteBooster needs to be flushed. The host is expected to issue a flush command by setting fWriteBoosterBufferFlushEn as '1'.
- DEVICE_LEVEL_EXCEPTION_OCCURRED – The device requests that the host takes action to examine the device level exception occurrence as described in 13.4.12.1. This bit is cleared once the attribute qDeviceLevelExceptionID is read.
- WRITEBOOSTER_RESIZE_HINT – the device recommends to increase or decrease the WriteBooster Buffer size. The host may read bWriteBoosterBufferResizeHint to identify hint information about which type of resize is recommended, then enable decrease or increase the WriteBooster Buffer size by setting the bWriteBoosterBufferResizeEn attribute.
- PINNED_WRITEBOOSTER_FULL – the device informs the host that the allocated buffer for pinned data in WriteBooster is full, so further pinned data may be written to normal storage.
- HEALTH_CRITICAL – Notify the host of a critical health condition. Following fields may change to reflect critical health condition and host may read following fields to get detailed information: bPreEOLInfo, bDeviceLifeTimeEstA, bDeviceLifeTimeEstB, bWriteBoosterBufferLifeTimeEst, bRPMBLifeTimeEst. Clear condition of the bit is met when host reads wExceptionEventStatus, which cause device to clear the HEALTH_CRITICAL.

In the Device Information field of the RESPONSE UPIU, the device will only indicate the events that were enabled by the host through writing to the wExceptionEventControl attribute. The event bits in the wExceptionEventStatus attribute and in the Device Information field of the Response UPIU are cleared by the device when the clear conditions are met.

13.4.12.1 Device Level Exception Event Notification

This feature provides a notification to the host if a device-level exception occurs.

The device issues the DEVICE_LEVEL_EXCEPTION_OCCURED exception event using the Exception Event Mechanism defined in 13.4.12 to inform the host of a device level exception event occurrence.

The host may read the device qDeviceLevelExceptionID attribute to discover the detailed exception type for the device level exception event. On read, the qDeviceLevelExceptionID attribute shall be reset to 0. For a detailed definition of the qDeviceLevelExceptionID attribute's values, refer to the device manufacturer datasheet.

This feature is optional. The device indicates whether the feature is supported or not by reading dExtendedUFSFeaturesSupport in the Device Descriptor.

13.4.13 Queue Depth Definition

Each logical unit is responsible for managing its own task set. Independently from the task set, the resources used for queueing tasks may either be statically allocated to each LU, so that the LU is capable of queueing new tasks up to a certain depth, or be shared by all LUs, so that queueing resources are dynamically allocated to LUs, depending on tasks received.

A device may implement one of the two queueing architectures described above. The device informs the host software on the policy implemented using read only parameters in the Device Descriptor and the Unit Descriptors.

The depth of a queue is defined as the number of pending commands which can be stored in the queue.

13.4.13.1 Shared Queue

In the shared queue architecture, the device has a fixed-depth queue where tasks are queued as they are received, regardless of their LUN designation.

When a COMMAND UPIU is received, resources are allocated from the shared queue and the command is accounted towards the queue depth limit. The host is expected to track the queue depth and not issue more commands than can be stored in the queue. If queue resources are unavailable, the device shall return a response with TASK SET FULL status.

QUERY REQUEST UPIUs, NOP OUT UPIUs, and TASK MANAGEMENT REQUEST UPIUs are not stored in the shared queue.

When this queueing architecture is implemented, the parameter bQueueDepth in the Device Descriptor shall indicate the depth of the queue. The value of bQueueDepth shall be equal to, or larger than, 1. The bLUQueueDepth parameters in all Unit Descriptors shall all be equal to 0, while bLUQueueDepth in RPMB Descriptor may be 0 or equal to the number of enabled RPMB regions (see 13.4.13.3).

13.4.13.2 Per-Logical Unit Queues

In the per-LU queueing architecture, the device implements separate fixed-depth queues, one queue for each LU, or, in other words, allocates a fixed number of queueing resources for each LU.

When a COMMAND UPIU is received, resources are allocated from the queue associated with its logical unit, as indicated by the LUN field in the UPIU Header. The command is accounted towards the depth limit of the respective queue. The host is expected to track the queue depths and not issue more commands than can be stored in their designated queue. If resources are unavailable for the desginated LU, the device shall return a response with TASK SET FULL status (even if queueing resources are available for other LUs). QUERY REQUEST UPIUs, NOP OUT UPIUs, and TASK MANAGEMENT REQUEST UPIUs are not stored in the LU queues.

When this queueing architecture is implemented, the bLUQueueDepth parameters in Unit Descriptors shall indicate the depth of the queue of each logical unit. If a logical unit is enabled, the value of bLUQueueDepth in its Unit Descriptor shall be equal to, or larger than, 1.

bQueueDepth parameter in the Device Descriptor shall be equal to 0.

NOTE For backward compatibility with previous revisions of the standard, if bLUQueueDepth for an LU is 0, and bQueueDepth is also 0, the queue depth should be treated by the host as unknown. The host is expected to infer the queue depth using software algorithms.

13.4.13.3 RPMB Well Known Logical Unit Queue

RPMB logical unit may use shared queue resources or may use its own separate queue, as implemented by the device manufacturer.

If the RPMB logical unit uses the shared queue resources, its bLUQueueDepth parameter shall be equal to 0. When a COMMAND UPIU is received, resources are allocated from the shared queue, and the command is accounted towards the queue depth limit. The host is expected to track the queue depth and not issue more commands than can be stored in the queue. If queue resources are unavailable, the device shall return a response with TASK SET FULL status.

If the RPMB logical unit uses a separate queue, its queue depth shall be equal to the value of bLUQueueDepth, except for 0. The host is expected to not issue more than one command to an RPMB region at any given time. Therefore, if more than the number of commands specified by bLUQueueDepth is issued to the RPMB LU, the device shall return a response with TASK SET FULL status.

It should be noted that, unlike other LUs, it is permitted that RPMB LU has a fixed depth queue while other LUs use a shared queue.

13.4.14 Device Life Span Mode

The intent of this mode is to improve the device life span by increasing the device endurance. Devices use mechanism like wear leveling etc. for improving the device life time which is limited to P/E cycle count given by a memory vendor. In this mode, device may use technology like lower programming voltage etc. for operations to increase the P/E cycle count and result in improving the device life.

Read and write operation performance in UFS are very high, However maximum operation speed is not required always e.g., when user is sleeping, device is in screen-off mode, downloading large size files/video and so on. During such scenarios, UFS host may indicate to device to use technology like lower programming voltage etc. for operations by enabling fDeviceLifeSpanModeEn flag. On disabling this flag, device uses normal voltage for operations. It is expected that the device will respond normally as soon as the flag is disabled.

There may be performance degradation in this mode, therefore fDeviceLifeSpanModeEn should be set only when the device is not actively used.

The improvement of device life span is dependent on device implementation.

13.4.14.1 Implementation

Device Life Span Mode Enable
fDeviceLifeSpanModeEn is the Flag to be used to enable or disable the execution of technology like lower programming voltage etc. for operations.

0 = Device Life Span Mode is disabled.

1 = Device Life Span Mode is enabled.

The default value of this flag is zero. Host can enable this flag depending on scenarios like screen-off, downloading large files etc.

13.4.15 Refresh Operation

In order to improve reliability (e.g., retention), the device data can be refreshed at the physical block level by erasing and re-programming its physical blocks.

The refresh operation mechanism provides a better capability for the host to control refresh operations explicitly with the refresh configuration, initiation, interruption, and progress management.

This feature is optional. The device indicates whether the feature is supported or not in bUFSFeaturesSupport parameter in Device descriptor. It aims at addressing reliability requirements.

It depends on device implementations how the reliability improves on this feature.

13.4.15.1 Configuration

13.4.15.1.1 Refresh Method

From refresh perspective, there are three types of blocks:

- Type 1: Physical blocks that don't contain data (i.e., unmapped physical blocks). Blocks of this type will not be refreshed.
- Type 2: Physical blocks that contain data but the device doesn't consider them to be in need of refresh
- Type 3: Physical blocks that contain data and the device considers them to be in need of refresh

The device shall support two refresh methods: Manual-Force or Manual-Selective.

- Manual-Force
  The device is obliged to refresh the amount of physical blocks as requested by the host, regardless whether these blocks need refresh or not. Only type 2 and type 3 blocks will be refreshed.
- Manual-Selective
  The device only refreshes the physical blocks that it considers to be in need of refresh.Specifically these are blocks of type 3.

The attribute bRefreshMethod needs to be initially set to either Manual-Force or Manual-Selective mode before initiating a device refresh operation.

Upon the host request, the device shall perform a refresh operation following the specified method (Manual-Force or Manual-Selective). In addition to the host initiated refresh modes, the device may implement an additional automatic refresh mode which is transparent to the host.

13.4.15.1.2 Refresh Unit

The host may configure bRefreshUnit attribute to specify the amount of physical blocks to be refreshed upon a single refresh request.

The device refreshes the amount of physical blocks specified by bRefreshUnit starting from the first physical block. In order to refresh the whole device (i.e., all physical blocks), the host has to send the refresh command one (bRefreshUnit = 01h: 100%) or more times (bRefreshUnit = 00h: Minimum refresh capability of Device).

In any case the device stops refreshing when dRefreshProgress reaches 100000 (100.000%).

13.4.15.1.3 Refresh Frequency

The device provides the refresh frequency indicator (bRefreshFreq attribute) in unit of month. The host should make sure that the whole device has been refreshed within the time period specified by the bRefreshFreq attribute. dRefreshTotalCount is incremented to indicate that the whole device refresh is completed.

Since the required refresh frequency depends on the use condition (e.g., temperature), bRefreshFreq attribute may be configured by OEM to let the host know how often it needs to send refresh requests.

13.4.15.2 Initiation and Interruption

The host initiates a refresh operation by setting fRefreshEnable to 1b and interrupts it by clearing fRefreshEnable to 0b. After interruption, fRefreshEnable can be also used to resume the refresh operation where it last stopped by setting it to 1b.

bRefreshStatus attribute provides information about a single refresh operation status.

The host should send a query request to set fRefreshEnable flag to one only if command queues are empty. A query request to set fRefreshEnable flag which is processed when device command queues are not empty may fail. If the request fails,, Query Response field in the QUERY RESPONSE UPIU shall be set to FFh ("General Failure"), the refresh operation shall not start, and the bRefreshStatus shall be set to 04h.

If a refresh operation is in progress, any request, other than Query Request (READ) and the refresh interruption described above, will fail.

Host should check the Device lifetime remained by reading bDeviceLifeTimeEst of Device Health Descriptor. Host should consider more carefully whether refresh operation will be issued or not since the refresh operation is consuming remained life time of device.

13.4.15.3 Progress Management

The device shall indicate the refresh progress with respect to its entire physical blocks. The two parameters are defined as progress monitors in Device Health Descriptor.

- dRefreshTotalCount
  Indicate how many times the device complete refresh for the entire device. Incremented by 1 when dRefreshProgress reach 100%.
- dRefreshProgress
  Indicate the refresh progress with respect to its entire physical blocks in %.

They work both in Manual-Force and Manual-Selective methods, and will not be changed by any operation (e.g., WRITE(10)) other than refresh operations.

It is host responsibility to keep pace of sending refresh requests based on:

- Refresh Frequency (bRefreshFreq)
- Refresh Unit (bRefreshUnit)
- Progress monitor (dRefreshProgress, dRefreshTotalCount)

For example, the host should send refresh request 1000 times during 6 month if

- bRefreshFreq = 06h (6 month)
- bRefreshUnit = 00h (e.g., 0.100% as Minimum refresh capability of Device)

bRefreshMethod (00h: Manual-Force, 01h: Manual-Selective) does not affect on how the progress monitor grows. Regardless of the actually refreshed blocks, dRefreshProgress is increased by the same amount. In particular this means that even if type 1 blocks (see 13.4.15.1.1) are not refreshed in Manual-Force mode and neither type 1 nor type 2 blocks are refreshed in Manual-Selective mode, dRefreshProgress is increased by the same amount.

13.4.15.3 Progress Management (cont'd)

When bRefreshMethod = 02h (Manual-Selective), even though some of physical blocks are not refreshed by device choice, dRefreshProgress should be incremented by the same amount.

For example, dRefreshProgress will be increased by 2% both in Manual-Force and Manual-Selective mode in the following conditions.

- Device has 100 physical blocks in total
- bRefreshUnit is set to 00h (e.g., 2% as Minimum refresh capability of Device)
- dRefreshProgress initially indicates 0%
- 1st physical block is empty
- 2nd physical block has data but does not need refresh

Case 1: bRefreshMethod = Manual-Force
Upon a single refresh command completion, dRefreshProgress will be increased by 2% even though the device ignores 1st physical block and refreshes 2nd physical block.

Case 2: bRefreshMethod = Manual-Selective
Upon a single refresh command completion, dRefreshProgress will be increased by 2% even though the device ignores both 1st and 2nd physical blocks.

13.4.16 Temperature Event Notification

The purpose of this feature is to provide notification to host in advance when UFS device temperature approaches defined upper and lower boundary of temperature. This feature is optional. Two bits in bUFSFeaturesSupport parameter specify the temperature event notification support (see 14.1.4.2)

When temperature of device is too high or too low that host's awareness is needed, device shall notify this situation to host by using wExceptionEventStatus Attribute in exception event mechanism defined in 13.4.12. When TOO_HIGH_TEMP in wExceptionEventStatus is raised, it is recommended for host to do throttling or other cooling activities for lowering device Tcase temperature. When TOO_LOW_TEMP in wExceptionEventStatus is raised, it is recommended for host to do activities for increasing device's Tcase temperature.

When this temperature alarming is raised, host may want to know temperature reported by device even though UFS device only could give rough temperature. In this purpose, device case rough temperature shall be provided through bDeviceCaseRoughTemperature attribute. Since the temperature sensor inside semiconductor device is not expected enough accurate, this temperature information shown in bDeviceCaseRoughTemperature is to provide the rough temperature range only. And there could some variation depending on location of measurement also, host should assume that this temperature information from device have around ± 10 ºC error range. Therefore, this temperature information should be referred by host only as rough device case temperature.

To give a temperature boundary information for too high or too low temperature alarming, bDeviceTooHighTempBoundary and bDeviceTooLowTempBoundary attribute is defined. The temperature alarming status field in wExceptionEventStatus shall be automatically cleared by device when device case temperature is going within boundary temperature range.

13.4.17 Performance Throttling Event Notification

The purpose of this feature is to provide notification to the host if the device is limiting performance. If the device needs to reduce performance, the host will be notified through the Exception Event Mechanism defined in 13.4.12. While the PERFORMANCE_THROTTLING exception event bit is set, the host should expect reduced performance from the device.

The host may read the device bThrottlingStatus attribute to discover why the device is operating at lower performance. Bits in bThrottlingStatus attribute will remain set while the condition exists.

This feature is optional. The device indicates whether the feature is supported or not by dExtendedUFSFeaturesSupport parameter in Device Descriptor.

How much the performance of the UFS device is reduced when a performance throttling event is notified depends on the device implementation.

13.4.18 WriteBooster

13.4.18.1 Overview

The write performance of TLC NAND is considerably lower than SLC NAND because the logically defined TLC bits require more programming steps and have higher error correction probability. To improve the write performance, part of the TLC NAND (normal storage) is configured as SLC NAND and used as write buffer, temporarily or permanently. Using SLC NAND as a WriteBooster Buffer enables the write request to be processed with lower latency and improves the overall write performance. Some portions of TLC NAND allocated for the user area are assigned as the WriteBooster Buffer. The data written in the WriteBooster Buffer can be flushed into TLC NAND storage by an explicit host command or implicitly while in hibernate (HIBERN8) state. Technologies other than TLC and SLC NAND may be used as normal storage and WriteBooster Buffer.

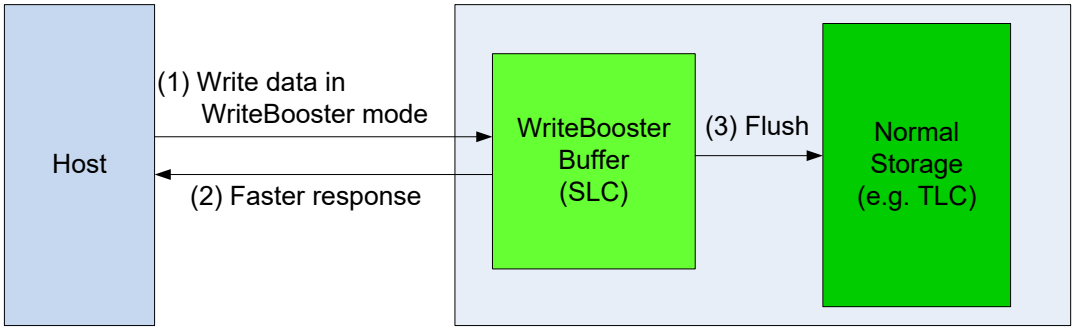


Figure 13.7 — Concept of WriteBooster Feature

Bit[8] of dExtendedUFSFeaturesSupport indicates if the device supports the WriteBooster feature.

There are two WriteBooster mode of operations: “LU dedicated buffer” mode and “shared buffer” mode. In the “LU dedicated buffer” mode, the WriteBooster Buffer is dedicated to a logical unit, while in the “shared buffer” mode all logical units share the same WriteBooster Buffer except well-known logical units. bSupportedWriteBoosterBufferTypes indicates which modes are supported by the device. In both WriteBooster mode of operations, the WriteBooster Buffer size is configurable.

13.4.18.1 Overview (cont'd)

There are two user space configuration modes: "user space reduction" and "preserve user space". With "user space reduction", the WriteBooster Buffer reduces the total configurable user space; while with "preserve user space", the total space is not reduced.

The WriteBooster feature shall be enabled for all WRITE commands when the fWriteBoosterEn flag is set to one. The WriteBooster feature shall be enabled for an individual WRITE command if that command's GROUP NUMBER field is set to 17.

bAvailableWriteBoosterBufferSize attribute indicates the available space in the WriteBooster Buffer. An exception event is triggered when there is the need to flush the WriteBooster Buffer: bit[5] of wExceptionEventStatus is set to indicate that data in WriteBooster Buffer should be flushed to normal storage.

There are two flags for controlling the WriteBooster Buffer flush operation. fWriteBoosterBufferFlushEn flag enables the flush operation: when it is set to one, the device shall flush the WriteBooster Buffer. fWriteBoosterBufferFlushDuringHibernate enables the flush operation during hibernate: the device initiates a WriteBooster Buffer flush operation whenever the link enters in the hibernate state.

bWriteBoosterBufferFlushStatus attribute provides the flush operation status.

bWriteBoosterBufferLifeTimeEst attribute indicates the estimated lifetime of the WriteBooster Buffer.

13.4.18.2 WriteBooster Configuration

Bit[8] of dExtendedUFSFeaturesSupport indicates if the device supports the WriteBooster feature. If the device does not support this feature, a query request that attempts to set a WriteBooster parameter in a Configuration Descriptor to a value different from zero shall fail, and the Query Response field in QUERY RESPONSE UPIU shall be set to "General Failure".

The WriteBooster Buffer can be configured in "LU dedicated buffer" mode or "shared buffer" mode according to the device capability. bSupportedWriteBoosterBufferTypes indicates which modes are supported by the device.

If bWriteBoosterBufferPreserveUserSpaceEn is set to 00h, the WriteBooster Buffer reduces the total user space that can be configured at provisioning. The amount of the reduction can calculated multiplying the WriteBooster Buffer size by the value indicated by bWriteBoosterBufferCapAdjFac. For example, the bWriteBoosterBufferCapAdjFac value for a TLC NAND storage device with a SLC NAND WriteBooster Buffer is 3; therefore, the total user capacitity that can be configured is reduced by 3 x WriteBoosterBufferCapacity.

Setting bWriteBoosterBufferPreserveUserSpaceEn to 01h avoids the reduction of the total user space that can be configured at provisioning, but it may result in lower performance, see 13.4.18.5.

13.4.18.2 WriteBooster Configuration (cont'd)

LU dedicated buffer mode

If the device supports the "LU dedicated buffer" mode, this mode is configured by setting bWriteBoosterBufferType to 00h. The logical unit WriteBooster Buffer size is configured by setting the dLUNumWriteBoosterBufferAllocUnits field of the related Unit Descriptor. Only a value greater than zero enables the WriteBooster feature in the logical unit. When bConfigDescrLock attribute is set to 01h, logical unit configuration can no longer be changed.

The maximum number of supported WriteBooster Buffers is defined in the bDeviceMaxWriteBoosterLU's parameter of the Geometry Descriptor. bDeviceMaxWriteBoosterLU is 01h, therefore the WriteBooster Buffer can be configured in only one logical unit.

Figure 13.8 and Figure 13.9 show an example of device configuration with a 2 GB WriteBooster Buffer.

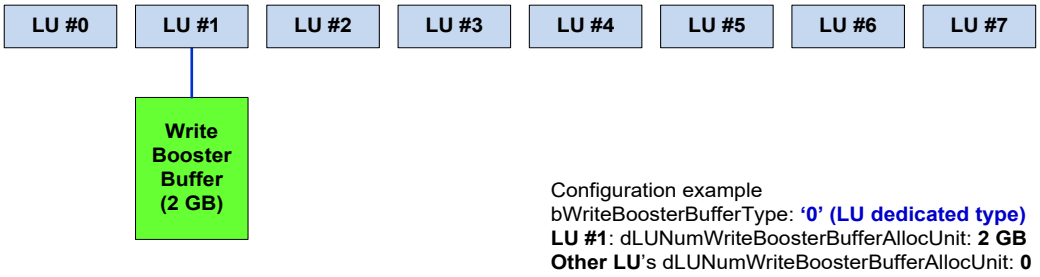


Figure 13.8 — Example of "LU Dedicated Buffer" Mode Configuration

The WriteBooster Buffer is available only for the logical units from 0 to 7 which are configured as "normal memory type" (bMemoryType = 00h) and "not Boot well known logical unit" (bBootLunID = 00h), otherwise the Query Request shall fail and the Query Response field shall be set to "General Failure".

Shared buffer mode

If the device supports the "shared buffer" mode, this mode is configured by setting bWriteBoosterBufferType to 01h. The WriteBooster Buffer size is configured by setting the dNumSharedWriteBoosterBufferAllocUnits field of the Device Descriptor.

NOTE The supported logical unit number is indicated by bMaxNumberLU.

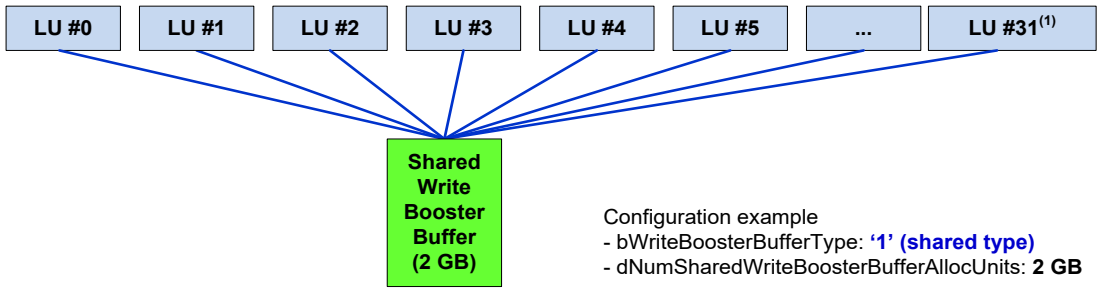


Figure 13.9 — Example of "shared buffer" Mode Configuration

NOTE If bWriteBoosterBufferType is set to 01h but dNumSharedWriteBoosterBufferAllocUnits is set to zero, the WriteBooster feature is disabled.

13.4.18.3 Writing Data to WriteBooster Buffer

If the fWriteBoosterEn flag is set to zero, data written to any logical unit is written in normal storage.

If the fWriteBoosterEn flag is set to one and the device is configured in “shared buffer” mode, data written to any logical unit is written in the shared WriteBooster Buffer.

If the fWriteBoosterEn flag is set to one and the device is configured in “LU dedicated buffer” mode, data written to the logical unit configured to use a dedicated buffer is written in the logical unit WriteBooster Buffer. Data written to any logical unit not configured to use a dedicated buffer is written in normal storage.

If Command-Level Data Write to WriteBooster Buffer is supported (see wExtendedWriteBoosterSupport in 13.4.22.2), a WRITE command with GROUP NUMBER field set to 17 indicates that the device shall process that specific WRITE command as though fWriteBoosterEn were set to one.

Writes to the WriteBooster Buffer may decrease the lifetime and the availability of the WriteBooster Buffer.

In the “LU dedicated buffer” mode, the device may write data from other LUs to the WriteBooster Buffer in case there are multiple pending commands while fWriteBoosterEn is set to one.

Whenever the endurance of the WriteBooster Buffer is consumed completely, a write command is processed as if WriteBooster feature was disabled. Therefore, it is recommended to set fWriteBoosterEn to one, only when WriteBooster performance is needed, so that WriteBooster feature can be used for a longer time.

In the shared buffer configuration, the data that may not need the performance of WriteBooster will be written to the WriteBooster Buffer when fWriteBoosterEn is one, there is higher probability to fill the WriteBooster Buffer and consume its endurance earlier.

If there is no available buffer, write data in WriteBooster mode will be stored in normal storage with normal write performance. The host can identify the available WriteBooster Buffer size by referring to the bAvailableWriteBoosterBufferSize attribute. This available buffer size is decreased by the WriteBooster operation and increased by the WriteBooster Buffer flush operation. The available buffer size can be increased by UNMAP commands.

The WriteBooster Buffer lifetime is indicated by the bWriteBoosterBufferLifeTimeEst attribute. If the value of bWriteBoosterBufferLifeTimeEst is equal to 0Bh (Exceeded its maximum estimated WriteBooster Buffer lifetime), a write command shall be processed as if the WriteBooster feature was disabled.

13.4.18.4 Flushing WriteBooster Buffer

When the entire buffer for WriteBooster is consumed, data will be written in normal storage instead of in the WriteBooster Buffer. The device informs the host when the WriteBooster Buffer is full or near full with the exception event WRITEBOOSTER_FLUSH_NEEDED, see 13.4.12.

The WRITEBOOSTER_FLUSH_NEEDED event mechanism is enabled by setting the WRITEBOOSTER_EVENT_EN bit of the wExceptionEventControl attribute.

There are two methods for flushing data from the WriteBooster Buffer to the normal storage: one is using an explicit flush command, the other enabling the flushing during link hibernate state. If the fWriteBoosterBufferFlushEn flag is set to one, the device shall flush the data stored in the WriteBooster Buffer to the normal storage. If fWriteBoosterBufferFlushDuringHibernate is set to one, the device flushes the WriteBooster Buffer data automatically whenever the link enters the hibernate (HIBERN8) state.

The time needed to flush the WriteBooster Buffer depends on the amount of data to be flushed. The bWriteBoosterBufferFlushStatus attribute indicates the status of the WriteBooster flush operation. The device shall execute the WriteBooster flush operation only when the command queue is empty. If the device receives a command while flushing the WriteBooster Buffer, the device may suspend the flush operation to expedite the processing of that command. Note that bWriteBoosterBufferFlushStatus will still indicate “Flush operation in progress” (01h) even if it has been temporarily suspended. After completing the host command, the device will resume flushing the data from the WriteBooster Buffer automatically. While the flushing operation is in progress, the device is in Active power mode.

The device shall stop the flushing operation if both fWriteBoosterBufferFlushDuringHibernate and fWriteBoosterBufferFlushEn are set to zero.

If the WriteBooster buffer becomes full while writing into a zone (see 13.4.23), a significant amount of data may have to be flushed before writing into the zone can continue. This can cause a latency spike. The initiator is responsible for avoiding these possible latency spikes.

13.4.18.5 User Space Modes

There are two user space configuration modes: “user space reduction” and “preserve user space”. With the “user space reduction”, the WriteBooster Buffer reduces the total configurable user space. While with the “preserve user space”, the total space is not reduced. However, the physical storage allocation may be smaller than total capacity of all logical units, since part of the physical storage is used for the WriteBooster Buffer.

When the physical storage allocated for the logical units is fully used, the device will start using the physical storage allocated for the WriteBooster Buffer. Therefore, the WriteBooster Buffer size may be less than what was initially configured. The current size of the WriteBooster Buffer can be discovered by reading the dCurrentWriteBoosterBufferSize attribute. Approximate available space in the WriteBooster Buffer can be calculated by multiplying the value of bAvailableWriteBoosterBufferSize (percentage of available WriteBooster Buffer) by the value of dCurrentWriteBoosterBufferSize (current WriteBooster Buffer size).

The bSupportedWriteBoosterBufferUserSpaceReductionTypes parameter of the Geometry Descriptor indicates which options are supported. Setting the bWriteBoosterBufferPreserveUserSpaceEn parameter of the Device Descriptor to 01h enables “preserve user space” mode.

13.4.18.5 User Space Modes (cont'd)

The disadvantage of the "preserve user space" mode is that there could be performance degradation when the physical storage used for the WriteBooster Buffer is returned to user space, since the device may adjust internal data structures as well as flush exising WriteBooster Buffer data. If the free storage is increased enough, the device may restore its decreased WriteBooster Buffer size from available physical storage. If the remaining storage is repeatedly increased and decreased, the WriteBooster Buffer could be repeatedly built from and returned to the user storage space and performance degradation may occur.

The amount of performance degradation due to returning the storage used for WriteBooster Buffer to user storage space is device specific. This is because the condition for migration between user space and the WriteBooster Buffer depends on device capacity and vendor's migration policy, which includes the granularity and frequency of migration, what percentage of free user space remains from which the migration starts, etc. For more details, see the device datasheet.

13.4.18.6 WriteBooster Buffer Resize

In preserve user space mode, without host involvement, the device may move allocation units between normal memory and WriteBooster Buffer based on the device's physical utilization. However, during these allocation unit changes, READ and WRITE commands may experience performance degradation. The host may initiate a Resize operation to prevent performance degradation. The device provides a hint to the host to recommend Resize operations, but the host selects the timing and type of Resize operations to be performed.

WriteBooster Buffer size can be decreased or increased during runtime when the host initates a WriteBooster Resize operation. This feature is supported only in preserve user space mode. If the WriteBooster Resize operation is initiated in user space reduction mode, Query Response shall return "General Failure"

As shown in Figure 13.10, the host initiates a Resize operation to increase or decrease the WriteBooster Buffer Size based on the information provided in the device's bWriteBoosterBufferResizeHint attribute. The host initiates a Resize operation by issuing a QUERY REQUEST UPIU with a write request for the bWriteBoosterBufferResizeEn attribute.

When there is a recommendation about resize for the WriteBooster Buffer, the device informs the host by the WRITEBOOSTER_RESIZE_HINT exception mechanism. It is enabled by setting the WRITEBOOSTER_RESIZE_HINT_EN bit of the wExceptionEventControl attribute.

The device may adjust the WriteBooster Buffer size when the command queue is empty. When size adjustment is completed, the device updates dCurrentWriteBoosterBufferSize. However, if the WriteBooster Buffer size cannot be increased or decreased due to device internal status, the WriteBooster Buffer size is not changed.

13.4.18.6 WriteBooster Buffer Resize (cont'd)

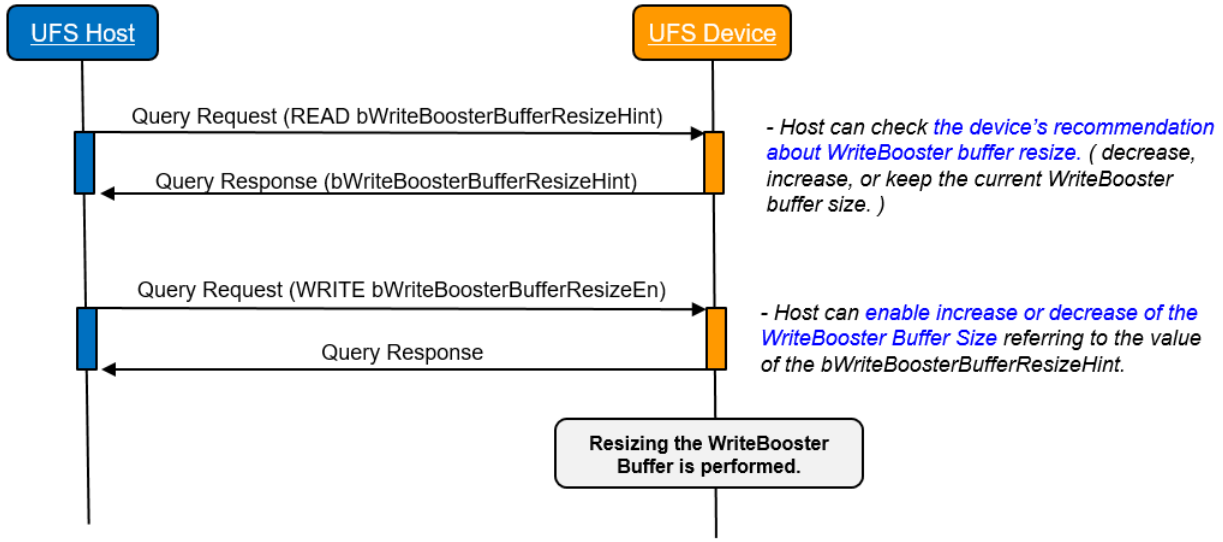


Figure 13.10 – (Example) Initiating a WriteBooster Buffer Resize Operation

As shown in Figure 13.11, the host can check the progress of the WriteBooster Buffer Resize operation by reading the bWriteBoosterBufferResizeStatus attribute. While a WriteBooster Resize is in progress, any QUERY REQUEST attempting to write bWriteBoosterBufferResizeEn shall not change the value nor interfere with the resize in progress and the device shall reply with a QUERY RESPONSE with Query Response of Success.

The amount of increase or decrease in WriteBooster Buffer is determined by the device. The host can check the updated WriteBooster Buffer size after completing those resize operation by reading the dCurrentWriteBoosterBufferSize attribute. The WriteBooster Buffer size shall not exceed the size indicated by the dNumSharedWriteBoosterBufferAllocUnits or dLUNumWriteBoosterBufferAllocUnits descriptor.

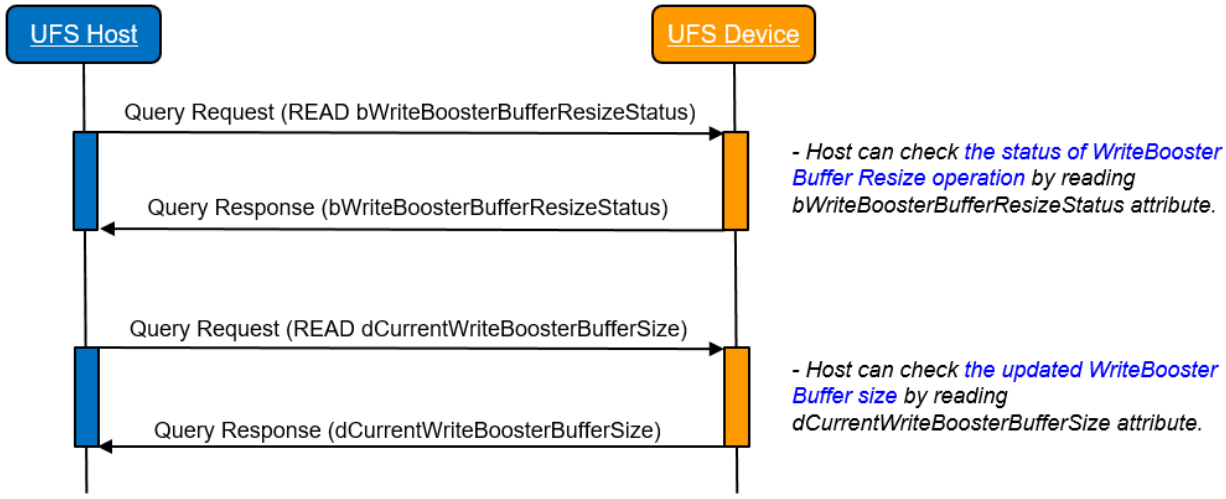


Figure 13.11 – (Example) Checking the Status of Resize Operation and Updated WriteBooster Buffer Size

13.4.18.7 Partial Flush Modes of WriteBooster Buffer

If the data is expected to be accessed soon or frequently, it is better to keep it in the WriteBooster Buffer for as long as possible in consideration of overall performance.

Using the partial flush mode, the host can indicate the data which is not to be moved out from the WriteBooster Buffer. The overall performance can be improved since higher speed access for those data can be continued.

There are two partial flush modes selected by the bWriteBoosterBufferPartialFlushMode attribute:

- FIFO(First-In First-Out) mode
- Pinned mode (supported only in preserved user space option of WriteBooster)

13.4.18.7.1 FIFO Partial Flush Mode

If bWriteBoosterBufferPartialFlushMode is 0x1, then FIFO mode partial flush mode is selected.

In the FIFO partial flush mode, the data written later are excluded from the WriteBooster flush operation, and the dCurrentFIFOSizeForWriteBoosterPartialFlushMode indicates the data size to be excluded from the WriteBooster flush operation.

The dMaxFIFOSizeForWriteBoosterPartialFlushMode value shall not be set to a value greater than dNumSharedWriteBoosterBufferAllocUnits or dLUNumWriteBoosterBufferAllocUnits. If the host attempts to write a value higher than what indicated by dNumSharedWriteBoosterBufferAllocUnits or dLUNumWriteBoosterBufferAllocUnits, the value shall not be changed and the QUERY RESPONSE UPIU shall have Query Response field set to “Invalid VALUE”.

13.4.18.7.2 Pinned Partial Flush Mode

If bWriteBoosterBufferPartialFlushMode is 0x2, then Pinned mode partial flush mode is selected.

In the Pinned Partial flush mode, the data of WRITE command with GROUP NUMBER set to 18h is intended to be written to the Pinned WriteBooster Buffer, and the data written in the Pinned WriteBooster Buffer is excluded from WriteBooster flush operation.

The Pinned WriteBooster Buffer size is indicated by dPinnedWriteBoosterBufferNumAllocUnits.

The minimum size recommendation for Non-Pinned WriteBooster Buffer area in Pinned Partial Flush Mode is indicated by dNonPinnedWriteBoosterBufferMinNumAllocUnits. Even if the WriteBooster buffer size is reduced, the size of the non-pinned area is recommended to be maintained larger than dNonPinnedWriteBoosterBufferMinNumAllocUnits size if possible. For example, if only the non-pinned buffer of this size remains in the device and the WriteBooster buffer size needs to be further reduced, the pinned data area is returned first to maintain the non-pinned WB buffer size indicated by dNonPinnedWriteBoosterBufferMinNumAllocUnits as long as possible.

13.4.18.7.2 Pinned Partial Flush Mode (cont'd)

The sum of

dPinnedWriteBoosterBufferNumAllocUnits value + dNonPinnedWriteBoosterBufferMinNumAllocUnits value

shall not be set to a value greater than

dNumSharedWriteBoosterBufferAllocUnits, or dLUNumWriteBoosterBufferAllocUnits.

If the host attempts to write a value higher than what is indicated by

dNumSharedWriteBoosterBufferAllocUnits or dLUNumWriteBoosterBufferAllocUnits,

the value shall not be changed and the QUERY RESPONSE UPIU shall have Query Response field set to "Invalid VALUE".

When the WriteBooster Buffer size decrease, the Pinned WriteBooster Buffer size can be decreased less than the size indicated by the dPinnedWriteBoosterBufferNumAllocUnits.

Therefore, the host can check the currently allocated size for pinned WriteBooster Buffer by reading the dPinnedWriteBoosterBufferCurrentAllocUnits attribute. The currently available portion in the currently allocated Pinned WriteBooster Buffer is indicated by the bPinnedWriteBoosterBufferAvailablePercentage attribute.

If the Pinned WriteBooster Buffer is not available(i.e full or not configured), even data of WRITE command with GROUP NUMBER set to 18h can be written to the Non-Pinned WriteBooster Buffer area, or to the normal storage if the Non-Pinned WriteBooster Buffer area is also not available(i.e full or not configured).

When the pinned WriteBooster buffer is full, the device notifies the host with the PINNED_WRITEBOOSTER_BUFFER_FULL exception mechanism. It is enabled by setting the PINNED_WRITEBOOSTER_EVENT_EN bit of the wExceptionEventControl attribute.

When the WriteBooster Buffer size is increased by the WriteBooster Buffer Resize operation or the WriteBooster Buffer size is implicitly increased in user space preservation mode, the Pinned WriteBooster Buffer cannot be increased to exceed the size indicated by the dNumSharedWriteBoosterBufferAllocUnits or dLUNumWriteBoosterBufferAllocUnits descriptor.

When the WriteBooster Buffer size is decreased, either or both Pinned Buffer size and non-pinned Buffer size can be decreased depending on the device's internal status.

When the WriteBooster Buffer size is decreased, the WriteBooster Buffer for the amount of non-pin data indicated by dNonPinnedWriteBoosterBufferMinAllocUnits remains as long as possible and shall be reduced at the end.

The pinned data is released by setting the fUnpinEn flag as 0x1 or by changing the bWriteBoosterBufferPartialFlushMode to 0x0(No partial flush mode) or 0x1(FIFO partial flush mode). The released pinned data is flushed by WriteBooster flush operation.

The cumulative amount written to the Pinned WriteBooster Buffer is reported by the dPinnedWriteBoosterCumulativeWrittenSize attribute.

13.4.19 Host Initiated Defragmentation

13.4.19.1 Overview

In storage devices, the fragmentation of Logical-to-Physical (L2P) mapping is inevitable because of the NAND flash-based memory's inherent structure and it can affect the performance and device lifetime. Therefore, NAND-based conventional memory devices, including UFS devices, may include memory maintenance mechanisms such as Garbage Collection to reduce fragmentation and secure the free space in the device.

This Host Initiated Defrag (HID) feature consists of the explicit analysis of storage and explicit execution of defragmentation which, for example, physically collects data distributed in many NAND blocks into one or a small number of NAND blocks in NAND-based memory device.. The host has a better overview of the optimal time for defragmentation, therefore a more effective memory maintenance can be done if the HID operations are initiated by the host.

This Host Initiated Defragmentation feature allows the host to check the status of whether defragmentation is needed or not and enable the device's internal defrag operation explicitly. I.e. host can enable or disable defragmentation analysis or defragmentation execution.

13.4.19.2 Supportability

The supportability of the Host Initiated Defrag (HID) feature is indicated by the Bit [13] of dExtendedUFSFeaturesSupport in Device Descriptor.

If this feature is not supported, all Query Requests related HID features shall fail and the Query Response field shall be set to "Invalid IDN".

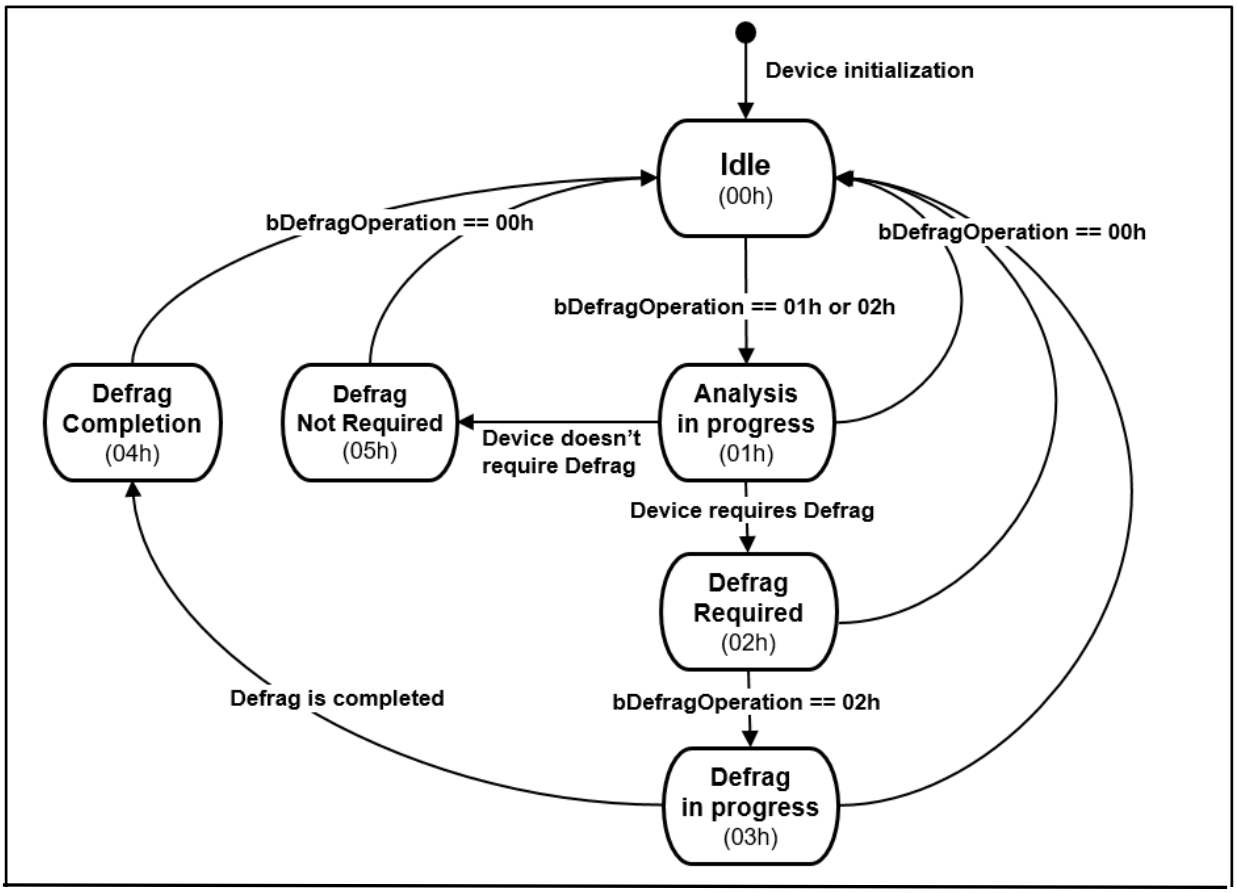
13.4.19.3 Checking the Need or Status of Analysis or Defragmentation

The host can read the bHIDState attribute to check whether the HID analysis operation or the HID Defrag operation is required or in progress.

- bHIDState attribute
  - Read Only Attribute
  - This attribute can be set to one of the following values:
    - 00h: Idle (default mode after device initialization. Analysis operation is needed because there is no valid fragmentation status information of the device.)
    - 01h: Analysis in Progress
    - 02h: Defragmentation is Required
    - 03h: Defragmentation is In Progress
    - 04h: Defragmentation is Completed
    - 05h: Defragmentation is Not Required
    - Other values are reserved.

Figure 13.12 shows a state transition example that describes the state of the bHIDState attribute and the transition between the states. The bHIDState attribute may be updated when the device receives any write-type (WRITE, Query Requests (WRITE Request), FORMAT UNIT, UNMAP, WRITE BUFFER) command that changes medium status. The bHIDState attribute is updated when the host sets bDefragOperation to operate HID analysis or defrag.

13.4.19.3 Checking the Need or Status of Analysis or Defragmentation (cont'd)



NOTE If any UFS command for changing the media status of a UFS device is received during HID analysis operation or HID defrag operation processing, the processing HID operation may be terminated. The host can read from time to time the value of bHIDState to verify the operation was terminated or the device managed to resume the HID operation.

Figure 13.12 – (Example) Transition of bHIDState Attribute

- The default value, i.e. after power on or reset, of the bHIDState attribute is 00h (idle state).
- When bDefragOperation (00h) is received from any state, it transitions to Idle (00h) State.
- If the host sets bDefragOperation to 01h (HID analysis is enabled) or 02h (HID analysis and HID defrag are enabled), then the bHIDState attribute shall be updated to 01h (Analysis is in Progress).
- After the device completed HID analysis, the device shall update bHIDState to 02h (Defrag Required) or 05h (Defrag Not Required) depending on whether the HID defrag operation is needed or not.
- If the bHIDState is 02h(Defrag Required) and bDefragOperation is 02h, the device shall update bHIDState to 03h (Defrag in Progress) to start the Defrag operation. After completing the HID Defrag operation, the device shall update bHIDState to 04h (Defrag Completion).

13.4.19.3 Checking the Need or Status of Analysis or Defragmentation (cont'd)

After the host reads the bHIDState value when it is 04h (Defrag Completion) or 05h (Defrag Not Required), the following parameters shall be initialized:

- bHIDState value to 00h (Idle)
- bHIDProgressRatio value to 00h (0%)
- bDefragOperation value to 00h (HID operations are disabled)
- dHIDAvailableSize value to FFFFFFFFh (indicating no valid information available about the fragmented size).

Additionally, after the host reads bHIDProgressRatio when it is 64h (100%), all the above parameters shall be initialized together.

13.4.19.4 Size to be Defragmented

After completing the HID analysis operation, the host can check the total fragmented size in the device by reading dHIDAvailableSize attribute.

- dHIDAvailableSize attribute
  - The attribute indicates the size of all fragmented data to be moved to completely defragment all data during the HID defrag operation.
  - The attribute set to be a multiple of 4 KB unit.
  - The attribute has Read Only access property (i.e. updated only by the device, and cannot be written by the host.)
  - The attribute will be set to 0xFFFFFFFFh when HID analysis is needed.
  - The attribute is updated only when the device has completed the HID analysis operation initiated by the host.

The defrag execution time for a large dHIDAvailableSize value may take a long time and affect I/O. To avoid doing all defragmentation during a single operation, the host sets dHIDsize to limit the amount of data the device should move during one defragmentation operation.

- dHIDSize attribute
  - The attribute is set by the host to configure the size to be defragmented by next defrag operation.
  - The attribute set to be a multiple of 4 KB unit.
  - The default value is 0xFFFFFFFFh indicating that the device defrag as much as possible.
  - The attribute has Persistent access property (i.e. value is kept after the power cycle or any type of reset event.)
  - The value indicates the size to be defragmented by an HID defrag operation.
  - The host can set the value of this attribute less than, or equal to the value of dHIDAvailableSize.
  - If the host set dHIDSize bigger than dHIDAvailableSize, then the device will perform defragmentation as much as the amount indicated by dHIDAvailableSize.

13.4.19.5 Enables and Disable HID Analysis or HID Defrag Operations

The HID analysis and the HID defrag operations are implemented using Query Functions and related Attributes. In particular, the bDefragOperation attribute can enable or disable HID analysis and HID defrag operations.

- bDefragOperation attribute
  - This attribute has one of the following values:
    00h: HID operations (i.e. HID analysis and HID defrag) are disabled
    01h: HID analysis is enabled
    02h: HID analysis and HID defrag are enabled
    Other values are reserved.
  - If the host sends a Query Request to set the bDefragOperation attribute to 01h or 02h so that HID analysis or HID defrag can be performed while the device command queue is not empty, the Query Request shall be processed after completing all previous commands in the command queue. The device will respond to the QUERY request but will not start the analysis till after the queue is empty. When the command queue is empty and the HID operation can be performed, the device performs the HID operation based on the most recently received bDefragOperation value.
  - This attribute can be set to 0 by the host to stop the ongoing HID analysis or HID defrag operation.
  - This attribute is automatically set to 0 by the UFS device when the operation is completed.
  - If any UFS command for changing the media status of a UFS device is received during HID analysis operation or HID defrag operation processing, the processing HID operation may be terminated and the value of bDefragOperation, bHIDProgressRatio, and bHIDState are initialized to 0. The host can read from time to time the value of bHIDState to verify the operation was terminated or the device managed to resume the HID operation
  - If the host enables the HID analysis operation or HID defrag operation during the purge operation, the HID analysis or HID defrag request shall fail and the Query Response field in the QUERY RESPONSE UPIU shall be set to 0xFFh ("General Failure").
  - This attribute can be set only after the UFS initialization phase (i.e. fDeviceInit cleared to zero by UFS device). If a Query Request to write bDefragOperation attribute is issued before the initialization phase is completed, the request shall fail and the Query Response field in the QUERY RESPONSE UPIU shall be set to 0xFFh (“General Failure”).

When the host sets the bDefragOperation attribute to 01h, the device starts the analysis to calculate the total fragmented amount and reports the result as the dHIDAvailableSize attribute.

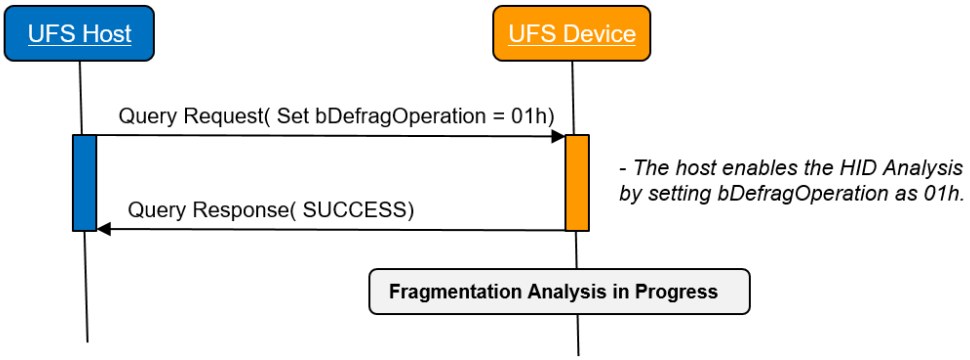


Figure 13.13 – (Example) HID Analysis Operation

13.4.19.5 Enables and Disable HID Analysis or HID Defrag Operations (cont'd)

When the HID analysis is completed, the device updates dHIDAvailableSize to indicate the total fragmented size in the device. The host can set dHIDSize attribute to set the size to be defragmented by an HID defrag operation.

When the bDefragOperation attribute is set to 02h, the device shall start executing the HID for the defragmentation.

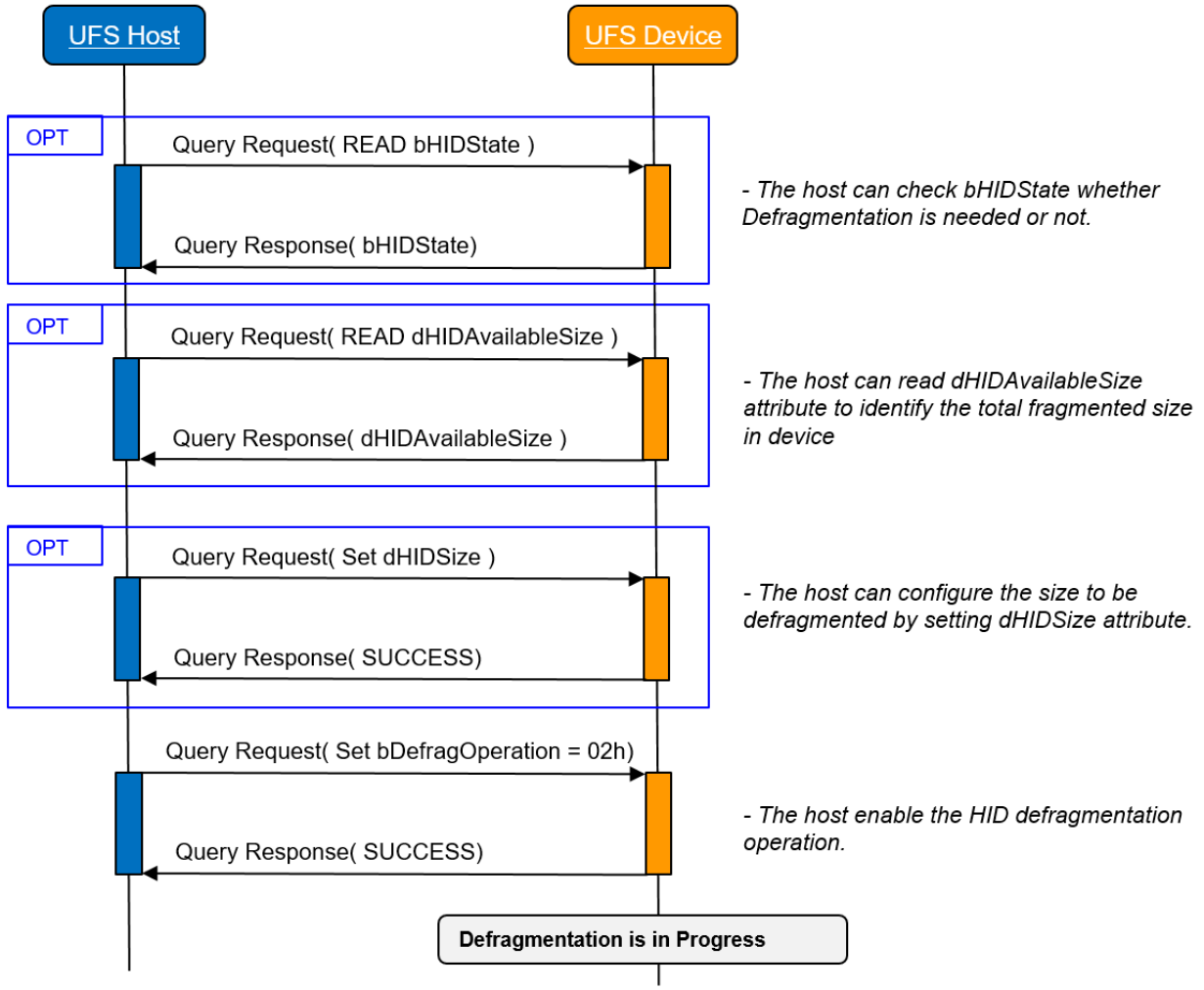


Figure 13.14 – (Example) HID Defrag Operation

13.4.19.5 Enables and Disable HID Analysis or HID Defrag Operations (cont'd)

When the bDefragOperation attribute is set to 02h in Idle state, the device performs the HID analysis operation and then starts executing defrag operation only if the analysis result is Defrag is Required. Note that if the device has already been analyzed, the device can start HID defrag operation without additional HID analysis operation.

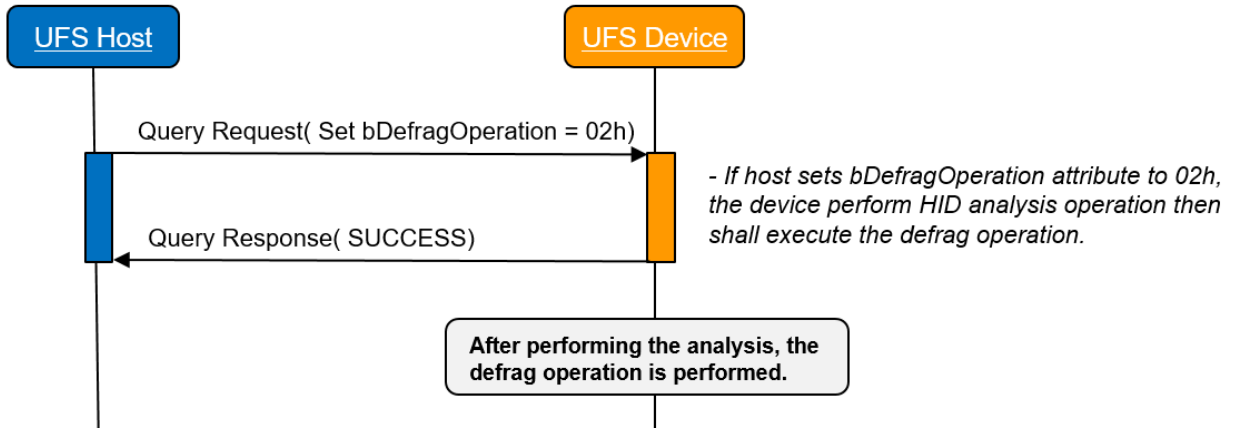


Figure 13.15 – (Example) HID Analysis & HID Defrag Operation

13.4.19.6 Monitoring the Progress of the Defrag Operation

The host can check defrag progress by reading the bHIDProgressRatio attribute.

This attribute indicates the ratio of the completed defragmentation size over the requested defragmentation size. The requested defragmentation size is calculated as the minimum value of the size indicated by dHIDSize and the size indicated by dHIDAvailableSize.

- bHIDProgressRatio attribute
  - The attribute has Read Only access property.
  - The value of the attribute is expressed in units of 1%.
  - The attribute is updated while the device performs a defrag operation. If the device completes the defrag operation, the value of this attribute shall be set to 64h (i.e. to indicate the 100% completion of the requested defragmentation size).

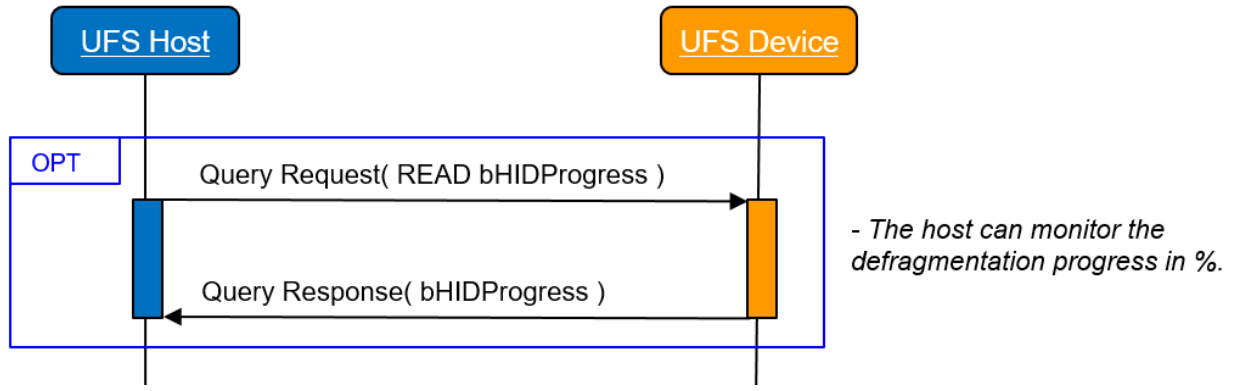


Figure 13.16 – (Example) Monitoring the Progress of Defragmenting

13.4.20 Fast Recovery Mode

13.4.20.1 Overview

Typically, the UFS host waits for the device's response to the requested command based on its timeout policy. This is not a one-time situation but is repeated multiple times to check the device status and proceed with the recovery procedure, resulting in a long wait time for the host side.

13.4.20.2 Introduction

To ease this burden, Fast Recovery Mode can quickly notice issue status of the device and request a device HW Reset from the host. The host can quickly perform a device HW Reset based on device hints and internal policy, resulting in fast recovery. It also provides the device idle time to perform background operations before the host initiates a HW Reset, reducing the waiting time for the host in situations where a HW Reset is necessary due to the device being unresponsive. The host can choose whether to utilize Fast Recovery Mode or proceed with the existing recovery process.

13.4.20.3 Implementation

When a non-recoverable hardware error occurs on the device due to internal or external issues while the device is performing an operation, the device will no longer respond and operate normally. The device determines that any operation cannot be completed and decides whether to send a hint requesting a HW Reset through RESPONSE UPIU to the host before losing its operation. After the host notices the device's internal status by checking FAST_RECOVERY_NEEDED bits of Device Information field in Response UPIU, the host also can determine how much time the device needs before the host proceeds with a HW Reset. The host determines whether to proceed with a HW Reset based on the device's hint or its own recovery policy.

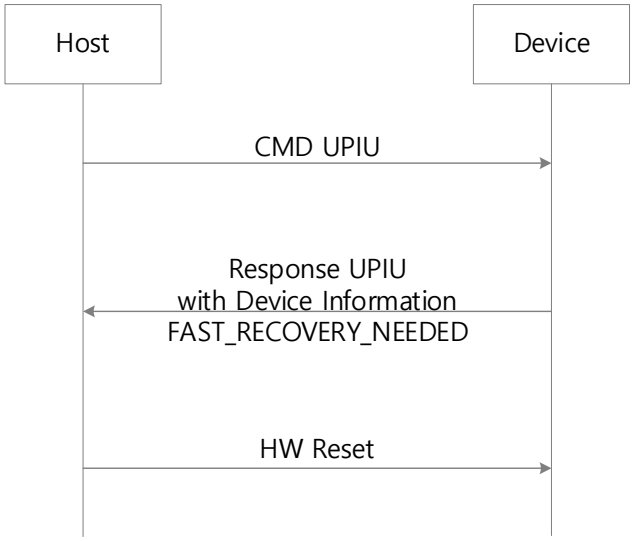


Figure 13.17 – FAST_RECOVERY_NEEDED from RESPONSE UPIU

13.4.21 Copy Offloading

The 3PC bit in the INQUIRY response informs host software whether or not copy offloading is supported. The Supported Commands third-party copy descriptor information tells the host which third-party copy commands are supported.

If the EXTENDED COPY command is supported, a single command is sufficient to offload a copy operation.

Copying data with the POPULATE TOKEN, RECEIVE ROD TOKEN INFORMATION and WRITE USING TOKEN commands happens as follows:

- List identifiers A and B are selected that are not in use by any active copy command.
- A POPULATE TOKEN command is submitted with list identifier A.
- After the POPULATE TOKEN command completes, a RECEIVE ROD TOKEN INFORMATION command is submitted with list identifier A, retrieving ROD C.
- After the RECEIVE ROD TOKEN INFORMATION command completes, a WRITE USING TOKEN command with list identifier B and ROD C is submitted.
- The host waits for the WRITE USING TOKEN command to complete.
- If the IMMED bit was set in the WRITE USING TOKEN command, the host polls for completion of the copy command with the RECEIVE ROD TOKEN INFORMATION command and list identifier B.
- The host receives the copy operation status with the RECEIVE ROD TOKEN INFORMATION command and list identifier B.
- List identifiers A and B are marked as not in use. Information about ROD C is discarded by the host.

13.4.22 Pre-Erase

13.4.22.1 Overview

Typically, a UFS device uses physical blocks by erasing first and then writing because erased blocks may only be stable for a limited time. As a result, when a sequential write needs to write to a new block, there may be additional latency (usually ms-level) while waiting for the erase.

When there is a large amount of data to be written, the Pre-Erase feature allows the host to notify the device to erase some physical blocks in advance. This can save block erase time during the writes, thereby improving sequential write performance and optimizing latency. The Pre-Erase feature is enabled only when WriteBooster is enabled, and this feature is applicable only for SLC blocks of WriteBooster.

13.4.22.2 Supportability

The supportability of the Pre-Erase feature is indicated by the bit[3] of wExtendedWriteBoosterSupport in Device Descriptor.

If this feature is not supported, all Query Requests related Pre-Erase features shall fail and the Query Response field shall be set to “Invalid IDN”.

13.4.22.3 Attributes related to Pre-Erase

The Pre-Erase operation is implemented using Query Functions and related Attributes. In particular, the bPreEraseSetCap attribute can be set to 0 by the host to disable the Pre-Erase feature.

- bPreEraseSetCap
  - This attribute indicates the capacity set by the host in GB to be erased by the device.
  - When this attribute is set, the device will pre-erase SLC blocks in idle time.
  - The value shall not exceed bPreEraseMaxCap.
  - If the host attempts to set bPreEraseSetCap＞bPreEraseMaxCap, then the device shall return 0xFA.
- bPreEraseCurCap
  - This attribute indicates the block capacity has been pre-erased in the device.
- bPreEraseMaxCap
  - This attribute indicates the max Pre-Erase capacity in GB can be set by the host.
  - The maximum value shall be equal to or less than the available WriteBooster Buffer size.

13.4.22.4 Pre-Erase Operation

If bit[3] of wExtendedWriteBoosterSupport of the Device Descriptor indicates the Pre-Erase feature is supported, the host can start the Pre-Erase operation.

Step 1: The host gets the pre-erased block capacity by reading bPreEraseCurCap.

Step 2: After the device responds with the pre-erased block capacity to the host, the host calculates the erase size and set the pre-erase size by sending the bPreEraseSetCap to the device.

Step 3: The device evaluates whether the pre-erase capacity can be satisfied based on its free capacity.

Step 4: If satisfied, the device will erase the blocks in idle time.

Step 5: When the WRITE commands arrive, the data can be directly written to the device without waiting the erase operation.

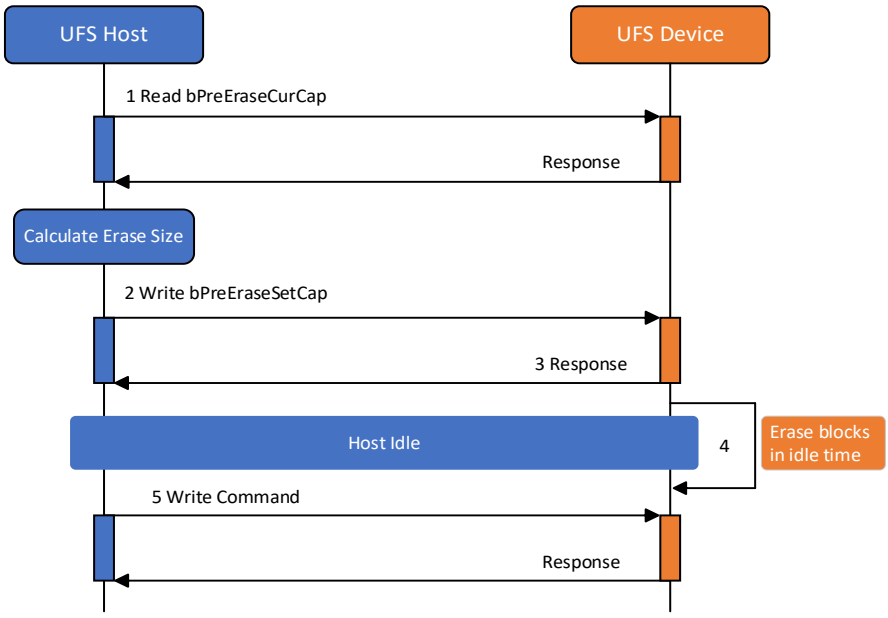


Figure 13.18 – Pre-Erase Operation

13.4.23 Zoned Logical Units

A UFS device declares its support for the Zoned Logical Units feature in the Zoned logical units bit of the Device Descriptor's dExtendedUFSFeaturesSupport field.

13.4.23.1 Overview

The purpose of the Zoned Storage for UFS standard is to enable higher bandwidth, lower latency and to reduce write amplification. These objectives are realized as follows:

- Zoned storage enables reducing the size of the L2P table for zoned logical units. If the L2P table does not fit in UFS device SRAM without using the zoned storage interface then the switch to zoned storage may make it possible to eliminate L2P paging and hence increases bandwidth and reduces latency.
- It is recommended that vendors make the zone size close to the size of the erase block size. "Close" means that the zone size is identical to the erase block size, a small integer multiple of the erase block size or that there are two or more zones per erase block. Device-side garbage collection and write amplification due to garbage collection are eliminated if the zone size is equal to or a multiple of the erase block size since garbage collection is moved from the UFS device to the host. If there are two or more zones per erase block then the UFS device still has to perform garbage collection.
- Host software can optimize read performance by allocating a contiguous LBA range per file.

Support for zoned logical units is based on the [ZBC] standard.

13.4.23.2 Zoned Block Device Model

The data on a block device is organized in logical blocks. A logical block is a contiguous range of bytes that is accessed and referenced as a unit. Each logical block has an integer index called the LBA. The logical blocks of a zoned block device are divided into zones. Each zone contains a contiguous range of logical blocks. Each logical block is included in exactly one zone.

Each zone has the following properties:

- A type. This standard only supports the sequential write required zone type.
- A zone condition: empty, implicit open, closed or full.
- A write pointer.

All zones across all zoned logical units shall have the same length. See also the qZoneSize parameter in the geometry descriptor.

A zoned UFS logical unit reports PERIPHERAL DEVICE TYPE = 14h in the INQUIRY response. All zones shall have the sequential write required zone type.

More information about zone types, zone conditions and the write pointer concept is available in [ZBC]. More information about the PERIPHERAL DEVICE TYPE field and the INQUIRY command are available in the UFS standard.

13.4.23..3 Open Zones

If a UFS device supports the Zoned logical unit feature (see dExtendedUFSFeaturesSupport), that device shall support at least 6 zones that are either open or closed. If host software opens a zone if this limit is met then this may result in reduced performance.

13.4.23.4 Logical Units

If a UFS device supports the Zoned logical unit feature (see dExtendedUFSFeaturesSupport), in addition to conventional and well-known logical unit configurations it shall also support zoned logical unit configurations. Zoned logical units are logical units that consist of multiple zones. Each zone is a consecutive range of LBAs and has the sequential write required zone type. All zones shall have the same size across all zoned logical units.

Writing data to all zones shall be supported.

For all logical unit types, the UFS device is responsible for wear leveling, bad block replacement and retrying reads if necessary.

A UFS device is not required to perform garbage collection for zoned logical units except if there are multiple zones per erase block.

13.4.23.5 Zoned Storage and Log-Structured File Systems (Informative)

This sub-clause is informational and not normative.

A log-structured file system is a file system that writes all modifications to disk sequentially in a log-like structure. New data is appended at the end of the log instead of overwriting data in place. Space occupied by logical blocks that hold data that is no longer current is reclaimed during a process called garbage collection.

Log-structured file systems are well suited for zoned storage. If data is written into multiple files simultaneously, the data from these files will be interleaved in the log. In other words, data from multiple files may end up in a single zone. During the garbage collection process the layout of files is optimized. Files that occupy dis-contiguous LBA ranges are transferred to a contiguous LBA range. It is the responsibility of the file system to perform garbage collection in such a way that it does not cause latency spikes for host software. See [1].

13.4.23.6 Sequential Write Required Zone Model, Processing Coordination

If the device supports SWR Processing Coordination, the device accepts write commands and processes each zone write command when its starting LBA is equal to the zone’s write pointer. If the device cannot process a write command with LBA equal to the zone’s write pointer, error handling is defined in [ZBC].

This feature is optional.

A UFS device indicates support by setting a flag in the dExtendedUFSFeaturesSupport parameter of the Device Descriptor.

13.5 UFS Cache

Cache is a temporary storage space in a UFS device. The cache should in typical case reduce the access time (compared to an access to the medium) for both write and read. The cache is not directly accessible by the host but is a separate element in a UFS device. This temporary storage space may be utilized also for some implementation specific operations like as an execution memory for the memory controller and/or as storage for an address mapping table etc. but which definition is out of scope of this standard.

The implementation of the cache is optional for the UFS devices but the related commands shall be implemented so that compatibility with host software driver is seamless independent from the implementation. Devices may explicitly indicate that cache is supported by setting INQUIRY Data VPD page (see [SPC]) parameter V_SUP = 1b. V_SUP = 0b means that there may or may not be cache implemented. INQUIRY Data VPD page parameter NV_SUP shall be set to 0b.

The cache is a device level cache and applies for all LUs generically. Data written to and read from a Boot W-LU and RPMB W-LU shall not be cached due to the specific nature of these LUs.

The cache is expected to be volatile by nature. Data in the cache is not expected to remain data valid over power cycles or HW/SW resets. The UFS device is expected to manage the cache so that it shall not be possible to read stale data from the device.

While the cache is implemented the device server may utilize the cache during write and read operations for storing data which an application client may request later. The algorithm to manage the cache is out of scope of this standard and is left for the implementation. There are parameters related to the management of the cache defined in CDBs (see the following bullets) and in 11.4.2.3, Caching Mode Page.

- The disable page out (DPO) bit in the CDB of write, read and verify commands allows the application client to influence the replacement of the logical blocks in the cache (e.g., in case the cache is full). Setting the DPO bit to 1 means that the device server should not replace the existing logical blocks in the cache with the new logical blocks written or read. When the DPO and FUA bits are set to one, write and read operations effectively bypass the cache.
- The force unit access (FUA) bit in the CDB of write and read commands enables the application client to access the medium. Setting the FUA bit to 1 means that the device server shall perform the write to the medium before completing the command and to read logical blocks from the medium (not from the cache).

During write operations the device server may use the cache to store data that is to be written to the medium at a later time (write-back caching) and thus the command may complete prior to logical blocks being written to the medium. This means also that such data may get lost if sudden power loss or reset occurs. There is also possibility of an error occurring during the actual write operation to the medium later. If an error occurred during such write operation it may be reported as a deferred error on a later command.

An UNMAP operation or any other operation which affects data of a logical block in the medium shall cause the device server to update potential data related to the logical block in the cache accordingly.

It is recommended that the host synchronizes the cache before initiating a PURGE operation.

When a VERIFY command is processed, both force unit access and synchronize cache operation are implied.

13.5 UFS Cache (cont'd)

Following commands shall be implemented by the device server to enable application client to control the behavior of the cache:

- PRE-FETCH commands: see 11.3.17 and 11.3.18.
- SYNCHRONIZE CACHE commands: see 11.3.22 and 11.3.23.

13.6 Production State Awareness (PSA)

13.6.1 Introduction

UFS device can utilize knowledge about its production status and adjust internal operations accordingly. For example, content which was loaded into the storage device prior to device soldering might be corrupted, at a higher probability than in regular mode. The UFS device could use "special" internal operations for loading content prior to device soldering which would reduce production failures and use "regular" operations post-soldering.

The sensitivity for device soldering is a property of the logical unit, some logical units may be sensitive to device soldering while some logical units may not be sensitive to it. Before loading the data to the device the host should read bPSASensitive, to identify the LU which are sensitive to device soldering.

Pre-loaded data is data which is loaded on the device after device configuration is completed (bConfigDescrLock is set and reset of the device), and before device soldering to the host platform. The combined maximum amount of data which could be pre-loaded to all sensitive LUs is device specific and defined by dPSAMaxDataSize attribute.

13.6.2 PSA flow

PSA feature is based on the device capability to uniquely identify which data is written before the soldering process. The PSA flow may be initiated only if all LBAs in logical units with bPSASensitive = 01h are unmapped. In case the host does not know if LBAs are unmapped, then it should set bPSAState 'Off', send an UNMAP command for the entire LBA range of each LU with bPSASensitive set to 01h, to unmap that data and re-start the PSA flow as described in the following.

Depicted in Figure 13.19 is the PSA flow. To start the PSA flow, the host first checks if PSA feature is supported by the device (see bUFSFeaturesSupport parameter in Device descriptor).

The host is expected to set dPSADataSize to indicate amount of data it plans to pre-load in all logical units with bPSASensitive = 01h. In case the host tries to set dPSADataSize > dPSAMaxDataSize, then the device shall return a General failure error.
The host writes bPSAState attribute to 'Pre-soldering' and then pre-loads the data in the logical units through WRITE commands.

The host should count towards dPSAMaxDataSize limit only data written through a WRITE command to a LU with bPSASensitive descriptor set to '1'. During PSA flow the host is not expected to write data to the same LBA more than once, in such case device behavior may be undefined.

13.6.2 PSA Flow (cont'd)

Once the host finishes pre-loading all LU (wrote total of dPSADataSize amount of data) the host is expected to change the state of bPSAState from 'Pre-soldering' to 'Loading Complete' to indicate to the device that pre-loading of data is complete.

Prior to soldering, at 'Loading Complete' bPSAState state, device may stop using special internal operations and resume regular operations. Therefore, the host should not write data to the device as data may be corrupted during soldering; a WRITE Command in this situation may result in an error.

After the setting of bPSAState to 'Loading Complete', the device may be soldered. The device shall set bPSAState to 'Soldered' during the processing of the first WRITE command after a power-up occurred with bPSAState = 'Loading Complete'.

13.6.2 PSA Flow (cont'd)

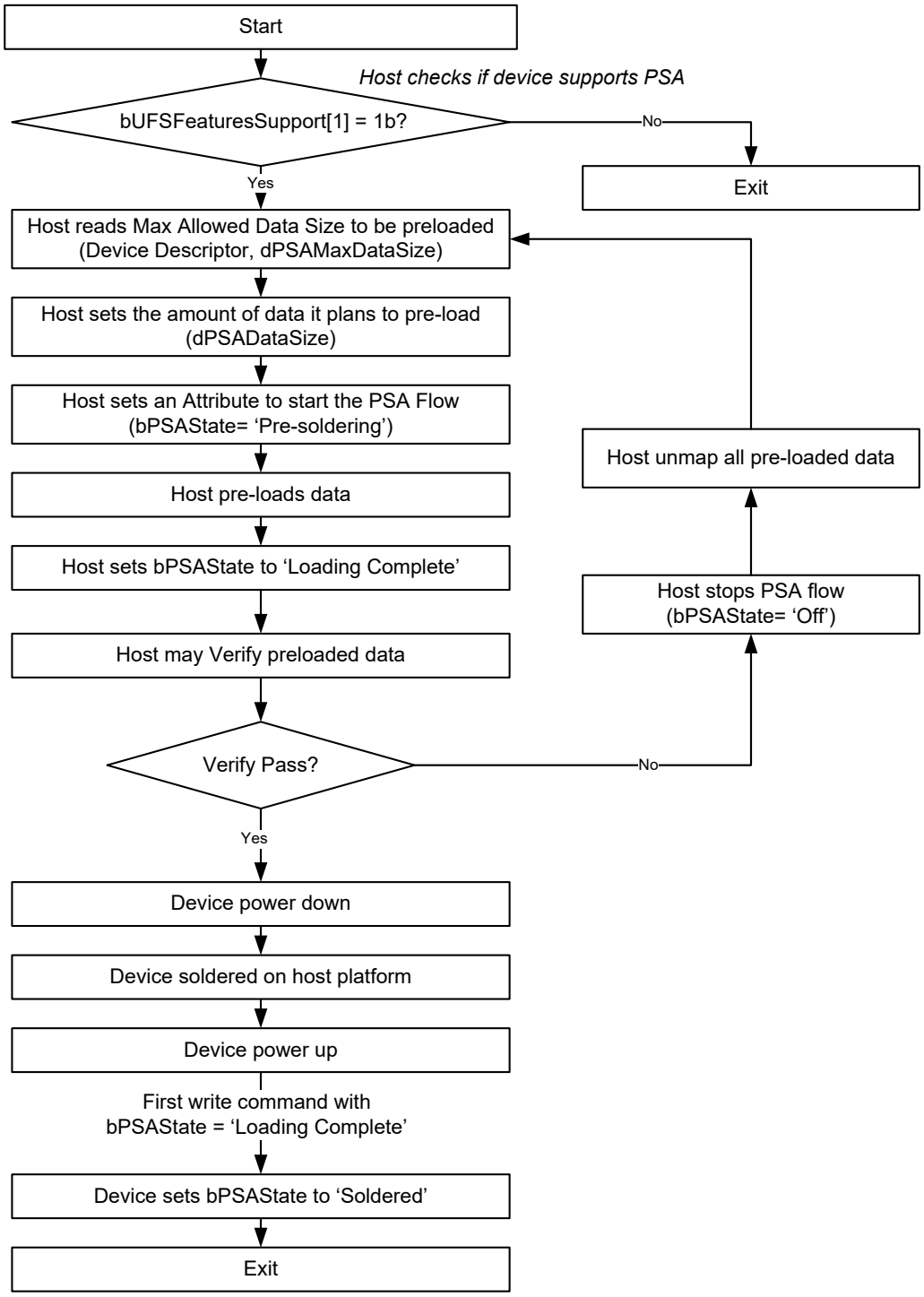


Figure 13.19 — PSA Flow

13.6.2 PSA flow(cont'd)

Depicted in Figure 13.20 is the PSA state machine which describes the different states of the bPSAState attribute and the transitions between its states.

Host misbehavior of writing, during pre-soldering phase, more data than indicated by dPSADataSize may result in data corruption during device soldering.

At any time before setting bPSAState to 'Soldered' the host may re-start the PSA flow by switching bPSAState to 'Off', unmap all sensitive data and set bPSAState to 'Pre-soldering'.

A change in bPSAState attribute may involve additional operations by the device which may require some time. bPSAStateTimeout indicates the maximum allowed timeout in which the device may return a response.

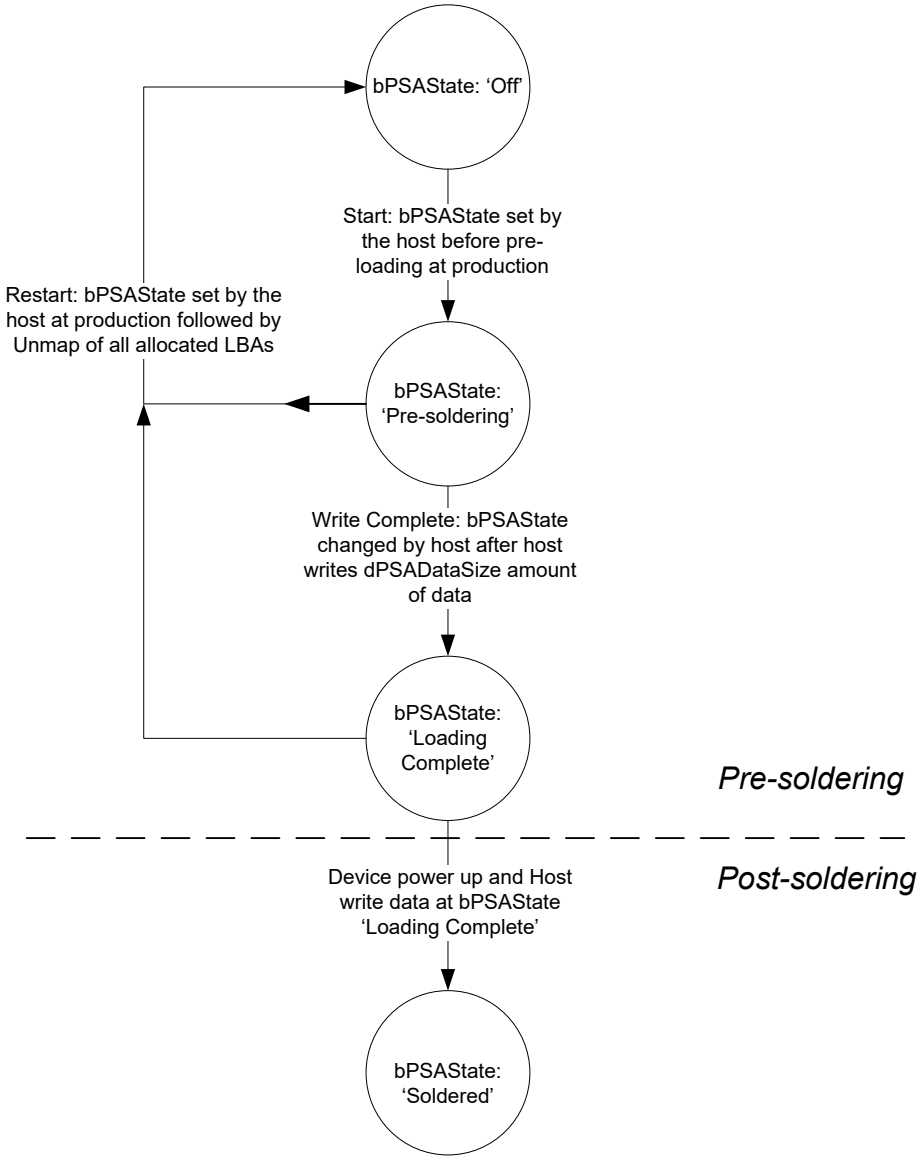


Figure 13.20 — PSA State Machine

14 UFS Descriptors, Flags And Attributes

14.1 UFS Descriptors

A descriptor is a data structure with a defined format. Descriptors are accessed via QUERY REQUEST UPIU packets. Descriptors are independently addressable data structures. Descriptors may be stand alone and unique per device or they may be interrelated and linked in a hierarchical fashion to other descriptors by parameters defined within the top-level descriptor. Descriptors can range in size from 2 bytes through 255 bytes. A 2 byte descriptor is an empty descriptor. All descriptors have a length value as their first element. This length represents the entire length of the descriptor, including the length byte. All descriptors have a type identification as their second byte. If a parameter in a Descriptor has a size greater than one byte, the byte containing the MSB is stored at the lowest offset and the byte containing the LSB is stored at the highest offset (i.e., big-endian byte ordering). A descriptor can be partially read, but starting point is always the offset 00h.

A Descriptor is a block or page of parameters that describe something about a Device. For example, there are Device Descriptors, Configuration Descriptors, Unit Descriptors, etc. In general, all Descriptors are readable, some may be write once, others may have a write protection mechanism. The Configuration Descriptor is writeable and allows modification of the device configuration set by the manufacturer.

Table 14.1 specifies the descriptor identification values.

Table 14.1 — Descriptor Identification Values

Descriptor IDN	Descriptor Type
00h	DEVICE
01h	CONFIGURATION
02h	UNIT
03h	Reserved
04h	INTERCONNECT
05h	STRING
06h	Reserved
07h	GEOMETRY
08h	POWER
09h	DEVICE HEALTH
0Ah	Reserved for File Based Optimization (FBO) Extension Specification
0Bh ... EFh	Reserved
F0h ... FFh	Reserved for Vendor Specific. Content and function of these Descriptors may vary based on Manufacturer Name String Descriptor and Product Revision Level String Descriptor. For detailed definition of these Descriptors please refer to the device manufacturer datasheet.

14.1.1 Descriptor Types

Descriptors are classified into types, as indicated in Table 14.1. Some descriptors are singular entities, such as a Device descriptor. Others can have multiple copies, depending upon the quantity defined, such as UNIT descriptors. See Figure 14.1.

14.1.2 Descriptor Indexing

Each descriptor type has an index number associated with it. The index value starts at zero and increments by one for each additional descriptor that is instantiated. For example, the first and only Device Descriptor has an index value of 0. The first String Descriptor will have an index value of 0. The Nth String Descriptor will have an index value of N-1.

14.1.3 Accessing Descriptors and Device Configuration

Descriptors are accessed by requesting a descriptor IDN and a descriptor INDEX. For example, to request the fifth Unit descriptor, the request would reference TYPE UNIT (numeric value = 2) and INDEX 4 (numeric value = N-1).

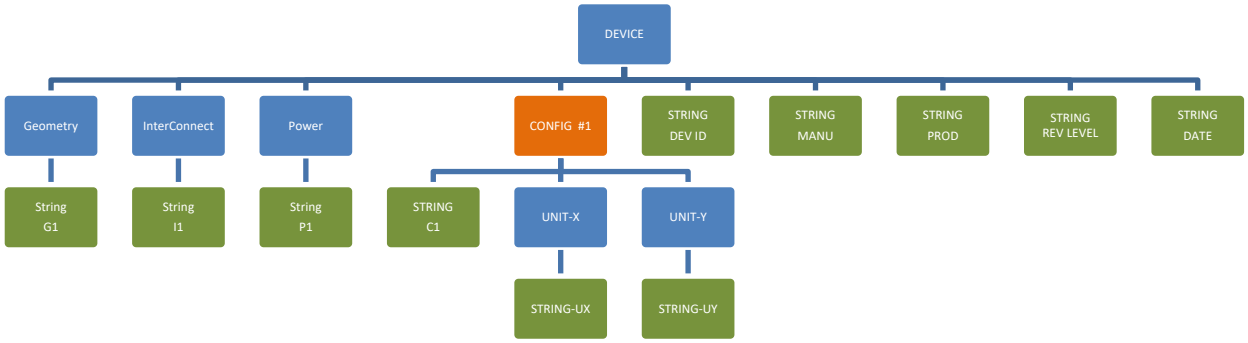


Figure 14.1 — Descriptor Organization

All descriptors are read-only, except the Configuration Descriptors and the OEM_ID String Descriptor which are: readable, writeable if bConfigDescrLock attribute value is equal to 00h.

In particular, the Configuration Descriptors allow modification of the device configuration set by the manufacturer. Parameter settings in the Configuration Descriptors are used to calculate and populate the parameter fields in the Device Descriptor and the Unit Descriptors – an internal operation by the device. The host may write the Configuration Descriptors multiple times if bConfigDescrLock attribute is equal to zero.

A device that supports 8 logical units has only one Configuration Descriptor, while a device that supports 32 logical units has four Configuration Descriptors. The host may write only the Configuration Descriptors related to the logical units to be configured, and avoid to send write descriptor query requests for the Configuration Descriptors that do not need to be changed.

14.1.3 Accessing Descriptors and Device Configuration (cont'd)

The bConfDescContinue parameter in Configuration Descriptor indicates the end of a sequence of write descriptor query requests during device configuration. In particular, if bConfDescContinue is set to one, the current query request will be followed by another one, and the device shall not start internal operations to implement the new configuration. If bConfDescContinue is set to zero, the current query request is the last one, and the device shall start internal operations to implement the new configuration.

For example, to configure LU 0, LU 1, LU 2, LU 18 and LU 20 the following write descriptor query request may be sent.

1) write descriptor query request with INDEX = 0, bConfDescContinue = 01h, Device Descriptor parameters, logical unit parameters from 0 to 7

2) write descriptor query request with INDEX = 2, bConfDescContinue = 00h, Device Descriptor parameters, logical unit parameters from 16 to 23

The latency of a write descriptor request with bConfDescContinue is set to zero may be significantly longer than a query request with bConfDescContinue set to one. If bConfDescContinue = 00h, then the device shall send a QUERY RESPONSE UPIU only after completing the configuration process.

If bConfDescContinue = 0h, then the Query Response field in the QUERY RESPONSE UPIU shall be set to "Success" only if

- parameters in Device Descriptor and Unit Descriptors have been updated successfully
- logical units configuration has been completed successfully
- logical units are ready for operation (read, write, etc.)

If the query request fails, it shall be possible to repeat the configuration procedure sending a new sequence of write descriptor query requests.

The device may not be able to properly process SCSI commands while updating its configuration, therefore Host should not send them.

The Configuration Descriptor for same index may be sent more than once by host (eg, Configuration Descriptor with Index = 00h) with bConfDescContinue = 01h. The last received values of each Configuration Descriptor index before bConfDescContinue set to zero, shall be considered as valid configuration.

If power cycle happens while bConfDescContinue is 01h, then all the configuration descriptor values shall be reset to previous successful configuration value or if there is no previous configuration then they shall be reset to MDV values as per UFS Specification.

Once the device has been configured, the setting of bConfigDescrLock to one permanently locks the device configuration. Some devices may be configured multiple times during system setup or system development. The details and restrictions related to multiple device configurations are vendor specific and out of scope for this standard.

NOTE This version of the standard does not require a power cycle to complete the device configuration.

14.1.3.1 Read Descriptor

A Query Request operation with READ DESCRIPTOR opcode is sent by the host to the device. The host builds a QUERY REQUEST UPIU places a READ DESCRIPTOR opcode within the UPIU, sets the appropriate values in the required fields and sends that UPIU to the target device.

Upon reception of the QUERY REQUEST UPIU the device will decode the READ DESCRIPTOR opcode field and retrieve the descriptor indicated by the DESCRIPTOR IDN field, the INDEX field and the SELECTOR field . When the device is ready to return data to the host the device will construct a QUERY RESPONSE UPIU, set the appropriate fields and place the entire retrieved descriptor within a single Data Segment area of the QUERY RESPONSE UPIU.

Upon transmission of the QUERY RESPONSE UPIU the device will consider the Query Request operation complete.

Upon reception of the QUERY RESPONSE UPIU the host will retrieve the requested descriptor from the Data Segment area, and decode the Status and other fields within the UPIU and take the appropriate completion action. Upon reception of the QUERY RESPONSE UPIU the host will consider that the Query Request operation is complete.

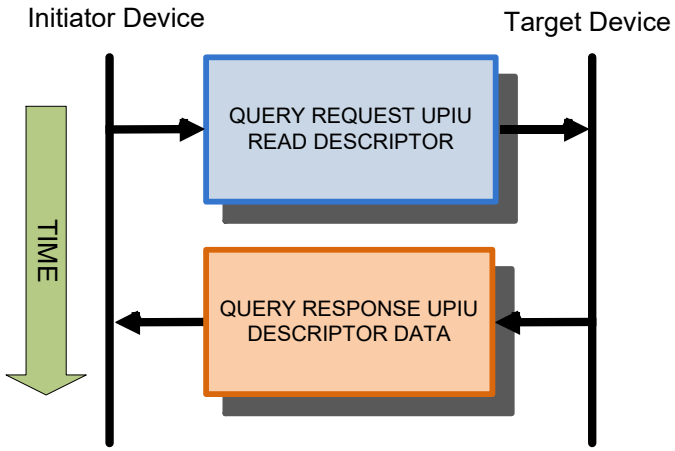


Figure 14.2 — Read Request Descriptor

14.1.3.2 Write Descriptor

A Query Request operation with WRITE DESCRIPTOR opcode is sent by the host to the device. The host builds a QUERY REQUEST UPIU places a WRITE DESCRIPTOR opcode within the UPIU, sets the appropriate values in the required fields and sends that UPIU to the target device. A QUERY REQUEST UPIU with a WRITE DESCRIPTOR opcode will additionally include a data segment that contains the descriptor data the host is sending to the device.

Upon reception of the QUERY REQUEST UPIU the device will decode the WRITE DESCRIPTOR opcode field and access the descriptor indicated by the DESCRIPTOR IDN field, the INDEX field and the SELECTOR field. The device will extract the descriptor data within the data segment of the UPIU and will internally overwrite the addressed descriptor with the extracted data. When the device has completed this process it will then notify the host of the success or failure of this operation by returning to the host a QUERY RESPONSE UPIU with the RESPONSE field containing the response code
.
After the transmission of the QUERY RESPONSE UPIU the device will consider the operation complete. Upon reception of the QUERY RESPONSE UPIU the host will decode the RESPONSE field and other fields within the UPIU and take the appropriate completion action. Upon reception of the QUERY RESPONSE UPIU the host will consider that the operation is complete.

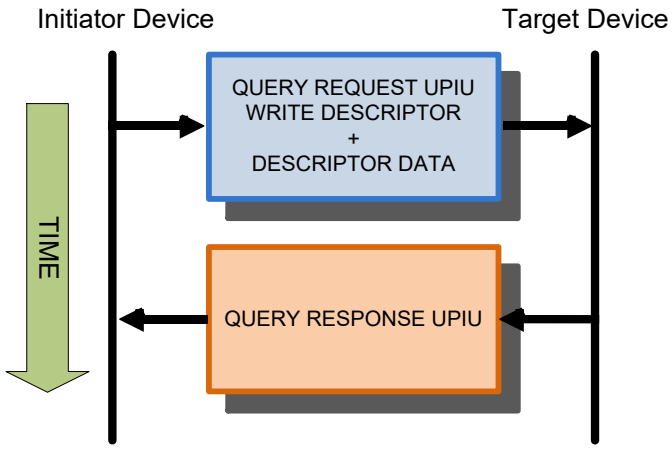


Figure 14.3 — Write Request Descriptor

14.1.4 Descriptor Definitions

14.1.4.1 Generic Descriptor Format

The format of all descriptors begins with a header which contains the length of the descriptor and a type value that indentifies the specific type of descriptor. The length value includes the length of the header plus any additional data.

Table 14.2 — Generic Descriptor Format

DEVICE DESCRIPTOR			
Offset	**Size**	**Name**	**Description**
0	1	bLength	Size of this descriptor inclusive = N
1	1	bDescriptorIDN	Descriptor Type Identifier
2 … N-1	N-2	DATA	Descriptor Information

For unit descriptors, the format includes an indication for the unit being addressed.

Table 14.3 — Logical Unit Descriptor Format

UNIT DESCRIPTOR			
Offset	**Size**	**Name**	**Description**
0	1	bLength	Size of this descriptor inclusive = N
1	1	bDescriptorIDN	Descriptor Type Identifier
2	1	bUnitIndex	Unit index 00h to the number of LU specified by bMaxNumberLU
3 ... N-1	N-3	DATA	Descriptor Information

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 **is** a table of Device Descriptor names and attributes that runs consecutively through the next several pages.

14.1.4.2 Device Descriptor (cont'd)

Table 14.4 — Device Descriptor

DEVICE DESCRIPTOR					
Offset	**Size**	**Name**	**MDV (1)**	**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(3)	Number of Logical Units bNumberLU does not include well known logical units.
07h	1	bNumberWLU	04h	No	Number of Well known Logical Units
08h	1	bBootEnable	00h	Yes	Boot Enable Indicate whether the device is enabled for boot. 00h: Boot feature disabled 01h: Bootable feature enabled 02h: Permanent-bootable feature enabled Others: Reserved

14.1.4.2 Device Descriptor (cont'd)

Table 14.4 — Device Descriptor

DEVICE DESCRIPTOR					
Offset	**Size**	**Name**	**MDV (1)**	**User Conf.**	**Description**
09h	1	bDescrAccessEn	00h	Yes	Descriptor Access Enable Indicate whether the Device Descriptor can be read after the partial initialization phase of the boot sequence 00h: Device Descriptor access disabled 01h: Device Descriptor access enabled Others: Reserved
0Ah	1	bInitPowerMode	01h	Yes	Initial Power Mode bInitPowerMode defines the Power Mode after device initialization or hardware reset 00h: UFS-Sleep Mode 01h: Active Mode Others: Reserved
0Bh	1	bHighPriorityLUN	7Fh	Yes	High Priority LUN bHighPriorityLUN defines the high priority logical unit. Valid values are: from 0 to the number of LU specified by bMaxNumberLU, and 7Fh. If this parameter value is 7Fh all logical units have the same priority.
0Ch	1	bSecureRemovalType	00h	Yes	Secure Removal Type 00h: information removed by an erase of the physical memory 01h: information removed by overwriting the addressed locations with a single character followed by an erase. 02h: information removed by overwriting the addressed locations with a character, its complement, then a random character. 03h: information removed using a vendor define mechanism. Others: Reserved
0Dh	1	bSecurityLU	01h	No	Support for security LU 00h: not supported 01h: RPMB Others: Reserved

14.1.4.2 Device Descriptor (cont'd)

Table 14.4 — Device Descriptor

DEVICE DESCRIPTOR					
Offset	**Size**	**Name**	**MDV (1)**	**User Conf.**	**Description**
0Eh	1	bBackgroundOpsTermLat	Device specific	No	Background Operations Termination Latency bBackgroundOpsTermLat defines the maximum latency for the termination of ongoing background operations. When the device receives a COMMAND UPIU with a transfer request, the device shall start the data transfer and send a DATA IN UPIU or a RTT UPIU within the latency declared in bBackgroundOpsTermLat. The latency is expressed in units of 10 ms (e.g., 01h = 10 ms, FFh = 2550 ms). The latency is undefined if the value of this parameter is zero.
0Fh	1	bInitActiveICCLevel	00h	Yes	Initial Active ICC Level bInitActiveICCLevel defines the bActiveICCLevel value after power on or reset. Valid range from 00h to 0Fh.
10h	2	wSpecVersion	0500h	No	Specification version Bits[15:8] = Major version in BCD format Bits[7:4] = Minor version in BCD format Bits[3:0] = Version suffix in BCD format Example: version 3.21 = 0321h
12h	2	wManufactureDate	Device specific	No	Manufacturing Date BCD version of the device manufacturing date, i.e., August 2010 = 0810h
14h	1	iManufacturerName	Device specific	No	Manufacturer Name Index to the string which contains the Manufacturer Name.
15h	1	iProductName	Device specific	No	Product Name Index to the string which contains the Product Name.
16h	1	iSerialNumber	Device specific	No	Serial Number Index to the string which contains the Serial Number.
17h	1	iOemID	Device specific	No	OEM ID Index to the string which contains the OEM ID.

14.1.4.2 Device Descriptor (cont'd)

Table 14.4 — Device Descriptor

DEVICE DESCRIPTOR					
Offset	**Size**	**Name**	**MDV (1)**	**User Conf.**	**Description**
18h	2	wManufacturerID	Device specific	No	Manufacturer ID Manufacturer ID as defined in JEP106C, 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	Frequency and method of Real-Time Clock update. Bits [15:10] Reserved Bits [9] TIME_BASELINE 0b: Time elapsed from the previous dSecondsPassed update. 1b: Absolute time elapsed from January 1st 2010 00:00. NOTE If the host device has a Real Time Clock it should use TIME BASELINE = '1'. If the host device has no Real Time Clock it should use TIME BASELINE = '0'. Bits [8:6] TIME_UNIT 000b = Undefined 001b = Months 010b = Weeks 011b = Days 100b = Hours 101b = Minutes 110b = Reserved 111b = Reserved Bits [5:0] TIME_PERIOD If TIME_UNIT is 0 TIME_PERIOD is ignored and the period between RTC update is not defined. All fields are configurable by the host.

14.1.4.2 Device Descriptor (cont'd)

Table 14.4 — Device Descriptor

DEVICE DESCRIPTOR					
Offset	**Size**	**Name**	**MDV (1)**	**User Conf.**	**Description**
1Fh	1	bUFSFeaturesSupport	Device specific	No	UFS Features Support This field indicates which features are supported by the device. A feature is supported if the related bit is set to one. bit[0]: Field Firmware Update (FFU) bit[1]: Production State 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 Performance Booster(HPB) Extension Standard Others: Reserved Bit 0 shall be set to one.
20h	1	bFFUTimeout	Device specific	No	Field Firmware Update Timeout The maximum time, in seconds, that access to the device is limited or not possible through any ports associated due to execution of a WRITE BUFFER command. A value of zero indicates that no timeout is provided.
21h	1	bQueueDepth	Device specific	No	Queue Depth 0: The device implements the per-LU queueing architecture. 1… 255: The device implements the shared queueing architecture. This parameter indicates the depth of the shared queue. If bLUQueueDepth > 0 for any LU (except RPMB LU), then bQueueDepth shall be 0.
22h	2	wDeviceVersion	Device specific	No	Device Version This field provides the device version.

14.1.4.2 Device Descriptor (cont'd)

Table 14.4 — Device Descriptor

DEVICE DESCRIPTOR					
Offset	**Size**	**Name**	**MDV (1)**	**User Conf.**	**Description**
24h	1	bNumSecureWPArea	Device specific	No	Number of Secure Write Protect Areas This value specifies the total number of Secure Write Protect Areas supported by the device. The value shall be equal to or greater than bNumberLU and shall not exceed 32 (bNumberLU ≤ bNumSecureWPArea ≤ 32).
25h	4	dPSAMaxDataSize	Device specific	No	PSA Maximum Data Size This parameter specifies the maximum amount of data that may be written during the pre-soldering phase of the PSA flow. The value indicates the total amount of data for all logical units with bPSASensitive = 01h. Value expressed in units of 4 Kbyte.
29h	1	bPSAStateTimeout	Device specific	No	PSA State Timeout This parameter specifies the command maximum timeout for a change in bPSAState state. 00h means undefined. Otherwise, the formula to calculate the max timeout value is: Production State Timeout = 100 µs * 2^bPSAStateTimeout For example: 01h means 100 µs x 2^1 = 200 µs 02h means 100 µs x 2^2 = 400 µs 17h means 100 µs x 2^23 = 838.86 s
2Ah	1	iProductRevisionLevel	Device specific	No	Product Revision Level Index to the string which contains the Product Revision Level
2Bh	5	Reserved	-	-	Reserved
30h	16	Reserved	-	-	Reserved for Unified Memory Extension standard
40h	3	Reserved	-	-	Reserved for Host Performance Booster (HPB) Extension Standard
43h	10	Reserved	-	-	Reserved

14.1.4.2 Device Descriptor (cont'd)

Table 14.4 — Device Descriptor

DEVICE DESCRIPTOR					
Offset	**Size**	**Name**	**MDV (1)**	**User Conf.**	**Description**
4Dh	2	wExtendedWriteBoosterSupport	Device specific	No	Extended WriteBooster Support This field indicates which extended WriteBooster features are supported by the device. This field is valid only when WriteBooster is enabled, i.e. bit [8] for WriteBooster Enable in the dExtendedUFSFeaturesSupport is one. A feature is supported if the related bit is set to one. bit[0]: WriteBooster Buffer Resize bit[1]: FIFO Partial Flush Mode bit[2]: Pinned Partial Flush Mode bit[3]: Pre-Erase bit[4]: Command-Level Data Write to WriteBooster Buffer Others: Reserved

14.1.4.2 Device Descriptor (cont'd)

Table 14.4 — Device Descriptor

DEVICE DESCRIPTOR					
Offset	**Size**	**Name**	**MDV (1)**	**User Conf.**	**Description**
4Fh	4	dExtendedUFSFeaturesSupport	Device specific	No	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. 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) Extension Specification bit[8]: WriteBooster bit[9]: Performance Throttling bit[10]: Advanced RPMB bit[11]: Zoned logical units bit[12]: Device Level Exception Warning bit[13]: HID (Host Initiated Defragmentation) bit[14]: Barrier bit[15]: Clear Error History functionality bit[16]: EXT_IID bit[17]: Reserved for File Based Optimization (FBO) Extension Specification bit[18]: Fast Recovery Mode bit[19]: RPMB Authenticated Vendor Command bit[20]: SWR Processing Coordination bit[21~31]: Reserved

14.1.4.2 Device Descriptor (cont'd)

Table 14.4 — Device Descriptor

DEVICE DESCRIPTOR					
Offset	**Size**	**Name**	**MDV (1)**	**User Conf.**	**Description**
53h	1	bWriteBoosterBufferPreserveUserSpaceEn	0	Yes	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. Others: Reserved
54h	1	bWriteBoosterBufferType	0	Yes	WriteBooster Buffer Type 00h: LU dedicated buffer type 01h: Single shared buffer type
55h	4	dNumSharedWriteBoosterBufferAllocUnits	0	Yes	The WriteBooster Buffer size for the shared WriteBooster Buffer configuration. The dNumSharedWriteBoosterBufferAllocUnits value shall be calculated using the following equation: $\mathrm{dNumSharedWriteBoosterBufferAllocUnits} = \mathrm{CEILING}\left(\frac{\mathrm{WriteBoosterBufferCapacity} \times 1}{\mathrm{bAllocationUnitSize} \times \mathrm{dSegmentSize} \times 512}\right)$ 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. If this value is zero, then the shared WriteBooster is not configured for this device.

14.1.4.2 Device Descriptor (cont'd)

Table 14.4 — Device Descriptor

DEVICE DESCRIPTOR					
Offset	**Size**	**Name**	**MDV (1)**	**User Conf.**	**Description**
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.					

a) wManufacturerID

This parameter contains manufacturer identification information for the device manufacturer. The Manufacturer ID is defined by JEDEC in Standard Manufacturer's identification code [JEP106]. The wManufacturerID consists of two fields: Manufacturer ID Code and Bank Index.

Table 14.5 — wManufacturerID Definition

Bit / Byte	7	6	5	4	3	2	1	0
0	Bank Index							
1	Manufacturer ID Code							

b) Bank Index

This field contains an index value of the bank that contains the manufacturer identification code. The Bank Index value shall be equal to the number of the continuation fields that precede the manufacturer identification code as specified by [JEP106].

c) Manufacturer ID Code

Manufacturer identification code as defined by JEDEC in Standard Manufacturer's identification code [JEP106].

14.1.4.3 Configuration Descriptor

The device configuration set by the manufacturer can be modified by writing the Configuration Descriptor. In particular, the Configuration Descriptor allows to configure parameters included in the Device Descriptor, RPMB Unit Descriptor, and Unit Descriptors. The Configuration Descriptor can be written if bConfigDescrLock attribute value is equal to 00h. If bConfigDescrLock attribute value is 01h the Configuration Descriptor is locked and a write request shall fail.

There are up to four Configuration Descriptors.

The first Configuration Descriptor is addressed by setting DESCRIPTOR IDN = 01h, INDEX = 00h and SELECTOR = 00h in a QUERY REQUEST UPIU, and used to change configurable parameters of Device Descriptor, RPMB Unit Descriptor, and the first eight Unit Descriptors (LU 0 to LU 7).

The second Configuration Descriptor is addressed by setting DESCRIPTOR IDN = 01h, INDEX = 01h and SELECTOR = 00h in a QUERY REQUEST UPIU, and used to change configurable parameters of the next eight Unit Descriptors (LU 8 to LU 15).

The third Configuration Descriptor is addressed by setting DESCRIPTOR IDN = 01h, INDEX = 02h and SELECTOR = 00h in a QUERY REQUEST UPIU, and used to change configurable parameters of the next eight Unit Descriptors (LU 16 to LU 23).

The fourth Configuration Descriptor is addressed by setting DESCRIPTOR IDN = 01h, INDEX = 03h and SELECTOR = 00h in a QUERY REQUEST UPIU, and used to change configurable parameters of the last eight Unit Descriptors (LU 24 to LU 31).

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

14.1.4.3 Configuration Descriptor (cont'd)

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

Table 14.9 shows the Configuration Descriptor format with DESCRIPTOR IDN = 01h, INDEX = 03h and SELECTOR = 00h: the lower address space is used for Configuration Descriptor header, then there are eight address spaces for user configurable parameters included in the Unit Descriptors of LU 24 to LU 32. Parameter offsets are defined based on:

- Offset of the Unit Descriptor 0 configurable parameters within the Configuration Descriptor (B).
- Length of Unit Descriptor configurable parameters (L).

Values for B and L are stored in the Device Descriptor parameters bUD0BaseOffset and bUDConfigPLength, respectively.

Table 14.9 — Configuration Descriptor Format (INDEX = 03h)

Offset	Description
00h … (B-1)h	Configuration Descriptor header
(B)h … (B+L-1)h	Unit Descriptor 24 configurable parameters
(B+L)h … (B+2*L-1)h	Unit Descriptor 25 configurable parameters
…	…
(B+7*L)h … (B+8*L-1)h	Unit Descriptor 31 configurable parameters
NOTE 1 B is offset of the Unit Descriptor 0/8/16/24 configurable parameters within the Configuration Descriptor, L is the total size of the configurable Unit Descriptor parameters.	

14.1.4.3 Configuration Descriptor (cont'd)

Table 14.10 defines Configuration Descriptor header, Device Descriptor configurable parameters, and RPMB Unit Descriptor configurable parameters within the Configuration Descriptor with INDEX = 00h. See 14.1.5.3 for details about configurable parameters.

Table 14.10 — Configuration Descr. Header and Device Descr. Conf. Parameters (INDEX = 00h)

Configuration Descriptor Header and Device Descriptor configurable parameters				
Offset	**Size**	**Name**	**MDV (1,2)**	**Description**
00h	1	bLength	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 14.1.5.2, 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.

14.1.4.3 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 (1)**	**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 The column "MDV" (Manufacturer Default Value) specifies parameter values after device manufacturing.

14.1.4.3 Configuration Descriptor (cont'd)

Table 14.12 defines the Unit Descriptors user configurable parameters within the Configuration Descriptor. See 14.1.5.5.

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	
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 is a table of Geometry Descriptor names and attributes that runs consecutively through the next several pages.

14.1.4.4 Geometry Descriptor (cont'd)

Table 14.13 — Geometry Descriptor

GEOMETRY DESCRIPTOR				
Offset	**Size**	**Name**	**Value**	**Description**
00h	1	bLength	69h	Size of this descriptor
01h	1	bDescriptorIDN	07h	Geometry Descriptor Type Identifier
02h	1	bMediaTechnology	00h	Reserved
03h	1	Reserved	00h	Reserved
04h	8	qTotalRawDeviceCapacity	Device specific	Total Raw Device Capacity Total memory quantity available to the user to configure the device logical units (RPMB excluded). It is expressed in unit of 512 bytes
0Ch	1	bMaxNumberLU	Device specific	Maximum number of Logical Unit supported by the UFS device 00h: 8 Logical units 01h: 32 Logical units Others: Reserved
0Dh	4	dSegmentSize	Device specific	Segment Size Value expressed in unit of 512 bytes
11h	1	bAllocationUnitSize	Device specific	Allocation Unit Size Value expressed in number of Segments Each logical unit can be allocated as a multiple of Allocation Units.
12h	1	bMinAddrBlockSize	Device specific	Minimum addressable block size Value expressed in unit of 512 bytes. Its minimum value is 08h, which corresponds to 4 Kbyte.
13h	1	bOptimalReadBlockSize	Device specific	Optimal Read Block Size Value expressed in unit of 512 bytes. This is an optional parameter. A value of zero indicates that this information is not available. If not zero, bOptimalReadBlockSize shall be equal to or greater than bMinAddrBlockSize.
14h	1	bOptimalWriteBlockSize	Device specific	Optimal Write Block Size Value expressed in unit of 512 bytes. bOptimalWriteBlockSize shall be equal to or greater than bMinAddrBlockSize.
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

14.1.4.4 Geometry Descriptor (cont'd)

Table 14.13 — Geometry Descriptor

GEOMETRY DESCRIPTOR				
Offset	**Size**	**Name**	**Value**	**Description**
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 in Normal RPMB mode, 4 KB of data in Advanced RPMB mode) allowed in Security Protocol In and Security Protocol Out (i.e., associated with a single command UPIU) In Normal RPMB mode, 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 In Advanced RPMB mode, If the data to be transferred is larger than bRPMB_ReadWriteSize x 4K bytes, the host will transfer it using multiple Security Protocol In/Out commands. This value shall be changed to the appropriate value based on RPMB configuration.
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.

14.1.4.4 Geometry Descriptor (cont'd)

Table 14.13 — Geometry Descriptor

GEOMETRY DESCRIPTOR				
Offset	**Size**	**Name**	**Value**	**Description**
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 for both DATA IN and DATA OUT UPIUs 01h: out-of-order data transfer is supported by the device for both DATA IN and DATA OUT UPIUs 02h: out-of-order data transfer is supported by the device for DATA IN UPIUs only, and DATA OUT UPIUs must be transferred in-order 03h: out-of-order data transfer is supported by the device for DATA OUT UPIUs only, and DATA IN UPIUs must be transferred in-order All others: Reserved
1Ah	1	bMaxContexIDNumber	Device specific	Max. available number of contexts which are supported by the device. Minimum number of supported contexts shall be 5.
1Bh	1	bSysDataTagUnitSize	Device specific	bSysDataTagUnitSize provides system data tag unit size, which can be calculated as in the following (in bytes) Tag Unit Size = $2^{\text{bSysDataTagUnitSize}}$ x bMinAddrBlockSize x 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: $\text{System Data Tag Resource Size} = \text{Tag Unit Size} \times floor\left(\frac{\text{qTotalRawDeviceCapacity x } 2^{\text{bSysDataTagResSize}-10}}{Tag\ Unit\ Size}\right)$ The range of valid bSysDataTagResSize values is from 0 to 6. Values in range of 07h to FFh are reserved. The formula covers a range from about 0.1% to 6.25% of the device capacity

14.1.4.4 Geometry Descriptor (cont'd)

Table 14.13 — Geometry Descriptor

GEOMETRY DESCRIPTOR				
Offset	**Size**	**Name**	**Value**	**Description**
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.
1Eh	2	wSupportedMemoryTypes	Device specific	Supported Memory Types Bit map which represents the supported memory types. bit 0: Normal memory type bit 1: System code memory type bit 2: Non-Persistent memory type bit 3: Enhanced memory type 1 bit 4: Enhanced memory type 2 bit 5: Enhanced memory type 3 bit 6: Enhanced memory type 4 bit 7: Reserved … bit 14: Reserved bit 15: RPMB memory type A value one means that the corresponding memory type is supported. Bit 0 and bit 15 shall be one for all UFS device.
20h	4	dSystemCodeMaxNAllocU	Device specific	Max Number of Allocation Units for the System Code memory type Maximum available quantity of System Code memory type for the entire device. Value expressed in number of Allocation Unit

14.1.4.4 Geometry Descriptor (cont'd)

Table 14.13 — Geometry Descriptor

GEOMETRY DESCRIPTOR				
Offset	**Size**	**Name**	**Value**	**Description**
24h	2	wSystemCodeCapAdjFac	Device specific	Capacity Adjustment Factor for the System Code memory type This parameter is the ratio between the capacity obtained with the Normal memory type and the capacity obtained with the System Code memory type for the same amount of allocation units. CapacityAdjFactor = $Capacity_{NormalMem}$ / $Capacity_{SystemCode}$ If bCapAdjFacRepresentation == 0h wSystemCodeCapAdjFac = INTEGER(256 × CapacityAdjFactor) Else wSystemCodeCapAdjFac [15:8] = numerator(CapacityAdjFactor) wSystemCodeCapAdjFac [7:0] = denominator(CapacityAdjFactor)
26h	4	dNonPersistMaxNAllocU	Device specific	Max Number of Allocation Units for the Non-Persistent memory type Maximum available quantity of Non-Persistent memory type for the entire device. Value expressed in number of Allocation Unit
2Ah	2	wNonPersistCapAdjFac	Device specific	Capacity Adjustment Factor for the Non-Persistent memory type This parameter is the ratio between the capacity obtained with the Normal memory type and the capacity obtained with the Non-Persistent memory type for the same amount of allocation units. CapacityAdjFactor = $Capacity_{NormalMem}$ / $Capacity_{NonPersist}$ If bCapAdjFacRepresentation == 0h wNonPersistCapAdjFac = INTEGER(256 × CapacityAdjFactor) Else wNonPersistCapAdjFac [15:8] = numerator(CapacityAdjFactor) wNonPersistCapAdjFac [7:0] = denominator(CapacityAdjFactor)

14.1.4.4 Geometry Descriptor (cont'd)

Table 14.13 — Geometry Descriptor

GEOMETRY DESCRIPTOR				
Offset	**Size**	**Name**	**Value**	**Description**
2Ch	4	dEnhanced1MaxNAllocU	Device specific	Max Number of Allocation Units for the Enhanced memory type 1 Maximum available quantity of Enhanced memory type 1 for the entire device Value expressed in number of Allocation Unit
30h	2	wEnhanced1CapAdjFac	Device specific	Capacity Adjustment Factor for the Enhanced memory type 1 This parameter is the ratio between the capacity obtained with the Normal memory type and the capacity obtained with the Enhanced memory type 1 for the same amount of allocation units. CapacityAdjFactor = $Capacity_{NormalMem}$ / $Capacity_{Enhanced1}$ If bCapAdjFacRepresentation == 0h wEnhanced1CapAdjFac = INTEGER(256 × CapacityAdjFactor) Else wEnhanced1CapAdjFac [15:8] = numerator(CapacityAdjFactor) wEnhanced1CapAdjFac [7:0] = denominator(CapacityAdjFactor)
32h	4	dEnhanced2MaxNAllocU	Device specific	Max Number of Allocation Units for the Enhanced memory type 2 Maximum available quantity of Enhanced memory type 2 for the entire device Value expressed in number of Allocation Unit

14.1.4.4 Geometry Descriptor (cont'd)

Table 14.13 — Geometry Descriptor

GEOMETRY DESCRIPTOR				
Offset	**Size**	**Name**	**Value**	**Description**
36h	2	wEnhanced2CapAdjFac	Device specific	Capacity Adjustment Factor for the Enhanced memory type 2 This parameter is the ratio between the capacity obtained with the Normal memory type and the capacity obtained with the Enhanced memory type 2 for the same amount of allocation units. CapacityAdjFactor = $Capacity_{NormalMem}$ / $Capacity_{Enhanced2}$ If bCapAdjFacRepresentation == 0h wEnhanced2CapAdjFac = INTEGER(256 × CapacityAdjFactor) Else wEnhanced2CapAdjFac [15:8] = numerator(CapacityAdjFactor) wEnhanced2CapAdjFac [7:0] = denominator(CapacityAdjFactor)
38h	4	dEnhanced3MaxNAllocU	Device specific	Max Number of Allocation Units for the Enhanced memory type 3 Maximum available quantity of Enhanced memory type 3 for the entire device Value expressed in number of Allocation Unit
3Ch	2	wEnhanced3CapAdjFac	Device specific	Capacity Adjustment Factor for the Enhanced memory type 3 This parameter is the ratio between the capacity obtained with the Normal memory type and the capacity obtained with the Enhanced memory type 3 for the same amount of allocation units. CapacityAdjFactor = $Capacity_{NormalMem}$ / $Capacity_{Enhanced3}$ If bCapAdjFacRepresentation == 0h wEnhanced3CapAdjFac = INTEGER(256 × CapacityAdjFactor) Else wEnhanced3CapAdjFac [15:8] = numerator(CapacityAdjFactor) wEnhanced3CapAdjFac [7:0] = denominator(CapacityAdjFactor)

14.1.4.4 Geometry Descriptor (cont'd)

Table 14.13 — Geometry Descriptor

GEOMETRY DESCRIPTOR				
Offset	**Size**	**Name**	**Value**	**Description**
3Eh	4	dEnhanced4MaxNAllocU	Device specific	Max Number of Allocation Units for the Enhanced memory type 4 Maximum available quantity of Enhanced memory type 4 for the entire device Value expressed in number of Allocation Unit
42h	2	wEnhanced4CapAdjFac	Device specific	Capacity Adjustment Factor for the Enhanced memory type 4 This parameter is the ratio between the capacity obtained with the Normal memory type and the capacity obtained with the Enhanced memory type 4 for the same amount of allocation units. CapacityAdjFactor = $Capacity_{NormalMem}$ / $Capacity_{Enhanced4}$ If bCapAdjFacRepresentation == 0h wEnhanced4CapAdjFac = INTEGER(256 × CapacityAdjFactor) Else wEnhanced4CapAdjFac [15:8] = numerator(CapacityAdjFactor) wEnhanced4CapAdjFac [7:0] = denominator(CapacityAdjFactor)
44h	4	dOptimalLogicalBlockSize	Device specific	Optimal Logical Block Size bit [3:0]: Normal memory type bit [7:4]: System code memory type bit [11: 8]: Non-Persistent memory type bit [15:12]: Enhanced memory type 1 bit [19:16]: Enhanced memory type 2 bit [23:20]: Enhanced memory type 3 bit [27:24]: Enhanced memory type 4 bit [31:28]:Reserved The optimal logical block size for each memory type can be calculated from the related dOptimalLogicalBlockSize field as indicated in the following: Optimal Logical Block Size = 2^ (dOptimalLogicalBlockSize field) x bMinAddrBlockSize x 512 byte
48h	5	Reserved	-	Reserved for Host Performance Booster (HPB) Extension Standard

14.1.4.4 Geometry Descriptor (cont'd)

Table 14.13 — Geometry Descriptor

GEOMETRY DESCRIPTOR				
Offset	**Size**	**Name**	**Value**	**Description**
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	Number of maximum WriteBooster Buffer supported by the device. In this version of the standard, the valid value of this field is 1. Other values are reserved.
54h	1	bWriteBoosterBufferCapAdjFac	Device specific	Capacity Adjustment Factor for the WriteBooster Buffer memory type. This value provides the LBA space reduction multiplication factor when WriteBooster Buffer is configured in user space reduction mode. Therefore, this parameter applies only if bWriteBoosterBufferPreserveUserSpaceEn is 00h. For "LU dedicated buffer" mode, the total user space is decreased by the following amount: bWriteBoosterBufferCapAdjFac * dLUNumWriteBoosterBufferAllocUnits * bAllocationUnitSize * dSegmentSize * 512 byte'. For "shared buffer" mode, the total user space is decreased by the following amount: bWriteBoosterBufferCapAdjFac * dNumSharedWriteBoosterBufferAllocUnits * bAllocationUnitSize * dSegmentSize * 512 byte. The value of this parameter is 3 for TLC NAND when SLC mode is used as WriteBooster Buffer. 2 for MLC NAND.
55h	1	bSupportedWriteBoosterBufferUserSpaceReductionTypes	Device specific	The supportability of user space reduction mode and preserve user space mode. 00h: WriteBooster Buffer can be configured only in user space reduction type. 01h: WriteBooster Buffer can be configured only in preserve user space type. 02h: Device can be configured in either user space reduction type or preserve user space type. Others : Reserved

14.1.4.4 Geometry Descriptor (cont'd)

Table 14.13 — Geometry Descriptor

GEOMETRY DESCRIPTOR				
Offset	**Size**	**Name**	**Value**	**Description**
56h	1	bSupportedWriteBoosterBufferTypes	Device specific	The supportability of WriteBooster Buffer type. 00h: LU based WriteBooster Buffer configuration 01h: Single shared WriteBooster Buffer configuration 02h: Supporting both LU based WriteBooster Buffer and Single shared WriteBooster Buffer configuration Others: Reserved
57h	1	bMaxZonedLUCount	Device specific	Maximum number of zoned logical units. (see 13.4.23)
58h	8	qZoneAllocationUnit	Device specific	The smallest number of bytes that can be erased by an erase operation from a zoned logical unit. (see 13.4.23)
60h	8	qZoneSize	Device specific	Size of a zone in bytes. Additionally, the zone size shall either be a divisor or a multiple of qZoneAllocationUnit. (see 13.4.23)
68h	1	bCapAdjFacRepresentation	Device specific	CapacityAdjFactor representation format 00h: Legacy 256ths 01h: Simple Fraction Others: Reserved
NOTE 1 The Capacity Adjustment Factor value for Normal memory type is one.				

14.1.4.5 Unit Descriptor

The Unit Descriptor 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 is a table of Unit Descriptor names and attributes.

14.1.4.5 Unit Descriptor (cont'd)

Table 14.14 — Unit Descriptor

UNIT DESCRIPTOR					
Offset	**Size**	**Name**	**MDV (1)**	**User Conf. (2)**	**Description**
00h	1	bLength	2Dh	No	Size of this descriptor
01h	1	bDescriptorIDN	02h	No	Unit Descriptor Type Identifier
02h	1	bUnitIndex	00h to the number of LU specified by bMaxNumberLU	No	Unit Index
03h	1	bLUEnable	00h	Yes	Logical Unit Enable 00h: Logical Unit disabled 01h: Logical Unit enabled 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	bPSASensitive	Device specific	No(3)	00h: LU is not sensitive to soldering 01h: LU is sensitive to soldering Others: Reserved

14.1.4.5 Unit Descriptor (cont'd)

Table 14.14 — Unit Descriptor

UNIT DESCRIPTOR					
Offset	**Size**	**Name**	**MDV (1)**	**User Conf. (2)**	**Description**
08h	1	bMemoryType	00h	Yes	Memory Type bMemoryType defines logical unit memory type. 00h: Normal Memory 01h: System code memory type 02h: Non-Persistent memory type 03h: Enhanced memory type 1 04h: Enhanced memory type 2 05h: Enhanced memory type 3 06h: Enhanced memory type 4 Others: Reserved
09h	1	bDataReliability	00h	Yes	Data Reliability bDataReliability defines the device behavior when a power failure occurs during a write operation to the logical unit 00h: the logical unit is not protected. Logical unit's entire data may be lost as a result of a power failure during a write operation 01h: logical unit is protected. Logical unit's data is protected against power failure. Others: Reserved
0Ah	1	bLogicalBlockSize	0Ch	Yes	Logical Block Size The size of addressable logical blocks is equal to the result of exponentiation with as base the number two and as exponent the bLogicalBlockSize value: $2^{bLogicalBlockSize}$ (i.e., bLogicalBlockSize = 0Ch corresponds to 4 Kbyte Logical Block Size). Its minimum value is 0Ch, which corresponds to 4 Kbyte
0Bh	8	qLogicalBlockCount	00h	Yes(4)	Logical Block Count Total number of addressable logical blocks in the logical unit
13h	4	dEraseBlockSize	00h(5)	No	Erase Block Size Optimal granularity for erase and discard operations. Value in number of Logical Blocks

14.1.4.5 Unit Descriptor (cont'd)

Table 14.14 — Unit Descriptor

UNIT DESCRIPTOR					
Offset	**Size**	**Name**	**MDV (1)**	**User Conf. (2)**	**Description**
17h	1	bProvisioningType	00h	Yes	Provisioning Type 00h: Thin Provisioning is disabled (default) 02h: Thin Provisioning is enabled and TPRZ = 0 03h: Thin Provisioning is enabled and TPRZ = 1 10h: Thin Provisioning is disabled, TPRZ=0 and zoned logical unit support is enabled. Others: Reserved
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 bMaxContexIDNumber. Bits [6:4]: LARGE_UNIT_MAX_MULTIPLIER_M1 is the highest multiplier that can be configured for Large Unit contexts, minus one. Large Unit contexts may be configured to have a multiplier in the range: 1 ≤ multiplier ≤ (LARGE_UNIT_MAX_MULTIPLIER_M1 + 1) This field is read only. Bit [15:7]: Reserved.
22h	1	bLargeUnitGranularity_M1	Device specific	No	Granularity of the Large Unit, minus one. Large Unit Granularity = 1MB * (bLargeUnitGranularity_M1 + 1)
23h	6	Reserved	-	-	Reserved for Host Performance Booster (HPB) Extension Standard

14.1.4.5 Unit Descriptor (cont'd)

Table 14.14 — Unit Descriptor

UNIT DESCRIPTOR					
Offset	**Size**	**Name**	**MDV (1)**	**User Conf. (2)**	**Description**
29h	4	dLUNumWriteBoosterBufferAllocUnits	00h	Yes	The WriteBooster Buffer size for the Logical Unit. The dLUNumWriteBoosterBufferAllocUnits value shall be calculated using the following equation: $\mathrm{dLUNumWriteBoosterBufferAllocUnits} = \mathrm{CEILING}\left(\frac{\mathrm{WriteBoosterBufferCapacity} \times 1}{\mathrm{bAllocationUnitSize} \times \mathrm{dSegmentSize} \times 512}\right)$ 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. The Logical unit among LU0 ~ LU7 can be configured for WriteBooster Buffer. Otherwise, whole WriteBooster Buffer configuration in this device is invalid.

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.

14.1.4.6 RPMB Unit Descriptor

In a QUERY REQUEST UPIU, the RPMB Unit Descriptor is addressed setting: DESCRIPTOR IDN = 02h, INDEX = C4h, and SELECTOR = 00h.

Table 14.15 is a table of RPMB Unit Descriptor names and attributes that runs consecutively through the next several pages.

14.1.4.6 RPMB Unit Descriptor (cont'd)

Table 14.15 — RPMB Unit Descriptor

RPMB UNIT DESCRIPTOR					
Offset	**Size**	**Name**	**MDV (1)**	**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) N: Queue depth available in RPMB LU. Only 1 task per region may be queued at any given time. N shall be 1 to 4 according to the number of regions enabled in bRPMBRegionEnable[0:3]. For example, when RPMB regions 1, 2, and 3 are enabled, N is set to 4.
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: If set to 1, Advanced RPMB Mode is enabled Bit-5: If set to 1, RPMB Purge Operation is enabled Bit-6 to Bit-7: Reserved.

14.1.4.6 RPMB Unit Descriptor (cont'd)

Table 14.15 — RPMB Unit Descriptor

RPMB UNIT DESCRIPTOR					
Offset	**Size**	**Name**	**MDV (1)**	**User Conf.**	**Description**
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., In Normal RPMB, bLogicalBlockSize = 08h corresponds to 256 byte Logical Block Size. In Advanced RPMB, bLogicalBlockSize = 0Ch correnponds to 4 KB Logical) This value shall be changed to the appropriate value based on RPMB configuration.
0Bh	8	qLogicalBlockCount	Device specific	No	Logical Block Count Total number of addressable logical blocks in the RPMB LU. For Normal RPMB, Logical Block Count shall be a multiple of 512. For Advanced RPMB, Logical Block Count shall be a multiple of 32. (i.e., 128 Kbyte) This value shall be changed to the appropriate value based on RPMB configuration.
13h	1	bRPMBRegion0Size	Device Specific	Yes	RPMB Region 0 Size RPMB region 0 size is defined in 128 KB unit (00h: 0 KB, 01h: 128 KB, … , 80h: 16384 KB).
14h	1	bRPMBRegion1Size	00h	Yes	RPMB Region 1 Size RPMB region 1 size is defined in 128 KB unit (00h: 0 KB, 01h: 128 KB, … , 80h: 16384 KB).
15h	1	bRPMBRegion2Size	00h	Yes	RPMB Region 2 Size RPMB region 2 size is defined in 128 KB unit (00h: 0 KB, 01h: 128 KB, … , 80h: 16384 KB).
16h	1	bRPMBRegion3Size	00h	Yes	RPMB Region 3 Size RPMB region 3 size is defined in 128 KB unit (00h: 0 KB, 01h: 128 KB, … , 80h: 16384 KB).
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.

14.1.4.7 Power Parameters Descriptor

This descriptor contains information about the power capabilities and power states of the device. In a QUERY REQUEST UPIU, the Power Parameters Descriptor is addressed setting: DESCRIPTOR IDN = 08h, INDEX = 00h, and SELECTOR = 00h.

Table 14.16 — Power Parameters Descriptor

POWER PARAMETERS DESCRIPTOR				
Offset	**Size**	**Name**	**Value**	**Description**
00h	1	bLength	62h	Size of this descriptor
01h	1	bDescriptorIDN	08h	Power Parameters Descriptor Type Identifier
02h	2	wActiveICCLevelsVCC[0]	Device specific	Maximum VCC current value for bActiveICCLevel = 0
04h	2	wActiveICCLevelsVCC[1]	Device specific	Maximum VCC current value for bActiveICCLevel = 1
…	…	…	…	…
20h	2	wActiveICCLevelsVCC[15]	Device specific	Maximum VCC current value for bActiveICCLevel = 15
22h	2	wActiveICCLevelsVCCQ[0]	Device specific	Maximum VCCQ current value for bActiveICCLevel = 0
24h	2	wActiveICCLevelsVCCQ[1]	Device specific	Maximum VCCQ current value for bActiveICCLevel = 1
…	…	…	…	…
40h	2	wActiveICCLevelsVCCQ[15]	Device specific	Maximum VCCQ current value for bActiveICCLevel = 15
42h	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

14.1.4.8 Interconnect Descriptor

The Interconnect Descriptor contains the MIPI M-PHY® specification version number and the MIPI UniPro® specification version number. In a QUERY REQUEST UPIU, the Interconnect Descriptor is addressed setting: DESCRIPTOR IDN = 04h, INDEX = 00h, and SELECTOR = 00h.

Table 14.17 — Interconnect Descriptor

INTERCONNECT DESCRIPTOR				
Offset	**Size**	**Name**	**Value**	**Description**
00h	1	bLength	06h	Size of this descriptor
01h	1	bDescriptorIDN	04h	Interconnect Descriptor Type Identifier
02h	2	bcdUniproVersion	0200h	MIPI UniPro® version number in BCD format Example: version 3.21 = 0321h
04h	2	bcdMphyVersion	0500h	MIPI M-PHY® version number in BCD format Example: version 3.21=0321h

14.1.4.9 Manufacturer Name String Descriptor

This descriptor contains the UNICODE, left justified, manufacturer name string.

The content of the descriptor shall be identical to the content of the “VENDOR IDENTIFICATION” field in Inquiry Response Data. The length of the descriptor shall be 12h (18 decimal), containing exactly 8 UNICODE characters, to match “VENDOR IDENTIFICATION” field in Inquiry Response Data.

In a QUERY REQUEST UPIU, the Manufacturer Name String Descriptor is addressed setting: DESCRIPTOR IDN = 05h, INDEX = iManufacturerName (Device Descriptor parameter), and SELECTOR = 00h.

Table 14.18 — Manufacturer Name String

MANUFACTURER NAME STRING				
Offset	**Size**	**Name**	**Value**	**Description**
00h	1	bLength	12h	Size of this descriptor
01h	1	bDescriptorIDN	05h	String Descriptor Type Identifier
02h	2	UC[0]		Unicode string character
04h	-	-		-
10h	2	UC[7]		Unicode string character

14.1.4.10 Product Name String Descriptor

This descriptor contains the UNICODE, left justified, product name string.

The content of the descriptor shall be identical to the content of the “PRODUCT IDENTIFICATION” field in Inquiry Response Data. The length of the descriptor shall be 22h (34 decimal), containing exactly 16 UNICODE characters, to match “PRODUCT IDENTIFICATION” field in Inquiry Response Data.

In a QUERY REQUEST UPIU, the Product Name String Descriptor is addressed setting: DESCRIPTOR IDN = 05h, INDEX = iProductName (Device Descriptor parameter), and SELECTOR = 00h.

Table 14.19 — Product Name String

PRODUCT NAME STRING DESCRIPTOR				
Offset	**Size**	**Name**	**Value**	**Description**
00h	1	bLength	22h	Size of this descriptor
01h	1	bDescriptorIDN	05h	String Descriptor Type Identifier
02h	2	UC[0]		Unicode string character
04h	-	-		-
20h	2	UC[15]		Unicode string character

14.1.4.11 OEM ID String Descriptor

This descriptor contains the UNICODE OEM ID string that may consist of up to 126 UNICODE characters. Number of UNICODE characters is calculated by $(LENGTH - 2) \div 2$.

In a QUERY REQUEST UPIU, the OEM ID String Descriptor is addressed setting: DESCRIPTOR IDN = 05h, INDEX = iOemID (Device Descriptor parameter), and SELECTOR = 00h.

The OEM ID String Descriptor identifies the OEM name (e.g. name of Phone Manufacturer, or name of Car Manufacturer, etc.).

OEM_ID String descriptor is: readable, writeable if bConfigDescrLock attribute value is equal to 00h.

Table 14.20 — OEM_ID String

OEM ID STRING DESCRIPTOR				
Offset	**Size**	**Name**	**Value**	**Description**
00h	1	bLength	LENGTH	Size of this descriptor
01h	1	bDescriptorIDN	05h	String Descriptor Type Identifier
02h	2	UC[0]		Unicode string character
04h	-	-		-
LENGTH-2	2	UC[(LENGTH-2)÷2-1]		Unicode string character

14.1.4.12 Serial Number String Descriptor

This descriptor contains the UNICODE serial number string that may consist of up to 126 UNICODE characters. Number of UNICODE characters is calculated by $(LENGTH - 2) \div 2$.

In a QUERY REQUEST UPIU, the Serial Number String Descriptor is addressed setting: DESCRIPTOR IDN = 05h, INDEX = iSerialNumber (Device Descriptor parameter), and SELECTOR = 00h.

Note: The Serial Number field contains a value that uniquely identifies the device. Uniqueness is defined relative to the value of wManufacturerID, and the combination of wManufacturerID and Serial Number is unique for each device.

Table 14.21 — Serial Number String Descriptor

SERIAL NUMBER STRING DESCRIPTOR				
Offset	**Size**	**Name**	**Value**	**Description**
00h	1	bLength	LENGTH	Size of this descriptor
01h	1	bDescriptorIDN	05h	String Descriptor Type Identifier
02h	2	UC[0]		Unicode string character
04h	-	-		-
LENGTH-2	2	UC[(LENGTH-2)÷2-1]		Unicode string character

14.1.4.13 Product Revision Level String Descriptor

This descriptor contains the UNICODE, left justified, product revision level string. The content of the descriptor shall be identical to the content of the "PRODUCT REVISION LEVEL" field in Inquiry Response Data. The length of the descriptor shall be Ah, containing exactly 4 UNICODE characters, to match "PRODUCT REVISION LEVEL" field in Inquiry Response Data.

14.1.4.13 Product Revision Level String Description (cont'd)

In a QUERY REQUEST UPIU, the Product Revision Level String Descriptor is addressed setting: DESCRIPTOR IDN = 05h, INDEX = iProductRevisionLevel (Device Descriptor parameter), and SELECTOR = 00h.

Table 14.22 — Product Revision Level String

PRODUCT REVISION LEVEL STRING DESCRIPTOR				
Offset	**Size**	**Name**	**Value**	**Description**
00h	1	bLength	0Ah	Size of this descriptor
01h	1	bDescriptorIDN	05h	String Descriptor Type Identifier
02h	2	UC[0]		Unicode string character
04h	-	-		-
08h	2	UC[3]		Unicode string character

14.1.4.14 Device Health Descriptor

Device Health Descriptor provides information related to the health of the device. In a QUERY REQUEST UPIU, the Device Health Descriptor is addressed by setting: DESCRIPTOR IDN = 09h, INDEX = 00h and SELECTOR = 00h.

Table 14.23 — Device Health Descriptor

Offset	Size	Name	Value	Description
00h	1	bLength	2Dh	Size of this descriptor
01h	1	bDescriptorIDN	09h	Device Health Descriptor Type Identifier
02h	1	bPreEOLInfo	Device specific	Pre End of Life Information This field provides indication about device life time reflected by average reserved blocks. 00h: Not defined 01h: Normal 02h: Warning. Consumed 80% of reserved blocks. 03h: Critical. Consumed 90% of reserved blocks. Others: Reserved
03h	1	bDeviceLifeTimeEstA	Device specific	This field provides an indication of the device life time based on the amount of performed program/erase cycles. The calculation method is vendor specific and referred as method A. 00h: Information not available 01h: 0% - 10% device life time used 02h: 10% - 20% device life time used 03h: 20% - 30% device life time used 04h: 30% - 40% device life time used 05h: 40% - 50% device life time used 06h: 50% - 60% device life time used 07h: 60% - 70% device life time used 08h: 70% - 80% device life time used 09h: 80% - 90% device life time used 0Ah: 90% - 100% device life time used 0Bh: Exceeded its maximum estimated device life time Others: Reserved

14.1.4.14 Device Health Descriptor (cont'd)

Table 14.23 — Device Health Descriptor (cont'd)

Offset	Size	Name	Value	Description
04h	1	bDeviceLifeTimeEstB	Device specific	This field provides an indication of the device life time based on the amount of performed program/erase cycles. The calculation method is vendor specific and referred as method B. 00h: Information not available 01h: 0% - 10% device life time used 02h: 10% - 20% device life time used 03h: 20% - 30% device life time used 04h: 30% - 40% device life time used 05h: 40% - 50% device life time used 06h: 50% - 60% device life time used 07h: 60% - 70% device life time used 08h: 70% - 80% device life time used 09h: 80% - 90% device life time used 0Ah: 90% - 100% device life time used 0Bh: Exceeded its maximum estimated device life time Others: Reserved
05h	32	VendorPropInfo	Device specific	Reserved for Vendor Proprietary Health Report
25h	4	dRefreshTotalCount	Device specific	Total Refresh Count Indicate how many times the device complete refresh for the entire device. Incremented by 1 when dRefreshProgress reach 100000 (100.000%).
29h	4	dRefreshProgress	Device specific	Refresh Progress Indicate the refresh progress in %. dRefreshProgress will indicate 0.000% ~ 100.000% in dec. dRefreshProgress = 100000 (dec) when it complete 100.000%. dRefreshProgress = 1000 (dec) when it complete 1.000%. When this value reach 100000 (100.000%) 1. Device stops refreshing even if it did not complete the number of units specified by bRefreshUnit 2. dRefreshProgress shall be reset to 0 3. dRefreshTotalCount shall be incremented by 1 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.

14.1.4.15 Vendor Specific Descriptor

The detailed format of the Vendor Specific Descriptor is defined by the device manufacturer.

In a QUERY REQUEST UPIU, the Vendor Specific Descriptor is addressed by setting: DESCRIPTOR IDN = F0h to FFh. The value of INDEX and SELECTOR are determined by the vendor. For detailed definition, please refer to the device manufacturer datasheet.

Table 14.24 — Vendor Specific Descriptor

Offset	Size	Name	Value	Description
00h	1	bLength	Device Specific	Size of this descriptor = N
01h	1	bDescriptorIDN	F0h-FFh	Descriptor Type Identifier
02h	N-2	DATA	Device specific	Device specific. For detailed definition, please refer to the device manufacturer datasheet.

14.2 Flags

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

Read access property (read or write only) and write access property (read only, write only, persistent, etc.) are defined for each flag. Table 14.25 describes the supported access properties for flags.

Table 14.25 — 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.

Table 14.26 is a table of Flag Descriptor names and attributes that runs consecutively through the next several pages.

14.2 Flags (cont'd)

Table 14.26 — Flags

FLAGS					
IDN	**Name**	**Type**	**Type[1] / # Ind.[2] / # Sel.[3]**	**Default**	**Description**
00h	Reserved				
01h	fDeviceInit	Read / Set only	D	0	Device Initialization Host set fDeviceInit flag to initiate device initialization after boot process is completed. Device resets flag when device initialization is completed. 0b: Device initialization completed or not started yet. 1b: Device initialization in progress.
02h	fPermanentWPEn	Read / Write once	D	0	Permanent Write Protection Enable fPermanentWPEn enables permanent write protection on all logical units configured as permanent protected; it cannot be toggled or cleared once it is set. 00h: Permanent write protection disabled 01h: Permanent write protection enabled
03h	fPowerOnWPEn	Read / Power on reset	D	0	Power On Write Protection Enable fPowerOnWPEn enables the write protection on all logical units configured as power on write protected. If fPowerOnWPEn is equal to one and the device receives a Query Request to clear or toggle this flag, the Query Request shall fail and Response field shall be set to "F8h" (Parameter already written). The device shall set fPowerOnWPEn to zero in the event of power cycle or hardware reset. 0b: Power on write protection disabled. 1b: Power on write protection enabled.
04h	fBackgroundOpsEn	Read / Volatile	D	1	Background Operations Enable 0b: Device is not permitted to run background operations. 1b: Device is permitted to run background operations.

14.2 Flags (cont'd)

Table 14.26 — Flags

FLAGS					
IDN	**Name**	**Type**	**Type[1] / # Ind.[2] / # Sel.[3]**	**Default**	**Description**
05h	fDeviceLifeSpanModeEn	Read / Volatile	D	0	Device Life Span Mode 0b: Device Life Span Mode is disabled. 1b: Device Life Span Mode is enabled. For more details, see 13.4.14.
06h	fPurgeEnable	Write only / Volatile	D	0	Purge Enable 0b: Purge operation is disabled. 1b: Purge operation is enabled. This flag shall only be set when the command queue of all logical units are empty and the bPurgeStatus is 00h (Idle). fPurgeEnable is automatically cleared by the UFS device when the operation completes or an error condition occurs. fPurgeEnable can be cleared by the host to interrupt an ongoing purge operation.
07h	fRefreshEnable	Write only / Volatile	D	0	Refresh Enable 0b: Refresh operation is disabled. 1b: Refresh operation is enabled. 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.
08h	fPhyResourceRemoval	Read / Persistent	D	0	Physical Resource Removal The host sets this flag to one to indicate that the dynamic capacity operation shall commence upon device EndPointReset or hardware reset. The device shall reset this flag to zero after completion of dynamic capacity operation. The host cannot reset this flag.

14.2 Flags (cont'd)

Table 14.26 — Flags

FLAGS					
IDN	**Name**	**Type**	**Type[1] / # Ind.[2] / # Sel.[3]**	**Default**	**Description**
09h	fBusyRTC	Read Only	D	0	Busy Real Time Clock 0b : Device is not executing internal operation related to RTC 1b: Device is executing internal operation related to RTC When this flag is set to one, it is recommended for the host to not send commands to the device
0Ah	Reserved	-	-	-	Reserved for Unified Memory Extension standard
0Bh	fPermanentlyDisableFwUpdate	Read / Write once	D	0	Permanently Disable Firmware Update 0b: The UFS device firmware may be modified 1b: The UFS device shall permanently disallow future firmware updates to the UFS device
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[4]	0	WriteBooster Enable 0b: WriteBooster is not enabled. 1b: WriteBooster is enabled.
0Fh	fWriteBoosterBufferFlushEn	Read / Volatile	A/LU/0 or D[4]	0	Flush the data in WriteBooster Buffer to the user area of storage. 0b: Flush operation is not performed. 1b: Flush operation is performed.
10h	fWriteBoosterBufferFlushDuringHibernate	Read / Volatile	A/LU/0 or D[4]	0	Flush WriteBooster Buffer during hibernate state. 0b: Device is not allowed to flush the WriteBooster Buffer during link hibernate state. 1b: Device is allowed to flush the WriteBooster Buffer during link hibernate state.
11h	Reserved	-	-	-	Reserved for Host Performance Booster (HPB) Extension Standard

14.2 Flags (cont'd)

Table 14.26 — Flags

FLAGS					
IDN	**Name**	**Type**	**Type[1] / # Ind.[2] / # Sel.[3]**	**Default**	**Description**
12h	Reserved	-	-	-	Reserved for Host Performance Booster (HPB) Extension Standard
13h	fUnpinEn	Read / Volatile	D	0	Enable unpin the pinned data 0h: Unpin Disable (pinned data is not flushed by WriteBooster Buffer flush operation) 1h: Unpin Enable (pinned data is flushed by WriteBooster Buffer flush operation)
14h-7Fh	Reserved	-	-	-	
80h-FFh	fVendorSpecific	Device Specific	Device Specific	Device Specific	These flags are reserved for vendor specific usage. Content and function of these flags may vary based on Manufacturer Name String Descriptor and Product Revision Level String Descriptor. For detailed definition of these flags please refer to the device manufacturer datasheet.

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 does not matter.

14.3 Attributes

An Attribute is a parameter that represents a specific range of numeric values that can be written or read. For example, the maximum Data In data packet size would be an attribute. Attribute size can be from 1-bit to 64-bit. Attributes of the same type can be organized in arrays, each element of them identified by an index. For example, in case of parameter that is logical unit specific, the LUN would be used as index.

Read access property (read or write only) and write access property (read only, write once, persistent, etc.) are defined for each attribute. Table 14.27 on the following consecutive pages describes the supported access properties for attributes

Table 14.27 — Attributes Access Properties

Access Property		Description
Read	Read	The attribute can be read.
	Write only	The attribute cannot be read.
Write	Read only	The attribute cannot be written.
	Write once	The attribute can be written only one time, the value is kept after power cycle or any type of reset.
	Persistent	The attribute can be written multiple times, the value is kept after power cycle or any type of reset event.
	Volatile	The attribute can be written multiple times. The attribute is set to the default value after power cycle or any type of reset event.
	Power on reset	The attribute is set to the default value after power cycle or hardware reset event. The attribute can written only one time after a power cycle or hardware reset, and it cannot be re-written until the next power cycle or hardware reset.

Table 14.28 is a table of Attribute Descriptor names and attributes that runs consecutively through the next several pages.

14.3 Attributes (cont'd)

Table 14.28 — Attributes

ATTRIBUTES							
IDN	Name	Access Property	Size	Type[1] / # Ind.[2] / # Sel.[3]	MDV[4]	Description	NOTE
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. When bBootEnable = 02h permanent-bootable is enabled. In this mode, a Query Request to change bBootLunEn value from 01h or 02h to 00h shall fail and and the Query Response field shall be set to "General failure (FFh)".	
01h	Reserved	-	-	-	-	Reserved for Host Performance Booster (HPB) Extension Standard	
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

14.3 Attributes (cont'd)

Table 14.28 — Attributes

ATTRIBUTES							
IDN	Name	Access Property	Size	Type[1] / # Ind.[2] / # Sel.[3]	MDV[4]	Description	NOTE
04h	bOutOfOrderDataEn	Read / Write once	1 byte	D	00h	Out of Order Data transfer Enable 00h: Out-of-order data transfer is disabled for both DATA IN and DATA OUT UPIUs 01h: Out-of-order data transfer is enabled for both DATA IN and DATA OUT UPIUs 02h: Out-of-order data transfer is enabled for DATA IN UPIUs only, and is disabled for DATA OUT UPIUs 03h: Out-of-order data transfer is enabled for DATA OUT UPIUs only, and is disabled for DATA IN UPIUs Others: Reserved This bit shall have effect only when bDataOrdering declares device support for the specific UPIUs enablement	
05h	bBackgroundOpStatus	Read only	1 byte	D	00h	Background Operations Status Device health status for background operation 00h: Not required 01h: Required, not critical 02h: Required, performance impact 03h: Critical. Others: Reserved	
06h	bPurgeStatus	Read only	1 byte	D	00h	Purge Operation Status 00h: Idle (purge operation disabled) 01h: Purge operation in progress 02h: Purge operation stopped prematurely 03h: Purge operation completed successfully 04h: Purge operation failed due to logical unit queue not empty 05h: Purge operation general failure. Others: Reserved When the bPurgeStatus is equal to the values 02h, 03h, 04h or 05h, the bPurgeStatus is automatically cleared to 00h (Idle) the first time that it is read.	

14.3 Attributes (cont'd)

Table 14.28 — Attributes

ATTRIBUTES							
IDN	**Name**	**Access Property**	**Size**	**Type[1]** / **# Ind.[2]** / **# Sel.[3]**	**MDV[4]**	**Description**	**NOTE**
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 sets are empty.	7, 8
08h	bMaxDataOutSize	Read / Persistent	1 byte	D	See Des-cription	Maximum Data-Out Size The maximum number of bytes that can be requested with a READY TO TRANSFER UPIU shall not be greater than the value indicated by this attribute. Value expressed in number of 512-byte units. bMaxDataOutSize shall not exceed the bMaxOutBufferSize parameter. bMaxDataOutSize = bMaxOutBufferSize when the UFS device is shipped. This parameter can be written by the host only when all LU task sets are empty.	8
09h	dDynCapNeeded	Read only	4 bytes	A / Number of LU specified by bMaxNumberLU (LUN) / 0	0000 0000h	Dynamic Capacity Needed The amount of physical memory needed to be removed from the physical memory resources pool of the particular logical unit, in units of bOptimalWriteBlockSize.	9

14.3 Attributes (cont'd)

Table 14.28 — Attributes

ATTRIBUTES							
IDN	Name	Access Property	Size	Type[1] / # Ind.[2] / # Sel.[3]	MDV[4]	Description	NOTE
0Ah	bRefClkFreq	Read / Persistent	1 byte	D	03h	Reference Clock Frequency value 0h:19.2 MHz 1h: 26 MHz 2h: 38.4 MHz 3h: 52 MHz (default) Others: Reserved This attribute value is used only when LSS is low. It is ignored when LSS is high	10
0Bh	bConfigDescrLock	Read / Write once	1 byte	D	00h	Configuration Descriptor Lock 0h: Configuration Descriptor not locked 1h: Configuration Descriptor locked Others: Reserved	
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 sets are empty.	
0Dh	wExceptionEventControl	Read / Volatile	2 bytes	D	0000h	Exception Event Control This attribute enables the setting of the EVENT_ALERT bit of Device Information field, which is contained in the RESPONSE UPIU. EVENT_ALERT is set to one if at least one exception event occured (wExceptionEventStatus[i]) and the corresponding bit in this attribute is one (wExceptionEventControl[i]). bit[0]: DYNCAP_EVENT_EN bit[1]: SYSPOOL_EVENT_EN bit[2]: URGENT_BKOPS_EN bit[3]: TOO_HIGH_TEMP_EN bit[4]: TOO_LOW_TEMP_EN bit[5]: WRITEBOOSTER_EVENT_EN bit[6]: PERFORMANCE_THROTTLING_EN bit[7]: DEVICE_LEVEL_EXCEPTION_EN bit[8]: WRITEBOOSTER_RESIZE_HINT_EN bit[9]: HEALTH_CRITICAL_EN bit[10]: PINNED_WRITE_BOOSTER_EVENT_EN bit[11~15]: Reserved	

14.3 Attributes (cont'd)

Table 14.28 — Attributes

ATTRIBUTES							
IDN	**Name**	**Access Property**	**Size**	**Type[1]** / **# Ind.[2]** / **# Sel.[3]**	**MDV[4]**	**Description**	**NOTE**
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]: DEVICE_LEVEL_EXCEPTION_OCCURRED bit[8]: WRITEBOOSTER_RESIZE_HINT bit[9]: HEALTH_CRITICAL bit[10]: PINNED_WRITE_BOOSTER_FULL bit[11~15]: Reserved	
0Fh	dSecondsPassed	Write only / Volatile	4 bytes	D	0000 0000h	Bits[31:0]: Seconds passed from TIME BASELINE (see wPeriodicRTCUpdate in Device Descriptor)	

14.3 Attributes (cont'd)

Table 14.28 — Attributes

ATTRIBUTES							
IDN	**Name**	**Access Property**	**Size**	**Type[1]** / **# Ind.[2]** / **# Sel.[3]**	**MDV[4]**	**Description**	**NOTE**
10h	wContextConf	Read / Volatile	2 bytes	A / Number of LU specified by bMaxNumberLU (LUN) / 15 (ID)	0000h	INDEX specifies the LU number. SELECTOR specifies the Context ID within the LU. Valid values are 01h – Fh. **Bit[15:8]: Reserved** **Bit[7:6]: Reliability mode** 00b: MODE0 (normal) 01b: MODE1 (non-Large Unit, reliable mode or Large Unit unit-by-unit mode) 10b: MODE2 (Large Unit, one-unit-tail mode) 11b: Reserved **Bit[5:3]: Large Unit multiplier** If Large Unit context is set, this field defines the Large Unit size, else it is ignored **Bit[2]: Large Unit context** 0b: Context is not following Large Unit rules 1b: Context follows Large Unit rules **Bit [1:0]: Activation and direction mode** 00b: Context is closed and it is no longer active 01b: Context is configured and activated as a write-only context. 10b: Context is configured and activated as a read-only context 11b: Context is configured and activated as a read/write context	
11h	Obsolete	-	-	-	-	-	
12h	Reserved	-	-	-	-	Reserved for Unified Memory Extension standard	
13h	Reserved	-	-	-	-	Reserved for Unified Memory Extension standard	

14.3 Attributes (cont'd)

Table 14.28 — Attributes

ATTRIBUTES							
IDN	Name	Access Property	Size	Type[1] / # Ind.[2] / # Sel.[3]	MDV[4]	Description	NOTE
14h	bDeviceFFUStatus	Read Only	1 byte	D	00h	Device FFU Status 00h: No information 01h: Successful microcode update 02h: Microcode corruption error 03h: Internal error 04h: Microcode version mismatch 05h-FEh: Reserved 0FFh: General Error	11
15h	bPSAState	Read / Persistent	1 byte	D	Device specific	00h: 'Off'. PSA feature is off. 01h: 'Pre-soldering'. PSA feature is on, device is in the pre-soldering state. 02h: 'Loading Complete' PSA feature is on. The host will set to this value after the host finished writing data during pre-soldering state. 03h: 'Soldered'. PSA feature is no longer available. Set by the Device to indicate it is in post-soldering state. This attribute unchangeable after it is in 'Soldered' state.	
16h	dPSADataSize	Read / Persistent	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 µs … FFh: 255 µs	12, 13

14.3 Attributes (cont'd)

Table 14.28 — Attributes

ATTRIBUTES							
IDN	Name	Access Property	Size	Type[1] / # Ind.[2] / # Sel.[3]	MDV[4]	Description	NOTE
18h	bDeviceCaseRoughTemperature	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 ºC~170 ºC) 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 Celsius. (i.e., 20 ºC~170 ºC) 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 Celsius. (i.e., -79 ºC~0 ºC) 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	

14.3 Attributes (cont'd)

Table 14.28 — Attributes

ATTRIBUTES							
IDN	**Name**	**Access Property**	**Size**	**Type[1]** / **# Ind.[2]** / **# Sel.[3]**	**MDV[4]**	**Description**	**NOTE**
1Ch	bWriteBoosterBufferFlushStatus	Read only	1 byte	A/LU/0 or D	0	Flush operation status of WriteBooster Buffer. 00h: idle. Device is not flushing the WriteBooster Buffer: either the WriteBooster Buffer is empty or a flush has not been initiated 01h: Flush operation in progress. The WriteBooster Buffer is not yet empty and a flush has been initiated. 02h: Flush operation stopped prematurely. The WriteBooster Buffer is not empty and the host stopped the in-progress flush. 03h: Flush operation completed successfully. 04h: Flush operation general failure Others : Reserved 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.	15

14.3 Attributes (cont'd)

Table 14.28 — Attributes

ATTRIBUTES							
IDN	Name	Access Property	Size	Type[1] / # Ind.[2] / # Sel.[3]	MDV[4]	Description	NOTE
1Dh	bAvailableWriteBoosterBufferSize	Read only	1 byte	A/LU/0 or D	0	Available WriteBooster Buffer Size This available buffer size is decreased by WriteBooster operation and increased by flush operation. Value expressed in unit of 10% granularity 00h: 0% buffer remains. 01h: 10% buffer remains. 02h~09h: 20%~90% buffer remains 0Ah: 100% buffer remains Others : Reserved The % reported by the attributes is remaining portion of the current WriteBooster Buffer size indicated by the dCurrentWriteBoosterBufferSize attribute. If bWriteBoosterBufferPartialFlushMode is set to 02h (i.e. Pinned Mode), this attribute indicates the WriteBooster available buffer size excluding the currently allocated size for the Pinned Write Booster Buffer.	15

14.3 Attributes (cont'd)

Table 14.28 — Attributes

ATTRIBUTES							
IDN	Name	Access Property	Size	Type[1] / # Ind.[2] / # Sel.[3]	MDV[4]	Description	NOTE
1E	bWriteBoosterBufferLifeTimeEst	Read only	1 byte	A/LU/0 or D	Device specific	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. The detailed calculation method is vendor specific. 00h: Information not available (WriteBooster Buffer is disabled) 01h: 0% - 10% WriteBooster Buffer life time used 02h: 10% - 20% WriteBooster Buffer life time used 03h: 20% - 30% WriteBooster Buffer life time used 04h: 30% - 40% WriteBooster Buffer life time used 05h: 40% - 50% WriteBooster Buffer life time used 06h: 50% - 60% WriteBooster Buffer life time used 07h: 60% - 70% WriteBooster Buffer life time used 08h: 70% - 80% WriteBooster Buffer life time used 09h: 80% - 90% WriteBooster Buffer life time used 0Ah: 90% - 100% WriteBooster Buffer life time used 0Bh: Exceeded its maximum estimated WriteBooster Buffer life time (write commands are processed as if WriteBooster feature was disabled) Others: Reserved	15

14.3 Attributes (cont'd)

Table 14.28 — Attributes

ATTRIBUTES							
IDN	Name	Access Property	Size	Type[1] / # Ind.[2] / # Sel.[3]	MDV[4]	Description	NOTE
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. If bWriteBoosterBufferPartialFlushMode is set to 02h (i.e. Pinned Mode), this attribute indicates the WriteBooster current buffer size excluding the currently allocated size for the Pinned Write Booster Buffer.	15
20h	bPreEraseSetCap	Write only / Volatile	1 byte	D	00h	Pre-erase capacity in GB. When set, UFS will pre-erase SLC blocks in idle time. This value shall not exceed bPreEraseMaxCap.	
21h	bPreEraseCurCap	Read only	1 byte	D	00h	The number of GB available for writing without NAND erasure.	
22h	bPreEraseMaxCap	Read only	1 byte	D	Device Specific	The maximum number of GB which may be set for Pre-Erase	
23h-29h	Reserved	-	-	-	-	-	
2Ah	bEXTIIDEn	Read/Write once	1 byte	D	00h	EXT_IID Enable 00h:EXT_IID is ignored 01h: EXT_IID is valid Others: Reserved	

14.3 Attributes (cont'd)

Table 14.28 — Attributes

ATTRIBUTES							
IDN	Name	Access Property	Size	Type[1] / # Ind.[2] / # Sel.[3]	MDV[4]	Description	NOTE
2Bh	wHostHintCache Size	Read / Persistent	2 byte	D	0000h	This attribute is set by the host to indicate the host controller Hint Cache size. wHostHintCacheSize indicates the maximum sum of outstanding Hint Data Count fields that device is allowed to send to the host controller. A hint is considered outstanding when it was provided to the host controller and the corresponding DATA IN UPIU / RTT UPIU was not transferred to the host controller. If the device sends hint that exceeds wHostHintCacheSize, it may lead to performance degradation since host may not be able to process command in optimal manner. Maximum total number of outstanding Hint Count fields = 32 x 2^wHostHintCacheSize. For example, if wHostHintCacheSize = 2, Maximum total number of outstanding Hint Data Count fields is 128 units, each unit correspond to 4 KB of data.	
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.	

14.3 Attributes (cont'd)

Table 14.28 — Attributes

ATTRIBUTES							
IDN	**Name**	**Access Property**	**Size**	**Type[1]** / **# Ind.[2]** / **# Sel.[3]**	**MDV[4]**	**Description**	**NOTE**
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	

14.3 Attributes (cont'd)

Table 14.28 — Attributes

ATTRIBUTES							
IDN	Name	Access Property	Size	Type[1] / # Ind.[2] / # Sel.[3]	MDV[4]	Description	NOTE
2Fh	bRefreshMethod	Read / Persistent	1 byte	D	00h	Refresh Method This parameter specifies the refresh operation method. 00h : Not defined 01h: Manual-Force 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). 02h: Manual-Selective 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. Others: Reserved	14
30h	qTimestamp	Write only	8 byte	D	00h	Timestamp in nanoseconds since January 1, 1970 Coordinated Universal Time (UTC)	
31h-33h	Reserved	-	-	-	-	Reserved for File Based Optimization (FBO) Extension Specification	
34h	qDeviceLevelExceptionID	Read only	8 bytes	D	Vendor specific	This attribute is to indicate the device level exception ID. For a detailed definition of this attribute, refer to the device manufacturer datasheet.	

14.3 Attributes (cont'd)

Table 14.28 — Attributes

ATTRIBUTES							
IDN	Name	Access Property	Size	Type[1] / # Ind.[2] / # Sel.[3]	MDV[4]	Description	NOTE
35h	bDefragOperation	Read / Volatile	1 byte	D	00h	HID Operations 00h: HID analysis and HID defrag operation are disabled. 01h: HID analysis is enabled. 02h: HID analysis and HID defrag operation are enabled. Others: Reserved When this attribute is set to 01h, the device starts to analyze the device's fragmentation status. When this attribute is set to 02h, the device starts the HID analysis operation and then executes the defragmentation operation. Note that if the device has already been analyzed, the device may start executing the HID defrag operation without further HID analysis operation. This attribute is automatically set to 0 by the device when the operation is completed or a stop condition occurs.	
36h	dHIDAvailableSize	Read only	4 byte	D	Vendor specific	HID Available Size The total fragmented size in the device is reported through this attribute. - FFFFFFFFh: Indicating no valid information available about the fragmented size. Analysis required. - Other values: Total fragmented size in units of 4 Kbyte When the HID analysis is completed, the device updates this attribute to indicate the total fragmented size in the device. The HID defrag operation is completed if the defragmentation is performed as much as the requested defragmentation size. (The requested defragmentation size is calculated as the minimum value of the size indicated by dHIDSize and the size indicated by dHIDAvailableSize.).	

14.3 Attributes (cont'd)

Table 14.28 — Attributes

ATTRIBUTES							
IDN	Name	Access Property	Size	Type[1] / # Ind.[2] / # Sel.[3]	MDV[4]	Description	NOTE
37h	dHIDSize	Read / Persistent	4 byte	D	FFFFFFFFh	HID Size The host sets the size to be defragmented by an HID defrag operation. The default value is 0xFFFFFFFFh indicating that the device defrag as much as possible. The value expressed in units of 4 Kbyte.	
38h	bHIDProgressRatio	Read only	1 byte	D	00h	HID Progress Ratio Defrag progress is reported by this attribute. I.e this attribute indicates the ratio of the completed defrag size over the requested defrag size. (The requested defrag size is calculated as the minimum value of the size indicated by dHIDSize and the size indicated by dHIDAvailableSize.) Value expressed in units of 1%. 00h: 0% 01h: 1% … 64h: 100% Others: reserved If bHIDState is set to 0h, this attribute is initialized to 00h. If bDefragOperation is set to 01h or 02h, this attribute is initialized to 00h. When this attribute is 64h, the attribute is cleared to 00h automatically right after reading this attribute by the host.	

14.3 Attributes (cont'd)

Table 14.28 — Attributes

ATTRIBUTES							
IDN	Name	Access Property	Size	Type[1] / # Ind.[2] / # Sel.[3]	MDV[4]	Description	NOTE
39h	bHIDState	Read only	1 byte	D	00h	HID State The HID state is reported by this attribute. 00h: Idle (Analysis Required because there is no valid information about fragmentation of the device) 01h: Analysis in Progress 02h: Defrag Required 03h: Defrag In Progress 04h: Defrag Completed 05h: Defrag is not required Others: Reserved If the host set bDefragOperation to 0 (i.e. disable HID operation), the bHIDState is set to idle state from any HID state. The bHIDState may be cleared to 00h after receiving any write-type command changing the medium status of the device. In this case, the host can read from time to time the value of bHIDState to verify the operation was terminated or the device managed to resume the HID operation.	
3Ah - 3Bh	Reserved	-	-	-	-	-	

14.3 Attributes (cont'd)

Table 14.28 — Attributes

ATTRIBUTES							
IDN	Name	Access Property	Size	Type[1] / # Ind.[2] / # Sel.[3]	MDV[4]	Description	NOTE
3Ch	bWriteBoosterBufferResizeHint	Read only	1 Byte	A/LU/0 or D	0	WriteBooster Buffer Resize Hint information This field indicates hint information about which type of resize for WriteBooster Buffer is recommended. 00h: Recommend keep the buffer size 01h: Recommend to decrease the buffer size 02h: Recommend to increase the buffer size Others: Reserved This field shall be cleared automatically when the device receives the WriteBooster Resize command.	15
3Dh	bWriteBoosterBufferResizeEn	Write only / Volatile	1 Byte	A/LU/0 or D	0	Enable WriteBooster Buffer Resize operation The host can decrease or increase the WriteBooster Buffer size by setting this attribute. 00h: Idle (There is no resize operation) 01h: Decrease WriteBooster Buffer Size 02h: Increase WriteBooster Buffer Size Others: Reserved The amount of increase or decrease in WriteBooster Buffer is determined by the device. If the WriteBooster Buffer size cannot be increased or decreased, the device will not change the WriteBooster Buffer size.	15

14.3 Attributes (cont'd)

Table 14.28 — Attributes

ATTRIBUTES							
IDN	Name	Access Property	Size	Type[1] / # Ind.[2] / # Sel.[3]	MDV[4]	Description	NOTE
3Eh	bWriteBoosterBufferResizeStatus	Read Only	1 Byte	A/LU/0 or D	0	Resize operation status of the Write Booster Buffer 00h: Idle (resize operation is not issued) 01h: Resize operation in progress 02h: Resize operation completed successfully 03h: Resize operation general failure Others: Reserved When a Resize operation is completed, the updated size of WriteBooster Buffer is set to dCurrentWriteBoosterBufferSize attribute. When the bWriteBoosterBufferResizeStatus is equal to the values 02h or 03h, the status value is cleared to 00h automatically right after bWriteBoosterBufferResizeStatus is read. If the host requests the resize operation, the status values of 00h, 02h, or 03h shall be set to 01h.	15
3Fh	bWriteBoosterBufferPartialFlushMode	Read / Persistent	1 Byte	D	0	WriteBooster Buffer Partial Flush Mode 00h: No partial flush 01h: FIFO(first-in-first-out) mode 02h: Pinned mode This field indicates the partial flush mode when WriteBooster Buffer Flush is performed. When this value is set to 01h, the latest written data as much as the FIFO WriteBooster Buffer is not flushed from the WriteBooster flush operation. When this value is set to 02h, the data in the Pinned WriteBooster Buffer is not flushed from the WriteBooster flush operation. When this value is set to 00h, all data in WriteBooster Buffer can be flushed.	

14.3 Attributes (cont'd)

Table 14.28 — Attributes

ATTRIBUTES							
IDN	Name	Access Property	Size	Type[1] / # Ind.[2] / # Sel.[3]	MDV[4]	Description	NOTE
40h	dMaxFIFOSizeForWriteBoosterPartialFlushMode	Read / Persistent	4 Bytes	A/LU/0 or D	0	Maximum FIFO Size for WriteBooster FIFO partial flush mode. Note that in the case of "preserve user space mode", depending on available user space remaining at run-time, storage blocks allocated for the WriteBooster Buffer can be returned to user space. In such a case, the currently allocated WriteBooster FIFO Buffer size is indicated by dCurrentFIFOSizeForWriteBoosterPartialFlushMode which can be less than indicated by this attribute. The value of the attribute is expressed in the unit of Allocation Units	15
41h	dCurrentFIFOSizeForWriteBoosterPartialFlushMode	Read Only	4 Bytes	A/LU/0 or D	0	Current FIFO Size for WriteBooster FIFO partial flush mode. Device shall manage this attribute to be less than dCurrentWriteBoosterBufferSize. The value of the attribute is expressed in the unit of Allocation Units	15
42h	dPinnedWriteBoosterBufferCurrentAllocUnits	Read Only	4 Bytes	D	0	The currently allocated size of Pinned WriteBooster Buffer which is excluded from the flush operation When the WriteBooster Buffer size decreases, the Pinned WriteBooster Buffer size may be decreased and can be less than the size by the dPinnedWriteBoosterBufferNumAllocUnits attribute. If this value is 00h (i.e. there is no Pinned buffer area), even the write command with Group Number set to 18h can be written to the Non-Pinned WriteBooster Buffer area, or to the normal storage if the Non-Pinned WriteBooster Buffer area is full or not configured. The value is expressed in the unit of Allocation Units.	

14.3 Attributes (cont'd)

Table 14.28 — Attributes

ATTRIBUTES							
IDN	Name	Access Property	Size	Type[1] / # Ind.[2] / # Sel.[3]	MDV[4]	Description	NOTE
43h	bPinnedWriteBoosterBufferAvailablePercentage	Read Only	1 Byte	D	0	Available Pinned WriteBooster Buffer size in Percentage This attribute indicates the remaining portion over the current Pinned WriteBooster Buffer size indicated by the dPinnedWriteBoosterBufferCurrentAllocUnits attribute. The value expressed in 10% granularity 00h: 0% buffer remains. 01h~09h: 10%~90% buffer remains. 0Ah: 100% buffer remains. Others: Reserved This value is decreased by the WRITE command whose Group Number is set to 18h and increased by WriteBooster flush operation.	
44h	dPinnedWriteBoosterCummulativeWrittenSize	Read Only	4 Bytes	D	0	Cumulative size is written in the Pinned Write Booster Buffer. When this value reaches its maximum value (0xffffffff), it does not increase. The value is expressed in the unit of 10 MB.	
45h	dPinnedWriteBoosterBufferNumAllocUnits	Read / Persistent	4 Bytes	A/LU/0 or D	0	Size for Pinned WriteBooster Buffer area in Pinned partial flush mode. This value shall be in the unit of Allocation Units.	15
46h	dNonPinnedWriteBoosterBufferMinNumAllocUnits	Read / Persistent	4 Bytes	A/LU/0 or D	0	Minimum size for Non-Pinned WriteBooster Buffer area in Pinned partial flush mode. This value shall be in the unit of Allocation Units.	15

14.3 Attributes (cont'd)

Table 14.28 — Attributes

ATTRIBUTES							
IDN	Name	Access Property	Size	Type[1] / # Ind.[2] / # Sel.[3]	MDV[4]	Description	NOTE
47h	qTxEQGnSettings	Read / Persistent	8 Bytes	A/HS-GEARs/ 2	0	High-Speed Gear TX Equalization settings of UFS Host and UFS Device. This attribute provides UFS Host a non-volatile memory area to store optimal TX Equalization settings for any supported High-Speed Gears. bit[63:56]: RFU bit[55:52]: Host TX Logical LANE 1 DeEmphasis bit[51:48]: Host TX Logical LANE 0 DeEmphasis bit[47:40]: RFU bit[39:36]: Host TX Logical LANE 1 PreShoot bit[35:32]: Host TX Logical LANE 0 PreShoot bit[31:24]: RFU bit[23:20]: Device TX Logical LANE 1 DeEmphasis bit[19:16]: Device TX Logical LANE 0 DeEmphasis bit[15:8]: RFU bit[7:4]: Device TX Logical LANE 1 PreShoot bit[3:0]: Device TX Logical LANE 0 PreShoot	16,17
48h	wTxEQGnSettingsExt	Read / Persistent	2 Bytes	A/HS-GEARs/ 2	0	High-Speed Gear TX Equalization settings extension. This attribute provides UFS Host a non-volatile memory area to store Pre-Coding enable indications. bit[15:6]: RFU bit[5]: PreCodeEn for Device RX Logical LANE 1 bit[4]: PreCodeEn for Device RX Logical LANE 0 bit[3:2]: RFU bit[1]: PreCodeEn for Device TX Logical LANE 1 bit[0]: PreCodeEn for Device TX Logical LANE 0	16,17
49h-7Fh	Reserved	-	-	-	-	-	

14.3 Attributes (cont'd)

Table 14.28 — Attributes

ATTRIBUTES							
IDN	**Name**	**Access Property**	**Size**	**Type[1] / # Ind.[2] / # Sel.[3]**	**MDV[4]**	**Description**	**NOTE**
80h-FFh	VendorSpecific	Device Specific	Device Specific	Device Specific	Device Specific	These attributes are reserved for vendor specific usage. Content and function of these attributes may vary based on Manufacturer Name String Descriptor and Product Revision Level String Descriptor. For detailed definition of these attributes please refer to the device manufacturer datasheet.	

14.3 Attributes (cont'd)

Table 14.28 — Attributes

ATTRIBUTES							
IDN	Name	Access Property	Size	Type[1] / # Ind.[2] / # Sel.[3]	MDV[4]	Description	NOTE

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, 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.

NOTE 16 HS-GEARs is the amount of valid values for the INDEX field in REQUEST/RESPONSE UPIU. The attribute of HS-GEARn is addressed setting INDEX = n-1, for example the attribute of HS-GEAR1 is addressed setting INDEX = 00h.

NOTE 17 qTxEQGnSettings and wTxEQGnSettingsExt include the definition of two SELECTOR values. The use of these SELECTOR values is implementation specific. For example, they could be used for default and backup copies of EQ settings.

15 UFS Mechanical Standard

UFS discrete and multi-chip ballouts are defined in [JESD21].

Annex A (Informative) Dynamic Capacity Host Implementation Example

A.1 Overview

This standard defines the Dynamic Capacity feature to enable a UFS device to regain at least partial functionality at the end-of-life where the defect level of the storage medium has accumulated to the point where the device can no longer maintain normal functionality.

Dynamic Capacity operation allows the device, at the direction of the host system, to remove physical memory resources from the resource pool dedicated for data storage and re-task the resources for device internal utility, thus restoring device functionality.

The net result of the Dynamic Capacity operation is a reduction of the usable storage space in the physical medium while the logical address space remains the same as before – i.e., the physical storage is less than the logical address space. It is the responsibility of the host system to keep track of the reduction in physical storage to maintain normal operation in its file system.

This application note outlines a method for the host file system to account for the reduction in physical storage with minimal impact.

A.2 Method Outline

1. The host system receives notification from the device in the Device Information parameter in Response UPIU that the device requires physical memory to be freed up from the storage space to continue operation.
2. The host reads the bDynCapNeeded[LUN] attributes and the bOptimalWriteBlockSize parameter in the Device Descriptor to determine how much physical memory resources needs to be freed up in each logical unit.
3. The host identifies the logical block address range(s) in the file system where the data can be discarded/erased to free up the physical memory resources. The host then uses the UNMAP command to unmap (deallocate) the LBA range(s), and initiates the Dynamic Capacity operation by setting the fPhyResourceRemoval flag and resetting the UFS device.
4. The host can mark the particular LBA range(s) as unusable in its file system by the means of dummy file(s) to ensure these LBAs will not be used in future write operations. The unusable LBAs marked by dummy file(s) match the reduction of physical storage, therefore from the host system perspective, the file system is intact.
5. The host can further backup the unusable LBA information by storing the information in the system area in case the file system of the main data storage logical unit is corrupted.

Annex B (Informative) Reference Clock Measurement Procedure

In order to verify that the Jitter of the Reference Clock meets the requirements of Table 6.5, two procedures are outlined in this annex, one for the measurement of the Random Jitter, the other for the measurement of the Deterministic Jitter.

Previous versions of this standard used the phase noise parameter as a measure of the quality of the reference clock signal. This parameter has been converted to random RMS jitter in this version of the standard so that it can be measured with a real-time Digital Storage Oscilloscope's (DSO) jitter separation tool.

B.1 Random RMS Jitter Measurement Procedure

The Table B.1 represents the maximum random RMS jitter amount by different UFS reference clock frequencies.

Table B.1 — Maximum Random RMS Jitter Amount

Reference Clock Frequency [MHz]	Integrated Phase Noise Power (Max) [dBc]	RMS Jitter (Max) [ps]
19.2	-66	5.9
26.0	-66	4.6
38.4	-66	3.5
52.0	-66	2.8

In order to comply with the UFS Standards' jitter parameters, it is mandated that the random RMS jitter value of the DSO shall be less or equal than the value of Table B.1. To perform this test that complies with the phase noise parameter, it is recommended to use the TIE (time interval error) bandpass filter that is directly correlated to the specification, from 50 kHz to the Nyquist frequency of the reference clock, when extracting random RMS jitter value.

Note that, to correlate measurement difference between phase noise measurement equipment and jitter measurement equipment (DSO), the DSO should have a bandwidth limit to 2 GHz so that DSO cannot add noise from wider bandwidth input above 2 GHz, the DSO has a sampling rate between 10 Gsa/s to 20 Gsa/s.

For consistency purpose, it is necessary to specify what kind of reference clock waveforms are to be used for this measurement. The reference clock can be a burst type of waveform or a continuous type of waveform depending on the operation, this measurement should be performed while DUT sends a continuous reference clock signal and acquires at least 500 Kcycles of the clock period.

Note that this test is required for the DUT that supports HS Gear 3, 4, and 5 speed only. When DUT operates at HS Gear5 speed, the 19.2 MHz and 26.0 MHz reference frequencies shall not be used, so the test is performed on the 38.4 MHz and the 52.0 MHz reference cases.

The value of the reference clock's random RMS jitter for all supported reference clock frequency should be less than or equal to the maximum value specified in Table B.1 above in order to be considered conforming.

B.1 Random RMS Jitter Measurement Procedure (cont'd)

Test Setup (see B.4).

Test Procedure

1. Connect the DUT's reference clock output to the DSO with a cable.
2. Configure the DUT to transmit a continuous reference clock signal.
3. Capture more than 500 Kcycles of clock period waveform.
4. Measure the RJ rms value as described above.
5. Repeat steps 1 to 4 for other reference clock frequencies (if supported).

Observable Results
For all test cases:

- Verify that every reference clock's RJ rms is below or equal to the RMS jitter value.

B.2 Deterministic Jitter Measurement Procedure

The UFS specification needs to test the reference clock's deterministic jitter for UFS devices that supports HS Gear3, 4, and 5 speed. This document covers how to test the UFS reference clock deterministic portion of jitter with a digital storage oscilloscope (DSO).

Same as in the previous random jitter test, it is recommended to use the TIE bandpass filter that is directly correlated to the specification, from 50 KHz to the Nyquist frequency of the reference clock, when extracting random RMS jitter value.

Note that, to correlate measurement results of random jitter, DSO should have a bandwidth limit to 2 GHz so that DSO cannot add noise from wider bandwidth input above 2 GHz, while DSO has a sampling rate between 10 Gsa/s to 20 Gsa/s.

For consistency purpose, it is necessary to specify what kind of reference clock waveforms are to be used for this measurement. Because, the reference clock can be a burst type of waveform or a continuous type of waveform depending on the operation, this measurement should be performed while DUT sends a continuous reference clock signal and acquires at least 500 Kcycles of the clock period.

The value of the reference clock's Deterministic jitter for all supported reference clock frequency should be less than or equal to 15 ps in order to be considered conformant.

Test Setup (see B.4).

Test Procedure

1. Connect the DUT's reference clock output to the DSO with a cable.
2. Configure the DUT to transmit a continuous reference clock signal.
3. Capture more than 500 Kcycles of clock period waveform.
4. Measure deterministic jitter (DJ) value as described above.
5. Repeat steps 1 to 4 for other reference clock frequencies (if supported).

Observable Results
For all test cases:

- Verify that every reference clock's DJ is below or equal to 15 ps.

B.3 Equipment Requirements

In order to perform the UFS reference clock test, the following resources are required:

- 1 x UFS Host DUT, properly mounted on an SMA based (or similar lab-grade connector) evaluation PCB.
- 1 x Real-Time Digital Storage Oscilloscope (DSO), 2 GHz bandwidth minimum with 10-20 GHz sampling rate.
- 1 x Short SMA test cable for connecting evaluation PCB and oscilloscope.

B.4 Test Setup

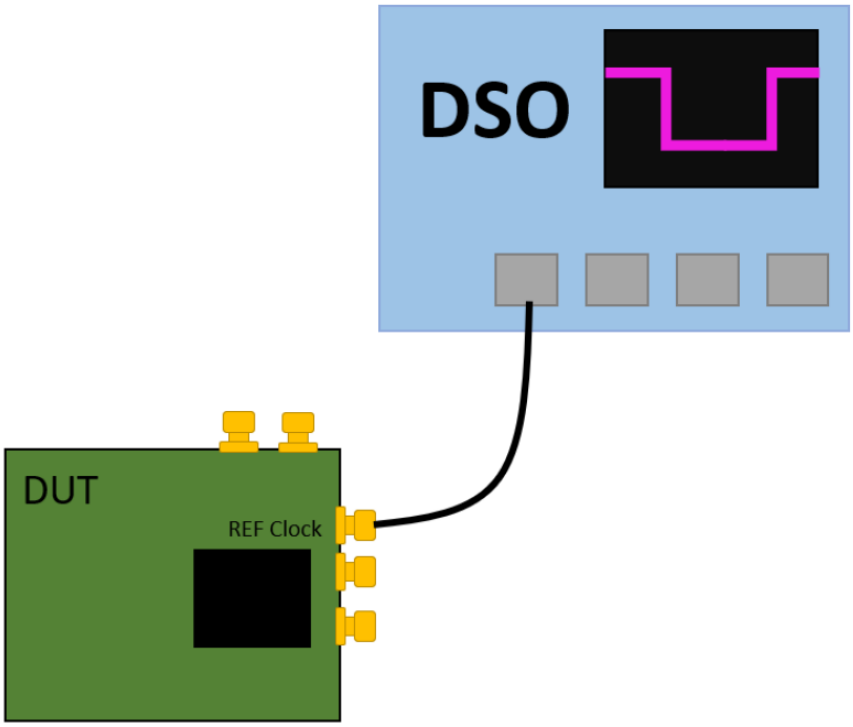


Figure B.1 — Test Setup

Notes on the Test Setup:

- The DUT is connected to the DSO using low loss 50 Ω SMA or equivalent lab grade cable.
- The cable measures the single-ended reference clock signal with respect to PCB ground (VSS).
- The DUT should be configured to transmit continuous reference clock signaling for consistency purpose.
- The DSO vertical gain should be optimized so that the signaling spans as much of the vertical height of the DSO screen as possible to reduce jitter measurement error.
- The DSO bandwidth should be limited to 2 GHz, so that the DSO does not include unnecessary noise from higher bandwidth.

Annex C (Informative) Reference Clock Detection in HS-LSS : An Implementation Example

C.1 Overview

This standard defines HS-LSS for faster initialization. In HS-LSS, the reference clock should be identified by device automatically.

This application note outlines a method for the UFS device to easily identify the current reference clock frequency given by the host at M-PHY layer initialization stage.

C.2 Method Outline

1. The device has two counters, called SYS_CNT and REF_CNT, and reset those counters at the same time.
2. The SYS_CNT counter is clocked by the UFS device internal system clock SYS_CLK.
3. The REF_CNT counter is clocked by Reference Clock REF_CLK provided by the host.
4. Since the UFS device knows its internal system clock frequency, the UFS device can identify the reference clock frequency given by the host, based on the ratio of the counter value of SYS_CNT and REF_CNT.

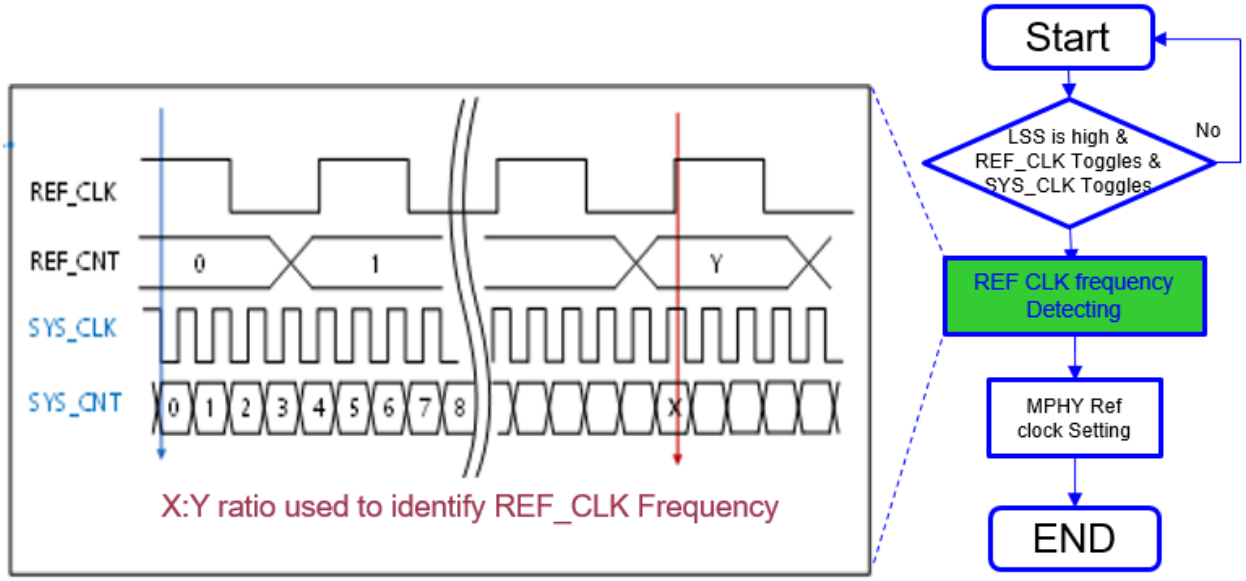


Figure C.1 — REF CLK Frequency Detection in HS-LSS

Annex D (Informative) Bibliography

[1] Rosenblum, Mendel, and John K. Ousterhout. "The Design and Implementation of a Log-structured File System." ACM Transactions on Computer Systems (TOCS) 10, no. 1 (1992): 26-52.

Annex E (Informative) Differences between Revisions

E.1 Differences Between JESD220H and its Predecessor JESD220G (December 2024)

It should be noted that the change of SCSI content from explicit to referenced in this revision of the standard (see E.1.3) had a substantial effect on the size of the document. 10 Figures were removed, as well as 100 Tables, along with their associated main body text. Annex D *(Informative) Board Design Guideline*, was also removed from the prior revision, replaced with a Bibliography. The page count dropped from 542 pages in the prior revision, JESD220G, to 444 pages in this revision, JESD220H.

E.1.1 NEW Features

The following items were added:

- Third party copy added via EXTENDED COPY, POPULATE TOKEN, WRITE USING TOKEN, and RECEIVE ROD TOKEN command support (see 11.3) with Third-Party Copy VPD page (see 11.5)
- HS-Gear6 adopted for increased bandwidth capability (see Table 6.7)
- Isolaged VCCQ_M power balls adapted from VCCQ, featuring more conditioned power for HS-G6 (see Table 6.1)
- Pre-erase feature (see 13.4.22)
- Descriptors, Flags, and Attributes Aggregation (see 10.7.8.13)
- Zoned Storage for UFS (see 13.4.23)
- Write Order Restore for ZUFS (see 13.4.23.6)
- Link Equalization Attribute Storage (see qTxEQGnSettings and wTxEQGnSettingsExt)
- Operating Voltage Clarifications (see 6.8)
- Per-command WriteBooster (see 13.4.18)

E.1.2 Changes to Clause 2 "Normative References"

- MIPI M-PHY 6.0
- MIPI UniPro 3.0
- SCSI Architecture Model 6
- SCSI Primary Commands 6
- SCSI Block Commands 5
- SCSI Zoned Block Commands 2 added
- Updated JEDEC standards to latest revision, JEP106BM and JESD21C

E.1.3 Changes to Features Already Defined in UFS 4.1

- Annex D “(Informative) Board Design Guideline” was removed, and replaced with a Bibliography
- UFS-DeepSleep power mode was removed
- RZQ1 nominal value defined (see Table 6.11)
- Cleanup of “shall” statements directed toward the host (various locations)
- Revision of SCSI content to reference the new SCSI normative references, and not copy, paraphrase, or duplicate content from those standards (various locations)
- Adaptation of GROUP NUMBER to include IO Advice Hints SCSI functionality (see 11.3.13)

E.2 Differences Between JESD220G and its Predecessor JESD220F (August 2022)

E.2.1 NEW Features

The following items were added:

- Device Level Exception Event (see 13.4.12.1)
- Host Initiated Defragmentation (see 13.4.20)
- Device Health Exception (see 13.4.12)
- WriteBooster buffer Resizing (13.4.18.6)
- Partial Flush Modes of WriteBooster (13.4.18.7)
- bBootLunEn attribute protection improvement (see 13.1.2)
- Authentication via RPMB in for vendor specific commands (see 12.4.7.3.9-10 and 12.4.7.4.9-10)
- Fast Recovery Mode (see 13.4.21)

E.2.2 Changes in Clause 2 "Normative References"

Updated the following references cited in this document:

- UniPro 2.0 publication date
- JEDEC publication references are updated to their respective latest revision and publication date

E.2.3 Changes in Features Already Defined in UFS 4.0

- VCCQ Absolute DC Max alignment to NAND interface standard (see 6.7)
- Defined the number of simultaneous operations allowed in RPMB (see 13.4.13)
- Added precision in Capacity Adjustment Factors (see 13.2.3 and 14.1.4.4)
- Authenticated Secure Write Protect Configuration Block Write LUN mismatch error in Advanced RPMB mode (see 12.4.7.4.5)
- 13.4.18.5 sub-clause title changed from Preserve User Space Option to User Space Modes

Minor editorial clarifications were added, typographic errors were corrected, and minor formatting changes were made to comply with *Style Manual for Standards and Other Publications of JEDEC*, *JM7A*, in addition to what is summarized above.

E.3 Changes between JESD220F and its Predecessor JESD220E (January 2020)

E.3.1 NEW Features or New Definitions

The following items were added:

- Optional LSS signal added for HighSpeed-LinkStartup control, see 6.1
- Electrostatic Discharge Sensitivity Characteristics added, see 6.1
- M-PHY HS reference clock operating requirements added, see 6.4
- REF_CLK frequency detection added for HS-LS, see 6.4
- Optional RZQ1 and RZQ2 added, see 6.6
- AC noise specification added, see 6.8
- EXT_IID added to extend Initiator ID, see 10.6.2(k)
- BARRIER Command added, see 11.3.29. 13.4.19
- RPMB Purge added, see 12.2.2.4.1
- Advanced RPMB using EHS added, see 12.4
- qTimestamp available for tagging Error Log entries, see 13.4.9
- Added Vendor Specific Descriptor, see 14.1.4.15
- Added Vendor Specific Flags, see 14.2
- Reserved Attribute 01h for HPB Extension, see 14.3
- Defined wHostHintCacheSize Attribute, see 14.3
- Added Vendor Specific Attributes, see 14.3
- Added Annex B (informative) Reference Clock Measurement Procedure
- Added Annex C (informative) Reference Clock Detection in HS-LSS
- Added Annex D (informative) Board Design Guideline

E.3.2 Changes in Clause 2 "Normative Reference"

Updated the following document references:

- M-PHY® specification: from version 4.1 to version 5.0
- Unified Protocol (UniPro®) specification: from version 1.8 to version 2.0
- Added UFS File Based Optimization (FBO) Extension Specification

E.3.3 Changes in Features Already Defined in UFS 3.1

- M-PHY HS-Gear5 is mandatory, see 4.1
- M-PHY LS support reduced to Gear1 only, see 4.1
- Support for VCC = 3.3V dropped, see 4.2, 6.3
- BER reduced from under 10-10 to 10-12, see 4.2
- Reference clock frequency of 52.0MHz reintroduced, now as default, see 6.4
- Reference clock phase noise parameter refined as Random Jitter and Deterministic Jitter, see 6.4
- Power Parameter element format change, see 7.4.3
- UFS PHY capability listed attributes reduced to set that differs from M-PHY 5.0, see 8.7
- UFS Target Port Identifiers updated, see 9.1
- Extended Header Segments enabled, see 10.6
- Out of order data sequencing with added hinting, see 10.7.14
- READ BUFFER command available at Boot Ready for Error Log, see 13.1.3.3
- Existing RPMB renamed Normal RPMB, see 12.4
- Clarified bDataOrdering in Geometry Descriptor, see 14.1.4.4
- RPMB bLogicalBlockSize and qLogicalBlockCount adapted for use with Advanced RPMB, see 14.1.4.6
- Further definition of OEM ID STRING DESCRIPTOR, see 14.1.4.11
- Clarified bOutOfOrderDataEn Attribute, see 14.3
- Moved (informative) Differences between Specification Revisions from Annex B to Annex E

Several clarifications were added and editorial changes were implemented in addition to what is summarized in this annex.

E.4 Changes between JESD220E and its Predecessor JESD220D (January 2018)

E.4.1 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 13.4.17 “Performance Throttling Event Notification”
- WriteBooster, see 13.4.18 “Write”

E.4.2 Changes in Clause 2 “Normative Reference”

Added the following documents:

- Application Note for UniPro® v1.8
- UFS Host Performance Booster (HPB) Extension Specification

E.4.3 Changes in Features Already Defined in UFS 3.0

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 13.4.12 “Exception Events Mechanism”.

Several clarifications were added and editorial changes were implemented in addition to what is summarized in this annex.

E.5 Changes between JESD220D and its Predecessor JESD220C (March 2016)

E.5.1 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(R) 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(R) 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

E.5.2 Changes in Clause 2 “Normative Reference”

- M-PHY(R) 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 Multi-chip Packages (MCP) and Discrete eMMC, e2MMC, and UFS, JESD21C

E.5.3 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 is summarized in this annex.

E.6 Changes between JESD220C and its Predecessor JESD220B (September 2013)

E.6.1 New Features or New Definitions

The following items were added

- Multi-initiator, see 10.6.2 “Basic Header Format”
- Command priority, see 10.7.1 “COMMAND UPIU”
- Field Firmware Update, see 11.3.28 “WRITE BUFFER Command”
- Vital product data parameters see 11.5
- Secure write protection, see
  - 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 13.4.13
- Production State Awareness, see 13.6
- Product Revision Level String Descriptor, see 14.1.4.13
- Device Health Descriptor, see 14.1.4.14

E.6.2 Changes in Clause 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

E.6.3 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.

E.6.3 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 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 sub-clause was in clause 13, while in UFS 2.1 it was moved in clause 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.

E.6.3 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 sub-clause, 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 is summarized in this annex.

E.7 Changes between JESD220B and its Predecessor JESD220A (June 2012)

E.7.1 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”

E.7.2 Changes in Clause 2 “Normative Reference”

- M-PHY^{SM} specification: from version 2.00.00 to version 3.0.
- Unified Protocol (UniPro^{SM}) specification: from version 1.41.00 to version 1.6.

E.7.3 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.”.

E.7.3 Changes in Features Already Defined in UFS 1.1 (cont'd)

- 7.4.1 "Device Power Modes", Clarified logical unit response to SCSI commands for each power mode. UFS-Sleep power mode: added the sentence: "VCC power supply should be restored before issuing START STOP UNIT command to request transition to Active power mode or PowerDown power mode.". Figure "Power Mode State Machine": added Powered On power mode, arrow from Active to Pre-Sleep with "bInitPowerMode = 00h", and some notes. Defined all power mode transitions. Added the sentence: "The effects of concurrent power mode changes requested by START STOP UNIT commands with the IMMED bit set to one are vendor specific.". Added the sentence: "A START STOP UNIT command with the IMMED bit set to zero causing a transition to Active, Sleep, or PowerDown power modes shall not complete with GOOD status until the device reaches the power mode specified by the command.". Added 7.4.1.9 "Device Well Known Logical Unit Responses to SCSI commands"
- New 7.4.4 "Logical Unit Power Condition"
- 8.6 "UFS PHY Attributes", Added the following M-PHY[SM] Attributes: TX_Hibern8Time_Capability, TX_Advanced_Granularity_Capability, TX_Advanced_Hibern8Time_Capability, TX_HS_Equalizer_Setting_Capability, RX_Hibern8Time_Capability, RX_PWM_G6_G7_SYNC_LENGTH_Capability, RX_HS_G2_SYNC_LENGTH_Capability, RX_HS_G3_SYNC_LENGTH_Capability, RX_HS_G2_PREPARE_LENGTH_Capability, RX_HS_G3_PREPARE_LENGTH_Capability, RX_Advanced_Granularity_Capability, RX_Advanced_Hibern8Time_Capability, RX_Advanced_Min_ActivateTime_Capability.
- 9.6.6 "UniPro Attributes", Added the UniPro[SM] PA_MaxDataLanes constant, PA_AvailTxDataLanes and PA_AvailRxDataLanes Attributes.
- Table 10.20 — 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."

E.7.3 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 × 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 is summarized above.

.

This page intentionally left blank.

STANDARD IMPROVEMENT FORM **JEDEC JESD220H**

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 10th Street North
Suite 240S
Arlington, VA 22201

Email: angies@jedec.org

1. I recommend changes to the following:

☐ Requirement, clause number _____

☐ Test method number _____ Clause number _____

The referenced clause number has proven to be:

☐ Unclear ☐ Too Rigid ☐ In Error

☐ Other _____

2. Recommendations for correction:

_____

_____

_____

_____

3. Other suggestions for document improvement:

_____

_____

_____

_____

Submitted by

Name: _____ Phone: _____

Company: _____ E-mail: _____

Address: _____

City/State/Zip: _____ Date: _____


JEDEC®