

**Working
Draft****Project
T10/BSR-INCITS 503**

**Revision 02a
February 19, 2016**

**Information technology -
SCSI Stream Commands - 5 (SSC-5)**

This is an internal working document of T10, a Technical Committee of Accredited Standards Committee INCITS (InterNational Committee for Information Technology Standards). As such this is not a completed standard and has not been approved. The contents may be modified by the T10 Technical Committee. The contents are actively being modified by T10. This document is made available for review and comment only.

Permission is granted to members of INCITS, its technical committees, and their associated task groups to reproduce this document for the purposes of INCITS standardization activities without further permission, provided this notice is included. All other rights are reserved. Any duplication of this document for commercial or for-profit use is strictly prohibited.

T10 Technical Editor:

Curtis Ballard
Hewlett Packard Enterprise
3404 E. Harmony Road
Fort Collins, CO 80528
USA

Telephone: 1-970-898-3013
Email: Curtis.Ballard@hpe.com

Reference number
ISO/IEC xxxx-xxx : 201x
BSR INCITS 503-201x

Points of Contact:**International Committee for Information Technology Standards (INCITS) T10 Technical Committee****T10 Chair**

John B. Lohmeyer
Avago Technologies
4420 Arrows West Drive
Colorado Springs, CO 80907-3444
USA

Telephone: 1-719-533-7560
Email: lohmeyer@t10.org

T10 Vice-Chair

Bill Martin
Samsung Semiconductor Inc
7213 Marblethorpe Drive
Roseville, CA 95747
USA

Telephone: 1-916-765-6875
Email: bill.martin@ssi.samsung.com

T10 Web Site <http://www.t10.org>

T10 Reflector To subscribe send e-mail to majordomo@T10.org with 'subscribe' in message body
To unsubscribe send e-mail to majordomo@T10.org with 'unsubscribe' in message body
Internet address for distribution via T10 reflector: T10@T10.org

INCITS Secretariat

1101 K Street NW
Suite 610
Washington, DC 20005
Email:incits@itic.org

Telephone: 1-202-737-8888
Facsimile: 1-202-638-4922

INCITS Web Site <http://www.INCITS.org>

Document Distribution Web Site <http://www.incits.org/standards-information/purchase-standards>

Revision Information

Changes in the SCSI standards family list, clause 1, are never marked with change bars. Changes in other clauses may be marked with change bars in minor (e.g., letter revisions such as SSC5r01c) revisions but are never marked with changes bars in major revisions.

1 Approved Documents Included

No T10 approved proposals are included in SSC-5 r0. SSC-5 r0 is identical to SSC-4 r03 except that editorial changes made by the ANSI Editor as part of the INCITS Public Review for SSC-4 appear in SSC-5 r0.

The following T10 approved proposals have been incorporated in SSC-5 up to and including this revision:

Table 1 — Incorporated T10 Approved Documents (in document number order)

Doc	In Rev	Document Title	Document Author
13-049r3	2	Logical Block Protection Length error example	Kevin Butt, IBM
13-266r6	1	SSC-5: Recommended Access Order	Kevin Butt, IBM
13-273r3	1a	SSC-5: Dynamic Runtime Information	Kevin Butt, IBM
14-148r2	1	SAM-5 SPC-5 SBC-4 SSC-5SPL-4 Redefinition of Peripheral device type	George Penokie, Avago Technologies
14-179r3	1a	CRC32C Polynomial for LBP Simplified	Kevin Butt, IBM
14-226r0	1	SSC-5 Removing CBCS	Curtis Ballard, Hewlett-Packard
14-227r0	1	SSC-5 Resolving editorial issues with Recommended Access Order	Curtis Ballard, Hewlett-Packard
15-082r1	1a	SSC-5 Modify 13-273 Dynamic Runtime Attributes for Service Actions	Curtis Ballard, Hewlett-Packard
15-080r1	2	Informational Exception MP clean-up	Kevin Butt, IBM
15-296r1	2a	SSC-5 Logical Block Protection VPD page	Curtis Ballard, Hewlett Packard Enterprise

2 Revision History

2.1 Revision 0 (4 February, 2014)

Revision 0 of SSC-5 is substantially equal to revision 3 of SSC-4. The only differences arise from changes made in SSC-4 by the ANSI Editor during the INCITS Public Review process.

2.2 Revision 1 (11 September, 2014)

The following T10 approved proposals were incorporated in SSC-5 revision 1:

- 14-148r2 SAM-5 SPC-5 SBC-4 SSC-5SPL-4 Redefinition of Peripheral device type [George Penokie, Avago Technologies]
- 13-266r6 SSC-5: Recommended Access Order [Kevin Butt, IBM]
- 14-226r0 SSC-5 Removing CBCS [Curtis Ballard, Hewlett-Packard]
- 14-227r0 SSC-5 Resolving editorial issues with Recommended Access Order [Curtis Ballard, Hewlett-Packard]

2.3 Revision 1a (25 March, 2015)

The following editorial changes were incorporated in SSC-5 revision 1a:

Changed multi-byte fields in tables to match updated T10 documentation style

The following T10 approved proposals were incorporated in SSC-5 revision 1a:

- 13-273r3 SSC-5: Dynamic Runtime Information [Kevin Butt, IBM]
- 14-179r3 CRC32C Polynomial for LBP Simplified [Kevin Butt, IBM]
- 15-082r1 SSC-5 Modify 13-273 Dynamic Runtime Attributes for Service Actions [Curtis Ballard, Hewlett-Packard]

The following T10 approved proposals are not yet incorporated in SSC-5 revision 1a.

- 13-049r2 SSC-4: Logical Block Protection Length error example [Kevin Butt, IBM]
- 14-249r2 SPC-5, SBC-4, SAT-4, ADC-4, SPL-4 - Obsolete TMC and ETC bits [Ralph Weber, WD]

2.4 Revision 1b (27 March, 2015)

The following changes were incorporated in SSC-5 revision 1b:

- Corrected location and size of ATTRIBUTE REPORT TYPE in READ DYNAMIC RUNTIME ATTRIBUTE command parameters
- Corrected incorporation errors from 14-179r3CRC32C Polynomial for LBP Simplified
- Incorporated editorial changes requested by working group

2.5 Revision 2 (10 January, 2016)

The following changes were incorporated in SSC-5 revision 2:

- 13-049r3 Logical Block Protection Length error example [Kevin Butt, IBM]
- 15-080r1 Informational Exception MP clean-up [Kevin Butt, IBM]

2.6 Revision 2a (10 February, 2016)

The following editorial changes were incorporated in SSC-5 revision 2a:

Changed references to list items to new ISO documentation style
Changed dynamic runtime attribute names to all caps to match SPC MAM attributes style
Changed length field values in some structures from incorrect decimal format to hex
Added OPERATION CODE field descriptive text to commands missing that text
Corrected VPD page lengths so all were two bytes as required by SPC-5
Moved MAM attributes into parameters sub-clause to match decision in SPC

The following changes were incorporated in SSC-5 revision 2a:

15-296r1 SSC-5 Logical block Protection VPD page [Curtis Ballard, Hewlett Packard Enterprise]

**ANSI (r)
INCITS 503-201x**

Draft

**American National Standards
for Information Systems -**

SCSI Stream Commands - 5 (SSC-5)

Secretariat
Information Technology Industry Council

Approved mm dd yy

American National Standards Institute, Inc.

Abstract

This standard specifies the device model and functional requirements for the SCSI sequential-access device type. This standard permits the SCSI sequential-access device type to attach to computers and provides the definitions for their use.

This standard does not contain material related to any service delivery subsystem that is used to transport the commands, command parameter logical block, command response logical block, and status specified in this standard.

Draft

**American
National
Standard**

Approval of an American National Standard requires review by ANSI that the requirements for due process, consensus, and other criteria for approval have been met by the standards developer. Consensus is established when, in the judgement of the ANSI Board of Standards Review, substantial agreement has been reached by directly and materially affected interests. Substantial agreement means much more than a simple majority, but not necessarily unanimity. Consensus requires that all views and objections be considered, and that a concerted effort be made towards their resolution.

The use of American National Standards is completely voluntary; their existence does not in any respect preclude anyone, whether he or she has approved the standards or not, from manufacturing, marketing, purchasing, or using products, processes, or procedures not conforming to the standards.

The American National Standards Institute does not develop standards and will in no circumstances give an interpretation of any American National Standard. Moreover, no person shall have the right or authority to issue an interpretation of an American National Standard in the name of the American National Standards Institute. Requests for interpretations should be addressed to the secretariat or sponsor whose name appears on the title page of this standard.

CAUTION NOTICE: This American National Standard may be revised or withdrawn at any time. The procedures of the American National Standards Institute require that action be taken periodically to reaffirm, revise, or withdraw this standard. Purchasers of American National Standards may receive current information on all standards by calling or writing the American National Standards Institute.

CAUTION: The developers of this standard have requested that holders of patents that may be required for the implementation of the standard disclose such patents to the publisher. However, neither the developers nor the publisher have undertaken a patent search in order to identify which, if any, patents may apply to this standard. As of the date of publication of this standard, following calls for the identification of patents that may be required for the implementation of the standard, notice of one or more such claims has been received.

By publication of this standard, no position is taken with respect to the validity of this claim or of any rights in connection therewith. The known patent holder(s) has (have), however, filed a statement of willingness to grant a license under these rights on reasonable and nondiscriminatory terms and conditions to applicants desiring to obtain such a license. Details may be obtained from the publisher.

No further patent search is conducted by the developer or the publisher in respect to any standard it processes. No representation is made or implied that this is the only license that may be required to avoid infringement in the use of this standard.

Published by
American National Standards Institute, Inc.
25 West 43rd Street, New York, NY 10036

Copyright 20xx by Information Technology Industry Council (ITI)
All rights reserved.

No part of this publication may be reproduced in any form, in an electronic retrieval system or otherwise, without prior written permission of ITI, 1101 K Street NW, Suite 610, Washington, DC 20005.

Printed in the United States of America

Contents

	Page
Revision Information	iv
1 Approved Documents Included	iv
2 Revision History	iv
2.1 Revision 0 (4 February, 2014).....	iv
2.2 Revision 1 (11 September, 2014).....	v
2.3 Revision 1a (25 March, 2015)	v
2.4 Revision 1b (27 March, 2015)	v
2.5 Revision 2 (10 January, 2016)	v
2.6 Revision 2a (10 February, 2016).....	vi
Foreword	xxii
Introduction	xxiii
SCSI standards family	xxiv
1 Scope	1
2 Normative references	1
2.1 Normative references overview.....	1
2.2 Approved references	1
2.3 References under development.....	1
2.4 Other references	2
3 Definitions, acronyms, keywords, and conventions	3
3.1 Definitions.....	3
3.2 Acronyms	9
3.3 Keywords.....	10
3.4 Editorial Conventions	11
3.5 Notation Conventions	12
3.5.1 Notation for state diagrams	12
4 General Concepts	14
4.1 Overview	14
4.2 Sequential-access device model	14
4.2.1 Sequential-access device model overview	14
4.2.2 Volumes and medium	14
4.2.2.1 Volumes	14
4.2.2.2 Medium	15
4.2.2.3 Recording volume model	16
4.2.2.4 WORM volume model	18
4.2.2.5 Cleaning volume model	19
4.2.2.6 Microcode update volume model	19
4.2.3 Device entity.....	19
4.2.4 End-of-partition.....	23

4.2.5 Early-warning	23
4.2.6 Programmable early warning	24
4.2.7 Partitions within a recording volume	25
4.2.8 Logical objects	27
4.2.8.1 Logical objects within a partition	27
4.2.8.2 Logical object identifier	28
4.2.9 Logical files	28
4.2.9.1 Logical files within a partition	28
4.2.9.2 Logical file identifier	28
4.2.10 User data segments	29
4.2.10.1 User data segments overview.....	29
4.2.10.2 User data segment descriptors	29
4.2.11 Logical object access command types.....	29
4.2.11.1 Logical object access command types overview	29
4.2.11.2 Verify commands	29
4.2.12 Logical object access operation types	30
4.2.12.1 Write operations.....	30
4.2.12.2 Read operations.....	30
4.2.12.3 Verify operations	30
4.2.13 Object buffering.....	30
4.2.14 Synchronize operation behavior.....	31
4.2.15 Direction and position definitions	31
4.2.16 Write modes	32
4.2.16.1 Write mode introduction	32
4.2.16.2 Overwrite-allowed mode	32
4.2.16.3 Append-only mode.....	32
4.2.17 Compression of objects.....	36
4.2.17.1 Compression of objects overview	36
4.2.17.2 Interaction of the compression mode parameters.....	36
4.2.18 Error reporting	38
4.2.18.1 Overview	38
4.2.18.2 Stream commands sense data descriptor	39
4.2.18.3 Information sense data descriptor.....	39
4.2.18.4 Deferred check condition eligible commands	40
4.2.18.5 Deferred error affinity commands	40
4.2.18.6 Error conditions	40
4.2.19 Self-test operations	42
4.2.20 Write protection	42
4.2.20.1 Write protection introduction	42
4.2.20.2 Write protection additional sense code use	43
4.2.20.3 Software write protection for the device server.....	44
4.2.20.4 Associated write protection	44
4.2.20.5 Persistent write protection.....	44
4.2.20.6 Permanent write protection	44
4.2.21 Progress indication.....	45
4.2.22 Command queuing.....	46
4.2.22.1 Command queuing overview	46
4.2.22.2 Explicit address mode write sequences.....	46
4.2.23 Block address mode.....	46
4.2.23.1 Block address mode overview	46
4.2.23.2 Block address mode selection	47
4.2.23.3 Block address mode state diagrams.....	47
4.2.24 TapeAlert application client interface	55
4.2.24.1 TapeAlert introduction	55

4.2.24.2	TapeAlert usage model	58
4.2.24.3	TapeAlert flag activation and deactivation	60
4.2.24.4	WORM TapeAlert flags	65
4.2.24.5	TapeAlert Response log page	65
4.2.25	Dynamic runtime information	66
4.2.25.1	Dynamic runtime information overview	66
4.2.25.2	Dynamic runtime information timestamp	67
4.2.25.3	Writing dynamic runtime information into the SCSI device	67
4.2.25.4	Retrieving dynamic runtime information from the SCSI device	67
4.2.25.5	Management of dynamic runtime information	67
4.2.25.6	Dynamic Runtime Information Lifetime	68
4.2.26	Recommended access order	69
4.2.27	Reservations	70
4.2.28	Logical block protection.....	72
4.2.28.1	Logical block protection overview	72
4.2.28.2	Protection information on a volume	73
4.2.28.3	Logical blocks and protection information.....	74
4.2.28.4	Protection information for Recover Buffered Data	76
4.2.28.5	Processing logical blocks using logical block protection	77
4.2.28.6	File Verification of protection information	80
4.2.28.7	Verification of protection information to EOD	80
4.2.29	Logical block encryption.....	81
4.2.29.1	Logical block encryption overview	81
4.2.29.2	Encrypting logical blocks on the medium.....	81
4.2.29.3	Reading encrypted logical blocks on the medium.....	82
4.2.29.4	Exhaustive-search attack prevention	83
4.2.29.5	Keyless copy of encrypted logical blocks.....	83
4.2.29.6	Managing logical block encryption keys within the device entity	85
4.2.29.7	Logical block encryption capabilities	86
4.2.29.8	Key instance counters.....	87
4.2.29.9	Encryption mode locking	87
4.2.29.10	Nonce generation.....	87
4.2.29.11	Unauthenticated key-associated data (U-KAD) and authenticated key-associated data (A-KAD).....	88
4.2.29.12	Metadata key-associated data (M-KAD)	88
4.2.29.13	Wrapped key key-associated data (WK-KAD)	88
4.2.29.14	Logical block encryption information per I_T_L nexus	88
4.2.29.15	Logical block encryption parameters	90
4.2.29.16	Effects of reservation loss on logical block encryption parameters	91
4.2.29.17	Effects of reservation preempt on logical block encryption parameters.....	92
4.2.30	External data encryption control.....	93
4.2.30.1	External data encryption control overview	93
4.2.30.2	External data encryption control of data encryption capabilities	93
4.2.30.3	External data encryption control of logical block encryption parameters	95
4.2.30.4	External data encryption control - logical block encryption parameters exclusive control	99
4.2.30.5	External data encryption control error conditions.....	99
4.2.31	Logical block encryption key protection	100
4.2.31.1	Logical block encryption key protection overview	100
4.2.31.2	Logical block encryption key protection using security associations	100
4.2.31.3	Key wrapping using public key cryptography	100
4.2.31.4	Key wrapping using key manager specific methods	100
4.2.31.5	Encryption management attributes	101
4.2.32	Appending data to a volume containing encrypted logical blocks	102

5	Explicit address command descriptions for sequential-access devices	104
5.1	Summary of commands for explicit address mode	104
5.2	ERASE(16) command	109
5.3	READ(16) command	110
5.4	READ REVERSE(16) command	114
5.5	VERIFY(16) command	115
5.6	WRITE(16) command	116
5.7	WRITE FILEMARKS(16) command	118
6	Implicit address command descriptions for sequential-access devices	120
6.1	Summary of commands for implicit address mode	120
6.2	ERASE(6) command	124
6.3	LOCATE(10) command	125
6.4	READ(6) command	126
6.5	READ REVERSE(6) command	128
6.6	SPACE(6) command	129
6.7	VERIFY(6) command	132
6.8	WRITE(6) command	135
6.9	WRITE FILEMARKS(6) command	136
7	Common command descriptions for sequential-access devices	138
7.1	ALLOW OVERWRITE command	138
7.2	FORMAT MEDIUM command	139
7.3	GENERATE RECOMMENDED ACCESS ORDER command	141
7.3.1	GENERATE RECOMMENDED ACCESS ORDER command parameter data	142
7.3.2	GRAO - user data segment descriptor	143
7.4	LOAD UNLOAD command	144
7.5	LOCATE(16) command	145
7.6	PREVENT ALLOW MEDIUM REMOVAL command	147
7.7	READ BLOCK LIMITS command	148
7.7.1	READ BLOCK LIMITS command overview	148
7.7.2	READ BLOCK LIMITS block length data	149
7.7.3	READ BLOCK LIMITS maximum logical object identifier data	150
7.8	READ DYNAMIC RUNTIME ATTRIBUTE command	151
7.8.1	READ DYNAMIC RUNTIME ATTRIBUTE attribute report type	151
7.8.2	SUPPORTED ATTRIBUTES attribute report type	152
7.8.3	ATTRIBUTE VALUES FOR THIS I_T NEXUS attribute report type	153
7.8.4	ATTRIBUTE VALUES FOR ALL I_T NEXUSES attribute report type	154
7.9	READ POSITION command	156
7.9.1	READ POSITION command description	156
7.9.2	READ POSITION data format, short form	158
7.9.3	READ POSITION data format, long form	160
7.9.4	READ POSITION data format, extended form	161
7.10	RECEIVE RECOMMENDED ACCESS ORDER command	161
7.10.1	RECEIVE RECOMMENDED ACCESS ORDER command parameter data	162
7.10.1.1	UDS Limits page	162
7.10.1.2	RAO list	163
7.10.1.3	RRAO - user data segment descriptor	164
7.11	RECOVER BUFFERED DATA command	166
7.12	REPORT DENSITY SUPPORT command	167
7.12.1	REPORT DENSITY SUPPORT command description	167

7.12.2 REPORT DENSITY SUPPORT header	168
7.12.3 Density support report.....	168
7.12.4 Medium type support report	171
7.13 REWIND command	173
7.14 SET CAPACITY command.....	174
7.15 SPACE(16) command	175
7.16 WRITE DYNAMIC RUNTIME ATTRIBUTE command	178
 8 Parameters for sequential-access devices	181
8.1 Diagnostic parameters	181
8.2 Dynamic runtime attributes.....	181
8.2.1 Attribute format.....	181
8.2.2 Attribute identifier values	182
8.2.2.1 Attribute identifier values overview	182
8.2.2.2 Device type attributes	182
8.2.2.3 Target type attributes	188
8.2.2.4 Initiator type attributes	190
8.3 Log parameters	191
8.3.1 Log parameters overview.....	191
8.3.2 Sequential-Access Device log page.....	192
8.3.3 TapeAlert log page.....	194
8.3.4 Device Statistics log page	195
8.3.4.1 Device Statistics log page overview.....	195
8.3.4.2 Device statistics log parameter formats	198
8.3.4.3 Device statistics string data log parameter format	200
8.3.4.4 Device Statistics log parameters.....	200
8.3.5 Tape Diagnostic Data log page.....	204
8.3.6 Current Service Information log page	208
8.3.6.1 Current Service Information log page overview	208
8.3.6.2 Vendor-specific service information descriptor	210
8.3.6.3 DEVICE INFORMATION DESCRIPTOR.....	211
8.3.6.4 Volume information descriptor	213
8.3.6.5 TapeAlert flag specific information	214
8.3.7 Requested Recovery log page.....	215
8.3.7.1 Requested Recovery log page overview	215
8.3.7.2 Recovery procedures log parameter.....	216
8.3.8 Data Compression log page.....	218
8.3.8.1 Data Compression log page overview	218
8.3.8.2 Data compression counter log parameter format	220
8.3.8.3 Data Compression log parameters	220
8.3.9 Volume Statistics log page	222
8.3.9.1 Volume Statistics log page overview	222
8.3.9.2 Volume statistics log parameter formats	226
8.3.9.3 Volume statistics log parameters	228
8.4 Medium auxiliary memory attributes.....	236
8.4.1 Medium auxiliary memory device type attributes	236
8.4.2 Medium auxiliary memory medium type attributes	237
8.4.3 Medium auxiliary memory host type attributes	238
8.5 Mode parameters	238
8.5.1 Mode parameters overview.....	238
8.5.2 Data Compression mode page	242
8.5.3 Device Configuration mode page	246
8.5.4 Medium Partition mode page	251

8.5.5 Read-Write Error Recovery mode page	255
8.5.6 Informational Exceptions Control mode page	256
8.5.7 Medium Configuration mode page	262
8.5.8 Device Configuration Extension mode page	263
8.5.9 Control Data Protection mode page	266
8.6 Vital product data (VPD) parameters	267
8.6.1 VPD parameters overview and page codes	267
8.6.2 Sequential-access Device Capabilities VPD page	268
8.6.3 Manufacturer-assigned Serial Number VPD page	268
8.6.4 TapeAlert Supported Flags VPD page	269
8.6.5 Automation Device Serial Number VPD page	270
8.6.6 Data Transfer Device Element Address VPD page	270
8.6.7 Logical Block Protection VPD page	271
8.7 Security protocol parameters	272
8.7.1 Security protocol overview	272
8.7.2 SECURITY PROTOCOL IN command specifying Tape Data Encryption security protocol	272
8.7.2.1 SECURITY PROTOCOL IN command specifying Tape Data Encryption security protocol overview	272
8.7.2.2 Tape Data Encryption In Support page	274
8.7.2.3 Tape Data Encryption Out Support page	274
8.7.2.4 Data Encryption Capabilities page	275
8.7.2.5 Supported Key Formats page	283
8.7.2.6 Data Encryption Management Capabilities page	283
8.7.2.7 Data Encryption Status page	285
8.7.2.8 Next Block Encryption Status page	288
8.7.2.9 Get Encryption Management Attributes page	292
8.7.2.10 Random Number page	293
8.7.2.11 Device Server Key Wrapping Public Key page	293
8.7.3 SECURITY PROTOCOL OUT command specifying Tape Data Encryption security protocol	294
8.7.3.1 SECURITY PROTOCOL OUT command specifying Tape Data Encryption security protocol overview	294
8.7.3.2 Set Data Encryption page	296
8.7.3.3 SA Encapsulation page	307
8.7.3.4 Set Encryption Management Attributes page	308
8.7.4 SECURITY PROTOCOL IN and SECURITY PROTOCOL OUT descriptors	308
8.7.4.1 Tape Data Encryption security protocol descriptors overview	308
8.7.4.2 Tape Data Encryption descriptors	309
8.7.4.3 Wrapped Key descriptors	311
8.7.4.4 Encryption management attributes descriptor	312
Annex A (Informative) Security environment	317
A.1 Security environment overview	317
A.2 Security environment threats	318
Annex B (Informative) Example keyless copy operation flowchart	319
B.1 Example keyless copy operation flowchart overview	319
Annex C (Informative) Example application client use of volume coherency	322
Annex D (Informative) Sense data information for error conditions	323
D.1 Sense data error indications for read and write operations	323
D.2 INFORMATION field and position for read and write operations	325
D.3 Summary of length error conditions on read type commands	328

Annex E (Informative) Logical block protection CRC information.....	329
E.1 Reed-Solomon CRC	329
E.2 CRC32C (Castagnoli)	332
Annex F (Informative) Example recommended access order usage.....	337
Annex G (Informative) Transfer length examples with and without logical block protection.....	339
G.1 Logical block and transfer lengths when logical block protection is supported	339
G.2 General write rules	340
G.3 General read rules	340
G.4 Examples from figure G.1 using variable-block transfers and various SILI and BLOCK LENGTH settings... <td>341</td>	341
G.5 Examples from figure G.1 using fixed-block transfers and various BLOCK LENGTH settings.....	342

Figures

	Page
Figure 0 — SCSI document relationships	xxiv
Figure 1 — Example state diagram	13
Figure 2 — Example recording medium layout	16
Figure 3 — Example recording medium track layout	16
Figure 4 — Serpentine recording example	17
Figure 5 — Parallel recording example	17
Figure 6 — Helical scan recording example	18
Figure 7 — Example cleaning medium layout	19
Figure 8 — UML example of SCSI target device and device entity	21
Figure 9 — Logical end-of-partition example	23
Figure 10 — Early-warning example where LEOP is the same as EOP	24
Figure 11 — Early-warning example where LEOP is not the same as EOP	24
Figure 12 — Programmable early warning example where LEOP is the same as EOP	24
Figure 13 — Programmable early warning example where LEOP is not the same as EOP	25
Figure 14 — Partitioning example - one partition per track group	26
Figure 15 — Partitioning example - one partition per two track groups	26
Figure 16 — Partitioning example - two partitions per track group	27
Figure 17 — Append-only mode flowchart	35
Figure 18 — Block address mode state diagram, overview	48
Figure 19 — Block address mode state diagram, Idle state	49
Figure 20 — Block address mode state diagram, Explicit Address Mode - Neutral	51
Figure 21 — Block address mode state diagram, Explicit Address Mode - Write Capable	53
Figure 22 — Block address mode state diagram, Implicit Address Mode	54
Figure 23 — Dynamic runtime attributes scope	68
Figure 24 — Protection information shown in relationship to logical objects and format-specific symbols	73
Figure 25 — Example file verification of protection information	80
Figure 26 — Example verification of protection information to EOD	81
Figure A.1 — Simple security deployment environment	317
Figure B.1 — Example keyless copy operation flowchart part 1	320
Figure B.2 — Example keyless copy operation flowchart part 2	321
Figure F.1 — Example reordering of UDSs	338
Figure G.1 — Example protected logical block as it relates to write and read commands	339

Tables

	Page
Table 1 — Incorporated T10 Approved Documents (in document number order)	iv
Table 1 — Numbering conventions examples	12
Table 2 — Device entity attributes.....	22
Table 3 — ALLOW_OVERWRITE variable value definitions	33
Table 4 — Compression algorithm and compression enablement selection.....	37
Table 5 — Stream commands sense data descriptor	39
Table 6 — Information sense data descriptor.....	40
Table 7 — Error conditions and sense keys	41
Table 8 — Write protect additional sense code combinations.....	43
Table 9 — Commands providing progress indication without changing ready state	45
Table 10 — Commands changing ready state and providing progress indication	45
Table 11 — TapeAlert flags severity	56
Table 12 — TapeAlert flags	56
Table 13 — TapeAlert flag activation and deactivation conditions	60
Table 14 — Types of dynamic runtime attributes	66
Table 15 — Dynamic runtime attribute states	66
Table 16 — RAO PROCESS for generating recommended access order	69
Table 17 — SSC-5 commands that are allowed in the presence of various reservations.....	71
Table 18 — Logical block with no protection information format	74
Table 19 — Logical block for READ REVERSE with BYTORD=0b and no protection information format.....	75
Table 20 — Logical block with protection information format	75
Table 21 — Logical block for READ REVERSE with BYTORD=0b and protection information format.....	76
Table 22 — Data transferred during a RECOVER BUFFERED DATA command with the ROBO=0b	76
Table 23 — Data transferred during a RECOVER BUFFERED DATA command with the ROBO=1b	77
Table 24 — Default I_T_L nexus logical block encryption information.....	90
Table 25 — Logical block encryption parameters for encryption request policies.....	96
Table 26 — Logical block encryption parameters for decryption request policies.....	97
Table 27 — Logical block encryption parameters for encryption request indicator settings.....	97
Table 28 — Logical block encryption parameters for decryption request indicator settings.....	98
Table 29 — Logical block encryption period timer expired indicator	99
Table 30 — Explicit address command set for sequential-access devices	104
Table 31 — ERASE(16) command.....	109
Table 32 — METHOD field	110
Table 33 — READ(16) command.....	111
Table 34 — READ REVERSE(16) command.....	114
Table 35 — VERIFY(16) command	115
Table 36 — WRITE(16) command	116
Table 37 — WRITE FILEMARKS(16) command	118
Table 38 — Implicit address command set for sequential-access devices	120
Table 39 — ERASE(6) command.....	124
Table 40 — LOCATE(10) command	125
Table 41 — READ(6) command	126
Table 42 — READ REVERSE(6) command	128
Table 43 — SPACE(6) command	129
Table 44 — CODE field	129
Table 45 — VERIFY(6) command	132
Table 46 — WRITE(6) command	135
Table 47 — WRITE FILEMARKS(6) command	136
Table 48 — ALLOW OVERWRITE command	138
Table 49 — ALLOW OVERWRITE field definition	138
Table 50 — FORMAT MEDIUM command	139

Table 51 — FORMAT field	140
Table 52 — GENERATE RECOMMENDED ACCESS ORDER command	141
Table 53 — uds_type field	142
Table 54 — GENERATE RECOMMENDED ACCESS ORDER parameter data format	142
Table 55 — GRAO - user data segment descriptor format	143
Table 56 — LOAD UNLOAD command	144
Table 57 — LOCATE(16) command	146
Table 58 — DEST_TYPE field	146
Table 59 — PREVENT ALLOW MEDIUM REMOVAL command	147
Table 60 — PREVENT field	147
Table 61 — READ BLOCK LIMITS command	148
Table 62 — READ BLOCK LIMITS block length data	149
Table 63 — READ BLOCK LIMITS maximum logical object identifier data	150
Table 64 — READ DYNAMIC RUNTIME ATTRIBUTE command	151
Table 65 — READ DYNAMIC RUNTIME ATTRIBUTE attribute report type codes	152
Table 66 — READ DYNAMIC RUNTIME ATTRIBUTE with SUPPORTED ATTRIBUTES attribute report type pa- rameter list format	153
Table 67 — READ DYNAMIC RUNTIME ATTRIBUTE with ATTRIBUTE VALUES FOR THIS I_T NEXUS attribute report type parameter list format	154
Table 68 — READ DYNAMIC RUNTIME ATTRIBUTE with ATTRIBUTE VALUES FOR ALL I_T NEXUSES attri- bute report type parameter list format	155
Table 69 — READ POSITION command	156
Table 70 — READ POSITION service action codes	156
Table 71 — READ POSITION data format, short form	158
Table 72 — READ POSITION data format, long form	160
Table 73 — READ POSITION data format, extended form	161
Table 74 — RECEIVED RECOMMENDED ACCESS ORDER command	162
Table 76 — RAO List	163
Table 75 — UDS Limits page	163
Table 77 — RAO list status codes	164
Table 78 — RRAO - user data segment descriptor format	165
Table 79 — Additional information descriptor format	165
Table 81 — RECOVER BUFFERED data command	166
Table 80 — additional information type codes	166
Table 82 — REPORT DENSITY SUPPORT command	167
Table 83 — REPORT DENSITY SUPPORT header	168
Table 84 — Density support data block descriptor	169
Table 85 — Medium type descriptor	172
Table 86 — REWIND command	173
Table 87 — SET CAPACITY command	174
Table 88 — SPACE(16) command	175
Table 89 — Space positioning information	176
Table 90 — WRITE DYNAMIC RUNTIME ATTRIBUTE command	178
Table 91 — WRITE DYNAMIC RUNTIME ATTRIBUTE parameter list format	179
Table 92 — Diagnostic page codes	181
Table 93 — Dynamic runtime attribute format	181
Table 94 — Dynamic runtime attribute FORMAT field	182
Table 95 — Dynamic runtime attribute identifier range assignments	182
Table 96 — Device type attributes	183
Table 97 — I_T_L nexus identifying information format	184
Table 98 — RESERVATION INFORMATION dynamic runtime attribute value format	186
Table 99 — RESERVATION TYPE codes	186
Table 100 — REGISTRATION INFORMATION dynamic runtime attribute value format	187
Table 101 — PREVENT ALLOW MEDIUM REMOVAL INFORMATION dynamic runtime attribute value format....	

187	
Table 102 — Target type attributes	188
Table 103 — LAST ACCESS TIME target type attribute format	189
Table 104 — Initiator type attributes.....	190
Table 105 — Log page codes	191
Table 106 — Parameter codes for Sequential-Access Device log page.....	192
Table 107 — TapeAlert log page.....	194
Table 108 — TapeAlert parameter format.....	194
Table 109 — Device Statistics log page.....	195
Table 110 — Device Statistics log parameter codes	196
Table 111 — Device statistics data counter log parameter format.....	198
Table 112 — Device statistics medium type log parameter format	199
Table 113 — Device statistics medium type descriptor format.....	199
Table 114 — Device statistics string data log parameter format.....	200
Table 115 — Maximum recommended mechanism temperature exceeded codes	204
Table 116 — Tape Diagnostic Data log page.....	205
Table 117 — Tape diagnostic data log parameter format	205
Table 118 — Current Service Information log page	208
Table 119 — Service information log parameter format.....	209
Table 120 — Service information descriptor	210
Table 121 — SERVICE INFORMATION DESCRIPTOR TYPE field	210
Table 122 — Vendor-specific service information descriptor	210
Table 123 — Device information descriptor	211
Table 124 — DEC field	211
Table 125 — DEVICE REQUESTED RECOVERY field.....	212
Table 126 — Volume information descriptor	213
Table 127 — VIC field.....	213
Table 128 — VICQ field	214
Table 129 — TapeAlert flag specific information descriptor	214
Table 130 — Requested Recovery log page.....	216
Table 131 — REQUESTED RECOVERY LOG PARAMETER CODES	216
Table 132 — Requested recovery log parameter format	216
Table 133 — RECOVERY PROCEDURES	217
Table 134 — Data Compression log page	218
Table 135 — DATA COMPRESSION LOG PARAMETER CODES	219
Table 136 — Data compression counter log parameter format.....	220
Table 137 — Volume Statistics log page.....	223
Table 138 — Volume Statistics log subpage codes	223
Table 139 — Volume Statistics log parameter codes.....	224
Table 140 — Volume statistics data counter log parameter format.....	226
Table 141 — Volume statistics string data log parameter format.....	226
Table 142 — Volume statistics partition record log parameter format.....	227
Table 143 — Volume statistics partition record descriptor format	228
Table 144 — Maximum recommended tape path temperature exceeded codes.....	232
Table 145 — Mount history log parameter format	234
Table 146 — Mount history descriptor format	235
Table 147 — Medium auxiliary memory device type attributes	236
Table 148 — Medium auxiliary memory medium type attributes.....	237
Table 149 — Medium auxiliary memory host type attributes.....	238
Table 150 — VOLUME COHERENCY INFORMATION attribute format	238
Table 151 — Device-specific parameter	239
Table 152 — Buffered modes.....	239
Table 153 — SPEED field.....	239
Table 154 — Sequential-access density codes.....	240

Table 155 — Mode page codes and subpage codes	241
Table 156 — Data Compression mode page	242
Table 157 — Possible boundaries and resulting sense keys due to data compression.....	243
Table 158 — Compression algorithm identifiers.....	245
Table 159 — Device Configuration mode page.....	246
Table 160 — EOD DEFINED field	247
Table 161 — WTRE field	249
Table 162 — REWIND ON RESET field.....	250
Table 163 — Medium Partition mode page	251
Table 164 — PSUM field	252
Table 165 — MEDIUM FORMAT RECOGNITION field	254
Table 166 — PARTITIONING TYPE field	254
Table 167 — Read-Write Error Recovery mode page.....	255
Table 168 — Informational Exceptions Control mode page	256
Table 169 — Definitions for the combinations of values in DEXCPT bit and TASER bit.....	258
Table 170 — Method of reporting informational exceptions (MRIE) field.....	259
Table 171 — REPORT COUNT/TEST FLAG NUMBER field definition if the TEST bit is set to one	261
Table 172 — Use of the INTERVAL TIMER field and the REPORT COUNT/TEST FLAG NUMBER field based on the MRIE field	262
Table 173 — Medium Configuration mode page	262
Table 174 — WORM VOLUME LABEL RESTRICTIONS field	263
Table 175 — WORM VOLUME FILEMARKS RESTRICTIONS field	263
Table 176 — Device Configuration Extension mode page	264
Table 177 — SHORT ERASE MODE field	265
Table 178 — WRITE MODE field	265
Table 179 — Control Data Protection mode page	266
Table 180 — Logical block protection methods.....	266
Table 181 — Sequential-access device VPD page codes	267
Table 182 — Sequential-access Device Capabilities VPD page.....	268
Table 183 — Manufacturer-assigned Serial Number VPD page.....	268
Table 184 — TapeAlert Supported Flags VPD page.....	269
Table 185 — Automation Device Serial Number VPD page	270
Table 186 — Data Transfer Device Element Address VPD page	270
Table 187 — Logical Block Protection VPD page	271
Table 188 — Logical block protection method descriptor format	271
Table 189 — SECURITY PROTOCOL SPECIFIC field	273
Table 190 — Tape Data Encryption In Support page.....	274
Table 191 — Tape Data Encryption Out Support page	274
Table 192 — Data Encryption Capabilities page	275
Table 193 — EXTDECC field.....	275
Table 194 — CFG_P field	276
Table 195 — Logical block encryption algorithm descriptor	277
Table 196 — DECRYPT_C field.....	278
Table 197 — ENCRYPT_C field.....	279
Table 198 — AVFCLP field	279
Table 199 — NONCE_C field	279
Table 200 — DKAD_C field	281
Table 201 — EEMC_C field	281
Table 202 — RDMC_C field.....	282
Table 203 — Supported Key Formats page	283
Table 204 — Data Encryption Management Capabilities page.....	283
Table 205 — Data Encryption Status page	285
Table 206 — PARAMETERS CONTROL field	286
Table 207 — Next Block Encryption Status page.....	288

Table 208 — COMPRESSION STATUS field	289
Table 209 — ENCRYPTION STATUS field	290
Table 210 — Get Encryption Management Attributes page	292
Table 211 — Random Number page.....	293
Table 212 — Device Service Key Wrapping Public Key page	293
Table 213 — PUBLIC KEY TYPE field.....	294
Table 214 — SECURITY PROTOCOL SPECIFIC field	295
Table 215 — Set Data Encryption page	296
Table 216 — SCOPE field	297
Table 217 — CEEM field	297
Table 218 — RDMC field.....	298
Table 219 — ENCRYPTION MODE field	299
Table 220 — DECRYPTION MODE field	300
Table 221 — LOGICAL BLOCK ENCRYPTION KEY FORMAT field	301
Table 222 — KAD FORMAT CODE	302
Table 223 — KEY field format with KEY FORMAT field set to 00h.....	303
Table 224 — KEY field format with KEY FORMAT field set to 01h.....	304
Table 225 — KEY field format with KEY FORMAT field set to 02h.....	304
Table 226 — PARAMETER SET field	305
Table 227 — ECIES-HC REQUIREMENTS AND PARAMETERS FOR ECIES-KEM	306
Table 228 — ECIES-HC REQUIREMENTS AND PARAMETERS FOR ECIES-DEM.....	306
Table 229 — SA Encapsulation page.....	307
Table 230 — Set Encryption Management Attributes page	308
Table 231 — Tape Data Encryption descriptor format	309
Table 232 — KEY DESCRIPTOR TYPE field	309
Table 233 — AUTHENTICATED field	309
Table 234 — Wrapped Key descriptor format	311
Table 235 — WRAPPED KEY DESCRIPTOR TYPE field.....	311
Table 236 — Encryption management attributes descriptor format	312
Table 237 — ENCRYPTION MANAGEMENT ATTRIBUTES TYPE field	313
Table 238 — Desired key manager operation definition	313
Table 239 — Logical block encryption key selection criteria attribute format.....	314
Table 240 — Logical block encryption key selection criteria descriptor format	314
Table 241 — LOGICAL BLOCK ENCRYPTION KEY SELECTION CRITERIA TYPE field	315
Table 242 — Logical block encryption key wrapping attribute format	315
Table 243 — Logical block encryption key wrapping attribute descriptor format	316
Table 244 — LOGICAL BLOCK ENCRYPTION KEY WRAPPING ATTRIBUTE TYPE field	316
Table A.1 — Security environment threats	318
Table B.1 — Example keyless copy figure codes	319
Table D.1 — Sense data error indications for read and write operations	323
Table D.2 — INFORMATION field and position for read and write operations	325
Table D.3 — Summary of length error conditions on read type commands	328
Table E.1 — CRC32C test vector	335

Foreword

(This foreword is not part of American National Standard INCITS 503-201x.)

This standard specifies the external behavior of a device server that defines itself as a sequential-access device in the PERIPHERAL DEVICE TYPE field of the standard INQUIRY data. This device type is known as a stream device. This standard conforms to SCSI Architecture Model - 4 (T10/1683-D).

Requests for interpretation, suggestions for improvement and addenda, or defect reports are welcome. They should be sent to the INCITS Secretariat, InterNational Committee for Information Technology Standards, Information Technology Institute, 1101 K Street NW Suite 610, Washington, DC 20005.

This standard was processed and approved for submittal to ANSI by the InterNational Committee for Information Technology Standards (INCITS). Committee approval of the standard does not necessarily imply that all committee members voted for approval. At the time of its approval this standard, INCITS had the following members:

<<Insert INCITS member list>>

The INCITS Technical Committee T10 on SCSI Storage Interfaces, which reviewed this standard, had the following members:

<<Insert T10 member list>>

Introduction

The SCSI Stream Commands - 5 (SSC-5) standard is divided into eight clauses:

- Clause 1 is the scope.
- Clause 2 enumerates the normative references that apply to this standard.
- Clause 3 describes the definitions, acronyms, keywords, and conventions used in this standard.
- Clause 4 describes an overview and model of the sequential-access type device.
- Clause 5 describes the explicit address command set for sequential-access type devices.
- Clause 6 describes the implicit address command set for sequential-access type devices.
- Clause 7 describes the common command set for sequential-access type devices.
- Clause 8 describes the parameters for sequential-access type devices.

The annexes provide information to assist with implementation of this standard.

SCSI standards family

Figure 0 shows the relationship of this standard to the other standards and related projects in the SCSI family standards as of the publication of this standard.

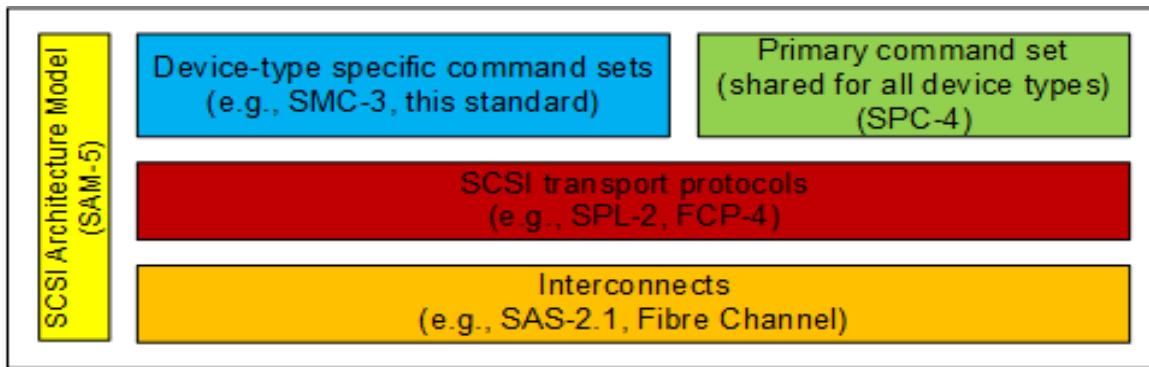


Figure 0 — SCSI document relationships

Figure 0 is intended to show the general relationship of the documents to one another and is not intended to imply a relationship such as a hierarchy, protocol stack, or system architecture.

The set of SCSI standards specifies the interfaces, functions, and operations necessary to ensure interoperability between conforming SCSI implementations. This standard is a functional description. Conforming implementations may employ any design technique that does not violate interoperability. See SAM-5 for more information about the relationships between the SCSI standards.

American National Standard**INCITS 503-201x****American National Standard
for Information Technology -****SCSI Stream Commands - 5 (SSC-5)****1 Scope**

This standard defines the command set extensions to facilitate operation of the sequential-access device type. This standard, implemented in conjunction with the requirements of the SCSI Architecture Model - 4 standard and the applicable clauses of the SCSI Primary Commands - 4 standard, fully specifies the standard command set for the sequential-access device type.

The objectives of this standard are to provide the following:

- a) permit an application client to communicate over a SCSI service delivery subsystem, with a logical unit that declares itself to be a sequential-access device in the PERIPHERAL DEVICE TYPE field of the standard INQUIRY data (see SPC-4);
- b) define commands unique to the sequential-access device type; and
- c) define commands to manage the operation of the sequential-access device type.

2 Normative references**2.1 Normative references overview**

The following standards contain provisions that, by reference in the text, constitute provisions of this standard. At the time of publication, the editions indicated were valid. All standards are subject to revision, and parties to agreements based on this standard are encouraged to investigate the possibility of applying the most recent editions of the standards listed below.

Copies of the ISO/IEC documents may be obtained from ANSI: approved ANSI standards, approved and draft international and regional standards (ISO, IEC), and approved foreign standards (including JIS, and DIN). For further information, contact ANSI Customer Service Department at 212-642-4900 (phone), 212-302-1286 (fax) or via the World Wide Web at <http://www.ansi.org>.

Copies of the DoD document may be obtained via the World Wide Web at <http://iase.disa.mil/policy.html#DoD>.

2.2 Approved references

ISO/IEC 14776-452, *SCSI Primary Commands - 2 standard (SPC-2)*

2.3 References under development

At the time of publication, the following referenced standards were still under development. For information on the current status of the document, or regarding availability, contact the relevant standards body or other organization as indicated.

ISO/IEC 14776-415, *SCSI Architecture Model - 5 standard (SAM-5)* [T10/2104-D]
ISO/IEC 14776-454, *SCSI Primary Commands - 4 standard (SPC-4)* [T10/1731-D]
ISO/IEC 14776-353, *SCSI Media Changer Commands - 3 standard (SMC-3)* [T10/1730-D]
ISO/IEC 14776-358, *SCSI Automation/Drive Interface Commands - 3 standard (ADC-3)* [T10/1895-D]

2.4 Other references

RFC 3447, *Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1*

NOTE 1 - Copies of IETF standards may be obtained through the Internet Engineering Task Force (IETF) at <http://www.ietf.org>.

NIST SP (Special Publication) 800-56A, *Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography*, March 2007

NIST SP (Special Publication) 800-57, *Recommendation for Key Management – Part 1: General*, March 2007

NOTE 2 - Copies of approved NIST standards may be obtained through the National Institute of Standards and Technology (NIST) at <http://csrc.nist.gov/publications/nistpubs/index.html>.

FIPS 140-2, *Security Requirements for Cryptographic Modules*, May 2001

FIPS 180-3, *Secure Hash Standard (SHS)*, October 2008

FIPS 186-3, *Digital Signature Standard (DSS)*, June 2009

FIPS 198-1, *The Keyed-Hash Message Authentication Code (HMAC)*, July 2008

NOTE 3 - Copies of the approved FIPS standards may be obtained through the National Institute of Standards and Technology (NIST) at <http://csrc.nist.gov/publications/fips/index.html>.

NIST SP (Special Publication) 800-88, *Guidelines for Media Sanitization*, September 2006

NOTE 4 - Copies of the NIST publications may be obtained through the National Institute of Standards and Technology (NIST) at <http://csrc.nist.gov/publications/PubsSPs.html>.

ISO/IEC18033-2, *Security techniques - Encryption algorithms - Part 2: Asymmetric ciphers*

ANSI X9.63:2001, *Public Key Cryptography for the Financial Services Industry, Key Agreement and Key Transport Using Elliptic Curve Cryptography*

ISO 1001:1986, *Information processing -- File structure and labelling of magnetic tapes for information exchange*

ISO/IEC18033-2, *Security techniques - Encryption algorithms - Part 2: Asymmetric ciphers*

3 Definitions, acronyms, keywords, and conventions

3.1 Definitions

3.1.1 additional sense code (see SPC-4)

combination of the ADDITIONAL SENSE CODE field and the ADDITIONAL SENSE CODE QUALIFIER field in sense data

3.1.2 application client (see SAM-5)

object that is the source of SCSI commands and task management requests

3.1.3 authorization white list

set of identifiers, typically public keys, for entities that are authorized to perform some operation

3.1.4 auxiliary memory

memory component that is accessible to the device server; this memory is usually non-volatile and independent of the main function of the device server

3.1.5 BOx

either beginning-of-medium (see 3.1.6) or beginning-of-partition (see 3.1.7)

3.1.6 beginning-of-medium (BOM)

extreme position along the medium in the direction away from the supply reel that is accessible by the read/write mechanism; this position and the beginning-of-partition position may be different positions

3.1.7 beginning-of-partition (BOP)

position at the beginning of the permissible recording region of a partition; this position may not coincide with a beginning-of-medium position

3.1.8 block address mode

mode of operation that the logical unit is currently supporting (see 4.2.23); the block address mode is either the explicit address mode (see 3.1.23) or the implicit address mode (see 3.1.32)

3.1.9 buffered mode

mode of logical block transfer in write operations that facilitates tape streaming; buffered mode is specified by a non-zero value (1h or 2h) in the BUFFER MODE field in the mode parameter header (see 8.5); buffered mode is the opposite of unbuffered mode (see 3.1.88)

3.1.10 byte

8-bit construct

3.1.11 cleaning volume

volume (see 3.1.95) used for cleaning the read/write mechanism (see 4.2.2.3)

3.1.12 command (see SAM-5)

request describing a unit of work to be performed by a device server

3.1.13 command descriptor block (CDB)

structure used to communicate commands from an application client to a device server; a command descriptor block may have a fixed length of up to 16 bytes or a variable length of between 12 and 260 bytes

3.1.14 common command

command that is contained in both the explicit and implicit address command sets

3.1.15 device entity (see 4.2.3)

object in a SCSI target device that performs operations on a volume, stores parameters, and communicates between device servers

3.1.16 device server (see SAM-5)

object within a logical unit that processes SCSI tasks according to the rules of task management

3.1.17 device type

device model implemented by the logical unit and indicated to the application client by the PERIPHERAL DEVICE TYPE field of the standard INQUIRY data (see SPC-4)

3.1.18 early-warning (EW)

physical mark or device computed position near but logically before the end-of-partition, independent of physical direction (see 4.2.5)

3.1.19 encrypted logical block

logical block containing data that has been subjected to a ciphering process by a device server

3.1.20 end-of-data (EOD)

format-specific recorded indication that no valid logical objects are recorded between this position and end-of-partition (see 3.1.22)

3.1.21 end-of-medium (EOM)

extreme position along the medium in the direction away from the take-up reel that is accessible by the read/write mechanism

3.1.22 end-of-partition (EOP)

position at the end of the permissible recording region of a partition

3.1.23 explicit address mode

mode of operation in which the logical unit is supporting the explicit address command set (see 3.1.24)

3.1.24 explicit address command set

command set in which read type and write type commands contain positioning information

3.1.25 explicit command

command contained only in the explicit address command set (see table 30)

3.1.26 Externally Encrypted Data Key (EEDK) (see 4.2.31.4.2)

structure that contains an identifier and a wrapped logical block encryption key

3.1.27 field

group of one or more contiguous bits, a part of a larger structure such as a CDB (see 3.1.13) or sense data (see 3.1.75).

3.1.28 filemark

special recorded logical object within a partition, not containing user data, that provides a segmentation scheme for the contents of a partition

3.1.29 fixed-block transfer

read type or write type command with the FIXED bit set to one

3.1.30 format label

vendor-specific series of contiguous logical objects beginning at BOP 0 and including at least one logical block containing information used to identify the recording volume (e.g., a label as defined in ISO 1001:1986 or two logical blocks followed by a filemark)

3.1.31 generic command

explicit command (see 3.1.25) that is not a read type or write type command

3.1.32 implicit address mode

mode of operation in which the logical unit is supporting the implicit address command set (see 3.1.33)

3.1.33 implicit address command set

command set in which read type and write type commands do not contain positioning information, and positioning is implied relative to the current position

3.1.34 implicit command

command contained only in the implicit address command set (see table 38)

3.1.35 information field

command-specific field in the sense data (see SPC-4)

3.1.36 I_T nexus

nexus between a SCSI initiator port and a SCSI target port (see SAM-5)

3.1.37 I_T nexus loss

condition resulting from the events defined by SAM-5 in which the SCSI device performs the operations described in SAM-5 and this standard

3.1.38 I_T nexus loss event

SCSI transport protocol specific event that triggers I_T nexus loss as described in SAM-5

3.1.39 Key Encrypting Key (KEK)

encryption key used to wrap the logical block encryption key in the EEDK (see 3.1.26)

3.1.40 Key Encrypting Key Selector (KEKS) (see 8.7.4.4.4.2)

encryption management attribute that is an identifier used to select a KEK

3.1.41 key manager (see 4.2.31.4)

entity that manages the distribution, creation, and access of keys for use in cryptographic systems

3.1.42 logical block

logical object that is a unit of data supplied or requested by an application client

3.1.43 logical block encryption parameters

set of parameters (see 4.2.29.15) that controls the logical block encryption and decryption processes in the device entity (see 3.1.15)

3.1.44 logical end-of-partition (LEOP)

position at the end of the permissible recording region of a partition that has been adjusted by device server capabilities (see 4.2.4)

3.1.45 logical file

zero or more contiguous logical blocks starting immediately after BOP or a filemark (see 4.2.9)

3.1.46 logical file identifier

unique identifier, within a partition, for a logical file (see 4.2.9)

3.1.47 logical identifier

logical object identifier or logical file identifier

3.1.48 logical object

logical block or a filemark (see 4.2.8)

3.1.49 logical object identifier

unique identifier, within a partition, for a logical object (see 4.2.8.2)

3.1.50 logical unit reset

logical unit action in response to a logical unit reset event in which the logical unit performs the operations described in SAM-5

3.1.51 logical unit reset event

event that triggers a logical unit reset from a logical unit as described in SAM-5

3.1.52 medium auxiliary memory (MAM) (see SPC-4)

auxiliary memory residing in a volume (e.g., a tape cartridge) that is accessible to the device server

3.1.53 message authentication code

information used to validate the integrity of encrypted logical blocks (see 3.1.19)

3.1.54 microcode update volume

volume (see 3.1.95) containing a microcode image on a recording medium (see 4.2.2.3)

3.1.55 native capacity

capacity assuming one-to-one compression (e.g., compression disabled), the medium is in good condition, and that the device recommended typical block size is used

3.1.56 nexus (see SAM-5)

relationship between two SCSI devices, and the SCSI initiator port and SCSI target port objects within those devices

3.1.57 nonce

unpredictable random value used only for a single instance or invocation of a cryptographic algorithm or protocol

3.1.58 one

logical true condition of a variable

3.1.59 overlength

incorrect-length condition that exists after processing a READ command if the length of the actual logical block read exceeds the requested transfer length in the command descriptor block or the mode header block size field, whichever is appropriate

3.1.60 overwrite

write operation that records a logical object in a logical position that is not an append point

3.1.61 page

regular parameter structure (or format) used by several commands and identified with a value known as a page code

3.1.62 partition

entire usable region for recording and reading in a volume or in a portion of a volume, defined in a vendor-specific or format-specific manner (see 4.2.7)

3.1.63 principal density code

principal density code is a density code selected by the device server; the logical unit specifies the principal density code by reporting a DEFLT bit set to one in the density support data block descriptor for supported densities in response to the REPORT DENSITY SUPPORT command (see 7.12); the selection of the principal density code is vendor specific

3.1.64 programmable-early-warning zone (PEWZ) (see 4.2.6)

zone within a partition that has its EOP side established at early warning and extends towards BOP for a distance indicated by the PEWS field (see 8.5.8)

3.1.65 recording volume

volume (see 3.1.95) containing a recording medium (see 4.2.2.3)

3.1.66 reservation loss

event caused by the release of a reserve/release method reservation (see SPC-2) or by the transition within the device server from the state where a persistent reservation holder exists to the state where a persistent reservation holder does not exist (see SPC-4)

3.1.67 SCSI device (see SAM-5)

device that contains one or more SCSI ports that are connected to a service delivery subsystem and supports a SCSI application protocol

3.1.68 SCSI domain (see SAM-5)

interconnection of two or more SCSI devices and a service delivery subsystem

3.1.69 SCSI initiator device

SCSI device containing application clients and SCSI initiator ports that originates device service and task management requests to be processed by a SCSI target device and receives device service and task management responses from the SCSI target devices

3.1.70 SCSI initiator port

SCSI initiator device object that acts as the connection between application clients and the service delivery subsystem through which requests, indications, responses, and confirmations are routed

3.1.71 SCSI port

SCSI device resident object that connects the application client, device server or task manager to the service delivery subsystem through which requests and responses are routed; a SCSI port is one of: a SCSI initiator port, or a SCSI target port

3.1.72 SCSI target device

SCSI device containing logical units and SCSI target ports that receives device service and task management requests for processing and sends device service and task management responses to SCSI initiator devices

3.1.73 SCSI target port

SCSI target device object that contains a task router and acts as the connection between device servers and task managers and the service delivery subsystem through which indications and responses are routed

3.1.74 security metadata

data used by security methods to enable user data to be returned in the form it existed prior to the application of the security methods (e.g., logical block encryption parameters, passwords, wrapped keys); security metadata may be used for vendor-specific security methods

3.1.75 sense data (see SPC-4)**3.1.76 sense key (see SPC-4)****3.1.77 service delivery subsystem (see SAM-5)**

part of an SCSI I/O system that transmits service requests to a logical unit or SCSI target device and returns logical unit or SCSI target device responses to a SCSI initiator device

3.1.78 spacing

act of positioning the medium on a sequential-access device while processing a SPACE command

3.1.79 status (see SAM-5)

one byte of response information sent from a device server to an application client upon completion of each command

3.1.80 synchronize operation

process of writing buffered logical objects to the medium (see 4.2.14)

3.1.81 user data segment (UDS)

A contiguous sequence of logical objects identified by the partition, the beginning logical object identifier, and the ending logical object identifier (see 4.2.10)

3.1.82 write sequence

one or more WRITE(16), WRITE FILEMARKS(16), or ERASE(16) commands delineated by the FCS and LCS bits (see 5.6, 5.7, and 5.2)

3.1.83 tape (see 4.2.2)

medium that interacts with the read/write mechanism

3.1.84 TapeAlert

device server capability that provides device diagnostic information using a standard interface

3.1.85 thread

process in which the medium is being engaged for positioning on a suitable transport mechanism (e.g., spooled on to a take-up reel or wrapped around the surface of a helical scan drum); after threading is complete, the tape device may begin positioning the medium to an initial position

3.1.86 track

contiguous line on the medium consisting of a pattern of recorded signals written by one write component

3.1.87 track group

set of tracks that are recorded at the same time

3.1.88 unbuffered mode

mode of operation where write data is written directly to the medium without being buffered; unbuffered mode is specified by a zero value (0h) in the BUFFER MODE field in the mode parameter header (see 8.5); unbuffered mode is the opposite of buffered mode (see 3.1.9)

3.1.89 underlength

incorrect-length condition that exists after processing a READ command if the requested transfer length in the command descriptor block or the mode header block size field, whichever is appropriate, exceeds the length of the actual logical block read

3.1.90 unencrypted logical block

logical block containing cleartext (i.e., logical block that has not been enciphered by the device server)

3.1.91 unthread

part of the unloading process in which the medium is being disengaged from the suitable transport mechanism (e.g., de-spooled from a take-up reel or unwrapped from around the surface of a helical scan drum)

3.1.92 variable-block transfer

read or write type command with the FIXED bit set to zero

3.1.93 vendor-specific control metadata

vendor-specific information stored on the recording volume outside the user data area(s) that is used to control or specify how the recording volume is being used by application clients (e.g., directory information, partition information, EOD locations, copies of data stored in a vendor-specific manner, volume serial number information, number of logical blocks on the medium)

3.1.94 Verify operation

verification of a single logical block. See (see 4.2.12.3).

3.1.95 volume (see 4.2.2)

medium together with its physical carrier

3.1.96 volume coherency set

set of information contained in logical objects including a volume coherency count (see annex C) for which coherency across an entire volume is desired

3.1.97 zero

logical false condition of a variable

3.2 Acronyms

A-KAD	authenticated key-associated data
ADC	Automation/Drive Interface Commands
ALDC	Adaptive Lossless Data Compression: ISO/IEC 15200:1996
BOM	beginning-of-medium
BOP	beginning-of-partition
CDB	command descriptor block
DCLZ	Data Compression according to Lempel and Ziv: ISO/IEC 11558:1992
DEM	data encapsulation mechanism
ECC	error correction code
ECC	elliptic curve cryptography
ECDSA	Elliptic Curve Digital Signature Algorithm
ECIES	Elliptic Curve Integrated Encryption Scheme
ECMA	European Computer Manufacturers Association
EOD	end-of-data
EOM	end-of-medium
EOP	end-of-partition

EW	early-warning
HC	hybrid cipher
I/O	input-output
ID	identifier
IDRC	Improved Data Recording Capability
INCITS	InterNational Committee for Information Technology Standards
KDF	key derivation function
KEM	key encapsulation mechanism
LSB	least significant bit
M	mandatory
M-KAD	metadata key-associated data
MA	MAC algorithm
MAC	message authentication code
MAM	medium auxiliary memory
MSB	most significant bit
NA	not applicable
O	optional
PEWZ	programmable-early-warning zone
RSA	Rivest-Shimir-Aadleman
Rsvd	reserved
SA	security association
SAM-5	SCSI Architecture Model - 5
SC	symmetric cipher
SCSI	small computer system interface
SDK	supplemental decryption keys
SMC-3	SCSI Media Changer Command - 3
SPC-2	SCSI Primary Commands - 2
SPC-4	SCSI Primary Commands - 4
SSC-4	SCSI Stream Commands - 4
SSC-5	SCSI Stream Commands - 5 (this standard)
UDS	user data segment
U-KAD	unauthenticated key-associated data

3.3 Keywords

3.3.1 invalid

A keyword used to describe an illegal or unsupported field or code value; receipt of an invalid field or code value shall be reported as an error.

3.3.2 ignored

Keyword used to describe an unused field or code value; the contents or value of an ignored field or code value shall not be examined by the receiving SCSI device and may be set to any value by the transmitting SCSI device

3.3.3 mandatory

Keyword indicating an item that is required to be implemented as defined in this standard

3.3.4 may

Keyword that specifies flexibility of choice with no implied preference (equivalent to "may or may not")

3.3.5 may not

Keyword that specifies flexibility of choice with no implied preference (equivalent to "may or may not")

3.3.6 obsolete

keyword indicating that an item was defined in prior SCSI standards but has been removed from this standard

3.3.7 optional

keyword that describes features that are not required to be implemented by this standard; however, if any optional feature defined by this standard is implemented, then it shall be implemented as defined in this standard

3.3.8 reserved

keyword referring to fields and code values that are set aside for future standardization; a reserved field shall be set to zero, or in accordance with a future extension to this standard; recipients are not required to check reserved fields for zero values; receipt of reserved code values in defined fields shall be reported as an error

3.3.9 shall

keyword indicating a mandatory requirement; designers are required to implement all such mandatory requirements to ensure interoperability with other products that conform to this standard

3.3.10 should

keyword indicating flexibility of choice with a strongly preferred alternative; equivalent to the phrase "it is strongly recommended"

3.3.11 vendor specific

items (e.g., fields, code values, etc.) that are not defined by this standard and may be vendor defined

3.4 Editorial Conventions

Certain words and terms used in this standard have a specific meaning beyond the normal English meaning. These words and terms are defined either in clause 3.1 or in the text where they first appear. Names of commands, statuses, sense keys, additional sense codes, and additional sense code qualifiers are in all uppercase (e.g., REQUEST SENSE). Lowercase is used for words having the normal English meaning.

The names of fields are in small uppercase (e.g., ALLOCATION LENGTH). If a field name is a concatenation of acronyms, uppercase letters may be used for readability (e.g., NORMACA). Normal case is used if the contents of a field are being discussed. Fields containing only one bit are usually referred to as the name bit instead of the name field.

Numbers that are not immediately followed by lower-case b or h are decimal values.

Numbers immediately followed by lower-case b (xxb) are binary values.

Numbers or upper case letters immediately followed by lower-case h (xxh) are hexadecimal values.

The most significant bit of a binary quantity is shown on the left side and represents the highest algebraic value position in the quantity.

If a field is specified as not meaningful or the field is to be ignored, the entity that receives the field shall not take any action based on the value of that field.

Lists sequenced by lowercase or uppercase letters show no ordering relationship between the listed items.

EXAMPLE 1 - The following list shows no relationship between the colors named:

- a) red, specificity one of the following colors:
 - A) crimson; or

- B) amber;
- b) blue; or
- c) green.

Lists sequenced by numbers show an ordering relationship between the listed items.

EXAMPLE 2 - The following list shows the order in which a page is meant to be read:

- 1) top;
- 2) middle; and
- 3) bottom.

Lists are associated with an introductory paragraph or phrase, and are numbered relative to that paragraph or phrase (i.e., all lists begin with an a) or 1) entry).

Lists sequenced by letters (e.g., a-red, b-blue, c-green) show no priority relationship between the listed items. Numbered lists (e.g., 1-red, 2-blue, 3-green) show a priority ordering between the listed items.

If a conflict arises between text, tables, or figures, the order of precedence to resolve the conflicts is text; then tables; and finally figures. Not all tables or figures are fully described in the text. Tables show data format and values. Notes do not constitute any requirements for implementors.

This standard uses the following conventions for representing decimal numbers:

- a) the decimal separator (i.e., separating the integer and fractional portions of the number) is a period;
- b) the thousands separator (i.e., separating groups of three digits in a portion of the number) is a space; and
- c) the thousands separator is used in both the integer portion and the fraction portion of a number.

Table 1 shows some examples of decimal numbers represented using various conventions.

Table 1 — Numbering conventions examples

French	English	This standard
0,6	0.6	0.6
3,141 592 65	3.14159265	3.141 592 65
1 000	1,000	1 000
1 323 462,95	1,323,462.95	1 323 462.95

3.5 Notation Conventions

3.5.1 Notation for state diagrams

All state diagrams use the notation shown in figure 1.

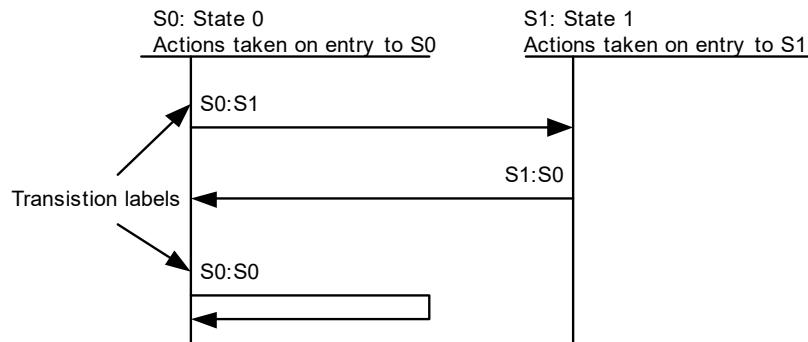


Figure 1 — Example state diagram

The state diagram is followed by a list of the state transitions, using the transition labels. Each transition is described in the list with particular attention to the conditions that cause the transition to occur and special conditions related to the transition. Using figure 1 as an example, the transition list might read as follows:

Transition S0:S1: This transition occurs if state S0 is exited and state S1 is entered.

Transition S1:S0: This transition occurs if state S1 is exited and state S0 is entered.

Transition S0:S0: This transition occurs if state S0 transitions to itself. It is particularly important to note that the actions taken if state S0 is entered are repeated every time this transition occurs.

A system specified in this manner has the following properties:

- time elapses only within discrete states;
- state transitions are logically instantaneous; and
- every time a state is entered, the actions of that state are started. Note that this means that a transition that points back to the same state restarts the actions from the beginning.

4 General Concepts

4.1 Overview

The sequential-access device type has the characteristic of primarily handling data in a sequential manner (i.e., a stream). This does not limit the device's ability to position randomly within the data although a sequential-access device is not truly random-access.

This standard describes two modes and associated command sets for communicating with a sequential-access device:

- a) implicit address mode. Commands to read and write on a sequential-access device do not contain any positioning information fields. Instead, the device position is normally determined by previous commands; and
- b) explicit address mode. Commands to read and write on a sequential-access device contain positioning information fields.

Commands are available for absolute and relative positioning. Writing to a sequential-access device may cause all data starting at the point at which the data is written to be invalidated. There may be restrictions on where write operations may be initiated. Reading or writing data as a long string of data, as in a stream, tends to be the most efficient.

4.2 Sequential-access device model

4.2.1 Sequential-access device model overview

Sequential-access devices are described herein from the point of view of a tape device. However, other implementations are not precluded.

Sequential-access devices optimize their use in storing or retrieving user data in a sequential manner. Since access is sequential, position changes typically take a long time compared to random-access devices.

4.2.2 Volumes and medium

4.2.2.1 Volumes

A volume is composed of:

- a) the medium (see 4.2.2.3);
- b) its physical carrier (e.g., reel, cartridge, cassette);
- c) MAM, if present; and
- d) any other elements contained in the physical carrier (e.g., cartridge memory).

Volumes have an attribute of being mounted or demounted on a suitable transport mechanism.

A volume is mounted when the device is physically capable of processing operations that involve interactions between the read/write element(s) of the device and the medium. The interactions between the read/write element(s) of the device and the medium may vary depending on the volume type (e.g., altering or detecting the magnetic polarization of a magnetically recordable medium or physical abrasion of the read/write element(s) for a cleaning medium). During operations involving a cleaning volume, some device servers position to a previously

unused location on the medium prior to performing the cleaning operation. For such technologies, the device server should consider the volume as mounted prior to positioning over the previously used locations on the cleaning volume. Some technologies do not position medium as part of the cleaning operation; for such technologies, the device server should consider the volume as mounted prior to beginning the cleaning operation. A volume is defined as demounted while it is being loaded, threaded, unloaded, unthreaded, or while not attached to the device.

A sequential-access device may support removable volumes. The sequential access device indicates that volumes are removable by setting the RMB bit to one in its standard INQUIRY data (see SPC-4).

A sequential-access device may support one or more of the following types of volumes:

- a) recording volumes (see 3.1.65);
- b) cleaning volumes (see 3.1.11); or
- c) microcode update volumes (see 3.1.54).

If the volume in a sequential-access device is removable, and a volume transitions from demounted to mounted, then the device server shall establish a unit attention condition with the additional sense code set to NOT READY TO READY CHANGE, MEDIUM MAY HAVE CHANGED.

The LOAD UNLOAD command (see 7.4) is used to mount or demount the volume.

The PREVENT ALLOW MEDIUM REMOVAL command (see 7.6) allows an application client to restrict the demounting of the removable volume.

A logical unit is in the ready state when medium access commands are able to be processed by the device server. A logical unit is in the not ready state if no volume is mounted or if any medium access command reports CHECK CONDITION status and a NOT READY sense key. The logical unit is in the not ready state during the transition from mounted to demounted, or demounted to mounted. Devices may have a physical control that places the device in the not ready state even if a volume is mounted.

4.2.2.2 Medium

The medium for tape devices consists of various widths and lengths of a flexible substrate. Most sequential-access devices use a recording medium for storage of logical objects and some sequential-access devices also use a cleaning medium for cleaning the read/write mechanism. A recording medium has the substrate coated with a semi-permanent magnetic material. A cleaning medium may use an abrasive substrate or other substrates designed for cleaning the read/write mechanism. The medium may be spooled onto single reels or encapsulated into cartridges containing both a supply reel and a take-up reel. Several American National Standards exist covering the construction of reels and cartridges for interchange as well as recording techniques for many of the format and density combinations.

For a tape device, a medium exists between two reels, the supply reel and take-up reel. The read/write mechanism interacts with the medium between the reels. As the medium is taken out of one reel, it passes by the read/write mechanism and into the other reel. Transferring data as a stream is most efficient, since the recording medium traverses the read/write mechanism producing a flow of data. To position to a given point requires moving the medium until the appropriate position is found.

The medium has two physical attributes called beginning-of-medium (BOM) and end-of-medium (EOM). Beginning-of-medium is at the end of the medium that is attached to the take-up reel. End-of-medium is at the end of the medium that is attached to the supply reel. In some cases, the medium is permanently affixed to one or both of the reel hubs. For a recording medium, BOM or EOM is not required to be related to the BOP or EOP of any partition.

4.2.2.3 Recording volume model

Sequential-access devices store logical objects on recording volumes (see 3.1.65). A recording volume contains a recording medium.

As shown in figure 2, a portion of the physical length of recording medium is not usable for recording data. For most recording volumes, a length of the recording medium is reserved between the take-up reel and the beginning-of-medium, and between the end-of-medium position and the supply reel.

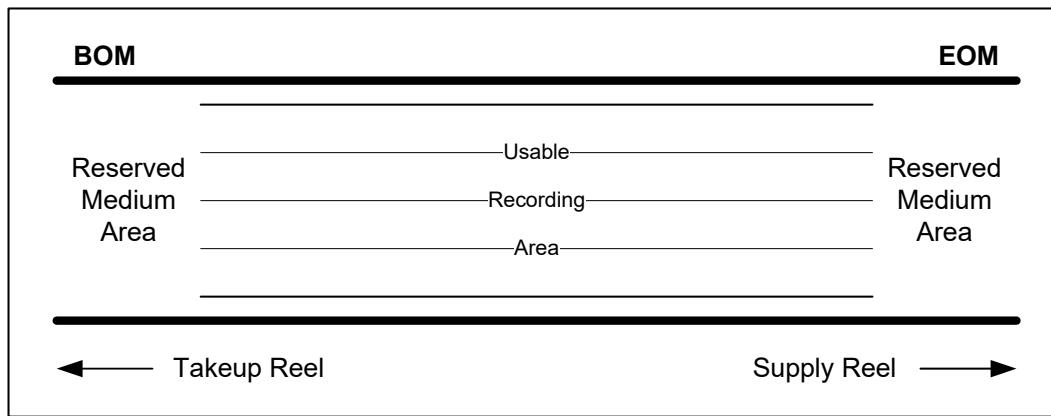


Figure 2 — Example recording medium layout

The position on the recording medium where write component(s) record a pattern of signals is called a track (see figure 3). A device writes or reads from one or more tracks at a time, depending on the format.

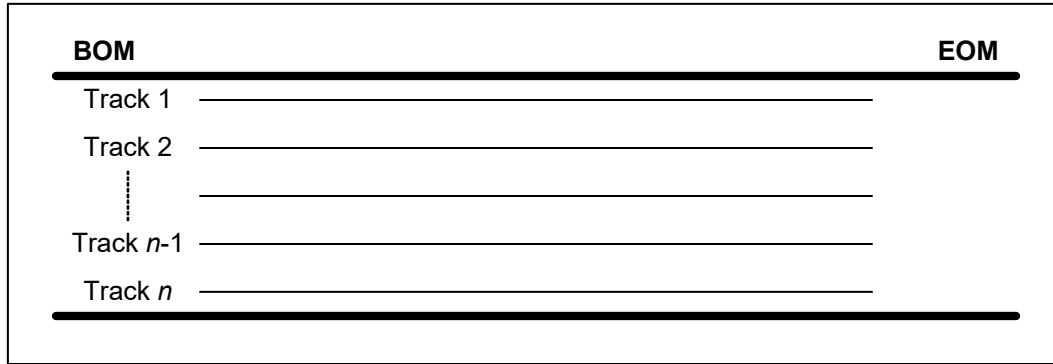


Figure 3 — Example recording medium track layout

On a new recording volume, recording of one or more tracks begins after mounting the recording volume and the medium is pulled from beginning-of-medium toward end-of-medium. The number of tracks written at one time is called a track group (TrkGrp). Track groups may be used by any recording format. Reading recorded volumes in the forward direction follows the same course of tracks as when writing.

In serpentine recording, not all tracks are recorded at the same time. At the end-of-medium or beginning-of-medium, the device reverses direction and begins recording the next track group. The process of reversing direction and recording the next track group may be repeated until all track groups are recorded. For

serpentine devices that record only one track at a time, each physical track represents one track group (see figure 4).

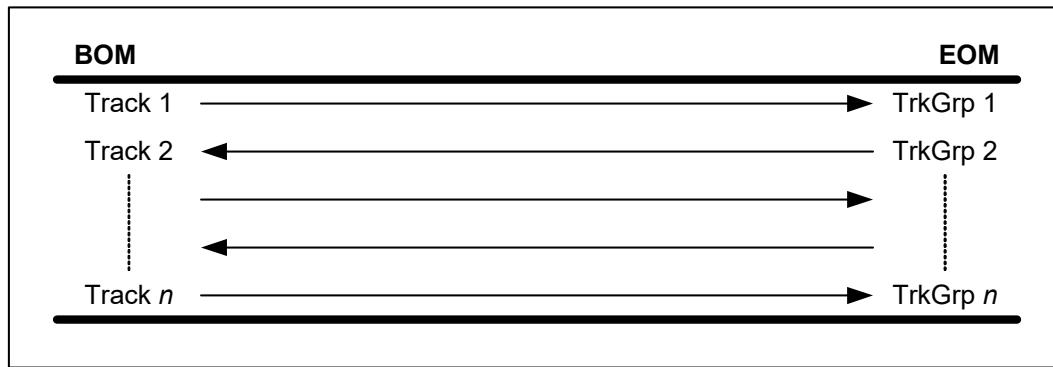


Figure 4 — Serpentine recording example

Some multi-track devices have only one track group, using a parallel storage format that supports the simultaneous recording of all available tracks (see figure 5).

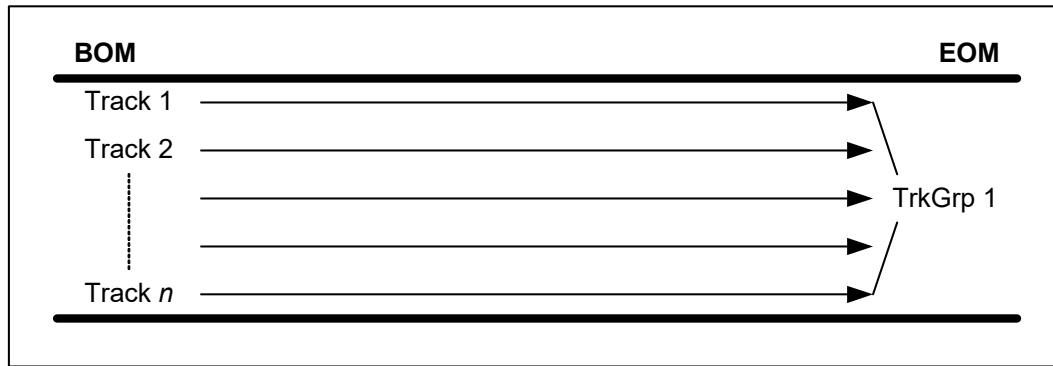


Figure 5 — Parallel recording example

The serpentine and parallel recording formats shown in the previous examples define tracks as longitudinal patterns of recorded information. One other storage format used by some devices records tracks diagonally across the recording medium. Recording of one or more tracks occurs at the same time. This recording technique is known as helical scan (see figure 6).

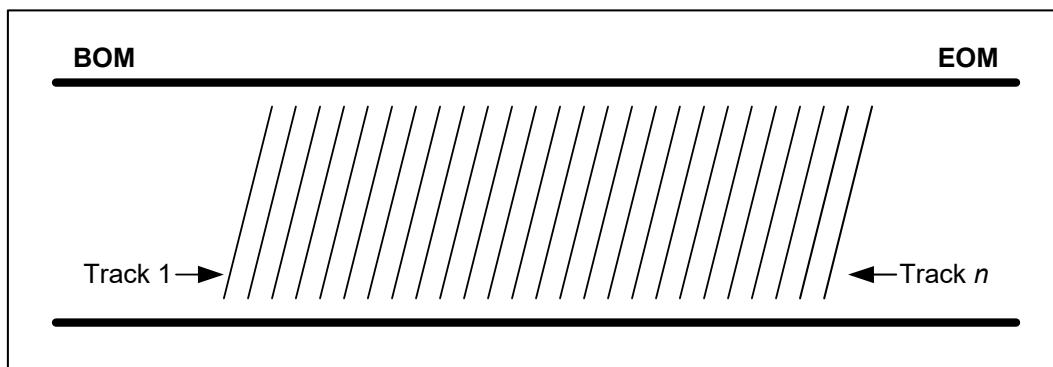


Figure 6 — Helical scan recording example

For most recording formats, a format identification in the form of a tone burst or some other recognizable pattern is recorded outside the user data area. The format identification is an attribute of a volume used for interchange purposes and is defined in applicable standards.

4.2.2.4 WORM volume model

Sequential-access devices may support a Write Once, Read Many (WORM) volume. A WORM volume is a kind of recording volume (see 4.2.2.3) that is used by the device server in a manner that prevents the overwriting, erasing or otherwise altering of logical objects (e.g., in a manner that complies with government regulations for electronic data retention).

The method by which a device server identifies a WORM volume is vendor-specific (e.g., the Medium Type attribute in the MAM data returning a Write Once Medium type value). The device server shall prevent modification (i.e., overwriting, erasing, or otherwise altering) of logical objects on a WORM volume except as specified in the Medium Configuration mode page (see 8.5.7). The prevention of logical object modification on WORM volumes shall apply to all commands and operations that modify logical objects (e.g., commands such as WRITE(6), ERASE(16), SET CAPACITY or operations such as those caused by a MODE SELECT command).

If prevention of logical object modification on WORM volumes would be violated by a command being processed, then the device server shall terminate the command with CHECK CONDITION status. The sense key shall be set to DATA PROTECT and the additional sense code shall be set to WORM MEDIUM - OVERWRITE ATTEMPTED. If a write protection is in effect for the device server, volume, or medium (see 4.2.20), it shall take precedence over the prevention of logical object modification on WORM volumes violation.

If a device server is unable to determine if a prevention of logical object modification on WORM volumes would be violated by a command being processed, then the device server shall terminate the command with CHECK CONDITION status, the sense key shall be set to DATA PROTECT, and the additional sense code shall be set to WORM MEDIUM - OVERWRITE ATTEMPTED. If a write protection is in effect for the device server, volume, or medium (see 4.2.20), it shall take precedence over the prevention of logical object modification on WORM volumes violation.

If a WORM volume is mounted and the device server supports the mounted WORM volume but does not support writing to the mounted WORM volume, then the device server shall treat the volume as write protected. A command that attempts to alter the WORM volume shall be terminated with CHECK CONDITION status with the sense key set to DATA PROTECT and the additional sense code set to CANNOT WRITE MEDIUM - INCOMPATIBLE FORMAT.

If a WORM volume is mounted and the device server does not support the mounted WORM volume, then the device server shall treat the volume as an incompatible medium. A medium access command, except a LOAD UNLOAD with the LOAD bit set to zero, shall be terminated with CHECK CONDITION status, with the sense key set to DATA PROTECT and the additional sense code set to INCOMPATIBLE MEDIUM INSTALLED.

4.2.2.5 Cleaning volume model

Sequential-access devices may support using a cleaning volume to clean the read/write mechanism. A cleaning volume may contain a cleaning medium. The cleaning medium may clean the read/write mechanism by abrading the surface or by other methods.

As shown in figure 7, a portion of the physical length of a cleaning medium is not usable for cleaning the read/write mechanism. For most cleaning volumes, a length of the cleaning medium is reserved between the take-up reel and the beginning-of-medium, and between the end-of-medium position and the supply reel. This is done to provide a sufficient tape wrap onto the reel hub and to ensure that interactions between the cleaning substrate and the read/write mechanism start in a section of the medium designed for cleaning the read/write mechanism.

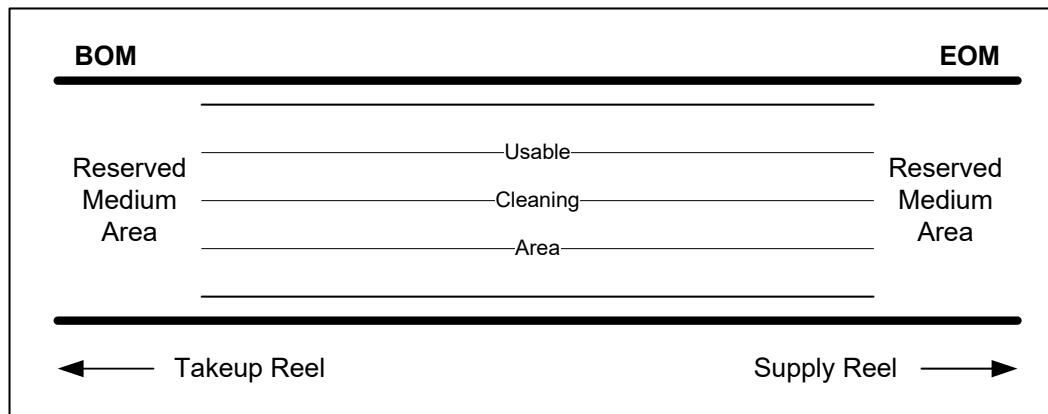


Figure 7 — Example cleaning medium layout

4.2.2.6 Microcode update volume model

Sequential-access devices may support updating microcode by using a microcode update volume. A microcode update volume shares most of its characteristics with a recording volume (see 4.2.2.3). A microcode update volume contains one or more pre-recorded microcode images and a signature (e.g., a header or MAM field) that identifies the volume as a microcode update volume. Sequential-access devices that support microcode update volumes may detect that the mounted volume is a microcode update volume and transition directly into the firmware update process as soon as the volume has been mounted. Upon completion of processing the microcode update volume, the sequential-access device may automatically demount the microcode update volume. Methods for converting microcode update volumes to any other volume type (e.g., recording volumes) are vendor specific.

The format of the microcode update volume is vendor specific.

4.2.3 Device entity

A sequential-access device contains one or more device entities. A device entity provides storage for values that are shared between multiple device servers and performs operations upon the volume (e.g., loading, unloading, positioning, writing, and reading the medium, and reading and writing medium auxiliary memory).

The device entity is controlled by various other entities, which may include:

- a) one or more SCSI device servers (e.g., where a device entity is associated with an automation device that performs medium movement, both a device server that implement the command set defined in this standard and a device server that implements another command set such as ADC-3 may control the device);
- b) an operator interface;
- c) a management interface; and
- d) a media changer.

A media changer may control the device entity by inserting a volume into or removing a volume from the device entity. While inserting a volume into or removing a volume from the device entity, the operator acts in the role of a media changer.

These entities perform operations that change various attributes of the device entity. These attributes affect the operations on a volume. Figure 8 shows in UML notation an example of the entities in a SCSI target device, and shows the attributes of the device entity.

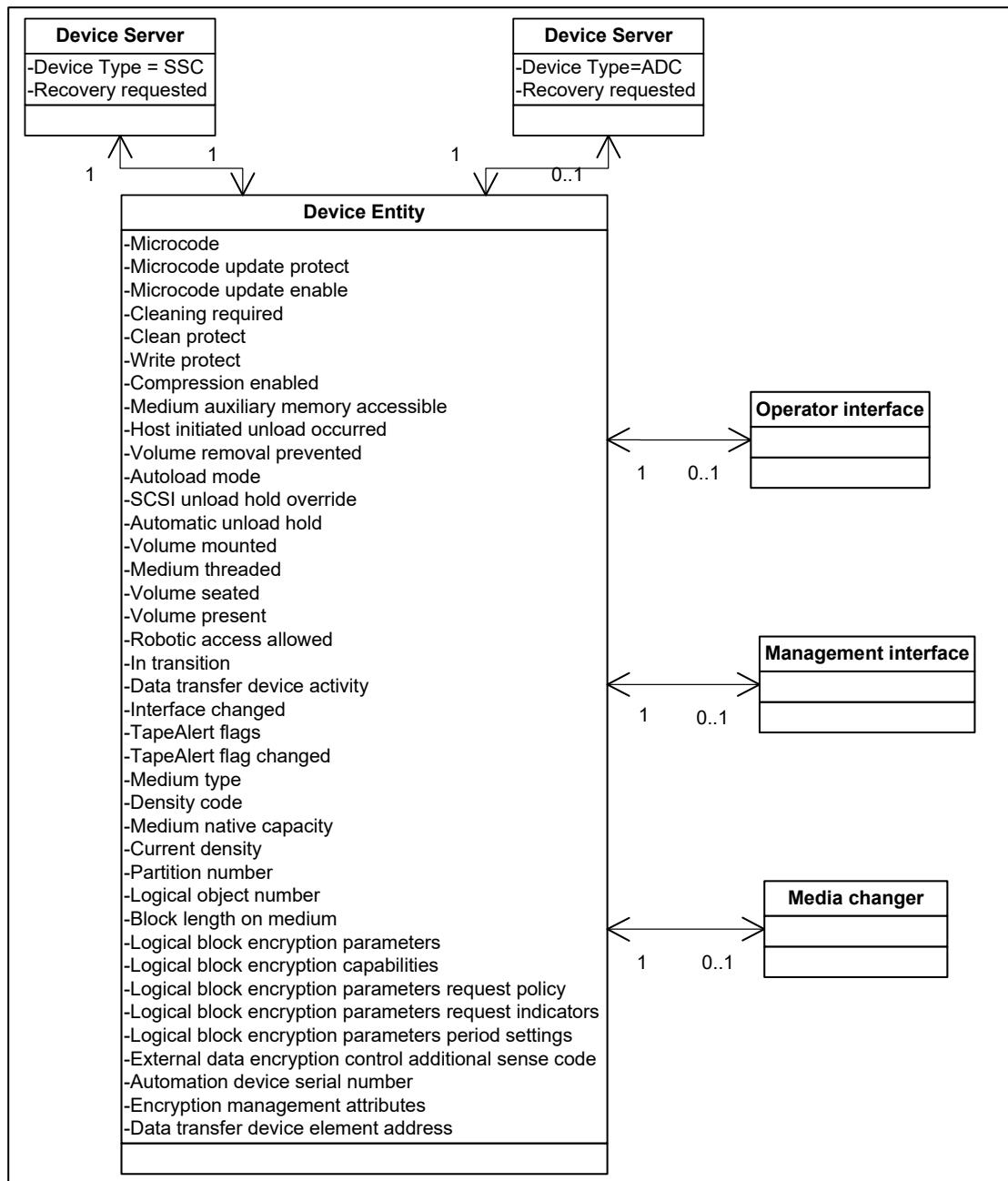


Figure 8 — UML example of SCSI target device and device entity

Table 2 specifies the standard that defines each attribute shown in figure 8.

Table 2 — Device entity attributes (part 1 of 2)

Attribute	Reference
Microcode	SPC-4
Microcode update protect	ADC-3
Microcode update enable	ADC-3
Cleaning required	ADC-3
Clean protect	ADC-3
Write protect	ADC-3
Compression enabled	ADC-3
Medium auxiliary memory accessible	ADC-3
Host initiated unload occurred	ADC-3
Volume removal prevented	ADC-3
Autoload mode	SPC-4
SCSI unload hold override	ADC-3
Automatic unload hold	ADC-3
Volume mounted	ADC-3
Medium threaded	ADC-3
Volume seated	ADC-3
Volume present	ADC-3
Robotic access allowed	ADC-3
In transition	ADC-3
Data transfer device activity	ADC-3
Interface changed	ADC-3
TapeAlert flags	Table 12
TapeAlert flag changed	ADC-3
Medium type	7.12.4
Density code	8.3.4.2.2
Medium native capacity ^a	7.12.3
Current density	ADC-3
Partition number	7.9.3
Logical object number	7.9.3
Block length on medium	SPC-4
Logical block encryption parameters	4.2.29.15

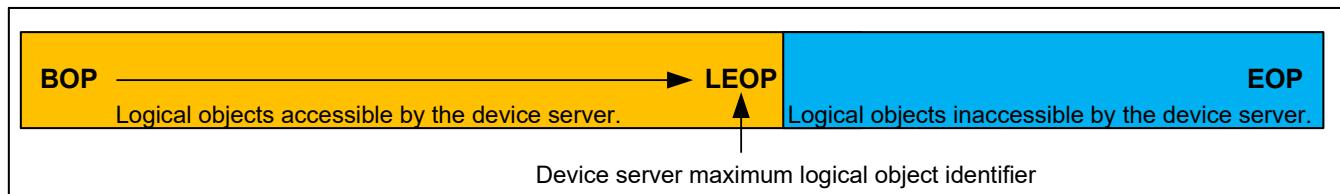
a) Medium native capacity is the value reported in the CAPACITY field of the density support data block descriptor if the MEDIA bit is one, and a SET CAPACITY command has not been used to affect the capacity of the medium.

Table 2 — Device entity attributes (part 2 of 2)

Attribute	Reference
Logical block encryption capabilities	4.2.29.7
Logical block encryption parameters request policy	4.2.30.3.2
Logical block encryption parameters request indicators	4.2.30.3.3
Logical block encryption parameters period settings	4.2.30.3.4
External data encryption control additional sense code	4.2.30.5
Automation device serial number	8.6.5
Encryption management attributes	4.2.31.5
Data transfer device element address	8.6.6
a) Medium native capacity is the value reported in the CAPACITY field of the density support data block descriptor if the MEDIA bit is one, and a SET CAPACITY command has not been used to affect the capacity of the medium.	

4.2.4 End-of-partition

The end of partition of a volume is represented by an end-of-partition (EOP) and a logical end-of-partition (LEOP). The EOP is determined by the format specification of the volume. Often the EOP is determined by the proximity to the end of the permissible recording region on the medium. The LEOP is determined by combining the EOP and any restrictions the device server has on use of that volume such as a maximum logical object identifier (see 7.7.3) that the device server supports (e.g., $2^{32}-1$). Figure 9 shows an example of a single partition volume that is mounted in a device server that has LEOP in a different location than EOP.

**Figure 9 — Logical end-of-partition example**

4.2.5 Early-warning

If writing, the application client needs an indication that it is approaching the end of the permissible recording area (i.e., end of the partition (see 4.2.7)). This position, called early-warning (EW), is typically reported to the application client at a position early enough for the device to write any buffered logical objects to the medium while still leaving enough room for additional recorded logical objects (see figure 10 and figure 11). Some American

National Standards include physical requirements for a marker placed on the medium to be detected by the device as early-warning. An early-warning example where LEOP is the same as EOP is shown in figure 10.

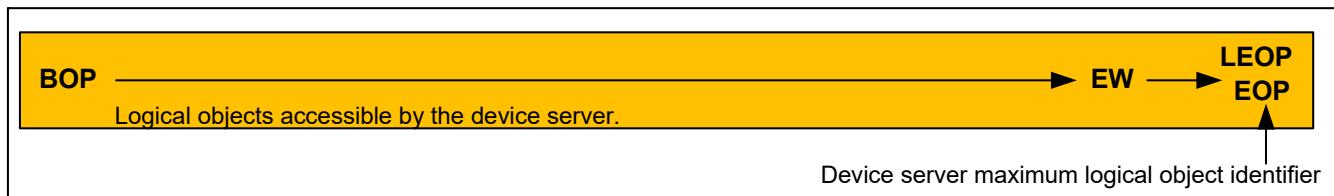


Figure 10 — Early-warning example where LEOP is the same as EOP

An early-warning example where LEOP is not the same as EOP is shown in figure 11.

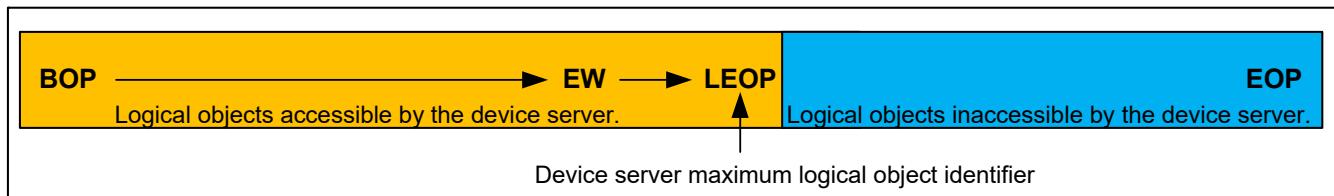


Figure 11 — Early-warning example where LEOP is not the same as EOP

Devices should report early warning to the application client while sufficient recording space is nominally available before LEOP to record logical objects in the object buffer(s) and some additional logical objects. A logical concept of early-warning may be required to signal the application client at an appropriate location prior to the physical marker, particularly for devices that implement object buffers.

4.2.6 Programmable early warning

If writing, the application client may need an indication prior to early warning (see 4.2.5) to allow for the application client to prepare to be ready for early warning (e.g., flush buffers in the application client).

Application clients that need this indication may set the PEWS field (see 8.5.8) to a value that creates a PEWZ of the specified size.

Application clients that need this indication may request the device server to create a zone called the programmable-early-warning zone (PEWZ) (see figure 12) by setting the PEWS field (see 8.5.8) to the requested size of the PEWZ. The EOP side of PEWZ is established at early-warning and extends towards BOP for a distance indicated by the PEWS field. A programmable early-warning example where LEOP is the same as EOP is shown in figure 12.

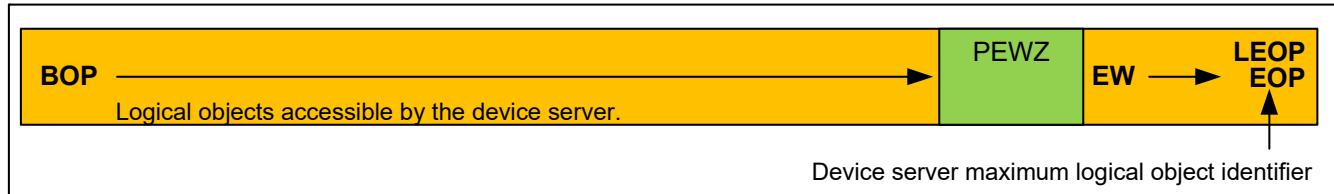


Figure 12 — Programmable early warning example where LEOP is the same as EOP

A programmable early-warning example where LEOP is not the same as EOP is shown in figure 13.

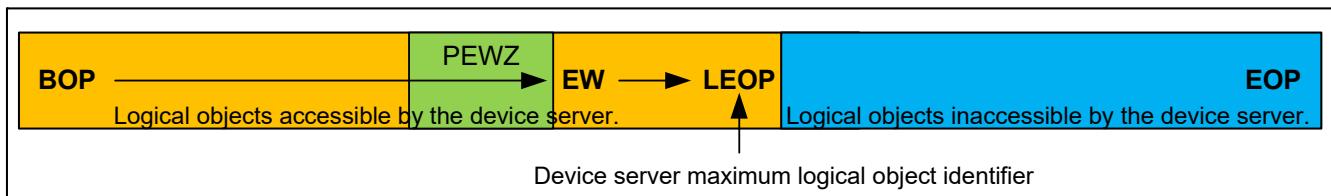


Figure 13 — Programmable early warning example where LEOP is not the same as EOP

The REW bit in the Device Configuration mode page (see 8.5.3) shall have no effect on the device server behavior in the PEWZ.

The device server shall return CHECK CONDITION status, with the sense key set to NO SENSE, the EOM bit set to one and the additional sense code set to PROGRAMMABLE EARLY WARNING DETECTED at the completion of a command that caused the medium to transition into the PEWZ if that command is:

- a) WRITE(6);
- b) WRITE(16);
- c) WRITE FILEMARKS(6); or
- d) WRITE FILEMARKS(16).

Encountering the PEWZ shall not cause the device server to perform a synchronize operation or terminate the command. If processing this command results in any other exception condition except early-warning, the CHECK CONDITION status associated with that exception condition shall be reported instead. If early-warning is crossed prior to the PROGRAMMABLE EARLY WARNING DETECTED additional sense being reported, the PROGRAMMABLE EARLY WARNING DETECTED additional sense shall be reported before the early-warning CHECK CONDITION.

If the PROGRAMMABLE EARLY WARNING DETECTED additional sense code was not reported, the next write in PEWZ or beyond early-warning for which it is possible to complete with GOOD status, shall return the programmable-early-warning CHECK CONDITION instead. This condition may occur if the PEWS field value is larger than the native capacity of the current partition.

If the PEWZ is entered and exited on the BOP side before the PROGRAMMABLE EARLY WARNING DETECTED additional sense code is returned, the device server shall not report CHECK CONDITION status with the additional sense code set to PROGRAMMABLE EARLY WARNING DETECTED.

4.2.7 Partitions within a recording volume

Partitions consist of one or more non-overlapped logical recording volumes, each with its own beginning and ending points, contained within a single physical recording volume. Each partition (n) within a recording volume has a defined beginning-of-partition (BOP n), an early-warning position (EW n), and an end-of-partition (EOP n).

All recording volumes have a minimum of one partition, called partition 0, which is the default data partition. For devices that support only one partition, the beginning-of-partition zero (BOP 0) may be equivalent to the beginning-of-medium and the end-of-partition zero (EOP 0) may be equivalent to the end-of-medium. For devices that support more than one partition, they shall be numbered sequentially starting with zero (i.e., beginning-of-partition 0).

At the successful completion of a mount, a recording volume is logically positioned to the beginning of the default data partition (BOP 0). If a REWIND command is received in any partition (x), the device positions to the beginning-of-partition of the current partition (BOP x).

Partitions on a recording volume may be recorded in any order and use any partition number unique to the physical recording volume. It is sufficient for a device to be able to locate a partition, given its partition number, or to determine that it does or does not exist on the recording volume. For interchange, information about which partitions are present on a recording volume may be stored on the recording volume in a format specified area, possibly unavailable to the application client, or the information may be an intrinsic attribute of the device implementation.

Figure 14 shows a possible partition implementation for a four-track serpentine recording device, assuming that each track group defines a partition.

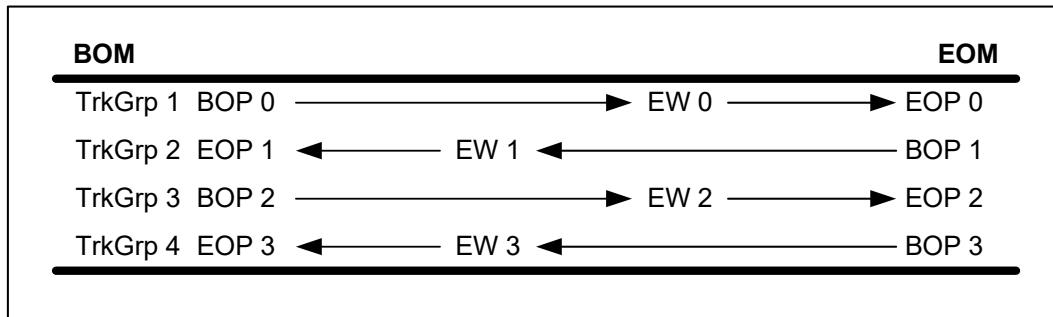


Figure 14 — Partitioning example - one partition per track group

Another possible partition implementation for this four-track serpentine recording device is shown in figure 15, using two track groups to define each partition.

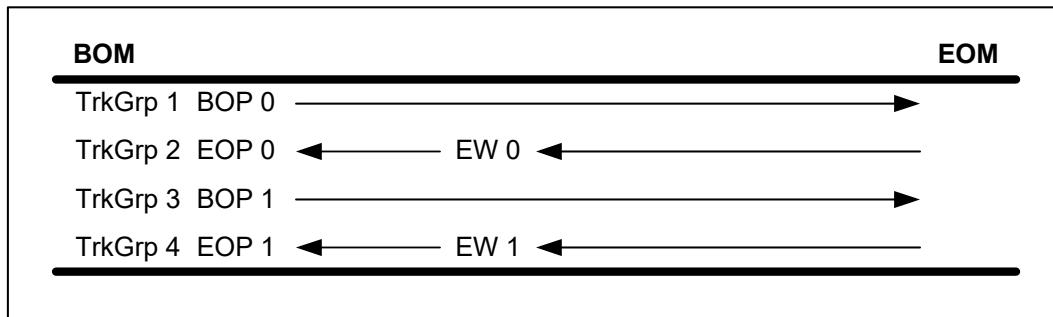


Figure 15 — Partitioning example - one partition per two track groups

The previous examples show the beginning and ending points for a partition aligned with physical bounds of the medium. This is not required for partitioning. It is sufficient for a device to be able to locate to and stay in any partition bounded by a BOP x and EOP x. In this case, it is possible for a device-recognizable attribute to be used to delineate the partitions. Figure 16 shows a possible two-partition implementation for a device with only one track group.

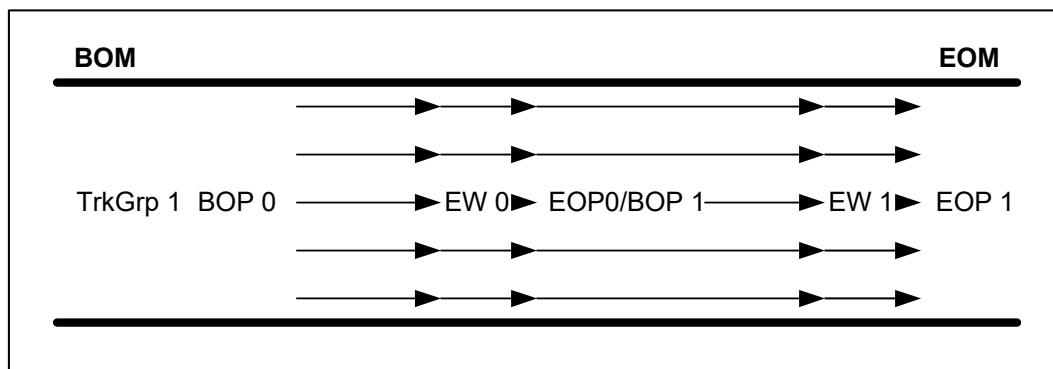


Figure 16 — Partitioning example - two partitions per track group

Three methods are defined in the MODE SENSE and MODE SELECT commands for managing partitions:

- device-defined fixed locations;
- device-defined based on an application client supplied number of partitions and a vendor-specific allocation algorithm; and
- definition by partition number and capacity by an application client.

4.2.8 Logical objects

4.2.8.1 Logical objects within a partition

The area between BOP n and EOP n on a recorded volume contains application client accessible logical objects. Logical objects are controlled and transferred between the application client and the medium using commands defined in this standard. Each logical object shall have a logical object identifier that is unique within a partition.

A device server that supports greater than 2^{32} logical object identifiers shall support setting the MLOI field of the READ BLOCK LIMITS command (see 7.7) to one.

The basic unit of data transferred by an application client is called a logical block. Logical blocks are stored according to the specifications of the format for the recording volume and may be recorded as portions of one or more physical blocks on the medium. The mapping between physical and logical blocks is the responsibility of the device server.

Filemarks are special recorded logical objects not containing user data. Proper recording and detection of filemarks is the responsibility of the device server. Application clients traditionally use filemarks to separate groups of user data from each other. Since some format standards do not define an explicit EOD, operating system software has often used conventions with filemarks to represent an EOD indication. In some implementations, the device's EOD definition may be specified by the application client using the Device Configuration mode page (see 8.5.3).

After logical objects have been written from BOP n , the partition is considered to contain a contiguous grouping of logical objects. Depending on the format, blank medium may be treated as an end-of-data indication, an error recovery area, or an unrecoverable medium error causing an interchange error. Unrecorded recording volumes, new or erased, may exhibit blank medium characteristics if an attempt is made to read or space the recording volume before data has been written.

A partition that has had logical objects written to it may contain more logical objects than a device server is capable of supporting (e.g., there are logical objects recorded beyond the maximum logical object identifier (i.e., beyond

LEOP)) or the device server does not support reporting READ POSITION data containing a location field larger than four bytes and there are logical objects recorded where the logical object identifiers require greater than four bytes to be represented).

Upon completion of mounting a recording volume, a device server capable of detecting logical objects after LEOP shall check to see if there are any logical objects after LEOP. If there are any logical objects after LEOP, then the device server shall treat that partition as write protected at all logical object identifiers except logical object identifier zero. If at least one partition is in this state, then the device server may treat each partition as write protected at all logical object identifiers except logical object identifier zero.

If a command that writes to a partition that contains logical objects recorded beyond LEOP is attempted, then that command shall be terminated with CHECK CONDITION status, the sense key set to DATA PROTECT, the additional sense code set to TOO MANY LOGICAL OBJECTS ON PARTITION TO SUPPORT OPERATION, and the sense data EOM bit set to one.

If a command that causes a read or locate beyond LEOP is attempted, then that command shall be terminated with CHECK CONDITION status, the sense key set to DATA PROTECT, and the additional sense code set to TOO MANY LOGICAL OBJECTS ON PARTITION TO SUPPORT OPERATION.

A sequential-access device may be capable of supporting fixed-block transfers or variable-block transfers. The concept of fixed or variable mode for writing and reading logical blocks only specifies the method by which the application client specifies the size of a logical block for transfer, and not the method of recording physical blocks on the medium. However, a device that supports only fixed-length physical blocks may only be capable of supporting logical blocks of the same length. The length of a logical block is always described in bytes. Refer to the READ BLOCK LIMITS command (see 7.7) for additional information about fixed-block transfers and variable-block transfers.

4.2.8.2 Logical object identifier

The logical object identifier value shall be a sequentially increasing number assigned to each logical object recorded in the partition starting with zero for the recorded logical object at BOP.

The READ POSITION command (see 7.9) may be used to determine a logical object identifier and the application client may use this value with a LOCATE command (see 6.3 and 7.5) or an explicit command (see clause 5) to position to the same location at some future time. If a READ POSITION command is enabled requesting a READ POSITION data format and the recording volume is positioned beyond the maximum value that the requested READ POSITION data format is able to represent, then the device server shall terminate the READ POSITION command with a CHECK CONDITION status. The sense key shall be set to DATA PROTECT and the additional sense code set to TOO MANY LOGICAL OBJECTS ON PARTITION TO SUPPORT OPERATION.

4.2.9 Logical files

4.2.9.1 Logical files within a partition

Application clients may use filemarks to separate groups of user data into logical files. A logical file shall contain zero or more contiguous logical blocks and shall begin with the next logical block following a filemark or BOP. Each logical file shall have a logical file identifier that is unique within the partition.

4.2.9.2 Logical file identifier

The logical file identifier value shall be a sequentially increasing number assigned to each logical file recorded in the partition starting with zero for the recorded logical file beginning at BOP.

The READ POSITION command may be used to determine a logical file identifier and the application client may use this value with a LOCATE(16) command to position to the BOP side of the same logical file at some future time.

4.2.10 User data segments

4.2.10.1 User data segments overview

Within a partition that has recorded logical objects, a contiguous sequence of logical objects (i.e., logical blocks and/or logical files) may be referenced as a user data segment (UDS).

4.2.10.2 User data segment descriptors

UDS descriptors are used to describe attributes of the UDS. There are two UDS descriptor formats, the GRAO - user data segment descriptor (see 7.3.2) and the RRAO - user data segment descriptor (see 7.10.1.3). Both descriptor formats contain the following:

- a) an application client specified name;
- b) a partition number;
- c) a beginning logical object identifier; and
- d) an ending logical object identifier.

Descriptors in response data may include additional information.

4.2.11 Logical object access command types

4.2.11.1 Logical object access command types overview

The following are logical object access command types:

- a) read commands;
- b) write commands;
- c) verify commands;
- d) positioning commands; and
- e) other commands (e.g., a FORMAT UNIT command or a SET CAPACITY command).

4.2.11.2 Verify commands

A verify command requests a sequence of verify operations (see 4.2.12.3) with a sequence termination condition specified by fields in the CDB. The verification sequence is a series of verify operations performed on each logical object beginning at the current logical position and ending if either a verification operation does not complete successfully or the expected sequence termination condition is met. The logical position on completion of the command is after the last logical object that was processed.

If the command indicates that the block length shall be checked and the length of the logical block does not match the expected block length, then the verification sequence shall terminate and the ILI bit shall be set to one in the sense data and the additional sense code shall be set to NO ADDITIONAL SENSE INFORMATION. Upon termination, the logical position shall be after the incorrect-length logical block (i.e., end-of-partition side). Refer to the READ(16) command (see 5.3) for a description of incorrect-length handling.

The verification sequence shall terminate unsuccessfully as follows:

- 1) if an unrecoverable read error is encountered;
- 2) if the logical end-of-partition is encountered;

- 3) if end-of-data is encountered and the VTE bit is set to zero;
- 4) if early-warning is encountered (i.e., if the device server is checking for early-warning due to the REW bit set to one in the Device Configuration mode page, see 8.5.3) and the VTE bit is set to zero;
- 5) if a filemark is encountered, the VBF bit is set to zero, and the VTE bit is set to zero;
- 6) if the logical block is unable to be processed due to incorrect encryption parameters;
- 7) if logical block integrity checks fail (e.g., a logical block protection check or a format specific integrity check);
- 8) if the command indicates that the block length shall be checked and an incorrect-length logical block is encountered;
- 9) if the BYTCMP bit is set to one and a byte-by-byte compare of the data on the medium and the data transferred from the application client does not match; or
- 10) if the logical block protection method used by the logical block is not the logical block protection method specified in the Control Data Protection mode page and the VLBPBM bit is set to one.

The status and sense data for each of the termination conditions are handled in the same manner as in the READ(16) command. Upon successful completion of a verify command, the logical position shall be after the last logical block verified (i.e., end-of-partition side).

4.2.12 Logical object access operation types

4.2.12.1 Write operations

A write operation may be used to record a logical object.

4.2.12.2 Read operations

A read operation may be used to read a previously recorded logical object.

4.2.12.3 Verify operations

A verify operation may be used to validate a previously recorded logical object.

The device server performs the following actions to accomplish a verify operation:

- 1) reads from the medium the physical block(s) that contain the logical object;
- 2) verifies that the physical block(s) pass validation of the format specific symbols that provide protection (e.g., ECC);
- 3) processes the logical object contained in the physical block(s) following the rules for reading a logical object as specified in 4.2.29.3.
- 4) verifies the protection information, if any, for the logical object being verified following the rules specified for verify in 4.2.28.5.2 and 4.2.28.5.3;
- 5) compares the length of the logical block, if indicated, as specified by the verify command; and
- 6) performs a byte-by-byte compare of the data on the medium and the data transferred from the application client, if indicated, as specified by the verify command.

4.2.13 Object buffering

A device may contain a temporary storage area capable of holding one or more logical objects (i.e., an object buffer). A device object buffer may include any combination of logical objects in the process of being written to the medium, or it may contain read-ahead logical objects transferred from the medium. Read-ahead logical objects are logical objects that are read from the medium by a device and placed in the object buffer in a vendor-specific manner.

A device with an object buffer may be capable of operating in either a buffered mode or an unbuffered mode. A device with no object buffer operates only in unbuffered mode. Either term is only applicable to the manner in which the device manages information to be written to the medium. Buffered mode is not applicable during read commands, regardless of whether read data passes through an object buffer.

A device operating in buffered mode may return GOOD status for write operations after all logical objects have been successfully transferred from the application client into the device object buffer. For devices operating in unbuffered mode, GOOD status is not returned until all requested logical objects are successfully recorded on the medium.

If a buffered WRITE FILEMARKS command is received with the IMMED bit set to one, GOOD status shall be returned as soon as the command is validated. For a WRITE FILEMARKS command with the IMMED bit set to zero, the device server shall perform a synchronize operation (see 4.2.14).

If an unrecoverable write error occurs while in buffered mode, the device generates an error condition to the current active command. If no command is active, the error may be reported on the next applicable operation as a deferred error (see SPC-4). For some implementations, auto contingent allegiance may be required. Refer to SAM-5 for a description of auto contingent allegiance.

The READ POSITION command may be used to determine the number and storage space of buffered logical objects not written before the unrecoverable error was encountered.

A device that encounters an unrecoverable error during a read-ahead operation shall not report the error unless the logical object in error is accessed by an application client.

Prior to performing some commands, the device server shall perform a synchronize operation (see 4.2.14) as stated in table 30 and table 38.

4.2.14 Synchronize operation behavior

Some commands (see table 30 and table 38) may require the device server to perform a synchronize operation (see 3.1.80). If a command requires a synchronize operation, the synchronize operation shall be performed prior to initiating any command-specific operations. Upon successful completion of the synchronize operation, no logical objects shall remain in the object buffer that have not been written to the medium. A synchronize operation shall have no effect on an object buffer that contains only read-ahead logical objects, or logical objects that have already been successfully written to the medium.

For a command specifying a download microcode operation (e.g., modes 04h, 05h, 06h), the device server shall perform a synchronize operation before performing the download operation.

For a MODE SELECT command specifying the Medium Partition mode page, the device server shall perform a synchronize operation before the logical unit partitions the volume.

For a SEND DIAGNOSTIC command, the device server shall perform a synchronization operation before any diagnostics that may affect the buffered logical objects, medium, or logical position, are initiated.

4.2.15 Direction and position definitions

For sequential-access devices, positioning has the connotation of logically being in, at, before, or after some defined place within a recording volume. Positioning requires that the position is capable of being repeated under the same circumstances. The orientation of usage for the four words (in, at, before, or after) is in one direction, from BOP x toward EOP x. All positioning defined below is worded from this perspective. Devices without object buffers have some physical position that relates to these logical positions. However, these definitions do not require the medium to have a physical position equivalent to the logical position unless explicitly stated.

The forward direction is defined as logically progressing from BOP x toward EOP x. The reverse direction is defined as logically progressing from EOP x toward BOP x. In serpentine devices, the logical forward or reverse direction has an alternating relationship to the physical motion of the medium.

The concept of being in some position means not being outside a defined region. The definition allows the position to be on the boundary of a defined region. If a recording volume is mounted, the logical position always begins at the beginning of the default data partition (BOP 0). If a recording volume is mounted and the medium motion is stopped, then the position is in some partition. While moving between partitions, there is no stable position.

The concept of being at some position specifies being positioned to a logical or physical extremity of a partition. A sequential-access device may be positioned at BOM, at BOP x, at EOD, at EOP x, or EOM, since these are stable positions at extremities of a partition.

The concept of being before some position specifies that there is some logical object or other defined point that may be encountered if moving toward EOP x, if the proper commands are issued. Being positioned before a particular logical block means that if the device receives a valid READ command, the logical block is transferred to the application client. This position may also be before EW x and EOP x, since these are defined points within any partition. However, if data has not been written to the end-of-partition, these points may not be accessible by the SCSI initiator device.

The concept of being after some position specifies that there is some logical object or other defined point on the BOP x side of the current position that may be encountered if the proper commands are issued. If a READ command for a single logical block has been successfully processed, then the logical position is after the transferred logical block.

4.2.16 Write modes

4.2.16.1 Write mode introduction

The write mode of the device entity specifies the allowable behaviors for altering logical objects on a mounted recording volume. If the write mode rules allow altering of logical objects, then the operation shall be processed following the write protection rules defined in 4.2.20.

4.2.16.2 Overwrite-allowed mode

Overwrite-allowed mode is used to allow alteration of any logical object on the medium. Overwrite-allowed mode is enabled or disabled using the WRITE MODE field of the Device Configuration Extension mode page (see 8.5.8). This mode is set in the device entity to enable device server behaviors. The overwrite-allowed mode does not modify the recording volume and no indication of whether overwrite-allowed mode is enabled or disabled is retained in the volume.

If overwrite-allowed mode is enabled in the drive and a command that causes a write is received, then that command shall be processed normally. If the mounted volume is a WORM volume, then a write type command shall be processed following the WORM rules (see 4.2.28) for that format.

NOTE 5 - Previous versions of this standard only defined one method of writing and did not define write modes.
The overwrite-allowed mode is the method of writing as defined in previous versions of this standard.

4.2.16.3 Append-only mode

Append-only mode is used to protect data from being accidentally overwritten.

Errors in the configuration of the environment may create a situation in which conflicting medium access commands are received (e.g., a REWIND command is received during a WRITE command from the same

application client or a different application client). If append-only mode is disabled, then it is possible for a different application client to reposition the recording volume causing the potential of an accidental overwrite of the medium. With append-only mode enabled, the medium shall not be overwritten.

Append-only mode is enabled or disabled using the WRITE MODE field of the Device Configuration Extension mode page (see 8.5.8). Append-only mode is only allowed to be enabled if a volume is not mounted or if the volume is located at BOP 0. Append-only mode is only allowed to be disabled if there is no volume mounted in the drive. Append-only mode is set in the device server to enable device server behaviors. The append-only mode does not modify the recording volume and no indication of whether append-only mode is enabled or disabled is retained in the volume.

If append-only mode is enabled in the drive a command that results in a write operation to a location that is not an append point shall be terminated with CHECK CONDITION status with the sense key set to DATA PROTECT and the additional sense code set to OPERATOR SELECTED WRITE PROTECT, and TapeAlert flag 09h shall be set. An append point shall be:

- a) the logical position zero if there are no logical objects beyond BOP;
- b) the current logical position if:
 - A) the current logical position is at BOP; and
 - B) there are only filemarks between the current logical position and EOD;
- c) the current logical position if:
 - A) the current logical position is between BOP and EOD;
 - B) there are only filemarks from the current logical position to EOD; and
 - C) there is at least one filemark immediately before the current logical position; or
- d) the current logical position if the current logical position is at EOD.

If append-only mode is supported, then the device server shall maintain an allow_overwrite variable. The allow_overwrite variable defines what operation is currently allowed if in append-only mode. The allow_overwrite variable values are defined in table 3.

Table 3 — ALLOW_OVERWRITE variable value definitions

Name	Description
Disabled	A write operation at a position that is not an append point is not allowed.
Current Position	A write operation is allowed at the position specified by the allow_overwrite_position variable
Format	An operation that modifies the format of the medium is allowed.

The allow_overwrite_position variable specifies the position (i.e., partition and logical object identifier) at which a write to a position that is not an append point is allowed.

Append-only mode may be used if accessing any recording volume (e.g., normal recording volume or WORM volume). An application client may overwrite data by using the ALLOW OVERWRITE command (see 7.1). The ALLOW OVERWRITE command specifies the logical position where the overwrite is to occur. After successfully processing an ALLOW OVERWRITE command, a write type command at the specified position is processed normally. If the position of the medium is changed or the recording volume is unmounted, then the device server shall set the allow_overwrite variable to Disabled (i.e., 0h) and the allow_overwrite_position variable to invalid. The ALLOW OVERWRITE command requires the partition number and the logical position to be passed in the CDB. If the position information passed in the ALLOW OVERWRITE command does not specify the current position of the medium, then the command is terminated with CHECK CONDITION status with the sense key set to ILLEGAL

REQUEST and the additional sense code set to SEQUENTIAL POSITIONING ERROR. If there is no volume loaded and the device server processes an ALLOW OVERWRITE command, then the command is terminated with CHECK CONDITION status with sense key set to NOT READY.

An ALLOW OVERWRITE command that returns GOOD status shall:

- a) set the allow_overwrite variable to the value in the ALLOW_OVERWRITE field of the ALLOW OVERWRITE command; and
- b) set the allow_overwrite_position variable to the current position.

An ALLOW OVERWRITE command that returns a CHECK CONDITION shall:

- a) set the allow_overwrite variable to Disabled (i.e., 0h); and
- b) set the allow_overwrite_position to invalid.

If append-only mode is enabled and a WORM volume is mounted, then a write type command shall be processed following:

- 1) the append-only rules; and
- 2) the rules for a WORM volume (see 4.2.2.4).

Figure 17 shows a representative flowchart of append-only mode behavior.

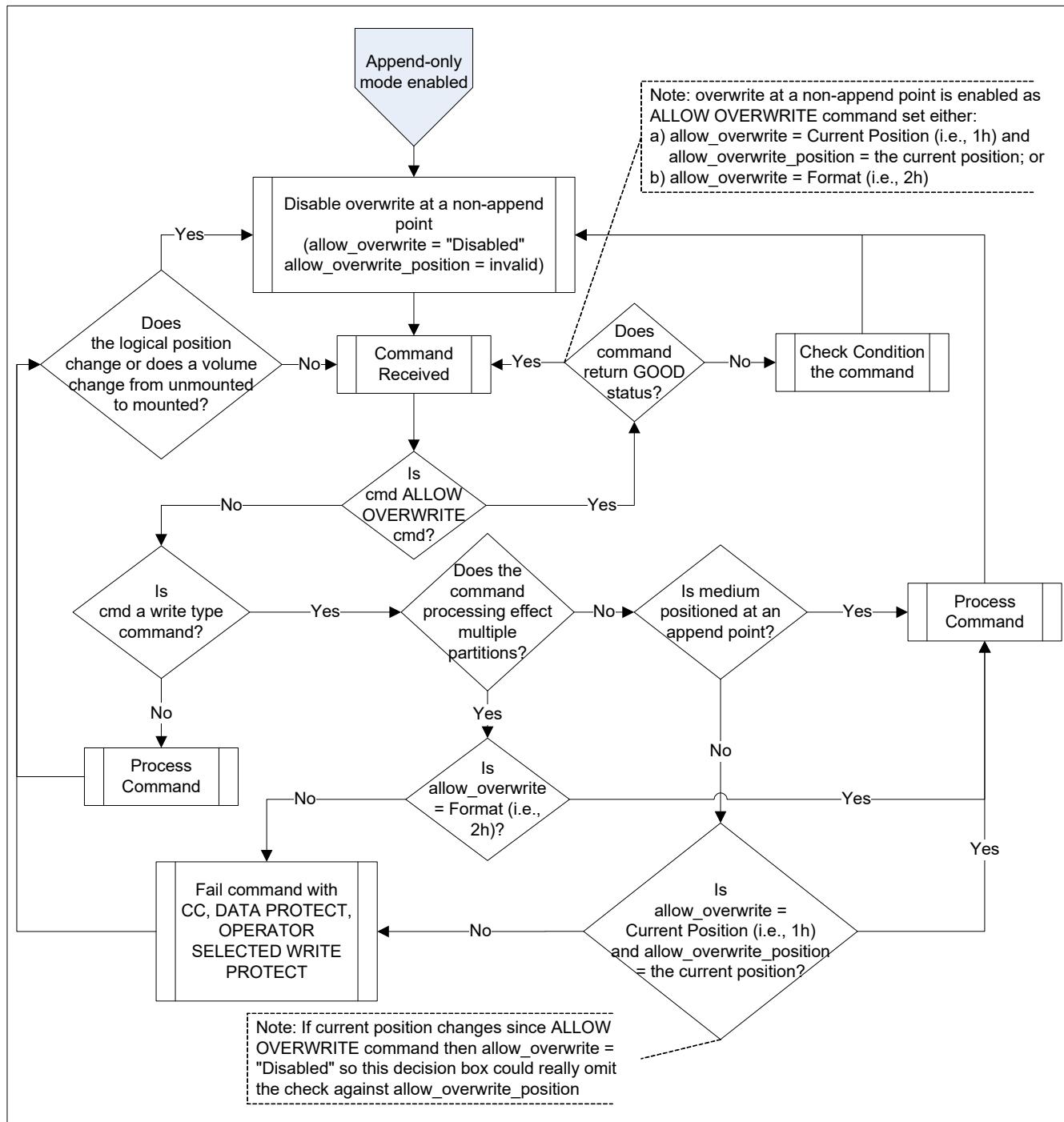


Figure 17 — Append-only mode flowchart

If the ALLOW OVERWRITE command is received by the device server and append-only mode is not enabled, then the command shall be rejected with CHECK CONDITION with the sense key set to ILLEGAL REQUEST and the additional sense code set to ILLEGAL COMMAND WHEN NOT IN APPEND-ONLY MODE.

The allow_overwrite variable shall be set to Disabled and the allow_overwrite_position variable shall be set to invalid if in append-only mode and:

- a) the WRITE MODE field of the Device Configuration Extension mode page (see 8.5.8) changes to a value of 01h (i.e., the write type operation shall only allow appends as specified for the append-only mode in 4.2.13.3);
- b) a change in logical position occurs;
- c) a volume changes state from unmounted to mounted;
- d) the CDB of a write type command is validated and the write processing begins; or
- e) an ALLOW OVERWRITE command returns a CHECK CONDITION.

4.2.17 Compression of objects

4.2.17.1 Compression of objects overview

Logical objects may be compressed before they are written to the medium. A device server may support a request from an application client that logical objects be compressed or not be compressed when stored to the medium. A device server that supports application client requests using this standard does so by supporting changeable fields in the Data Compression mode page (see 8.5.2) or by supporting changeable fields in the Device configuration mode page (see 8.5.3). Mechanisms outside the scope of this standard (e.g., the FCOMP bit in an ADC Device Server Configuration mode page RMC logical unit descriptor, see ADC-3) may also request that logical objects be compressed or not compressed when stored to the medium.

4.2.17.2 Interaction of the compression mode parameters

The DCE bit of the Data Compression mode page and the SELECT DATA COMPRESSION ALGORITHM field of the Device configuration mode page are the mode parameters that allow an application client to request that logical objects be compressed before being written to the medium. The DCE bit and the SELECT DATA COMPRESSION ALGORITHM field are different representations of compression enablement. The COMPRESSION ALGORITHM field of the Data Compression mode page and the SELECT DATA COMPRESSION ALGORITHM field of the Device configuration mode page are the mode parameters that allow an application client to request which compression algorithm to use if compression is enabled. The interaction between the DCE bit, the COMPRESSION ALGORITHM field, and the SELECT DATA COMPRESSION ALGORITHM field is specified in table 4. A Unit Attention condition for MODE PARAMETERS CHANGED shall not be established for the I_T nexus on which a MODE SELECT command is received, if mode parameters change as a result of:

- a) the DCE bit or COMPRESSION ALGORITHM field being changed due to the SELECT DATA COMPRESSION ALGORITHM field being changed in response to the MODE SELECT command; or
- b) the SELECT DATA COMPRESSION ALGORITHM field being changed due to the DCE bit or the COMPRESSION ALGORITHM field being changed in response to the MODE SELECT command.

Table 4 — Compression algorithm and compression enablement selection (part 1 of 2)

Setting in the MODE SELECT command			Requested compression state	Resulting value		
Value of the DCE bit	Value of the COMPRESSION ALGORITHM field	Value of the SELECT DATA COMPRESSION ALGORITHM field		Value of the DCE bit	Value of the COMPRESSION ALGORITHM field	Value of the SELECT DATA COMPRESSION ALGORITHM field ^b
0b	X	Not present	Disabled	0b	Value received in MODE SELECT	00h
1b	non-zero	Not present	Enabled	1b		01h
1b	0000 0000h	Not present	Disabled	1b	0000 0000h	00h
Not present		00h	Disabled	0b	Not changed	00h
Not present		01h	Enabled	1b	Device default value	01h
0b	X	00h	Disabled	0b	Value received in MODE SELECT	00h
0b	X	01h	Enabled	1b		01h
1b	non-zero	00h	Enabled	1b	Value received in MODE SELECT	01h
1b	0000 0000h	01h	Disabled ^{a)}	1b		00h
1b	non-zero	01h	Enabled	1b	Value received in MODE SELECT	01h
Xb	X	02h-7Fh	Reserved	Not changed - results in CHECK CONDITION status		

a) A value of 0000 0000h in the COMPRESSION ALGORITHM field identifies uncompressed data.
 b) The default compression algorithm of the device for this field is considered to be the compression algorithm specified in the COMPRESSION ALGORITHM field of the Data Compression mode page if received in the same MODE SELECT command.

Table 4 — Compression algorithm and compression enablement selection (part 2 of 2)

Setting in the MODE SELECT command			Requested compression state	Resulting value		
Value of the DCE bit	Value of the COMPRESSION ALGORITHM field	Value of the SELECT DATA COMPRESSION ALGORITHM field		Value of the DCE bit	Value of the COMPRESSION ALGORITHM field	Value of the SELECT DATA COMPRESSION ALGORITHM field ^b
0b	X	80h-FFh	Enabled	1b	0000 00<nn>h where <nn> is equal to the value received in the SELECT DATA COMPRESSION ALGORITHM field	Value received in MODE SELECT
Not present		80h-FFh	Enabled	1b	0000 00<nn>h where <nn> is equal to the value received in the SELECT DATA COMPRESSION ALGORITHM field	Value received in MODE SELECT
1b	0000 0000h	80h-FFh	Disabled ^a	1b	0000 0000h	00h
1b	non-zero	80h-FFh	Enabled	1b	Value received in MODE SELECT	01h

a) A value of 0000 0000h in the COMPRESSION ALGORITHM field identifies uncompressed data.
 b) The default compression algorithm of the device for this field is considered to be the compression algorithm specified in the COMPRESSION ALGORITHM field of the Data Compression mode page if received in the same MODE SELECT command.

4.2.18 Error reporting

4.2.18.1 Overview

Sequential-access devices compliant with this standard shall support both the fixed and descriptor sense data formats (see SPC-4). If fixed format sense data is specified, but the INFORMATION field value exceeds the maximum value allowed in the fixed format sense data, the VALID bit shall be set to zero. Refer to SPC-4 for a description of the sense data VALID bit and INFORMATION field contained in the REQUEST SENSE sense data. In addition, this standard describes the use of the INFORMATION field specific to the sequential-access device type.

4.2.18.2 Stream commands sense data descriptor

Table 5 defines the stream commands sense data descriptor for sequential-access devices.

Table 5 — Stream commands sense data descriptor

Bit Byte	7	6	5	4	3	2	1	0
0								DESCRIPTOR TYPE (04h)
1								ADDITIONAL LENGTH (02h)
2								Reserved
3	FILEMARK	EOM		ILI				Reserved

See SPC-4 for a description of the DESCRIPTOR TYPE and ADDITIONAL LENGTH fields.

For a description of FILEMARK bit usage see the:

- a) READ(16) (see 5.3) command;
- b) READ(6) (see 6.4) command;
- c) SPACE(6) (see 6.6) command;
- d) RECOVER BUFFERED data (see 7.11) command; and
- e) SPACE(16) (see 7.15) command.

For end-of-medium (EOM) bit usage see the:

- a) READ(16) (see 5.3) command;
- b) READ REVERSE(16) (see 5.4) command;
- c) WRITE(16) (see 5.6) command;
- d) WRITE FILEMARKS(16) (see 5.7) command;
- e) LOCATE(10) (see 6.3) command;
- f) READ(6) (see 6.4) command;
- g) READ REVERSE(6) (see 6.5) command;
- h) SPACE(6) (see 6.6) command;
- i) WRITE(6) (see 6.8) command;
- j) WRITE FILEMARKS(6) (see 6.9) command;
- k) LOCATE(16) (see 7.5) command;
- l) RECOVER BUFFERED data (see 7.11) command;
- m) SPACE(16) (see 7.15) command; and
- n) Device Configuration mode page (see 8.5.3).

For incorrect length indicator (ILI) bit usage see the:

- a) READ(16) (see 5.3) command;
- b) READ(6) (see 6.4) command; and
- c) Data Compression mode page (see 8.5.2).

4.2.18.3 Information sense data descriptor

If reporting descriptor format sense data, the device server shall include an information sense data descriptor (see table 6) for a unit attention condition established:

- a) by a TapeAlert specific informational exception condition; or

- b) by a TapeAlert flag meeting its threshold criteria if the corresponding ETC bit in the TapeAlert log parameter is set to one.

The information sense data descriptor format is specified in table 6.

Table 6 — Information sense data descriptor

Bit Byte	7	6	5	4	3	2	1	0
0	DESCRIPTOR TYPE (00h)							
1	ADDITIONAL LENGTH (0Ah)							
2	VALID (1b)	Reserved						
3	Reserved							
4	FLAG01h	FLAG02h	FLAG03h	FLAG04h	FLAG05h	FLAG06h	FLAG07h	FLAG08h
5	FLAG09h	FLAG0Ah	FLAG0Bh	FLAG0Ch	FLAG0Dh	FLAG0Eh	FLAG0Fh	FLAG10h
6	FLAG11h	FLAG12h	FLAG13h	FLAG14h	FLAG15h	FLAG16h	FLAG17h	FLAG18h
7	FLAG19h	FLAG1Ah	FLAG1Bh	FLAG1Ch	FLAG1Dh	FLAG1Eh	FLAG1Fh	FLAG20h
8	FLAG21h	FLAG22h	FLAG23h	FLAG24h	FLAG25h	FLAG26h	FLAG27h	FLAG28h
9	FLAG29h	FLAG2Ah	FLAG2Bh	FLAG2Ch	FLAG2Dh	FLAG2Eh	FLAG2Fh	FLAG30h
10	FLAG31h	FLAG32h	FLAG33h	FLAG34h	FLAG35h	FLAG36h	FLAG37h	FLAG38h
11	FLAG39h	FLAG3Ah	FLAG3Bh	FLAG3Ch	FLAG3Dh	FLAG3Eh	FLAG3Fh	FLAG40h

The VALID bit shall be set to one.

An active TapeAlert flag has the corresponding FLAGXX bit set to one. An inactive TapeAlert flag has the corresponding FLAGXX bit set to zero.

4.2.18.4 Deferred check condition eligible commands

If a device server has detected a deferred error (see SPC-4), then it queues the error condition for return to the correct SCSI initiator device per the buffered mode (see 4.2.18.6). A command to which a deferred error may be returned as sense data is called a deferred check condition (DCC) eligible command. The DCC eligible commands are specified in table 30 and in table 38.

4.2.18.5 Deferred error affinity commands

A deferred error affinity (DEA) command is a command that causes the device server to change which I_T nexus is used for reporting deferred errors (see 4.2.18.6). DEA commands are commands that are capable of causing a change to the logical position of the medium (see 4.2.2.2) or that are capable of changing the contents of the medium.

4.2.18.6 Error conditions

If any of the conditions specified in table 7 occur during the processing of a command or if a deferred error prevented the command from processing, the device server shall terminate the command with CHECK CONDITION status with the sense key set to the specified value and the additional sense code set to the appropriate value for the condition.

Table 7 specifies some error conditions and the applicable sense keys. Table 7 does not provide a complete list of all conditions that may cause the CHECK CONDITION status.

Table 7 — Error conditions and sense keys

Condition	Sense key
Unsupported option requested.	ILLEGAL REQUEST
Logical unit reset, I_T nexus loss, or volume change since last command from this I_T nexus.	UNIT ATTENTION
Self diagnostic failed.	HARDWARE ERROR
Unrecovered read error.	MEDIUM ERROR HARDWARE ERROR
Recovered read or write error.	RECOVERED ERROR
Attempt a WRITE, READ, READ REVERSE, VERIFY, or RECOVER BUFFERED data command with the FIXED ^a bit set to zero and variable-block transfers are not supported.	ILLEGAL REQUEST
Attempt a WRITE, READ, READ REVERSE, VERIFY, or RECOVER BUFFERED data command with the FIXED ^a bit set to zero and requested block length is not supported.	ILLEGAL REQUEST
Attempt a WRITE, READ, READ REVERSE, VERIFY, or RECOVER BUFFERED data command with the FIXED ^a bit set to one and MODE SENSE block length set to zero.	ILLEGAL REQUEST
Attempt to perform an erase, format, partition, set capacity, or write type operation on write protected medium.	DATA PROTECT
Deferred write error.	MEDIUM ERROR VOLUME OVERFLOW HARDWARE ERROR
Medium failed to thread or unthread during the process of mounting or demounting.	MEDIUM ERROR HARDWARE ERROR
a) See the READ(16) (see 5.3), READ REVERSE(16) (see 5.4), VERIFY(16) (see 5.5), WRITE(16) (see 5.6), READ(6) (see 6.4), READ REVERSE(6) (see 6.5), VERIFY(6) (see 6.7), WRITE(6) (see 6.8), and RECOVER BUFFERED data (see 7.11) commands for a description of the FIXED bit.	

The Read-Write Error Recovery mode page (see 8.5.5) current values specify behavior if an unrecoverable read or write error is encountered. If the Read-Write Error Recovery mode page is not implemented, the behavior is vendor specific.

In the case of a deferred error, the sense data VALID bit shall be set to zero.

In the case of a deferred error:

- a) if buffered mode 0h or buffered mode 1h is selected, then:
 - A) if the device server associates deferred errors to the I_T nexus over which the data associated with the deferred error was received, then the deferred error indication shall be returned in response to the first of:
 - a) a DCC eligible command (see 4.2.18.4) received over that I_T nexus; or
 - b) a command, received on any I_T nexus, that requires the logical unit to discard any unwritten buffered logical objects (i.e., buffered logical objects that the device is unable to continue

- successfully writing) before performing the requested operation (i.e., a LOAD UNLOAD command (see 7.4) or a REWIND command (see 7.13)); or
- B) if the device server does not associate deferred errors to the I_T nexus over which the data associated with the deferred error was received, then the deferred error indication shall be returned in response to the first of:
 - a) a DCC eligible command received on the I_T nexus on which the last DEA command (see 4.2.18.5) was received; or
 - b) a DEA command received on an I_T nexus that is a different I_T nexus than the one on which the last DEA command was received if that command is also a DCC eligible command. All deferred errors not returned to this command shall be set for reporting on this I_T nexus; or
 - b) if buffered mode 2h is selected, then the error shall be reported to the first of:
 - A) a DCC eligible command received on an I_T nexus with unwritten data in the object buffer; or
 - B) a command, received on any I_T nexus, that requires the logical unit to discard any unwritten buffered logical objects (i.e., buffered logical objects that the device is unable to continue successfully writing) before performing the requested operation (i.e., a LOAD UNLOAD command (see 7.4) or a REWIND command (see 7.13)).

In the case of a write attempt to write protected medium, the additional sense code specifies the cause of the DATA PROTECT sense key (see 4.2.20).

In the case of a medium thread or unthread failure, the additional sense code shall be set to MEDIUM THREAD OR UNTHREAD FAILURE. The sense key shall be set to MEDIUM ERROR or HARDWARE ERROR (see SPC-4).

4.2.19 Self-test operations

The logical position following the completion of a self-test (see SPC-4) is not specified by this standard.

4.2.20 Write protection

4.2.20.1 Write protection introduction

Write protection of the medium prevents the alteration of logical objects on the medium, and any change to the accessibility of logical objects on the medium, by commands issued to the device server. Write protection is usually controlled by the user of the volume through manual intervention (e.g., mechanical lock) or may result from hardware controls (e.g., tabs on the volume housing), conditions such as positioning within unrecoverable data, or software write protections. All sources of write protection are independent. If present, any write protection shall cause otherwise valid commands that request alteration of logical objects on the medium, or affect the accessibility of logical objects on the medium, to be rejected with a CHECK CONDITION status with the sense key set to DATA PROTECT (see 4.2.20.2). The device server shall process commands that request alteration of logical objects on the medium, or commands that may affect the accessibility of logical objects on the medium, only if all write protections are disabled.

Write protection does not prevent the alteration of MAM attributes (e.g., processing of a WRITE ATTRIBUTE command).

Hardware write protection results if a physical attribute of the drive or volume is changed to specify that writing shall be prohibited. Changing the state of the hardware write protection requires physical intervention, either with the drive or the volume. If allowed by the drive, changing the hardware write protection while the volume is mounted results in vendor-specific behavior that may include the writing of previously buffered logical objects.

Conditions such as positioning within unrecoverable data may result in a temporary write protection condition. To preserve future data integrity, the device server may reject any command that requires writing data to the medium if the recovery of the data is uncertain. A temporary write protection condition may be released by the device server at any time. Buffered logical objects may not be written to the medium (e.g., the application client unloads the

recording volume before the temporary write protection condition is removed). The exact behavior of the device server during a temporary write protection condition is vendor specific.

Software write protection results if either the device server, volume, or medium is marked as write protected by a command from the application client. Four optional means of setting a software write protection state are available to an application client through the Device Configuration and Control mode pages:

- a) software write protection for the device server across mounts;
- b) associated write protection for the currently mounted volume;
- c) persistent write protection of a volume across mounts; and
- d) permanent write protection of a volume across mounts.

The application client may control these write protections using the Control mode page (see SPC-4) and the Device Configuration mode page (see 8.5.3). All of the software write protection methods are optional. Changing the state of any software write protection shall not prevent previously buffered logical objects from transferring to the medium.

4.2.20.2 Write protection additional sense code use

The additional sense code associated with the DATA PROTECT sense key depends on the write protection in effect at the time. Table 8 specifies the preferred additional sense code for the given write protection. Alternatively, the generic additional sense code of WRITE PROTECTED may be returned by the device server.

Table 8 — Write protect additional sense code combinations

Cause of DATA PROTECT error	Additional sense code
Hardware Write Protection	HARDWARE WRITE PROTECTED
Permanent Write Protection	PERMANENT WRITE PROTECT
Persistent Write Protection	PERSISTENT WRITE PROTECT
Associated Write Protection	ASSOCIATED WRITE PROTECT
Software Write Protection	LOGICAL UNIT SOFTWARE WRITE PROTECTED
WORM volume mounted in a device that does not support WORM (see 4.2.28)	CANNOT WRITE MEDIUM - INCOMPATIBLE FORMAT

If more than one condition exists, the device server shall:

- a) report an additional sense code indicating a write protection not related to either:
 - A) a hardware write protect; or
 - B) one of the software write protections specified in 4.2.20.1; or
- b) report the applicable condition in order of:
 - 1) HARDWARE WRITE PROTECTED;
 - 2) PERMANENT WRITE PROTECT;
 - 3) PERSISTENT WRITE PROTECT;
 - 4) ASSOCIATED WRITE PROTECT;
 - 5) LOGICAL UNIT SOFTWARE WRITE PROTECTED; and
 - 6) CANNOT WRITE MEDIUM - INCOMPATIBLE FORMAT

Other conditions that may cause a command to be rejected with a DATA PROTECT sense key include:

- a) the current volume is maintained as read-only by the device server;
- b) the device server is only able to write from BOP or EOD and the current logical position is neither;

- c) the volume is a WORM volume (see 4.2.28) and an attempt to write in an unrecordable location is detected;
- d) the set of logical block encryption parameters in the device entity is not correct for the operation requested; and
- e) vendor-specific conditions.

4.2.20.3 Software write protection for the device server

Software write protection controls write protection for the device server. This method of write protection is optionally controlled from the Control mode page (see SPC-4) or the SWP bit in the Device Configuration mode page (see 8.5.3). Either or both methods may be implemented by the device server. If both methods are implemented, each control bit is independently set. Software write protection exists if either bit is non-zero. The state of software write protection for the device server shall not be recorded on the medium. The value of the SWP bit may be altered by the application client (if the SWP bit is changeable). The state of each control bit shall be set to its default state after a logical unit reset.

4.2.20.4 Associated write protection

Associated write protection controls write protection for the currently mounted volume as long as the current volume is mounted. The associated write protection state is controlled by the ASOCWP bit in the Device Configuration mode page (see 8.5.3). Associated write protection exists if the ASOCWP bit is non-zero. Associated write protection may be altered by the application client (if the ASOCWP bit is changeable) if a volume is mounted. If a volume is demounted or after a logical unit reset occurs, associated write protection shall be removed.

4.2.20.5 Persistent write protection

Persistent write protection controls write protection for the currently mounted volume. The persistent write protection state is controlled by the PERSWP bit in the Device Configuration mode page (see 8.5.3). If enabled, persistent write protection shall exist for the mounted volume until disabled by the application client. The state of persistent write protection shall be recorded with the volume and the persistent write protection shall only affect the application client accessible medium. The device server shall report the PERSWP bit as one if a mounted volume is marked with persistent write protection. If a volume is demounted or after a logical unit reset occurs, the device server shall report the PERSWP bit as zero prior to the mounting of a volume. The means for recording the state of persistent write protection for the volume may be specified in the applicable recording format standard or be vendor specific.

4.2.20.6 Permanent write protection

Permanent write protection controls write protection for the currently mounted volume. The permanent write protection state is controlled by the PRMWP bit in the Device Configuration mode page (see 8.5.3). If enabled, permanent write protection shall exist for the mounted volume until disabled by a vendor-specific method. The state of permanent write protection shall be recorded with the volume and the permanent write protection shall only affect the application client accessible medium. The device server shall report the PRMWP bit as one if a mounted volume is marked with permanent write protection. If a volume is demounted or after a logical unit reset occurs, the device server shall report the PRMWP bit as zero prior to the mounting of a volume. The means for recording the state of permanent write protection for the volume may be specified in the applicable recording format standard or be vendor specific. Permanent write protection shall not be removed by a MODE SELECT command using the PRMWP bit. Methods to remove this protection may exist and are vendor specific.

4.2.21 Progress indication

If the device server performs one of the commands specified in table 9 and the immediate bit is set to one, then the device server remains in the ready state, and progress indication information may be available.

Table 9 — Commands providing progress indication without changing ready state

Operation	Options	Subclause	Additional Sense Code
ERASE	LONG = 1	5.2, 6.2	ERASE OPERATION IN PROGRESS
LOCATE		5.2,6.3	LOCATE OPERATION IN PROGRESS
REWIND		7.13	REWIND OPERATION IN PROGRESS
SET CAPACITY		7.14	SET CAPACITY OPERATION IN PROGRESS
VERIFY		5.5,6.7	VERIFY OPERATION IN PROGRESS

While the device server is performing the operation, an application client may test the progress of the operation by interpreting the progress indication information in the SENSE KEY SPECIFIC field of the sense data. During the operation, the device server may report a sense key value of NO SENSE and additional sense code as specified in table 9. The device server should use the sense key specific function for progress indication to provide information on the completion of the operation.

If the device server performs one of the commands specified in table 10 and the immediate bit is set to one, then the device server transitions between the ready state and the not ready state, and progress indication information may be available.

Table 10 — Commands changing ready state and providing progress indication

Operation	Options	Subclause	Additional Sense Code
FORMAT MEDIUM		7.2	LOGICAL UNIT NOT READY, FORMAT IN PROGRESS
LOAD UNLOAD	LOAD = 1, EOT = 0	7.4	LOGICAL UNIT IS IN PROCESS OF BECOMING READY
LOAD UNLOAD	LOAD = 0, EOT = 1	7.4	LOGICAL UNIT NOT READY, OPERATION IN PROGRESS

While the device server is performing the operation, an application client may test the progress of the operation by interpreting the progress indication information in the SENSE KEY SPECIFIC field of the sense data. During the operation, the device server may report a sense key value of NOT READY and an additional sense code as specified in table 10. The sense key specific function for progress indication may be used by the device server to provide information on the completion of the operation.

NOTE 6 - A REQUEST SENSE command following a TEST UNIT READY command that results in CHECK CONDITION status may provide information that, if acted upon, may lead to unexpected conditions. For example, progress indication reporting is useful if a media changer is used to service a sequential-access device following a demount with IMMED=1b. A TEST UNIT READY command may receive CHECK CONDITION status and a NOT READY sense key reported in the subsequent sense data. This may imply that the demount is finished. If the application client ignores the progress indication information in the sense data, an EXCHANGE MEDIUM or MOVE MEDIUM command (see SMC-3) to move the demounted volume from the SCSI device may fail to grab the volume if the demount is still in progress.

4.2.22 Command queuing

4.2.22.1 Command queuing overview

Issuing multiple write commands facilitates streaming operations up to the limit of the number of outstanding commands supported by the application client and the device server.

An application client should wait for status to be returned for any MODE SELECT command before issuing the next command.

4.2.22.2 Explicit address mode write sequences

If operating in explicit address mode, write sequences (see 3.1.82) are used for write operations.

For explicit address mode write sequences, the following rules apply:

- a) for a write sequence consisting of more than one command, the FCS bit (see 5.2, 5.6, 5.7) shall be set to one in the first command of the write sequence. For all other commands within the write sequence, the FCS bit shall be set to zero;
- b) for a write sequence consisting of more than one command, the LCS bit (see 5.2, 5.6, 5.7) shall be set to one in the last command of the write sequence. For all other commands within the write sequence, the LCS bit shall be set to zero;
- c) for a write sequence consisting of only one command, the FCS bit and LCS bit shall be set to one;
- d) for a write sequence consisting of more than one command, the application client shall issue the commands in sequentially increasing logical object identifier order;
- e) the application client shall not issue a write sequence prior to receiving status for all outstanding read type commands; and
- f) the application client shall specify a Command Reference Number (see SAM-5) for each command in a write sequence.

4.2.23 Block address mode

4.2.23.1 Block address mode overview

If operating in implicit address mode, then space operations and commands to read from and write on the medium do not contain positioning information (i.e., a command is processed based on the medium position relative to the last command that was processed). As such, the application client does not contain explicit knowledge of the medium position. If an error occurs, to maintain data integrity, the application client is required to determine the medium position before re-issuing a command that affects the medium position.

If operating in explicit address mode, then space operations and commands to read from and write on the medium contain positioning information fields (i.e., a command is processed based on the medium position information specified in the command). As such, the application client contains explicit knowledge of the medium position. This results in enhanced error detection and recovery functionality that allows the application client to maintain data integrity while performing various operations without first determining the medium position. Some example operations include:

- a) the re-issuing of a command that affects the medium position;
- b) multi-path I/O; and
- c) command queuing.

The device server shall support explicit address mode only, implicit address mode only, or both explicit address mode and implicit address mode. At any instant, the device server shall be in or transitioning between one of the following block address mode states (see 4.2.23.3):

- a) A0:Idle;
- b) E0:Explicit Address Mode - Neutral;
- c) E1:Explicit Address Mode - Write Capable; or
- d) F0:Implicit Address Mode.

4.2.23.2 Block address mode selection

The block address mode shall be selected as follows:

- a) if the BAML bit in the Device Configuration mode page (see 8.5.3) is set to zero, then the setting of the BAM bit in the Device Configuration mode page shall be ignored and the block address mode shall be determined based on the first block address mode unique command that is received after a successful mount or successful completion of a command that positions the medium to BOP;
- b) if the BAML bit in the Device Configuration mode page is set to one and the BAM bit in the Device Configuration mode page is set to zero, the logical unit shall support implicit address mode; or
- c) if the BAML bit in the Device Configuration mode page is set to one and the BAM bit in the Device Configuration mode page is set to one, the logical unit shall support explicit address mode.

Prior to performing a block address mode change, the logical unit shall perform a synchronize operation (see 4.2.14).

If the device server receives a command that is not supported by the currently selected mode, the device server shall return CHECK CONDITION, the sense key shall be set to ILLEGAL REQUEST, and the additional sense code shall be set to:

- a) ILLEGAL COMMAND WHILE IN EXPLICIT ADDRESS MODE if the currently selected mode is the explicit address mode;
- b) ILLEGAL COMMAND WHILE IN IMPLICIT ADDRESS MODE if the currently selected mode is the implicit address mode; or
- c) ILLEGAL COMMAND WHILE IN WRITE CAPABLE STATE if the device server is in explicit address mode write capable state.

4.2.23.3 Block address mode state diagrams

For the block address mode state diagrams (see figure 19, figure 20, figure 21, and figure 22), the following terminology shall apply:

- a) explicit command: a command contained only in the explicit address command set (see table 30);
- b) implicit command: a command contained only in the implicit address command set (see table 38); and
- c) generic command: an explicit command that is not a read type or write type command (see table 30).

A common command containing a BAM bit (e.g., LOCATE(16)) shall be processed as either an explicit or implicit command based on the setting of the BAM bit contained in the common command.

The SPACE(16) command shall be processed as either an explicit or implicit command based on the setting of the PARAMETER LENGTH field in the command descriptor block.

Figure 18 provides an overview of the block address model state diagram. Refer to figure 19, figure 20, figure 21, and figure 22 for detailed descriptions of the block address model state diagram.

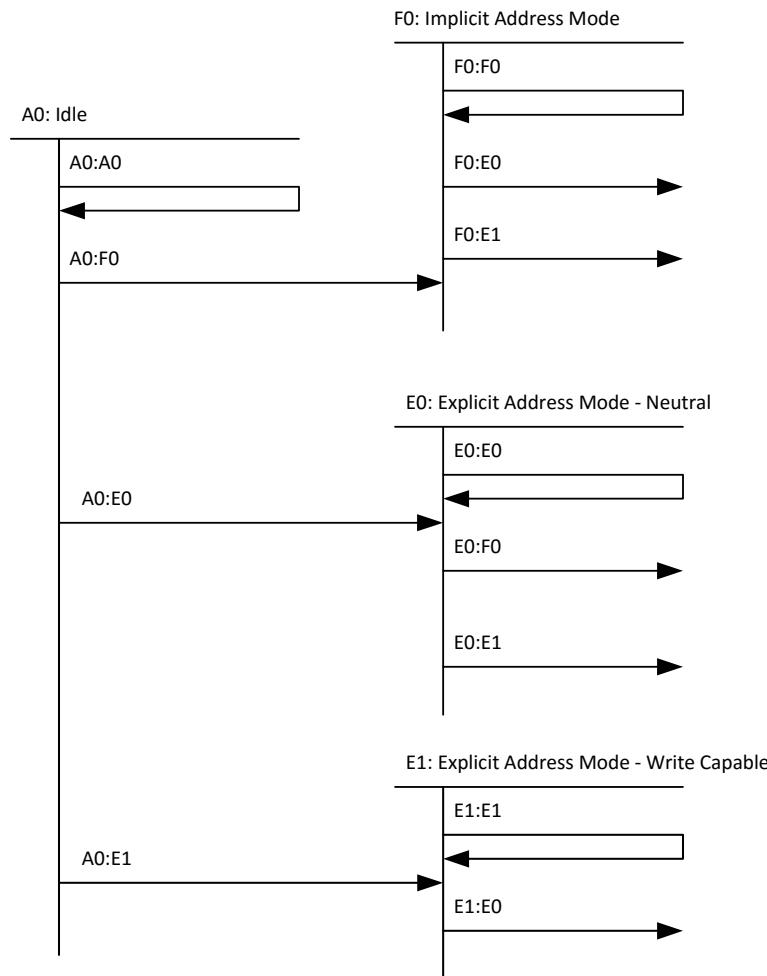


Figure 18 — Block address mode state diagram, overview

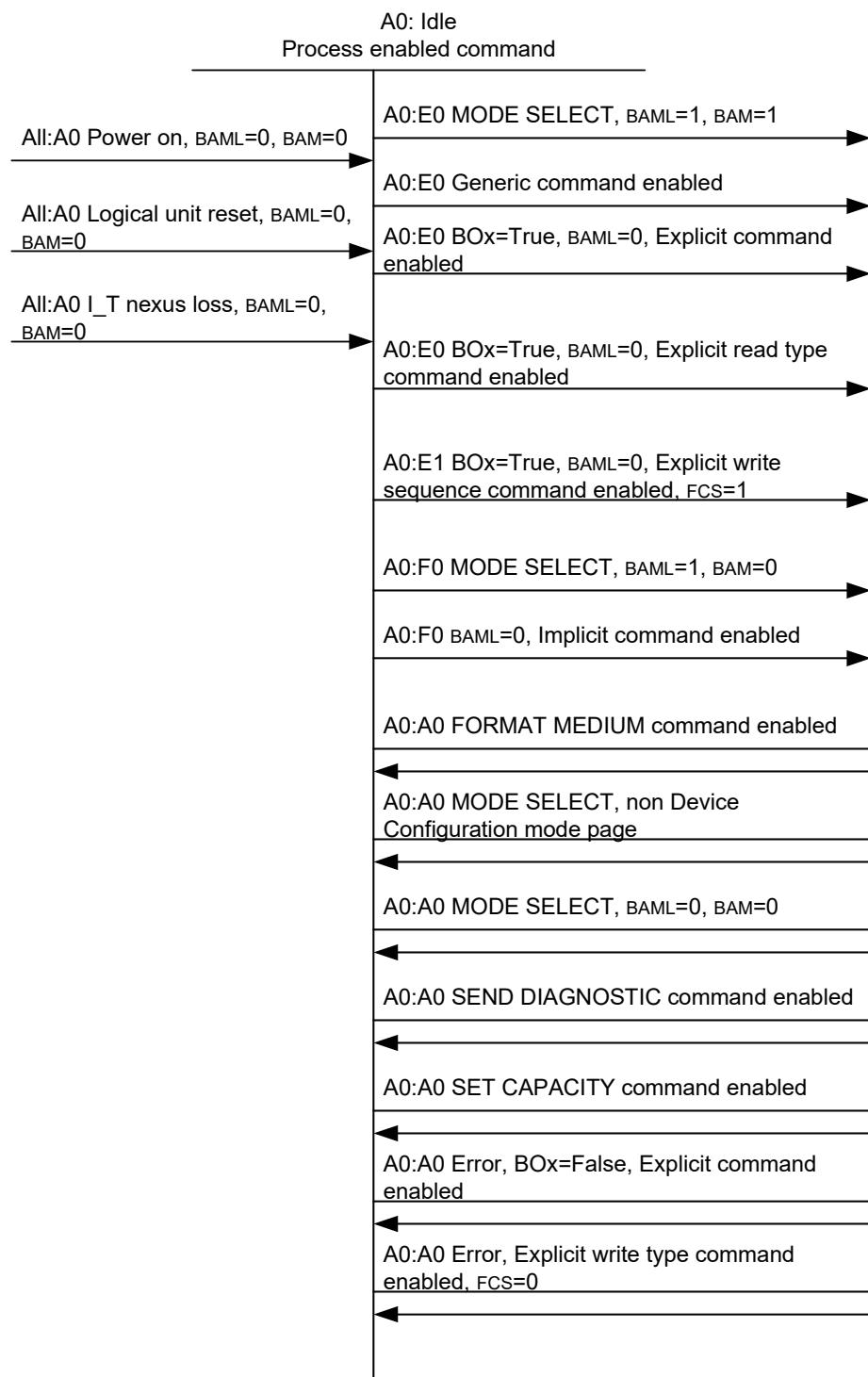


Figure 19 — Block address mode state diagram, Idle state

State A0:Idle: This is the idle state.

Transition All:A0: If the BAML bit is set to zero and the BAM bit is set to zero, then this transition shall occur as a result of:

- a) a power-on;
- b) a logical unit reset; or
- c) an I_T nexus loss event.

Transition A0:E0: This transition shall occur if:

- a) a MODE SELECT command specifying a Device Configuration mode page with the BAML bit set to one and the BAM bit set to one completes with GOOD status;
- b) a generic command is enabled;
- c) an explicit command is enabled, the medium position is at BOx, and the BAML bit is set to zero; or
- d) an explicit read type command is enabled, the medium position is at BOx, and the BAML bit is set to zero.

Transition A0:E1: This transition shall occur if an explicit write sequence command is enabled with:

- a) the FCS bit set to one;
- b) the medium position at BOx; and
- c) the BAML bit set to zero.

Transition A0:F0: This transition shall occur if:

- a) a MODE SELECT command specifying a Device Configuration mode page with the BAML bit set to one and the BAM bit set to zero completes with GOOD status; or
- b) an implicit command is enabled, the medium position is at BOx, and the BAML bit is set to zero.

Transition A0:A0: This transition shall occur if:

- a) a FORMAT MEDIUM command is enabled;
- b) a MODE SELECT command specifying a mode page other than the Device Configuration mode page is enabled;
- c) a MODE SELECT command specifying a Device Configuration mode page with the BAML bit set to zero and the BAM bit set to zero is enabled;
- d) a SEND DIAGNOSTIC command is enabled;
- e) a SET CAPACITY command is enabled;
- f) an explicit command is enabled and the medium position is not at BOx. In this case the device server shall return CHECK CONDITION status, the sense key shall be set to ILLEGAL REQUEST and the additional sense code shall be set to SEQUENTIAL POSITIONING ERROR; or
- g) an explicit write type command is enabled with the FCS bit set to zero. In this case the device server shall return CHECK CONDITION status, the sense key shall be set to ILLEGAL REQUEST and the additional sense code shall be set to INVALID FIELD IN CDB.

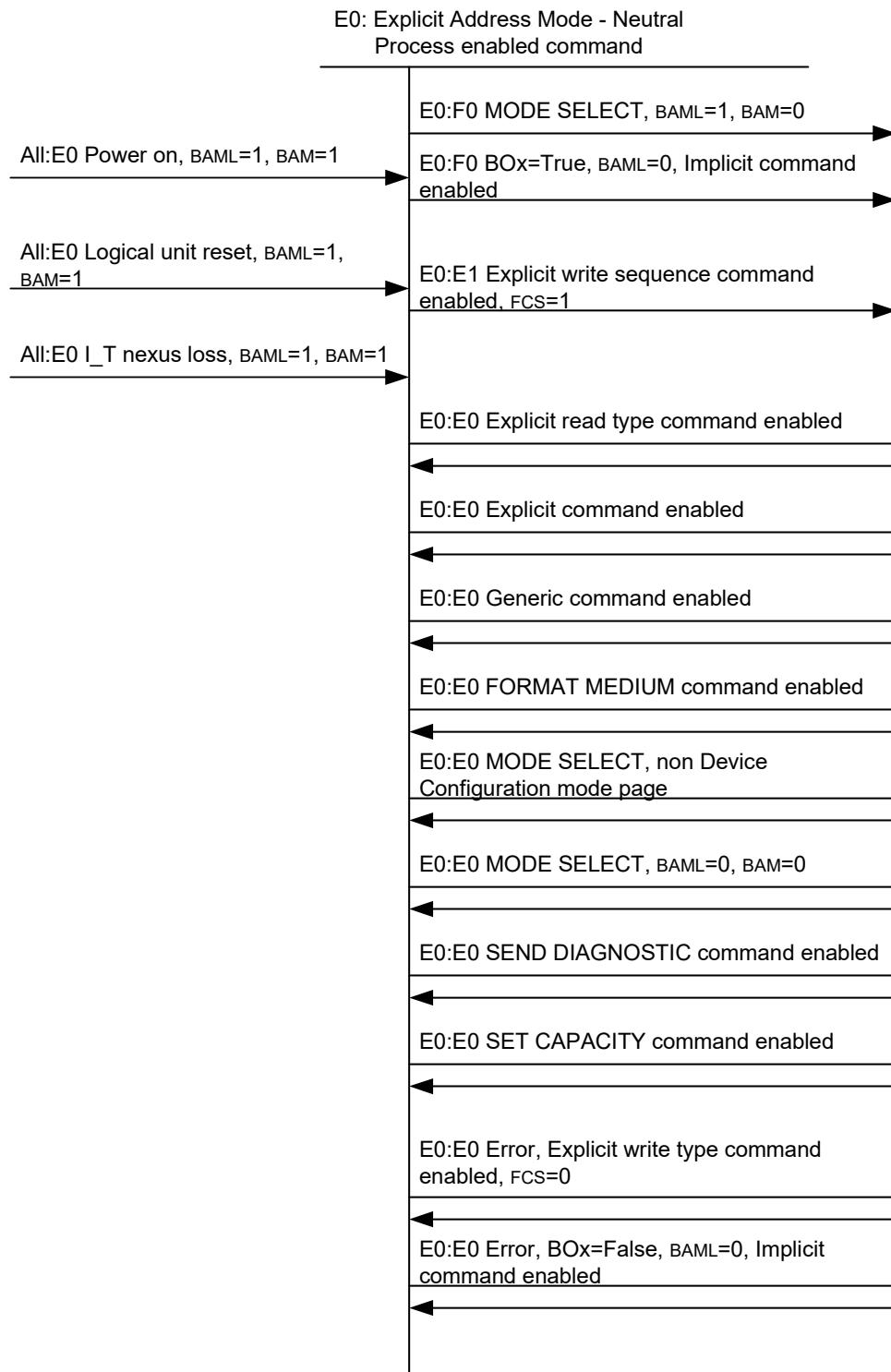


Figure 20 — Block address mode state diagram, Explicit Address Mode - Neutral

State E0:Explicit Address Mode - Neutral: This is the neutral state for explicit address mode.

Transition All:E0: If the BAML bit is set to one and the BAM bit is set to one, then this transition shall occur as a result of:

- a) a power-on;
- b) a logical unit reset; or
- c) an I_T nexus loss event.

Transition E0:F0: This transition shall occur if:

- a) a MODE SELECT command specifying a Device Configuration mode page with the BAML bit set to one and the BAM bit set to zero completes with GOOD status; or
- b) an implicit command is enabled, the medium position is at BOx, and the BAML bit is set to zero.

Transition E0:E1: This transition shall occur if an explicit write sequence command is enabled with the FCS bit set to one.

Transition E0:E0: This transition shall occur if:

- a) an explicit read type command is enabled;
- b) an explicit command is enabled;
- c) a generic command is enabled;
- d) a FORMAT MEDIUM command is enabled;
- e) a MODE SELECT command specifying a mode page other than the Device Configuration mode page is enabled;
- f) a MODE SELECT command specifying a Device Configuration mode page with the BAML bit set to zero and the BAM bit set to zero is enabled;
- g) a SEND DIAGNOSTIC command is enabled;
- h) a SET CAPACITY command is enabled;
- i) an explicit write type command is enabled with the FCS bit set to zero. In this case the device server shall return CHECK CONDITION status, the sense key shall be set to ILLEGAL REQUEST and the additional sense code shall be set to INVALID FIELD IN CDB; or
- j) an implicit command is enabled, the medium position is not at BOx, and the BAML bit is set to zero. In this case the device server shall return CHECK CONDITION status, the sense key shall be set to ILLEGAL REQUEST and the additional sense code shall be set to ILLEGAL COMMAND WHILE IN EXPLICIT ADDRESS MODE.

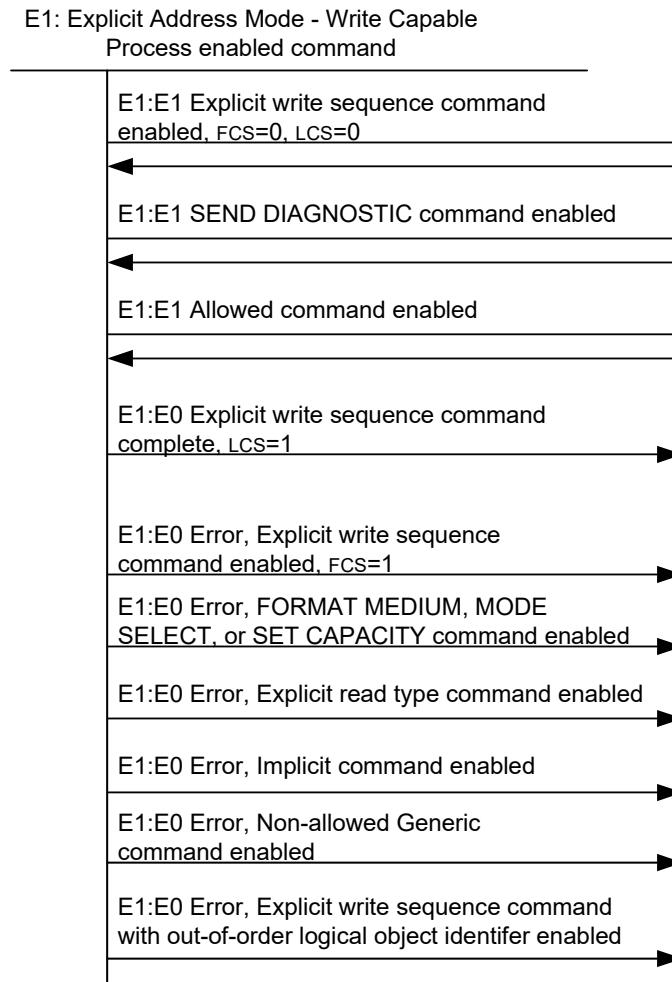


Figure 21 — Block address mode state diagram, Explicit Address Mode - Write Capable

State E1:Explicit Address Mode - Write Capable: This is the write capable state for explicit address mode.

Transition E1:E1: This transition shall occur if:

- a) an explicit write sequence command is enabled with the FCS bit set to zero and the LCS bit set to zero;
- b) a SEND DIAGNOSTIC command is enabled; or
- c) an allowed command (see table 30) is enabled.

Transition E1:E0: This transition shall occur if:

- a) an explicit write sequence command with the LCS bit set to one completed with GOOD status;
- b) an explicit write sequence command with the FCS bit set to one and the LCS bit set to one completed with GOOD status;
- c) an explicit write sequence command is enabled with the FCS bit set to one. In this case, the device server shall return CHECK CONDITION status, the sense key shall be set to ILLEGAL REQUEST, and the additional sense code shall be set to INVALID FIELD IN CDB;

- d) a FORMAT MEDIUM, MODE SELECT, or SET CAPACITY command is enabled. In this case, the device server shall return CHECK CONDITION status, the sense key shall be set to ILLEGAL REQUEST, and the additional sense code shall be set to ILLEGAL COMMAND WHILE IN WRITE CAPABLE STATE;
- e) an explicit read type command is enabled. In this case, the device server shall return CHECK CONDITION status, the sense key shall be set to ILLEGAL REQUEST, and the additional sense code shall be set to ILLEGAL COMMAND WHILE IN WRITE CAPABLE STATE;
- f) an implicit command is enabled. In this case, the device server shall return CHECK CONDITION status, the sense key shall be set to ILLEGAL REQUEST, and the additional sense code shall be set to ILLEGAL COMMAND WHILE IN WRITE CAPABLE STATE;
- g) a non-allowed generic command (see table 30) is enabled. In this case, the device server shall return CHECK CONDITION status, the sense key shall be set to ILLEGAL REQUEST, and the additional sense code shall be set to ILLEGAL COMMAND WHILE IN WRITE CAPABLE STATE; or
- h) an explicit write sequence command with an out-of-order logical object identifier is enabled. In this case, the device server shall return CHECK CONDITION status, the sense key shall be set to ILLEGAL REQUEST and the additional sense code shall be set to INVALID FIELD IN CDB.

Following an error condition that does not result in a transition out of write capable state, the following commands may be issued to transition from write capable state to neutral state:

- a) a WRITE(16) command with the LCS bit set to one and the TRANSFER LENGTH field set to zero; or
- b) a WRITE FILEMARKS(16) command with the LCS bit set to one, the IMMED bit set to zero, and the FILEMARK COUNT field set to zero.

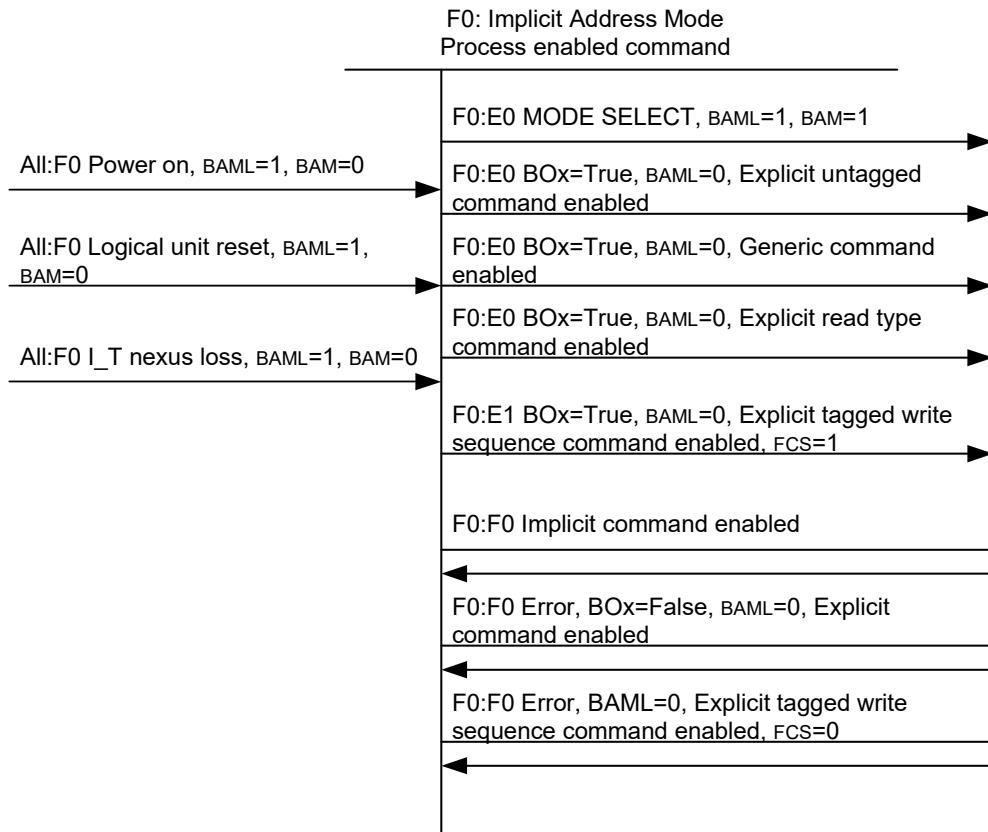


Figure 22 — Block address mode state diagram, Implicit Address Mode

State F0:Implicit Address Mode: This is the state for implicit address mode.

Transition All:F0: If the BAML bit is set to one and the BAM bit is set to zero, then this transition shall occur as a result of:

- a) a power-on;
- b) a logical unit reset; or
- c) an I_T nexus loss event.

Transition F0:E0: This transition shall occur if:

- a) a MODE SELECT command specifying a Device Configuration mode page with the BAML bit set to one and the BAM bit set to one completes with GOOD status;
- b) an explicit command is enabled, the medium position is at BOx, and the BAML bit is set to zero;
- c) a generic command is enabled, the medium position is at BOx, and the BAML bit is set to zero; or
- d) an explicit read type command is enabled, the medium position is at BOx, and the BAML bit is set to zero.

Transition F0:E1: This transition shall occur if an explicit write sequence command is enabled with the FCS bit set to one, the medium position is at BOx, and the BAML bit is set to zero.

Transition F0:F0: This transition shall occur if:

- a) an implicit command is enabled;
- b) an explicit command is enabled, the medium position is not at BOx, and the BAML bit is set to zero. In this case, the device server shall return CHECK CONDITION status, the sense key shall be set to ILLEGAL REQUEST, and the additional sense code shall be set to ILLEGAL COMMAND WHILE IN IMPLICIT ADDRESS MODE; or
- c) an explicit write sequence command is enabled with the FCS bit set to zero. In this case, the device server shall return CHECK CONDITION status, the sense key shall be set to ILLEGAL REQUEST, and the additional sense code shall be set to ILLEGAL COMMAND WHILE IN IMPLICIT ADDRESS MODE.

4.2.24 TapeAlert application client interface

4.2.24.1 TapeAlert introduction

TapeAlert provides an application client with the capability to receive notification of various events and conditions arising in the SCSI target device. This standard defines 64 unique TapeAlert flags for a sequential-access device. A service information log parameter (see table 117) is also defined for each TapeAlert flag that provides information necessary for an application client to decide appropriate error recovery procedures.

TapeAlert flag severity is specified in table 11. If the TapeAlert log page (see 8.3.3) is reported, then the TapeAlert flags have the default severities specified in table 11. If the Current Service Information log page is supported, then

the values reported in the DEVICE SEVERITY CODE field of the device information descriptor (see 8.3.6.3) and the VOLUME SEVERITY CODE field of the volume information descriptor (see 8.3.6.4) override the default severities.

Table 11 — TapeAlert flags severity

Value ^a	Severity	Definition
01h	Informational	No guidance about continued operation without corrective action is given by this standard. The condition should be logged and/or an operator informed.
06h	Retryable	The event that generated this information may be retried.
0Bh	Warning	The system may not be operating optimally. Continued operation without corrective action may cause a failure or raise critical TapeAlert flags. The condition should be logged and/or an operator informed.
10h	Critical	Either a failure has already occurred or a failure is imminent. Corrective action is required. The condition should be logged and/or an operator informed.
15h	Intervention required	If this condition is not corrected, a data loss failure may occur. The condition should be logged and/or an operator informed.
1Ah	Call service	Action by service personnel is required. The condition should be logged and service personnel informed.
all others	-	Reserved

a) The severity value may be reported in a Current Service Information log page (see 8.3.6).

Table 12 specifies the 64 TapeAlert flags for a sequential-access device. See table 13 for the TapeAlert flag activation and deactivation conditions.

Table 12 — TapeAlert flags (part 1 of 3)

Flag	Name	Type	Default severity
01h	Read warning	O	W
02h	Write warning	O	W
03h	Hard error	M	W
04h	Medium	M	C
05h	Read failure	M	C
06h	Write failure	M	C
07h	Medium life	O	W
08h	Not data grade	O	W
09h	Write protect	O	C
0Ah	Volume removal prevented	O	I

Type key: M=Mandatory
O=Optional
W=Warning
C=Critical
I=Informational

Table 12 — TapeAlert flags (part 2 of 3)

Flag	Name	Type	Default severity
0Bh	Cleaning volume	O	I
0Ch	Unsupported format	O	I
0Dh	Recoverable mechanical cartridge failure	O	C
0Eh	Unrecoverable mechanical cartridge failure	O	C
0Fh	Memory chip in cartridge failure	O	W
10h	Forced eject	O	C
11h	Read only format	O	W
12h	Tape directory corrupted on load	O	W
13h	Nearing medium life	O	I
14h	Cleaning required	O	C
15h	Cleaning requested	O	W
16h	Expired cleaning volume	O	C
17h	Invalid cleaning volume	O	C
18h	Retension requested	O	W
19h	Multi-port interface error on a primary port	O	W
1Ah	Cooling fan failure	O	W
1Bh	Power supply failure	O	W
1Ch	Power consumption	O	W
1Dh	Drive preventive maintenance required	O	W
1Eh	Hardware A	O	C
1Fh	Hardware B	M	C
20h	Primary interface	O	W
21h	Eject volume	O	C
22h	Microcode update fail	O	W
23h	Drive humidity	O	W
24h	Drive temperature	O	W
25h	Drive voltage	O	W
26h	Predictive failure	O	C
27h	Diagnostics required	O	W
28h to 2Eh	Obsolete		

Type key: M=Mandatory
 O=Optional
 W=Warning
 C=Critical
 I=Informational

Table 12 — TapeAlert flags (part 3 of 3)

Flag	Name	Type	Default severity
2Fh	External data encryption control - communication failure	O	W
30h	External data encryption control - key manager returned an error	O	W
31h	Diminished native capacity	M	I
32h	Lost statistics	O	W
33h	Tape directory invalid at unload	O	W
34h	Tape system area write failure	O	C
35h	Tape system area read failure	O	C
36h	No start of data	O	C
37h	Loading or threading failure	O	C
38h	Unrecoverable unload failure	O	C
39h	Automation interface failure	O	C
3Ah	Microcode failure	O	W
3Bh	WORM volume - integrity check failed	O	W
3Ch	WORM volume - overwrite attempted	O	W
3Dh to 40h	Reserved		
Type key: M=Mandatory O=Optional W=Warning C=Critical I=Informational			

4.2.24.2 TapeAlert usage model

4.2.24.2.1 TapeAlert usage model introduction

This standard specifies three methods for an application client to monitor activation of TapeAlert flags:

- a) polling either the TapeAlert log page or the TapeAlert Response log page;
- b) configuring the device server to establish an Informational exception condition upon activation of one or more TapeAlert flags; and
- c) establishing a threshold for one or more of the parameters in the TapeAlert log page.

An application client may use any of these methods or a mixture of them.

Prior to using the TapeAlert Response log page (see 4.2.24.5) with method (a), an application client should determine whether the device server supports the TapeAlert Response log page. An application client may determine if a device server supports a log page by issuing a LOG SENSE command with the PAGE CODE field set to 00h and examining the data returned.

4.2.24.2.2 TapeAlert polling usage model

The application client configures the device server for the TapeAlert polling usage model by:

- a) setting the TASER bit in the Device Configuration Extension mode page to one (see 8.5.8); and
- b) setting the ETC bit of every parameter in the TapeAlert log page to zero (see 8.3.3).

NOTE 7 - Devices that comply with earlier versions of this standard set the ETC bit in each TapeAlert log parameter to zero and do not allow the application client to change this value.

If using the TapeAlert polling usage model, the application client reads the TapeAlert log page or the TapeAlert Response log page without receiving notification from the device server that a TapeAlert flag has changed state. The application client may read the TapeAlert log page or the TapeAlert Response log page at any time (e.g., polled at a regular interval of 60 seconds). The application client should read either the TapeAlert log page or the TapeAlert Response log page:

- a) prior to mounting a recording volume and at the beginning of a data transfer sequence;
- b) immediately after detecting an unrecoverable error during the data transfer sequence;
- c) before demounting each recording volume; and
- d) at the end of a data transfer sequence.

4.2.24.2.3 TapeAlert informational exception usage model

The application client configures the device server for the TapeAlert informational exception usage model by:

- a) setting the TASER bit in the Device Configuration Extension mode page to zero (see 8.5.8);
- b) setting the DEXCPT bit in the Informational Exceptions Control mode page to zero and the TEST bit in the Informational Exceptions Control mode page to zero (see 8.5.6);
- c) setting the MREI field in the Informational Exceptions Control mode page to a supported value greater than zero (see 8.5.6); and
- d) setting the ETC bit of every parameter in the TapeAlert log page to zero (see 8.3.3).

NOTE 8 - Devices that comply with earlier versions of this standard set the ETC bit in each TapeAlert log parameter to zero and do not allow the application client to change this value.

If using the TapeAlert informational exception usage model, the device server generates an informational exception condition if a TapeAlert flag is activated. The device server does not generate an informational exception condition if a TapeAlert flag is deactivated. The method used by the device server to report the informational exception condition depends on the value of the MREI field (see SPC-4). If the device server returns descriptor format sense data (see SPC-4), the current state of all TapeAlert flags appears in the information sense data descriptor (see 4.2.18.3). If the device server returns fixed format sense data (see SPC-4), the application client should read the TapeAlert log page to retrieve the state of the TapeAlert flags.

If the TEST bit is set to zero, a device server reporting an informational exception condition for a TapeAlert flag sets the additional sense code to FAILURE PREDICTION THRESHOLD EXCEEDED.

4.2.24.2.4 TapeAlert threshold usage model

The application client configures the device server for the TapeAlert threshold usage model by:

- a) setting the TASER bit in the Device Configuration Extension mode page to one (see 8.5.8);
- b) setting to one the ETC bit of each parameter in the TapeAlert log page (see 8.3.3) for which the application client wishes to receive a unit attention condition;

- c) setting to zero the ETC bit of each parameter in the TapeAlert log page (see 8.3.3) for which the application client does not wish to receive a unit attention condition; and
- d) establishing a threshold value and a threshold met criteria (TMC) value for each TapeAlert log page parameter with the ETC bit set to one (see SPC-4).

NOTE 9 - Devices that comply with earlier versions of this standard set the ETC bit in each TapeAlert log parameter to zero and do not allow the application client to change this value. These devices do not support the TapeAlert threshold usage model.

If using the TapeAlert threshold usage model, then the device server establishes a unit attention condition if a TapeAlert log page parameter meets its threshold criteria. If the device server returns descriptor format sense data (see SPC-4), the current state of all TapeAlert flags appears in the information sense data descriptor (see 4.2.18.3). If the device server returns fixed format sense data (see SPC-4), the application client should read the TapeAlert log page to retrieve the state of the TapeAlert flags.

The threshold and TMC values determine whether the device server establishes a unit attention condition if a TapeAlert flag is activated or if a TapeAlert flag is deactivated.

4.2.24.3 TapeAlert flag activation and deactivation

The device server shall activate a TapeAlert flag upon detecting the condition or event specified in table 13. The device server may activate more than one TapeAlert flag for the same condition (e.g., the device server may activate an 02h, Write warning, TapeAlert flag and also activate an 08h, Not data grade, TapeAlert flag if the write warning causes the vendor specific algorithm to determine that the volume is not suitable for long-term data storage).

Table 13 — TapeAlert flag activation and deactivation conditions (part 1 of 5)

Flag	Name	Activation condition	Deactivation condition
01h	Read warning	A vendor specific read warning threshold is crossed.	After the device is no longer able to detect the presence of the volume, or start of next volume load.
02h	Write warning	A vendor specific write warning threshold is crossed.	After the device is no longer able to detect the presence of the volume, or start of next volume load.
03h	Hard error	An unrecoverable read/write/positioning error.	After the device is no longer able to detect the presence of the volume, or start of next volume load ^a .
04h	Medium	An unrecoverable read/write/positioning error due to a faulty medium.	After the device is no longer able to detect the presence of the volume, or start of next volume load ^a .
05h	Read failure	An unrecoverable read error due to either a faulty medium or faulty device hardware.	After the device is no longer able to detect the presence of the volume, or start of next volume load ^a .
06h	Write failure	An unrecoverable write/positioning error due to either a faulty medium or faulty device hardware.	After the device is no longer able to detect the presence of the volume, or start of next volume load ^a .
a) Device compliant with previous versions of this standard may deactivate this TapeAlert flag at completion of volume demount.			

Table 13 — TapeAlert flag activation and deactivation conditions (part 2 of 5)

Flag	Name	Activation condition	Deactivation condition
07h	Media life	A vendor specific algorithm determines that the volume has reached the end of life.	After the device is no longer able to detect the presence of the volume, or start of next volume load.
08h	Not data grade	A vendor specific algorithm determines that the volume is not suitable for long term data storage.	After the device is no longer able to detect the presence of the volume, or start of next volume load.
09h	Write protect	A command is terminated with a DATA PROTECT sense key because the device server detects that the media is write protected at the target logical object identifier (see 4.2.20).	After the device is no longer able to detect the presence of the volume, or start of next volume load.
0Ah	Volume removal prevented	A request to unload a volume (e.g., SCSI command or eject button pressed) was rejected because a prevention of volume removal condition exists (see 7.6).	After the device is no longer able to detect the presence of the volume, or after volume removal is allowed.
0Bh	Cleaning volume	As soon as the device server detects that a cleaning volume is present.	After the device is no longer able to detect the presence of the volume, or start of next volume load.
0Ch	Unsupported format	As soon as the device server detects that the format of the volume is not supported for reading or writing.	After the device is no longer able to detect the presence of the volume; start of next volume load; or format change.
0Dh	Recoverable mechanical cartridge failure	The device server has detected a mechanical volume failure and the volume is able to be unloaded.	After the device is no longer able to detect the presence of the volume; or start of next volume load.
0Eh	Unrecoverable mechanical cartridge failure	The device server has detected a mechanical volume failure and the volume is not able to be unloaded.	After service resolution.
0Fh	Memory chip in cartridge failure	The device server has detected an unrecoverable error in the memory chip in the volume.	After the device is no longer able to detect the presence of the volume; or start of next volume load.
10h	Forced eject	A volume is ejected due to a manual or forced eject while: <ol style="list-style-type: none"> volume removal was prevented; logical objects to be written are in the object buffer; or a command requesting logical objects is in process. 	Start of next volume load.
11h	Read only format	The device server detects that the volume format is not able to be written by this device. The device server may activate this flag during a mount or during an attempted write.	After the device is no longer able to detect the presence of the volume; start of next volume load; or format change.
a) Device compliant with previous versions of this standard may deactivate this TapeAlert flag at completion of volume demount.			

Table 13 — TapeAlert flag activation and deactivation conditions (part 3 of 5)

Flag	Name	Activation condition	Deactivation condition
12h	Tape directory corrupted on load	The device detects that the tape directory was corrupt when the volume was loaded.	After the device is no longer able to detect the presence of the volume; or start of next volume load.
13h	Nearing medium life	A vendor specific algorithm has determined that the medium is nearing end of life.	After the device is no longer able to detect the presence of the volume; or start of next volume load.
14h	Cleaning required	A vendor specific algorithm has determined that the device requires cleaning.	After successful cleaning or cause resolved.
15h	Cleaning requested	A vendor specific algorithm has determined that a preventative cleaning operation is recommended.	After successful cleaning.
16h	Expired cleaning volume	The device detects that the cleaning volume in the device has expired.	Start of next volume load.
17h	Invalid cleaning volume	The device detects that the cleaning volume is not valid for this device.	Start of next volume load.
18h	Retention requested	A vendor specific algorithm has determined that retensioning the medium is recommended.	After successful retension.
19h	Multi-port interface error	A vendor specific algorithm has determined that one of the ports of a primary port interface with multiple ports is operating outside of specified limits.	After a vendor specific algorithm has determined that all ports of a multi-port interface are operating within specified limits.
1Ah	Cooling fan failure	The device detects that a cooling fan is not operating within vendor specified limits.	After a vendor specific algorithm has determined that all cooling fans are operating within vendor specified limits.
1Bh	Power supply failure	The device detects that a power supply is not operating within vendor specified limits.	After a vendor specific algorithm has determined that all power supplies are operating within vendor specified limits.
1Ch	Power consumption	The device detects that the power consumption is outside of vendor specified limits.	After power consumption returns to within vendor specified limits.
1Dh	Drive preventative maintenance required	A vendor specific algorithm has determined that preventative maintenance is required.	After service resolution.
1Eh	Hardware A	A device detects a hardware fault.	After service resolution.
1Fh	Hardware B	An error during power on self test.	At power on event.

a) Device compliant with previous versions of this standard may deactivate this TapeAlert flag at completion of volume demount.

Table 13 — TapeAlert flag activation and deactivation conditions (part 4 of 5)

Flag	Name	Activation condition	Deactivation condition
20h	Primary interface	A vendor specific algorithm has determined that a primary port interface is operating outside of specified limits.	After a vendor specific algorithm has determined that all primary interfaces are operating within vendor specified limits.
21h	Eject volume	A vendor specific algorithm has determined that the volume should be demounted.	After volume is demounted.
22h	Microcode update fail	A microcode update process failed.	Start of next microcode update.
23h	Drive humidity	The device detects that the humidity is outside of the specified operational limits.	After humidity returns to within specification.
24h	Drive temperature	The device detects that the temperature is outside of the specified operational limits.	After temperature returns to within specification.
25h	Drive voltage	The device detects that a voltage is outside of the specified operational limits.	After voltage returns to within specification.
26h	Predictive failure	A vendor specific algorithm has determined that a failure may occur.	After service resolution.
27h	Diagnostics required	A vendor specific algorithm has determined that a diagnostic should be performed.	After service resolution.
28h - 2Eh	Obsolete		
2Fh	External data encryption control - communication failure	The device server detects a communication failure with an out of band encryption key manager.	After service resolution.
30h	External data encryption control - key manager returned an error	A request for service from an out of band encryption key manager is returned with an error from the encryption key manager.	The logical position of the medium is changed.
31h	Diminished native capacity	A write from BOP 0 occurs and the volume state is set to not allow partition 0 to use the full native capacity of the volume (e.g., the volume is partitioned or the available medium for use has been reduced by a SET CAPACITY command).	After the device is no longer able to detect the presence of the volume; start of next volume load; or if the volume state is changed to allow partition 0 to use the full native capacity of the recording volume.
a) Device compliant with previous versions of this standard may deactivate this TapeAlert flag at completion of volume demount.			

Table 13 — TapeAlert flag activation and deactivation conditions (part 5 of 5)

Flag	Name	Activation condition	Deactivation condition
32h	Lost statistics	The device server detects that an event has caused some or all of the statistics for the volume to be lost.	After the device is no longer able to detect the presence of the volume; or start of next volume load .
33h	Tape directory invalid at unload	The device detects that the tape directory is invalid when a volume is demounted.	After the device is no longer able to detect the presence of the volume; or start of next volume load.
34h	Tape system area write failure	The device detects a failure writing to the system area.	After the device is no longer able to detect the presence of the volume; or start of next volume load .
35h	Tape system area read failure	The device detects a failure reading from the system area.	After the device is no longer able to detect the presence of the volume; or start of next volume load.
36h	No start of data	The device is unable to locate a valid start of data.	After the device is no longer able to detect the presence of the volume; or start of next volume load.
37h	Loading or threading failure	A vendor specific retry reporting threshold for loading or threading failures is crossed.	After the device is no longer able to detect the presence of the volume; or start of next volume load.
38h	Unrecoverable unload failure	The device reaches the end of the allowed unload retries without successfully unload the volume.	After the device is no longer able to detect the presence of the volume; or after service resolution.
39h	Automation interface failure	The device detects a failure of the automation interface that is not caused by protocol violations or invalid commands.	After the automation interface returns to operation within vendor specified limits.
3Ah	Microcode failure	The device detects a firmware fault.	After service resolution.
3Bh	WORM Medium – Integrity Check Failed	The device detects that a WORM volume integrity check has failed.	After the device is no longer able to detect the presence of the volume; or start of next volume load.
3Ch	WORM Medium – Overwrite Attempted	The device server detects that an overwrite of a read-only logical object was attempted on a WORM volume.	After the device is no longer able to detect the presence of the volume; or start of next volume load.
3Dh - 40h	Reserved		

a) Device compliant with previous versions of this standard may deactivate this TapeAlert flag at completion of volume demount.

In addition to the activation conditions shown in table 13, vendor specific conditions may cause TapeAlert flags to be activated.

Initialization processing due to a power-on condition may activate some TapeAlert flags.

A device server that supports the TapeAlert Response log page (see 4.2.24.5) shall deactivate a TapeAlert flag upon detecting the condition or event specified for that flag in table 13. A device server that does not support the

TapeAlert Response log page should deactivate a TapeAlert flag upon detecting the condition or event specified for that flag in table 13. The device server shall deactivate all TapeAlert flags:

- a) upon processing a LOG SENSE command with the PAGE CODE field set to 2Eh if the TAPLSD bit is set to zero in the Device Configuration Extension mode page (see 8.5.8); or
- b) upon detecting a logical unit reset condition (see SAM-5).

The device server may deactivate any TapeAlert flag on a vendor-specific basis due to:

- a) processing a LOG SELECT command with the PCR bit set to one (see SPC-4); or
- b) processing a LOG SELECT command with the PARAMETER LIST LENGTH field set to zero and the PC field set to 11b.

If the device server deactivates a TapeAlert flag by processing a LOG SENSE command with the PAGE CODE field set to 2Eh, then the device server shall not activate the flag again until the device server:

- a) detects the deactivation condition specified in table 13;
- b) detects a logical unit reset condition; or
- c) processes a LOG SELECT command with the PCR bit set to one.

If the device server deactivates a TapeAlert flag through a mechanism other than a LOG SENSE command, then the device server may activate the flag the next time the activation condition is detected.

If the TAPLSD bit in the Device Configuration Extension mode page (see 8.5.8) is set to zero, the device server should deactivate flags on a per I_T nexus basis such that active flags are available for reading by other I_T nexus. If the TAPLSD bit in the Device Configuration Extension mode page (see 8.5.8) is set to one, the device server may deactivate flags on a per I_T nexus basis.

NOTE 10 - Backwards compatibility with previous versions of this standard is violated if the TAPLSD bit is set to zero and the device server deactivates TapeAlert flags on any basis other than per I_T nexus.

4.2.24.4 WORM TapeAlert flags

Two TapeAlert flags exist to support Write Once Read Many (WORM) volumes:

- a) 3Bh (i.e., WORM volume - integrity check failed); and
- b) 3Ch (i.e., WORM volume - overwrite attempted).

If the device server supports TapeAlert flag 3Bh, it shall activate that flag upon detecting that the integrity of the volume may be compromised. If the device server supports TapeAlert flag 3Ch, it shall activate that flag if an application client attempts to overwrite or erase user data on a WORM volume (see 4.2.28).

4.2.24.5 TapeAlert Response log page

The TapeAlert flags reported through the TapeAlert Response log page (see 8.3.1) represent states. The device server does not maintain unique TapeAlert information for each I_T nexus, and the state flags are not affected by SCSI port events (e.g., I_T nexus loss).

An application client is responsible for determining which TapeAlert flags have changed state between retrievals of the TapeAlert Response log page. The application client may maintain a state change history.

If the TAPLSD bit in the Device Configuration Extension mode page (see 8.5.8) is set to one, the device server shall maintain the value of the flags in the TapeAlert Response log page independently of the TapeAlert flags reported through the TapeAlert log page (see 8.3.3). A LOG SENSE command that retrieves the TapeAlert Response log

page shall not set the flags in that page to zero and shall not set the flags in the TapeAlert log page to zero. A LOG SENSE command that retrieves the TapeAlert log page shall not set the flags in the TapeAlert Response log page to zero.

4.2.25 Dynamic runtime information

4.2.25.1 Dynamic runtime information overview

Dynamic runtime information allows an initiator to set attributes about itself into a device server. These attributes are called dynamic runtime attributes. The device server then associates those attributes to the I_T_L nexus and uses the information and associations for enhanced data collection and debugging. This information and the associations may be added to device error logs and are provided for retrieval by an application client through the READ DYNAMIC RUNTIME ATTRIBUTE command (see 7.8). Device servers that support dynamic runtime attributes shall support the READ DYNAMIC RUNTIME ATTRIBUTE command and the WRITE DYNAMIC RUNTIME ATTRIBUTE command (see 7.16).

A dynamic runtime attribute is represented in the format described in 8.2.1 and is composed of:

- a) an I_T nexus index;
- b) an attribute identifier;
- c) an attribute format code;
- d) a bit indicating whether the identified attribute is read only;
- e) an attribute length specifying the number of bytes in the identified attribute value; and
- f) the value of the identified attribute.

There are three types of dynamic runtime attributes (see table 14).

Table 14 — Types of dynamic runtime attributes

Attribute Type	Scope	Attribute Source	Writable ^a	Reference
Device	SCSI Device	Set by the device server	No	8.2.2.2
Target	I_T Nexus	Set by the device server	No	8.2.2.3
Initiator	I_T Nexus	Set by the application client	Yes	8.2.2.4
a) with the WRITE DYNAMIC RUNTIME ATTRIBUTE command (see 7.16)				

Dynamic runtime attributes have the states shown in table 15.

Table 15 — Dynamic runtime attribute states

Attribute State	Description
Read Only	An application client may read the contents of the dynamic runtime attribute with the READ DYNAMIC RUNTIME ATTRIBUTE command, but an attempt to clear or change the dynamic runtime attribute using the WRITE DYNAMIC RUNTIME ATTRIBUTE command shall result in the command being terminated with CHECK CONDITION status with the sense key set to DATA PROTECT and the additional sense code set to WRITE PROTECTED. When in the read only state the READ ONLY bit (see 8.2.1) is one.
Unsupported	The device server does not support the dynamic runtime attribute and shall not return it in response to a READ DYNAMIC RUNTIME ATTRIBUTE command.

Table 15 — Dynamic runtime attribute states

Attribute State	Description
Nonexistent	An initiator attribute does not exist in the dynamic runtime attributes until a WRITE DYNAMIC RUNTIME ATTRIBUTE command creates it.
Read/Write	The dynamic runtime attribute has been created using the WRITE DYNAMIC RUNTIME ATTRIBUTE command. After the dynamic runtime attribute has been created, the contents may be altered using subsequent WRITE DYNAMIC RUNTIME ATTRIBUTE commands. A dynamic runtime attribute in the Read/Write state may be returned to the nonexistent state using a WRITE DYNAMIC RUNTIME ATTRIBUTE command with the attribute length set to zero. When in the Read/Write state the READ ONLY bit (see 8.2.1) is zero.

4.2.25.2 Dynamic runtime information timestamp

Some dynamic runtime attributes have a timestamp (see SPC-4) associated with them. If no timestamp is set by the SET TIMESTAMP command or by a method outside the scope of the standard, then the timestamp shall be based on power-on time.

4.2.25.3 Writing dynamic runtime information into the SCSI device

An application client may set attributes in the SCSI device using the WRITE DYNAMIC RUNTIME ATTRIBUTE command (see 7.16) to set one or more of the initiator type attributes defined in 8.2.2.4. The application client may write these values at any time and may change these values at any time. If an application client attempts to create new attributes by writing attributes that were previously in the non-existent state and the device server does not have the resources necessary to create those attributes, then the device server shall create the attributes for which it has resources and return CHECK CONDITION status, with the sense key set to RECOVERED ERROR, and the additional sense code set to ROUNDED PARAMETER.

Editors Note 1 - This requirement on returning ROUNDED PARAMETER is repeated in the command and neither place says what to do if none of the request attributes are able to be created due to a lack of resources.

4.2.25.4 Retrieving dynamic runtime information from the SCSI device

An application client may read attributes by using the READ DYNAMIC RUNTIME ATTRIBUTE command (see 7.8). The application client may request a single attribute or multiple attributes in a single command. The application client may read any attribute(see 8.2) that is in the Read Only state or in the Read/Write state.

4.2.25.5 Management of dynamic runtime information

Dynamic runtime attributes have either a scope of a SCSI device (i.e., device type attributes) or a scope of I_T nexus (i.e., target type attributes and initiator type attributes). Some Device Attributes may contain target type attributes and initiator type attributes. This relationship is shown in figure 23.

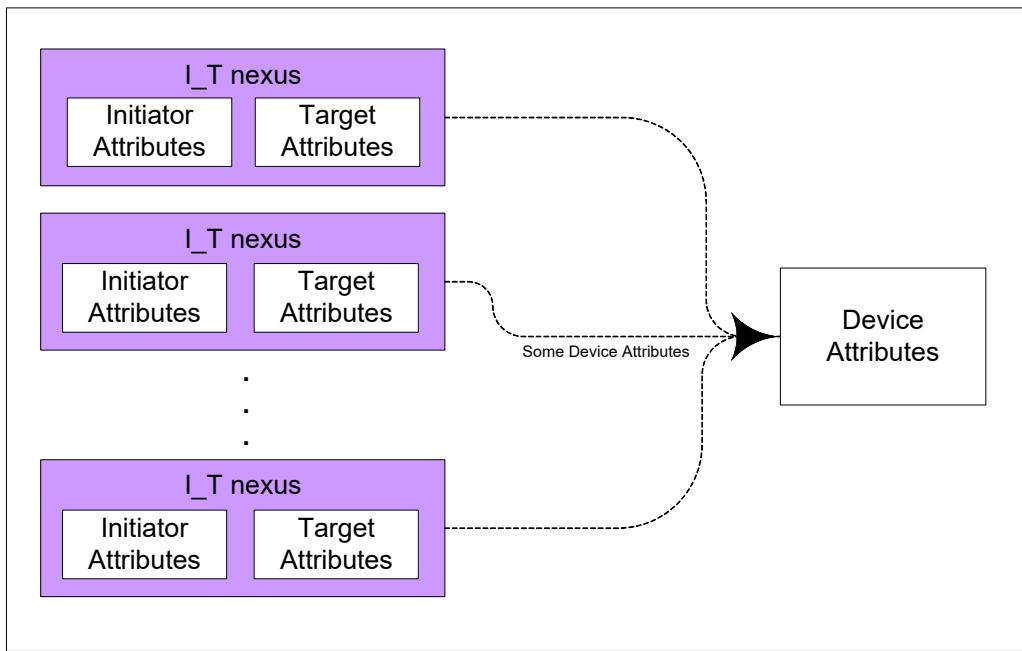


Figure 23 — Dynamic runtime attributes scope

For each dynamic runtime attribute (see 8.2) that the device server supports, if a command is received that changes a value contained in an attribute (e.g., Reserve, Persistent Reserve Out, Prevent/Allow Medium Removal), then the device server shall update that dynamic runtime attribute. If one or more of the Initiator type attributes that is a constituent of the attribute that is being updated is in the nonexistent state, then that attribute is not included, but attributes that exist are included. The TransportID of the I_T_L nexus and the target port identifier of the I_T_L nexus are presented as target type attributes.

4.2.25.6 Dynamic Runtime Information Lifetime

Dynamic Runtime Attributes are maintained separate from the device server's management of I_T nexuses. The I_T_L nexus identifying information (see 8.2.2.2.1) remains in existence inside dynamic runtime attributes even after the I_T nexus referenced is no longer known by the SCSI target port.

Device type attributes (see 8.2.2.2) are created or destroyed as described in the description of each device type attribute.

Target type attributes (see 8.2.2.3) are created by the device server if it detects communication from a new I_T nexus (e.g., Fibre Channel PLOGI/PRLI sequence) and are destroyed by the device server if it detects the disappearance of the I_T nexus (e.g., Fibre Channel LOGO).

Initiator type attributes (see 8.2.2.4) are created if a WRITE DYNAMIC RUNTIME ATTRIBUTE command is received requesting a new attribute and are destroyed if the device server detects the destruction of the target type attribute associated with the I_T nexus that created the Initiator type attribute or if a WRITE DYNAMIC RUNTIME ATTRIBUTE command is received requesting that initiator type attribute to be destroyed.

Dynamic Runtime Attributes do not persist across a power on event.

4.2.26 Recommended access order

A device server may accept a list of UDS descriptors (see 4.2.10.2), process the UDS descriptor list, and generate a Recommended Access Order (RAO) list using the GENERATE RECOMMENDED ACCESS ORDER command (see 7.3). The GENERATE RECOMMENDED ACCESS ORDER command allows the user to specify the process to use in generating the RAO list.

The RAO list is read with one or more RECEIVE RECOMMENDED ACCESS ORDER commands (see 7.10).

The device server may have a limit on the number of UDS descriptors that are supported. The RECEIVE RECOMMENDED ACCESS ORDER command and the UDS_LIMITS bit may be used to determine the number of supported UDS's and the maximum size of each UDS.

The RECEIVE RECOMMENDED ACCESS ORDER command returns the RAO list generated in the last successful GENERATE RECOMMENDED ACCESS ORDER command (see 7.3). The RAO list is generated based on the state of the mounted volume (e.g., logical position, logical objects on media, etc.) at the time the GENERATE RECOMMENDED ACCESS ORDER command is processed. The device server shall not update an RAO list except through the processing of a GENERATE RECOMMENDED ACCESS ORDER command. The device server shall clear the RAO list on a volume mount. The RAO list may be cleared by a hard reset or other vendor specific events.

The RAO PROCESS field (see table 16) specifies the process that shall be used to generate the recommended access order.

Table 16 — RAO PROCESS for generating recommended access order

Code	Description
000b	The device server shall not reorder the user data segment descriptors (see 7.3.2) passed in the GENERATE RECOMMENDED ACCESS ORDER parameter list and shall set the ESTIMATED LOCATE TIME TO UDS field in each of the user data segment descriptors of the RAO list to an estimation of the nominal time in seconds required to change position from the current position at the time the list is generated to the beginning logical object identifier of that user data segment descriptor (i.e., all the times are calculated from the same starting position).
001b	<p>The device server shall not reorder the user data segment descriptors passed in the GENERATE RECOMMENDED ACCESS ORDER parameter list and shall set the ESTIMATED LOCATE TIME TO UDS field of each user data segment descriptor to an estimation of the nominal time in seconds required to change position:</p> <ul style="list-style-type: none"> a) if this is the first user data segment descriptor in the RAO list, then from the current position at the time the list is generated to the beginning logical object identifier of the first user data segment descriptor; or b) if this is not the first user data segment descriptor in the RAO list, then from the ending logical object identifier of the previous user data segment descriptor to the beginning logical object identifier of this user data segment descriptor.
	<p>a) The following conditions may be considered or not considered by the device server in determining the value to place in the ESTIMATED LOCATE TIME TO UDS field:</p> <ul style="list-style-type: none"> a) read-ahead operations; b) application client behaviors; c) error recovery procedures; and d) other conditions.

Table 16 — RAO PROCESS for generating recommended access order

Code	Description
010b	The device server shall reorder the user data segment descriptors passed in the GENERATE RECOMMENDED ACCESS ORDER parameter list into a recommended access order and shall set the ESTIMATED LOCATE TIME TO UDS field of each user data segment descriptor to an estimation of the nominal time in seconds required to change position: <ol style="list-style-type: none"> if this is the first user data segment descriptor in the RAO list, then from the current position at the time the list is generated to the beginning logical object identifier of the first user data segment descriptor; or if this is not the first user data segment descriptor in the RAO list, then from the ending logical object identifier of the previous user data segment descriptor to the beginning logical object identifier of this user data segment descriptor.
011b	The device server shall not reorder the user data segment descriptors passed in the GENERATE RECOMMENDED ACCESS ORDER parameter list and shall set the ESTIMATED LOCATE TIME TO UDS field in each of the user data segment descriptors of the RAO list to an estimation of the nominal time in seconds required to change position from BOP 0 to the beginning logical object identifier of that user data segment descriptor (i.e., all the times are calculated from BOP 0).
100b-111b	Reserved <ol style="list-style-type: none"> The following conditions may be considered or not considered by the device server in determining the value to place in the ESTIMATED LOCATE TIME TO UDS field: <ol style="list-style-type: none"> read-ahead operations; application client behaviors; error recovery procedures; and other conditions.

4.2.27 Reservations

Reservation restrictions are placed on commands as a result of access qualifiers associated with the type of reservation. See SPC-4 for a description of reservations. The details of which commands are allowed under what types of reservations are described in table 17.

Application clients may use the OIR bit in the Device Configuration mode page (see 8.5.3) to require that the I_T nexus is participating in a reservation before processing commands that are defined:

- in SPC-2 as Conflict in the presence of reservations, except for the RESERVE, RELEASE, PERSISTENT RESERVATION IN and PERSISTENT RESERVATION OUT command;
- in table 17 as Conflict in the presence of persistent reservations; or
- in the table in SPC-4 that defines the commands that are allowed in the presence of various reservations as Conflict, except for the RESERVE, RELEASE, PERSISTENT RESERVATION IN and PERSISTENT RESERVATION OUT command.

Commands from I_T_nexus holding a reservation should complete normally. Table 17 specifies the behavior of commands from registered I_T_nexus in the presence of a registrants only or all registrants persistent reservation.

NOTE 11 - Due to the nature of the sequential-access device type, Write Exclusive and Write Exclusive, Registrants Only modes of reservation do not protect an application client's continuity of operations if using the implicit address command set. While these modes do protect unauthorized modification of data, they do not protect from medium position changes that may result in errors due to incorrect position. It is the responsibility of the

application client to manage this using means outside the scope of this specification. Application clients should use exclusive modes of reservation while accessing the volume to prevent interference from other applications.

For each command, this standard and the SPC-4 standard define the conditions that result in RESERVATION CONFLICT.

Table 17 — SSC-5 commands that are allowed in the presence of various reservations (part 1 of 2)

Command	Addressed LU has this type of persistent reservation held by another I_T_nexus				
	From any I_T_nexus		From registered I_T_nexus (RR all types)	From I_T_nexus not registered	
	Write Exclusive	Exclusive Access		Write Exclusive - RR	Exclusive Access - RR
ALLOW OVERWRITE	Conflict	Conflict	Allowed	Conflict	Conflict
ERASE(6)	Conflict	Conflict	Allowed	Conflict	Conflict
ERASE(16)	Conflict	Conflict	Allowed	Conflict	Conflict
FORMAT MEDIUM	Conflict	Conflict	Allowed	Conflict	Conflict
LOAD UNLOAD	Conflict	Conflict	Allowed	Conflict	Conflict
LOCATE(10)	Allowed	Conflict	Allowed	Allowed	Conflict
LOCATE(16)	Allowed	Conflict	Allowed	Allowed	Conflict
PREVENT ALLOW MEDIUM REMOVAL (Prevent=0)	Allowed	Allowed	Allowed	Allowed	Allowed
PREVENT ALLOW MEDIUM REMOVAL (Prevent<>0)	Conflict	Conflict	Allowed	Conflict	Conflict
READ(6)	Allowed	Conflict	Allowed	Allowed	Conflict
READ(16)	Allowed	Conflict	Allowed	Allowed	Conflict
READ BLOCK LIMITS	Allowed	Allowed	Allowed	Allowed	Allowed
READ POSITION	Allowed	Conflict	Allowed	Allowed	Conflict
READ REVERSE(6)	Allowed	Conflict	Allowed	Allowed	Conflict
READ REVERSE(16)	Allowed	Conflict	Allowed	Allowed	Conflict
RECOVER BUFFERED DATA	Allowed	Conflict	Allowed	Allowed	Conflict
REPORT DENSITY SUPPORT	Allowed	Allowed	Allowed	Allowed	Allowed
REWIND	Allowed	Conflict	Allowed	Allowed	Conflict
SET CAPACITY	Conflict	Conflict	Allowed	Conflict	Conflict

Key: **LU**=Logical Unit, **RR**=Registrants Only or All Registrants

Allowed: In the presence of a registrants only or an all registrants type persistent reservation, commands received from I_T nexus not holding the reservation or from I_T nexus not registered should complete normally.

Conflict: In the presence of a registrants only or an all registrants type persistent reservation, commands received from I_T nexus not holding the reservation or from I_T nexus not registered shall not be performed and the device server shall terminate the command with RESERVATION CONFLICT status.

Table 17 — SSC-5 commands that are allowed in the presence of various reservations (part 2 of 2)

Command	Addressed LU has this type of persistent reservation held by another I_T_nexus				
	From any I_T_nexus		From registered I_T_nexus (RR all types)	From I_T_nexus not registered	
	Write Exclusive	Exclusive Access		Write Exclusive - RR	Exclusive Access - RR
SPACE(6)	Allowed	Conflict	Allowed	Allowed	Conflict
SPACE(16)	Allowed	Conflict	Allowed	Allowed	Conflict
VERIFY(6)	Allowed	Conflict	Allowed	Allowed	Conflict
VERIFY(16)	Allowed	Conflict	Allowed	Allowed	Conflict
WRITE(6)	Conflict	Conflict	Allowed	Conflict	Conflict
WRITE(16)	Conflict	Conflict	Allowed	Conflict	Conflict
WRITE FILEMARKS(6)	Conflict	Conflict	Allowed	Conflict	Conflict
WRITE FILEMARKS(16)	Conflict	Conflict	Allowed	Conflict	Conflict

Key: LU=Logical Unit, RR=Registrants Only or All Registrants

Allowed: In the presence of a registrants only or an all registrants type persistent reservation, commands received from I_T nexus not holding the reservation or from I_T nexus not registered should complete normally.

Conflict: In the presence of a registrants only or an all registrants type persistent reservation, commands received from I_T nexus not holding the reservation or from I_T nexus not registered shall not be performed and the device server shall terminate the command with RESERVATION CONFLICT status.

4.2.28 Logical block protection

4.2.28.1 Logical block protection overview

A device server compliant with this standard may contain hardware or software that is capable of checking and/or generating protection information that is transferred with logical blocks between the device server and an application client. This protection information transferred with logical blocks is saved to the medium with each logical block and read from the medium with each logical block. This protection information is validated at the destination prior to completing the task thereby ensuring that the logical block has not been corrupted. This level of detection is not achievable by methods where the application client inserts vendor-specific data protection information in its data. The configuration of this capability is performed using the Control Data Protection mode page (see 8.5.9). A device server that supports using this protection information shall:

- a) set the PROTECT bit in standard inquiry (see SPC-4) to one;
- b) set the SPT field of the Extended INQUIRY Data VPD page (see SPC-4) to 001b; and
- c) set the value returned in the MAXIMUM BLOCK LENGTH LIMIT field of the READ BLOCK LIMITS command to a value that, if added to the largest value supported in the LOGICAL BLOCK PROTECTION INFORMATION LENGTH field of the Control Data Protection mode page, is less than or equal to the maximum length able to be represented in commands that transfer logical blocks between the application client and the device server.

A device server that supports logical block protection should support the Logical Block Protection VPD page (see 8.6.7) for reporting protection information methods supported by the device server.

4.2.28.2 Protection information on a volume

A recorded volume contains logical objects (see 4.2.8.1) and format-specific symbols. Logical objects are application client accessible. Format-specific symbols are used by the device server to provide methods for recording logical objects on the medium in a manner that allows them to be successfully read at a later date and may not be application client accessible. Often format-specific symbols contain information used to protect logical objects. A device server that supports protection information should include the protection information field as one of the format-specific symbols. If the protection information field is not included as one of the format-specific symbols then the protection information field shall be transformed by the device server into a format-specific symbol that provides at least the same level of protection as the protection information field. The format-specific symbol that is the protection information field or the transformed protection information field shall be written to the medium with each logical block. The protection information field is accessible by the application client if the LOGICAL BLOCK PROTECTION METHOD field of the Control Data Protection mode page for the I_T_L nexus through which that application client communicates is set to a non-zero value. A representation of logical objects and format-specific symbols is shown in figure 24.

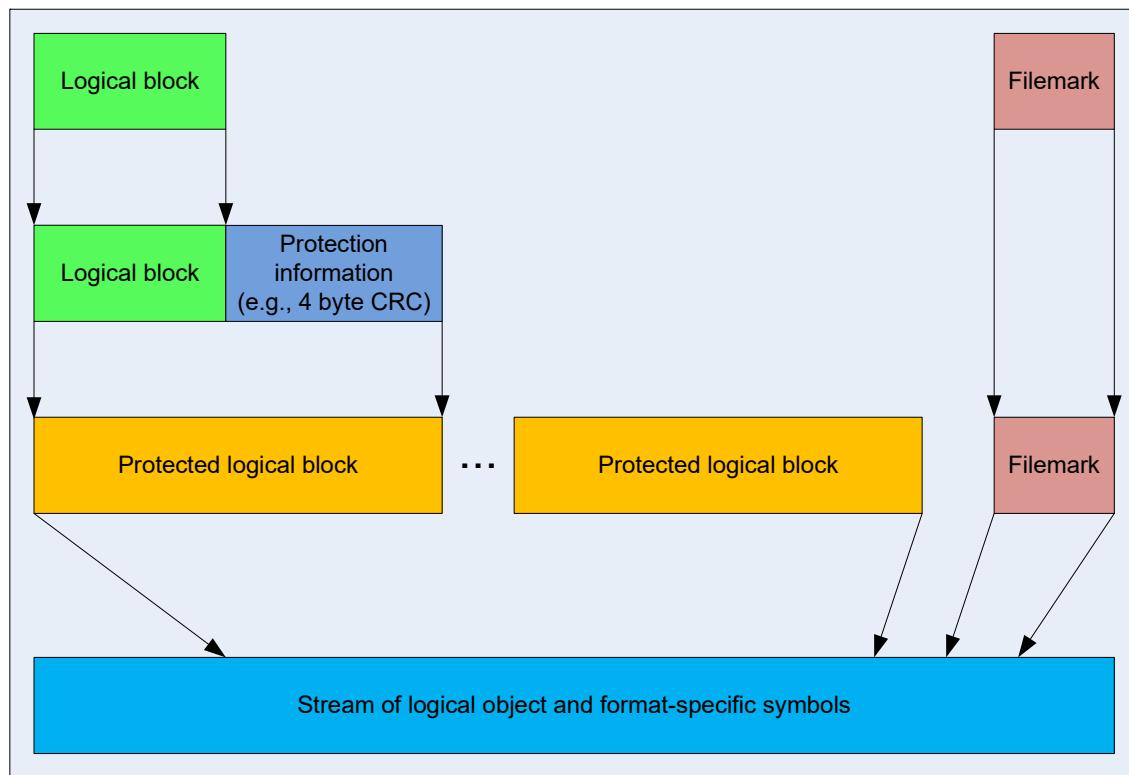


Figure 24 — Protection information shown in relationship to logical objects and format-specific symbols

A device server that supports a non-zero value in the LOGICAL BLOCK PROTECTION METHOD field of the Control Data Protection mode page shall generate the protection information or the format-specific symbol that provides the same level of protection as the protection information and add it to a logical block before recording the logical block to the medium if the command that transferred the logical block being recorded to medium was received on an I_T_L nexus for which:

- a) the LOGICAL BLOCK PROTECTION METHOD field of the Control Data Protection mode page is set to zero; or

- b) the LOGICAL BLOCK PROTECTION METHOD field of the Control Data Protection mode page is set to a non-zero value and the LBP_W bit of the Control Data Protection mode page is set to zero.

A device server that supports a non-zero value in the LOGICAL BLOCK PROTECTION METHOD field of the Control Data Protection mode page shall read the protection information or the format-specific symbol that provides the same level of protection as the protection information, if present, from the medium, validate it, and remove it from the logical block before transferring the logical block to the application client if the command that is requesting the transfer of a logical block being read was received on an I_T_L nexus for which:

- a) the LOGICAL BLOCK PROTECTION METHOD field of the Control Data Protection mode page is set to zero; or
- b) the LOGICAL BLOCK PROTECTION METHOD field of the Control Data Protection mode page is set to a non-zero value and the LBP_R bit of the Control Data Protection mode page is set to zero.

Protection information may be:

- a) compressed;
- b) encrypted; or
- c) included in byte counts in log parameters.

4.2.28.3 Logical blocks and protection information

If the LOGICAL BLOCK PROTECTION METHOD field of the Control Data Protection mode page is set to zero for a specific I_T_L nexus, then:

- a) a logical block transferred between the application client and the device server through that I_T_L nexus is formatted as specified in table 18 if that transfer occurs in response to a:
 - A) WRITE(6);
 - B) WRITE(16);
 - C) VERIFY(6) with the BYTCMP field set to one;
 - D) VERIFY(16) with the BYTCMP field set to one;
 - E) READ(6);
 - F) READ(16);
 - G) READ REVERSE(6) with the BYTORD bit set to one; or
 - H) READ REVERSE(16) with the BYTORD bit set to one; and
- b) a logical block transferred between the application client and the device server through that I_T_L nexus is formatted as specified in table 19 if that transfer occurs in response to a:
 - A) READ REVERSE(6) with the BYTORD bit set to zero; or
 - B) READ REVERSE(16) with the BYTORD bit set to zero.

The format for a logical block with no protection information is specified in table 18.

Table 18 — Logical block with no protection information format

Bit Byte	7	6	5	4	3	2	1	0
Data								
0	Most significant byte							
n-1	Least significant byte							
Key: n = the TRANSFER LENGTH field value in the CDB for variable-length transfers or the block length field value specified in the mode parameter header (see SPC-4) for fixed-block transfers.								

The format for a logical block for READ REVERSE with BYTORD bit set to zero and no protection information is specified in table 19.

Table 19 — Logical block for READ REVERSE with BYTORD=0b and no protection information format

Bit Byte	7	6	5	4	3	2	1	0
Data								
0	Least significant byte							
n-1	Most significant byte							
Key:								

n = the TRANSFER LENGTH field value in the CDB for variable-length transfers or the block length field value specified in the mode parameter header (see SPC-4) for fixed-block transfers.

If the logical block being transferred contains protection information (e.g., a read type command and the LBP_R bit is set to one), then:

- a) a logical block transferred between the application client and the device server through that I_T_L nexus is formatted as specified in table 20 if that transfer occurs in response to a:
 - A) WRITE(6);
 - B) WRITE(16);
 - C) VERIFY(6) with the BYTCMP field set to one;
 - D) VERIFY(16) with the BYTCMP field set to one;
 - E) READ(6);
 - F) READ(16);
 - G) READ REVERSE(6) with the BYTORD bit set to one; or
 - H) READ REVERSE(16) with the BYTORD bit set to one; and
- b) a logical block transferred between the application client and the device server through that I_T_L nexus is formatted as specified in table 21 if that transfer occurs in response to a:
 - A) READ REVERSE(6) with the BYTORD bit set to zero; or
 - B) READ REVERSE(16) with the BYTORD bit set to zero.

The format for a logical block with protection information is specified in table 20.

Table 20 — Logical block with protection information format

Bit Byte	7	6	5	4	3	2	1	0
Data								
0	Most significant byte							
n-x-1	Least significant byte							
Protection information								
n-x	Most significant byte							
n-1	Least significant byte							
Key:								

n = the TRANSFER LENGTH field value in the CDB for variable-length transfers or the block length field value specified in the mode parameter header (see SPC-4) for fixed-block transfers.

x = the LOGICAL BLOCK PROTECTION INFORMATION LENGTH field value in the Control Data Protection mode page.

The format for a logical block for READ REVERSE with BYTORD bit set to zero and protection information is specified in table 21.

Table 21 — Logical block for READ REVERSE with BYTORD=0b and protection information format

Bit Byte	7	6	5	4	3	2	1	0
Protection information								
0	Least significant byte							
n-x-1	Most significant byte							
Data								
n-x	Least significant byte							
n-1	Most significant byte							

Key:
n = the TRANSFER LENGTH field value in the CDB for variable-length transfers or the block length field value specified in the mode parameter header (see SPC-4) for fixed-block transfers.
x = the LOGICAL BLOCK PROTECTION INFORMATION LENGTH field value in the Control Data Protection mode page.

4.2.28.4 Protection information for Recover Buffered Data

In response to a RECOVER BUFFERED DATA command, the device server transfers unwritten logical blocks from the logical unit's object buffer to the application client. If the ROBO bit in the Device Configuration mode page (see 8.5.3) is set to zero, then the logical blocks are transferred in FIFO (first-in-first-out) order, which is the same order in which they were written to the object buffer (see table 22).

Table 22 — Data transferred during a RECOVER BUFFERED DATA command with the ROBO=0b

Bit Byte	7	6	5	4	3	2	1	0
0	(MSB)							
n-1								(LSB)
n	(MSB)							
s								(LSB)
⋮								
y	(MSB)							
x-1								(LSB)

Key:
n = the block length of the first logical block.
x = the number of bytes transferred by a single RECOVER BUFFERED DATA command.

If the ROBO bit in the Device Configuration mode page (see 8.5.3) is set to one, then the logical blocks are transferred in LIFO order (last-in-first-out), which is the opposite order from which they were written to the object buffer (see table 23).

Table 23 — Data transferred during a RECOVER BUFFERED DATA command with the ROBO=1b

Bit Byte	7	6	5	4	3	2	1	0
0	(MSB)							
n-1								(LSB)
:								
s	(MSB)							
r-1								(LSB)
r	(MSB)							
x-1								(LSB)

Key:
n = the block length of the last logical block.
x = the number of bytes transferred by a single RECOVER BUFFERED DATA command.

If the LOGICAL BLOCK PROTECTION METHOD field of the Control Data Protection mode page is set to zero for a specific I_T_L nexus, then the format of the logical block is as specified in table 18.

If the LOGICAL BLOCK PROTECTION METHOD field of the Control Data Protection mode page is set to a non-zero value for a specific I_T_L nexus and the RBDP bit is set to one, then the format of the logical block is as specified in table 20.

4.2.28.5 Processing logical blocks using logical block protection

4.2.28.5.1 Processing of conflicting logical block protection methods

A device server that supports more than one logical block protection method may have vendor-specific restrictions on when a specific logical block protection method may be used or when a transition between logical block protection methods may occur. A device server may support transformation between the logical block protection method requested by the application client and the protection information in the recorded format.

A device server shall not report an additional sense code set to LOGICAL BLOCK GUARD CHECK FAILED if the selected logical block protection method is unable to be used because of vendor-specific restrictions (e.g., the selected logical block protection method is not compatible with the protection information used by the logical block on the medium).

If the device server is unable to process a command using the selected logical block protection method with the protection information used by a logical block on the medium, then the device server shall terminate the command with CHECK CONDITION status with the sense key set to MISCOMPARE and the additional sense code set to LOGICAL BLOCK PROTECTION METHOD ERROR. The logical position shall be such that the logical block protection method may be changed and the command reissued on the same logical block without first changing position (e.g., for a READ REVERSE the logical position shall be on the EOP side of the logical block with the logical block protection method mismatch).

4.2.28.5.2 Protecting logical blocks transferred during writes

If the LBP_W bit of the Control Data Protection mode page (see 8.5.9) is set to one for a specific I_T_L nexus, then each logical block transferred from the application client through that I_T_L nexus contains protection information. The commands for which this applies are:

- a) WRITE(6);
- b) WRITE(16);
- c) VERIFY(6) with the BYTCMP bit set to one; and
- d) VERIFY(16) with the BYTCMP bit set to one.

See 4.2.28.5.3 for the behavior of a VERIFY(6) command or VERIFY(16) command with the BYTCMP bit set to zero.

For the WRITE(6) and WRITE(16) commands, the device server shall validate the protection information before the logical block is written to medium. If the FIXED bit in the CDB is set to one each logical block shall be validated before being written to the medium. If the validation of the protection information for a logical block fails, the processing of the command shall terminate prior to writing the failed logical block to the medium. If the validation of the protection information fails, then the device server shall report a CHECK CONDITION status with response code of current information or deferred error, the sense key set to HARDWARE ERROR and the additional sense code set to LOGICAL BLOCK GUARD CHECK FAILED.

For the VERIFY(6) and VERIFY(16) commands with the BYTCMP field set to one the protection information is validated prior to the byte-by-byte compare of the logical block on the medium and the logical block transferred from the application client. If the FIXED bit in the CDB is set to one each logical block shall be validated prior to the byte-by-byte compare of the logical block on the medium for that logical block and the data transferred from the application client for that logical block. If the validation of the protection information fails, the device server shall report a CHECK CONDITION status with response code of current information, the sense key set to HARDWARE ERROR and the additional sense code set to LOGICAL BLOCK GUARD CHECK FAILED. If the validation of the protection information does not fail for a logical block, the byte-by-byte compare of the logical block on the medium for that logical block and the data transferred from the application client for that logical block takes place. This byte-by-byte compare also includes the protection information on the medium and the protection information transferred from the application client.

A device server that supports the LOGICAL BLOCK PROTECTION METHOD field of the Control Data Protection mode page set to a non-zero value shall support the LBP_W bit of the Control Data Protection mode page set to one.

An application client that has set the LOGICAL BLOCK PROTECTION METHOD field of the Control Data Protection mode page to a non-zero value and the LBP_W bit of the Control Data Protection mode page to one shall add the protection information on each logical block before transferring that logical block and shall increase the TRANSFER LENGTH field by the length of the logical block protection information. The application client should add the protection information to the logical block at the earliest point possible. If the data has had the protection information added to the logical block at some point in the application client prior to the hardware that transfers the logical block, then the protection information should be validated when it is transferred. If the validation fails, then the application client should abort the command and report a status to the user that validation failed.

NOTE 12 - The device server treats the LOGICAL BLOCK PROTECTION INFORMATION field as the protection information. If the protection information is not added to the logical block, then the validation fails if the bytes used do not validate (e.g., the last 4-bytes of the logical block are treated as the CRC and the last 4-bytes of the logical block do not calculate as the CRC of the previous data).

4.2.28.5.3 Protecting logical blocks transferred during reads

If the LBP_R bit of the Control Data Protection mode page (see 8.5.9) is set to one for a specific I_T_L nexus, the protection information shall be read from the medium and transferred with the logical block to the application client on that I_T_L nexus. The commands for which this applies are:

- a) READ(6) command;
- b) READ(16) command;
- c) READ REVERSE(6) command; and
- d) READ REVERSE(16) command.

If the LBP_R bit of the Control Data Protection mode page is set to one for a specific I_T_L nexus, then the protection information, if any, shall be read from the medium and validated for the:

- a) VERIFY(6) command with the BYTCMP bit set to zero; and
- b) VERIFY(16) command with the BYTCMP bit set to zero.

See 4.2.28.5 for the behavior of a VERIFY(6) command or VERIFY(16) command with the BYTCMP bit set to one.

The protection information should be validated by the device server before sending status to the command that caused the transfer of the logical block. If the FIXED bit in the CDB is set to one each logical block shall be validated before being transferred to the application client. If the validation of the protection information for a logical block fails, the processing of the command shall terminate prior to transferring any additional blocks to the application client.

If, during the processing of a VERIFY(6) or VERIFY(16) command with the IMMED bit set to one, the validation of the protection information fails, then the device server shall establish a deferred error with a CHECK CONDITION status and sense data with a Response Code of Deferred error, the sense key set to HARDWARE ERROR, and the additional sense code set to LOGICAL BLOCK GUARD CHECK FAILED. If, during the processing of all other commands specified in this subclause, the validation of the protection information fails, the device server shall report a CHECK CONDITION status with response code of current information, the sense key set to HARDWARE ERROR, and the additional sense code set to LOGICAL BLOCK GUARD CHECK FAILED. A device server that supports the LOGICAL BLOCK PROTECTION METHOD field of the Control Data Protection mode page set to a non-zero value shall support the LBP_R bit of the Control Data Protection mode page set to one.

An application client that has set the LOGICAL BLOCK PROTECTION METHOD field of the Control Data Protection mode page to a non-zero value and the LBP_R bit of the Control Data Protection mode page to one should validate the protection information on each logical block at the latest point possible before using the data.

4.2.28.5.4 Protecting logical blocks transferred during a RECOVER BUFFERED DATA command

If the RBDP bit of the Control Data Protection mode page (see 8.5.9) is set to one for a specific I_T_L nexus, each logical block transferred between the device server and the application client on that I_T_L nexus during a RECOVER BUFFERED DATA command (see 7.11) shall include the protection information. The device server shall:

- a) read the protection information from the object buffer if it exists; or
- b) generate the protection information if it does not exist.

The protection information for each logical block should be validated before sending status to the command. If the validation of the protection information fails for any logical block, the device server shall terminate the command without transferring any additional logical blocks that may exist in the object buffer and report a CHECK CONDITION status with sense code of Current Sense, the sense key set to HARDWARE ERROR and the

additional sense code set to LOGICAL BLOCK PROTECTION ERROR ON RECOVER BUFFERED DATA. A device server shall support the RBDP bit of the Control Data Protection mode page set to one if it supports:

- a) the LOGICAL BLOCK PROTECTION METHOD field of the Control Data Protection mode page set to a non-zero value; and
- b) the RECOVER BUFFERED DATA command.

An application client that has set the LOGICAL BLOCK PROTECTION METHOD field of the Control Data Protection mode page to a non-zero value and the RBDP bit of the Control Data Protection mode page to one should validate the protection information on each logical block at the latest point possible before using the data.

4.2.28.6 File Verification of protection information

An application client may verify that protection information is present on each logical block on the medium between the current position and a specified number of filemarks from the current position and that the protection information validates correctly by using the VBF and VLBPM bits of the VERIFY(16) command (see 5.5) or the VERIFY(6) command (see 6.7) and setting to one the LBP_R bit of the Control Data Protection mode page (see 8.5.9). The device reads the medium verifying that each logical block between the current position and the n^{th} filemark is protected with the protection information using the LOGICAL BLOCK PROTECTION METHOD specified in the Control Data Protection mode page and that the protection information validates as shown in figure 25.

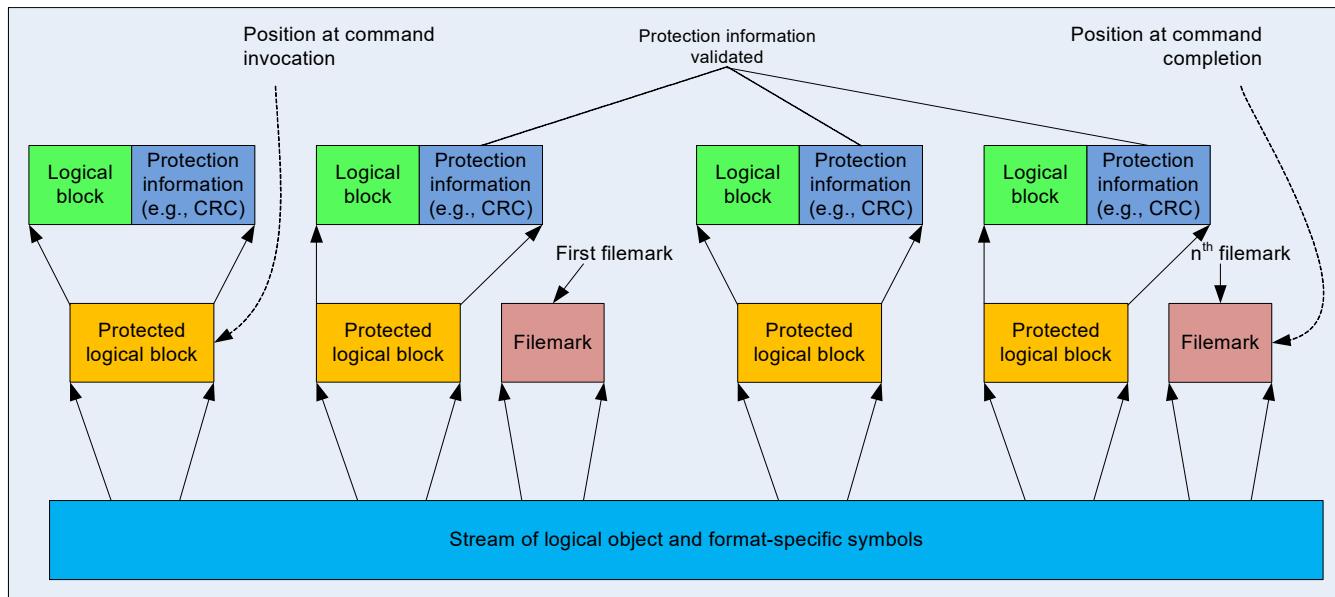


Figure 25 — Example file verification of protection information

4.2.28.7 Verification of protection information to EOD

An application client may verify that protection information is present on each logical block on the medium between the current position and EOD and that the protection information validates correctly by using the VTE and VLBPM bits of the VERIFY(16) command (see 5.5) or the VERIFY(6) command (see 6.7) and setting to one the LBP_R bit of the Control Data Protection mode page (see 8.5.9). The device reads each logical block between the current position and end-of-data verifying that each logical block is protected with the protection information using the LOGICAL

BLOCK PROTECTION METHOD specified in the Control Data Protection mode page and that the protection information validates as shown in figure 26.

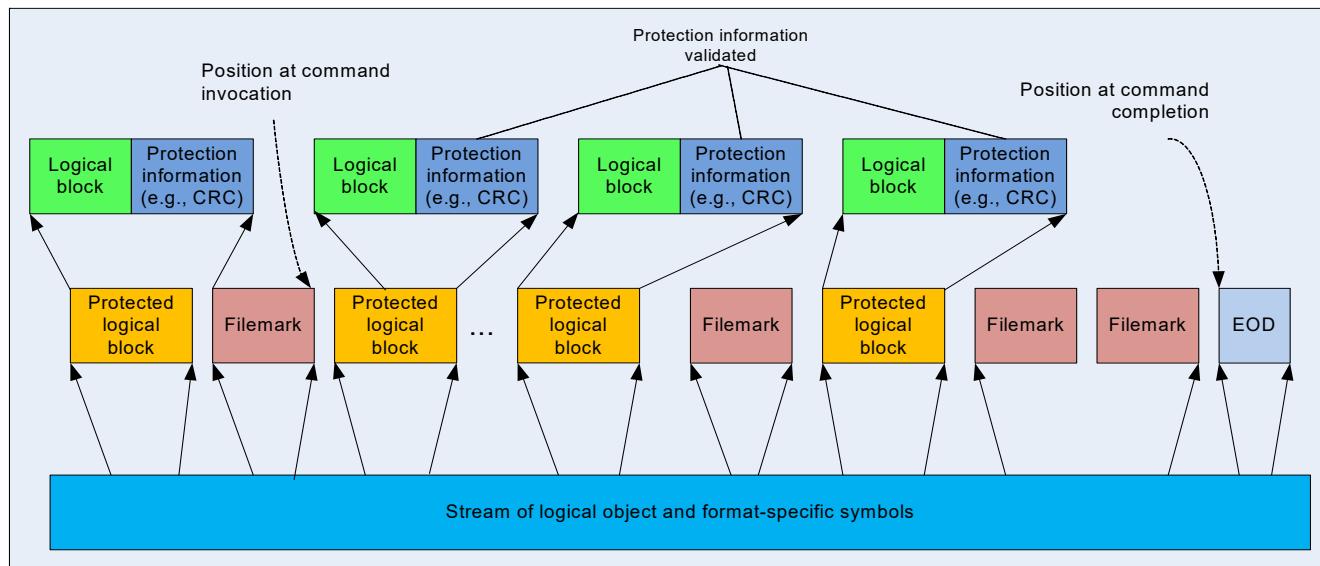


Figure 26 — Example verification of protection information to EOD

4.2.29 Logical block encryption

4.2.29.1 Logical block encryption overview

A device compliant with this standard may contain hardware or software that is capable of encrypting logical blocks as those blocks are stored on the medium, and decrypting logical blocks as those blocks are read from the medium, to provide security against unauthorized access to that data. The SECURITY PROTOCOL IN and SECURITY PROTOCOL OUT commands specifying the Tape Data Encryption security protocol (see 8.7.2 and 8.7.3) provide a means for the application client to monitor and control the encryption and decryption processes within the device entity. A device server that supports the SECURITY PROTOCOL OUT command shall also support the SECURITY PROTOCOL IN command.

The SECURITY PROTOCOL OUT command specifying the Tape Data Encryption security protocol is used to set logical block encryption parameters. The SECURITY PROTOCOL IN command specifying the Tape Data Encryption security protocol is used to discover the type of logical block security features supported by the device server, the current configuration of logical block security features, and status of the encryption and decryption processes.

4.2.29.2 Encrypting logical blocks on the medium

The application client controls the logical block encryption process by use of the SECURITY PROTOCOL OUT command specifying the Tape Data Encryption security protocol. Logical block encryption shall be managed within the device server on a per I_T_L nexus basis. The logical block encryption process is enabled for an I_T_L nexus upon successful completion of a SECURITY PROTOCOL OUT command that sends a Set Data Encryption page (see 8.7.3.2) with the ENCRYPTION MODE field set to ENCRYPT and with a valid logical block encryption key. If the logical block encryption scope parameter for an I_T_L nexus is set to PUBLIC (see 4.2.29.14), the logical block encryption process may be enabled by another I_T_L nexus that establishes a set of logical block encryption parameters with a logical block encryption scope of ALL I_T_NEXUS (see 4.2.29.14).

If logical block encryption is enabled for an I_T_L nexus and the mounted recording volume supports the selected encryption algorithm at the current logical position, all logical blocks received by the device server from that I_T_L nexus as part of a WRITE(6) or WRITE(16) command shall be encrypted before being recorded on the medium. Filemarks are logical objects that shall not be encrypted.

If logical block encryption is enabled for an I_T_L nexus and the mounted recording volume does not support the selected encryption algorithm at the current logical position, then the device server shall terminate a WRITE(6) or WRITE(16) command with CHECK CONDITION status, with the sense key set to DATA PROTECT, and the additional sense code set to ENCRYPTION PARAMETERS NOT USEABLE.

If logical block encryption is enabled for an I_T_L nexus and the mounted recording volume does not support the selected encryption algorithm at the current logical position, then the device server may terminate a WRITE FILEMARKS(6) or WRITE FILEMARKS(16) command with CHECK CONDITION status, with the sense key set to DATA PROTECT, and the additional sense code set to ENCRYPTION PARAMETERS NOT USEABLE.

4.2.29.3 Reading encrypted logical blocks on the medium

A recording volume may contain no encrypted logical blocks, all encrypted logical blocks, or a mixture of encrypted logical blocks and unencrypted logical blocks. The fact that logical blocks are encrypted shall not alter space or locate operations. The decryption mode shall be ignored if processing a filemark during a read or verify operation.

A device entity that supports encryption should be capable of distinguishing encrypted logical blocks from unencrypted logical blocks. The device server reports the capability of the device entity for distinguishing encrypted logical blocks from unencrypted logical blocks using the DELB_C bit in the logical block encryption algorithm descriptor (see 8.7.2.4). If the device entity is capable of distinguishing encrypted logical blocks from unencrypted logical blocks and the decryption mode is set to DISABLED, then an attempt to read or verify an encrypted logical block shall cause the device server to terminate the command with CHECK CONDITION status, with the sense key set to DATA PROTECT, and the additional sense code set to UNABLE TO DECRYPT DATA. The device entity shall establish the logical position at the BOP side of the encrypted logical block.

If the device entity is capable of distinguishing encrypted logical blocks from unencrypted logical blocks and the decryption mode is set to DECRYPT or RAW, then an attempt to read or verify an unencrypted logical block shall cause the device server to terminate the command with CHECK CONDITION status, with the sense key set to DATA PROTECT, and the additional sense code set to UNENCRYPTED DATA ENCOUNTERED WHILE DECRYPTING. The device entity shall establish the logical position at the BOP side of the unencrypted logical block.

A device entity that supports encryption and has been configured to decrypt the logical block may be capable of determining that the logical block encryption key is correct for an encrypted logical block. The correct logical block encryption key to use for this encrypted logical block may be either the logical block encryption key or one of the supplemental decryption keys (SDK). The method for determining which logical block encryption key or supplemental decryption key to use for decryption is vendor specific. If the device entity is capable of determining that the logical block encryption key is correct, the decryption mode is set to either DECRYPT or MIXED, and all of the logical block encryption keys provided are incorrect for the encrypted logical block, then an attempt to read or verify an encrypted logical block shall cause the device server to terminate the command with CHECK CONDITION status, with the sense key set to DATA PROTECT, and the additional sense code set to INCORRECT DATA ENCRYPTION KEY. The device entity shall establish the logical position at the BOP side of the encrypted logical block.

A device entity that supports encryption and has been configured to decrypt the logical block may be capable of validating the integrity of the logical block after decrypting it (i.e., that the decrypted logical block matches the logical block that was encrypted). If the device entity is capable of validating the integrity of the logical block after decrypting it and the decryption mode is set to either DECRYPT or MIXED, then an attempt to read or verify an encrypted logical block if the logical block fails the integrity validation process, shall cause the device server to

terminate the command with CHECK CONDITION status, with the sense key set to DATA PROTECT, and the additional sense code set to CRYPTOGRAPHIC INTEGRITY VALIDATION FAILED. The device entity shall establish the logical position at the BOP side the encrypted logical block.

A device entity that is capable of distinguishing encrypted logical blocks from unencrypted logical blocks and has been configured to decrypt the logical block should perform at least one of the following for each encrypted logical block that is decrypted:

- a) determine if the logical block encryption key is correct for the encrypted logical block; or
- b) validate the integrity of the logical block after decrypting it.

A device entity that is capable of determining if the logical block encryption key is correct for the encrypted logical block and validating the integrity of the logical block after decrypting it shall, for each encrypted logical block:

- 1) determine if the logical block encryption key is correct for the encrypted logical block; and
- 2) validate the integrity of the logical block.

4.2.29.4 Exhaustive-search attack prevention

To prevent an exhaustive-search attack from discovering the logical block encryption key or one of the supplemental decryption keys, the device server should provide a mechanism to prevent unlimited attempts at setting a logical block encryption key or supplemental decryption key(s) and then attempting to read the logical block. The use of such a mechanism may protect against an encryption algorithm being compromised.

If the device server has reached its limit on failed attempts to set the logical block encryption key or supplemental decryption key(s) and decrypt logical blocks, it shall disable encryption and decryption for all I_T_L nexus. All subsequent SECURITY PROTOCOL OUT commands specifying the Tape Data Encryption security protocol and with the SECURITY PROTOCOL SPECIFIC field set to Set Data Encryption page with the DECRYPTION MODE field or ENCRYPTION MODE field set to any value other than DISABLE shall be terminated with CHECK CONDITION status, with the sense key set to DATA PROTECT, and the additional sense code set to DATA DECRYPTION KEY FAIL LIMIT REACHED. This condition shall persist until the recording volume is demounted or a hard reset condition occurs.

4.2.29.5 Keyless copy of encrypted logical blocks

In some scenarios, it is desirable to copy logical blocks from one recording volume to another without needing knowledge of the encryption parameters used to encrypt the logical blocks on the recording volume.

A keyless copy logical unit (KCLU) controls configuration and data flows related to a recording volume that is either a source or destination for encrypted logical blocks being transferred without requiring application client knowledge of a logical block encryption key.

A keyless copy source logical unit (KCSLU) controls configuration and data flows related to the recording volume from which the encrypted logical block is copied without requiring device server knowledge of a logical block encryption key if the decryption mode is set to RAW. The device servers capability to act as a KCSLU for a specific encryption algorithm is indicated in the RDMC_C field of the Data Encryption Capabilities page (see 8.7.2.4).

A keyless copy destination logical unit (KCCLU) controls configuration and data flows related to the recording volume to which the encrypted logical block is being copied without requiring device server knowledge of a logical block encryption key if the encryption mode is set to EXTERNAL. The device servers capability to act as a KCCLU for a specific encryption algorithm is indicated in the EEMC_C field of the Data Encryption Capabilities page (see 8.7.2.4).

To accomplish a keyless copy operation, an application client sets the KCSLU decryption mode to RAW and the KCDLU encryption mode to EXTERNAL. The application client then reads one or more logical objects from the KCSLU and writes those logical objects to the KCDLU. During this process, if the KCSLU detects a mismatch between the key-associated data in the logical block encryption parameters and the key-associated data on the medium during a read operation, then the KCSLU returns a CHECK CONDITION status to the application client to notify it that some action is required. An example of this is shown in the informative flowchart in Annex B.

It shall not be considered an error if a Set Data Encryption page has the DECRYPTION MODE field set to RAW and any of the key-associated data descriptors required by the algorithm specified by the value in the ALGORITHM INDEX field are not present.

If the encryption algorithm in use by the KCSLU requires key-associated data to be included in the Set Data Encryption page if the ENCRYPTION MODE field is set to EXTERNAL, then an attempt to read or verify an encrypted logical block while the decryption mode is set to RAW shall cause the KCSLU to compare all key-associated data associated with each encrypted logical block that is read or verified to the corresponding key-associated data that are part of the current encryption parameters. Key-associated data required to be compared by the decryption algorithm that do not match or are not present shall cause the KCSLU to terminate the command with CHECK CONDITION status, with the sense key set to DATA PROTECT, and the additional sense code set to INCORRECT ENCRYPTION PARAMETERS. The KCSLU shall establish the logical position at the BOP side of the logical block.

If a KCDLU receives a SECURITY PROTOCOL OUT command with a Set Data Encryption page with the ENCRYPTION MODE field set to EXTERNAL, and any of the key-associated data descriptors required by the logical block encryption algorithm specified by the ALGORITHM INDEX field are not present or are not supported, then the KCDLU shall terminate the command with CHECK CONDITION, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

Some encryption algorithms provide a mechanism to record with encrypted logical blocks the setting of the encryption mode used to write the encrypted logical block. The device server reports if an encryption algorithm supports this mechanism by way of the EAREM bit in the algorithm descriptor (see 8.7.2.4). If the encryption algorithm provides this capability, then the device entity may support a mechanism to check during read and verify operations if the EXTERNAL encryption mode was used to write the encrypted logical block. The CEEM field in the Set Data Encryption page (see 8.7.3.2) provides the means to control this mechanism. If the decryption mode is set to DECRYPT or MIXED and the check external encryption mode logical block encryption parameter (see 4.2.29.15) is set to 10b (see table 217), then:

- 1) the device entity shall verify that each encrypted logical block that is processed for read and verify commands was written with the encryption mode set to ENCRYPT; and
- 2) if an attempt is made to read or verify an encrypted logical block that was written with the encryption mode set to EXTERNAL, the device server shall terminate the command with CHECK CONDITION status, with the sense key set to DATA PROTECT and the additional sense key set to ENCRYPTION MODE MISMATCH ON READ.

If the decryption mode is set to DECRYPT or MIXED and the check external encryption mode logical block encryption parameter is set to 11b (see table 217):

- 1) the device entity shall verify that each encrypted logical block that is processed for read and verify commands was written with the encryption mode set to EXTERNAL; and
- 2) if an attempt is made to read or verify an encrypted logical block that was written with the encryption mode set to ENCRYPT, then the device server shall terminate the command with CHECK CONDITION status, with the sense key set to DATA PROTECT and the additional sense key set to ENCRYPTION MODE MISMATCH ON READ.

The check external encryption mode logical block encryption parameter shall not affect space or locate operations. The check external encryption mode logical block encryption parameter shall not affect read or verify operations on filemarks and unencrypted logical blocks.

Some encryption algorithms provide a mechanism to record with encrypted logical blocks an indication that they are disabled for raw decryption mode operations. The device server reports if an encryption algorithm supports this mechanism by way of the RDMC_C field in the algorithm descriptor (see 8.7.2.4). If the decryption mode is set to RAW and the encryption algorithm supports this feature, then:

- 1) the device entity shall check the format-specific indication that disables raw decryption mode operations for each encrypted logical block that is processed for read and verify commands; and
- 2) if an attempt is made to read or verify an encrypted logical block that was disabled for raw decryption mode operations, then the device server shall terminate the command with CHECK CONDITION status, with the sense key set to DATA PROTECT and the additional sense key set to ENCRYPTED BLOCK NOT RAW READ ENABLED.

4.2.29.6 Managing logical block encryption keys within the device entity

To increase the security of logical block encryption keys, the logical block encryption parameters are volatile in the device entity and the logical block encryption keys are not reported to an application client.

A device server that supports logical block encryption shall support at least one of the key formats that are defined in this standard (see table 221).

A vendor-specific key reference is an identifier that is associated with a specific logical block encryption key. The method by which logical block encryption keys and their associated vendor-specific key references are made available to the device server is outside the scope of this standard. A device server that supports passing logical block encryption keys by vendor-specific key reference shall include the code for vendor-specific key reference format (see table 221) in the SUPPORTED KEY FORMATS LIST field in the Supported Key Formats page (see 8.7.2.5).

The device entity shall release the resources used to save a set of logical block encryption parameters (see 4.2.29.15) under the following conditions:

- a) the CKOD bit is set to one in the saved logical block encryption parameters and the volume is demounted;
- b) the CKORL bit is set to one and the logical block encryption scope is set to LOCAL in the saved logical block encryption parameters and the I_T_L nexus that established the set of logical block encryption parameters loses its reservation;
- c) the CKORL bit is set to one and the logical block encryption scope is set to ALL I_T NEXUS in the saved logical block encryption parameters and a reservation loss (see 3.1.66) occurs;
- d) the CKORP bit is set to one in the saved logical block encryption parameters and the device server processes a PERSISTENT RESERVE OUT command with a service action of either PREEMPT or PREEMPT AND ABORT;
- e) a microcode update is performed on the device; or
- f) a power on condition occurs.

The device entity may release the resources used to save a set of logical block encryption parameters if:

- a) a volume is mounted that does not support logical block encryption using the algorithm specified by the algorithm index logical block encryption parameter; or
- b) other vendor-specific events.

If a device server processes a Set Data Encryption page with the ENCRYPTION MODE field set to DISABLE and DECRYPTION MODE field set to DISABLE or RAW, the device entity shall:

- a) release any resources that it had allocated to store logical block encryption parameters for the I_T_L nexus associated with the SECURITY PROTOCOL OUT command and shall change the contents of all memory containing a key value associated with the logical block encryption parameters that are released; and
- b) establish a unit attention condition with the additional sense of DATA ENCRYPTION PARAMETERS CHANGED BY ANOTHER I_T NEXUS for all other I_T_L nexus that has its registered for logical block encryption unit attentions state set to one (see 4.2.29.14) and is affected by the loss of the logical block encryption key, (i.e., any I_T_L nexus that is using a logical block encryption scope of PUBLIC and the SCOPE field in the Set Data Encryption page is set to ALL I_T NEXUS).

If a device server processes a Set Data Encryption page that includes a logical block encryption key and the SDK bit is set to zero, then the device server shall establish a unit attention condition with the additional sense code set to DATA ENCRYPTION PARAMETERS CHANGED BY ANOTHER I_T NEXUS for all other I_T_L nexus that have their registered for logical block encryption unit attentions state set to one (see 4.2.29.14) and are affected by the change of the logical block encryption key (i.e., any I_T_L nexus that is using a logical block encryption scope of PUBLIC and the SCOPE field in the Set Data Encryption page is set to ALL I_T NEXUS), and the device entity shall:

- 1) release all resources that it had allocated to store key values set by previous Set Data Encryption pages from that I_T_L nexus and shall change the contents of all memory containing a key value associated with the logical block encryption parameters that are released; and
- 2) establish a set of logical block encryption parameters with the values from the Set Data Encryption page.

A device entity shall save at most one set of logical block encryption parameters with a logical block encryption scope of ALL I_T NEXUS. If a device server processes a Set Data Encryption page with the SCOPE field set to ALL I_T NEXUS, the device server shall establish a unit attention condition with the additional sense code set to DATA ENCRYPTION PARAMETERS CHANGED BY ANOTHER I_T NEXUS for all other I_T_L nexus that have their registered for logical block encryption unit attentions state set to one (see 4.2.29.14) and are affected by the change of the logical block encryption key (i.e., any I_T_L nexus that is using a logical block encryption scope of PUBLIC) and the device entity shall:

- a) release any resources that it had allocated to store logical block encryption parameters with a logical block encryption scope value of ALL I_T NEXUS and shall change the contents of all memory containing a key value associated with the logical block encryption parameters that are released; and
- b) establish a set of logical block encryption parameters with the values from the Set Data Encryption page and a logical block encryption scope value of ALL I_T NEXUS.

If a vendor-specific event occurs that changes or clears a set of logical block encryption parameters, the device server shall establish a unit attention condition with the additional sense of DATA ENCRYPTION PARAMETERS CHANGED BY VENDOR SPECIFIC EVENT for any I_T_L nexus that has its registered for logical block encryption unit attentions state set to one (see 4.2.29.14) and is affected by the change of the logical block encryption key.

4.2.29.7 Logical block encryption capabilities

A device entity that supports logical block encryption shall have a set of logical block encryption capabilities. The set of logical block encryption capabilities determine the values reported through a SECURITY PROTOCOL IN command specifying the Tape Data Encryption security protocol and the Data Encryption Capabilities page (see 8.7.2.4). The set of logical block encryption capabilities includes the set of logical block encryption algorithms supported by the device entity.

The set of logical block encryption capabilities includes some values that may be changed by a method outside the scope of this standard. The capabilities that may be changed include:

- a) the set of logical block encryption algorithms reported by the device server;
- b) encryption capable;
- c) decryption capable; and
- d) other vendor-specific logical block encryption capabilities.

4.2.29.8 Key instance counters

The device server shall keep a counter for each set of logical block encryption parameters that it is managing called the logical block encryption parameters key instance counter. The device entity shall keep a separate key instance counter called the device entity key instance counter. There may be a device entity key instance counter associated with each I_T_L nexus or there may be one global device entity key instance counter.

All key instance counters shall be set to zero if a hard reset condition occurs. Any other event that sets, clears, or changes a parameter in a set of logical block encryption parameters, except the supplemental decryption keys, shall cause the device entity key instance counter associated with the I_T_L nexus to be incremented. The value of the logical block encryption parameters key instance counter of the currently selected logical block encryption parameters for an I_T_L nexus is reported in the Data Encryption Status page of the SECURITY PROTOCOL IN command. The key instance counters are 32 bits and shall roll over to zero if incremented past their maximum value.

4.2.29.9 Encryption mode locking

There are conditions outside of the control of an application client that cause the device entity to release the resources used to save the logical block encryption parameters (see 4.2.29.6) or change the logical block encryption parameters used to control the encryption of logical blocks. Each of these conditions cause the device server to establish a unit attention condition to report the change of operating mode, but the unit attention condition may not always be reported to the application client through protocol bridges and driver stacks.

The LOCK bit in the Set Data Encryption page is set to one to lock the set of data encryption parameters established at the completion of the processing of the command to the I_T_L nexus through which the SECURITY PROTOCOL OUT command was issued. The set of logical block encryption parameters remains locked to that I_T_L nexus until a hard reset condition occurs or another SECURITY PROTOCOL OUT command including a Set Data Encryption page through the same I_T_L nexus is processed.

If the device server processes a WRITE(6) or WRITE(16) command through an I_T_L nexus that has its ITL_lock variable set to one, and the logical block encryption parameters key instance counter value has changed since the time it was locked (i.e., the logical block encryption parameters key instance counter does not equal the ITL_lock_key_instance_cnt for this I_T_L nexus), the device server shall terminate the command with CHECK CONDITION status, with the sense key set to DATA PROTECT, and the additional sense code set to DATA ENCRYPTION KEY INSTANCE COUNTER HAS CHANGED. All subsequent WRITE(6) and WRITE(16) commands shall also be terminated in this manner until a hard reset condition occurs or a SECURITY PROTOCOL OUT command including a Set Data Encryption page through the same I_T_L nexus is processed.

4.2.29.10 Nonce generation

For a given encryption algorithm, the device entity may:

- a) not require a nonce value;
- b) generate its own nonce value;
- c) require a nonce value or part of the nonce value be provided by the application client; or
- d) be configurable with respect to the source of the nonce value.

The device server reports its nonce value capability in the logical block encryption algorithm descriptor(s) (see 8.7.2.4). If the device server reports that it requires a nonce value from the application client and a Set Data Encryption page is processed that does not include a nonce value descriptor, the device server shall terminate the command with CHECK CONDITION, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INCOMPLETE KEY-ASSOCIATED DATA SET.

4.2.29.11 Unauthenticated key-associated data (U-KAD) and authenticated key-associated data (A-KAD)

Some encryption algorithms allow or require the use of additional data that is associated with the logical block encryption key and the logical block, but is not encrypted. Key-associated data may be authenticated by being included in the message authentication code (MAC) calculations for the encrypted logical block if such a MAC exists, or unauthenticated by not being included in these calculations.

The device server reports its capability with respect to key-associated data in the logical block encryption algorithm descriptor(s) DKAD_C field (see 8.7.2.4).

NOTE 13 - A key identifier or key reference may be stored in the U-KAD or A-KAD (see 8.7.3.2.1).

The U-KAD field is provided for applications that do not require the key-associated data to be protected by an MAC.

4.2.29.12 Metadata key-associated data (M-KAD)

Some encryption algorithms allow or require the use of additional data that is associated with the logical block encryption key and every logical block encrypted with that logical block encryption key. This data is contained in an M-KAD descriptor.

4.2.29.13 Wrapped key key-associated data (WK-KAD)

Some encryption algorithms allow or require the use of one or more EEDKs (see 4.2.31.4). The wrapped key key-associated data descriptor contains one EEDK. Up to MAXIMUM NUMBER OF EEDKS (see 8.7.2.4) number of WK-KAD may be transferred in the same page.

4.2.29.14 Logical block encryption information per I_T_L nexus

If the device server supports logical block encryption it shall maintain I_T_L nexus logical block encryption information on a per I_T_L nexus basis. This I_T_L nexus logical block encryption information shall contain the:

- a) ITL_scope (i.e., the logical block encryption scope for this I_T_L nexus);
- b) ITL_lock (i.e., the LOCK bit value for this I_T_L nexus);
- c) ITL_lock_key_instance_cnt (i.e., the logical block encryption parameters key instance counter value at the time the LOCK bit was set to one for this I_T_L nexus); and
- d) ITL_encryption_UA_reg (i.e., the registered for logical block encryption unit attentions state for this I_T_L nexus).

Device servers shall support setting the ITL_scope to any value of logical block encryption scope supported in the logical block encryption parameters. A device server that supports logical block encryption shall support an ITL_scope value of PUBLIC.

If the ITL_scope for this I_T_L nexus is set to PUBLIC it indicates that the logical block encryption parameters with a logical block encryption scope of ALL I_T NEXUS shall be used for commands received through this I_T_L nexus.

If a device server successfully completes the processing of a Set Data Encryption page with a logical block encryption scope of PUBLIC through this I_T_L nexus, then the device server shall:

- a) set the ITL_scope for this I_T_L nexus to PUBLIC;
- b) set the ITL_lock for this I_T_L nexus to the value of the LOCK bit in the Set Data Encryption page;
- c) set the ITL_lock_key_instance_cnt for this I_T_L nexus to:
 - A) zero if the LOCK bit in Set Data Encryption page is set to zero; or
 - B) the value of the logical block encryption parameters key instance counter in the ALL I_T_L_NEXUS logical block encryption parameters if the LOCK bit in the Set Data Encryption page is set to one; and
- d) set the ITL_encryption_UA_reg for this I_T_L nexus to one.

If a command is received through an I_T_L nexus with an ITL_scope of LOCAL, then the logical block encryption parameters with a logical block encryption scope of LOCAL for that I_T_L nexus shall be used for processing commands. If the resources used to save a set of logical block encryption parameters for an I_T_L nexus are released, then the ITL_scope for that I_T_L nexus shall be set to PUBLIC.

If a device server successfully completes the processing of a Set Data Encryption page with a logical block encryption scope of LOCAL through this I_T_L nexus, then the device entity shall:

- a) increment the device entity key instance counter associated with this I_T_L nexus;
- b) create a set of logical block encryption parameters with a logical block encryption scope of LOCAL for this I_T_L nexus; and
- c) set the logical block encryption parameters key instance counter to the value of the device entity key instance counter associated with this I_T_L nexus.

If a device server successfully completes the processing of a Set Data Encryption page with a logical block encryption scope of LOCAL through this I_T_L nexus, then the device server shall:

- a) set the ITL_scope for this I_T_L nexus to LOCAL;
- b) set the ITL_lock for this I_T_L nexus to the value of the LOCK bit in the Set Data Encryption page;
- c) set the ITL_lock_key_instance_cnt for this I_T_L nexus to:
 - A) zero if the LOCK bit in the Set Data Encryption is set to zero; or
 - B) the value of the logical block encryption parameters key instance counter of the logical block encryption parameters with a logical block encryption scope of LOCAL for this I_T_L nexus if the LOCK bit in the Set Data Encryption page is set to one; and
- d) set the ITL_encryption_UA_reg for this I_T_L nexus to one.

At most, one I_T_L nexus shall be assigned the logical block encryption scope of ALL I_T_NEXUS. If the device entity releases resources used to store a set of logical block encryption parameters with a logical block encryption scope of ALL I_T_NEXUS, it shall change the ITL_scope for the I_T_L nexus that established that set of logical block encryption parameters to PUBLIC. If a device server has a set of logical block encryption parameters with a logical block encryption scope of ALL I_T_NEXUS, and it successfully completes the processing of a Set Data Encryption page with a logical block encryption scope value of ALL I_T_NEXUS through a different I_T_L nexus than the one that established the set of logical block encryption parameters, the device entity shall change the ITL_scope for the I_T_L nexus that established the previous set of logical block encryption parameters to PUBLIC.

If a device server successfully completes the processing of a Set Data Encryption page with a logical block encryption scope of ALL I_T_NEXUS through this I_T_L nexus, then the device entity shall:

- a) increment the device entity key instance counter associated with this I_T_L nexus;
- b) perform the actions specified in 4.2.29.6 if a Set Data Encryption page with a logical block encryption scope of ALL I_T_NEXUS is received; and
- c) set the logical block encryption parameters key instance counter to the value of the device entity key instance counter associated with this I_T_L nexus.

Table 24 specifies the values assigned to the I_T_L nexus data encryption information when a power on condition occurs.

Table 24 — Default I_T_L nexus logical block encryption information

Parameter	Value when a power on condition occurs
ITL_scope	PUBLIC
ITL_lock	Zero
ITL_lock_key_instance_cnt	Zero
ITL_encryption_UA_reg	Zero

The ITL_encryption_UA_reg is a state variable that indicates if the device server shall establish unit attention conditions related to logical block encryption status for this I_T_L nexus. The device server shall set ITL_encryption_UA_reg to one for this I_T_L nexus if the device server processes a:

- a) SECURITY PROTOCOL IN command specifying the Tape Data Encryption protocol through this I_T_L nexus; or
- b) SECURITY PROTOCOL OUT command specifying the Tape Data Encryption protocol through this I_T_L nexus.

The device server shall set ITL_encryption_UA_reg to zero for an I_T_L nexus if an I_T nexus loss occurs for that I_T_L nexus. The device server shall set ITL_encryption_UA_reg to zero for all I_T_L nexus if the device server processes a logical unit reset.

4.2.29.15 Logical block encryption parameters

If a device server supports logical block encryption, then it shall have the ability to save in the device entity the following information as a set of logical block encryption parameters associated with the logical block encryption scope:

- a) for SCSI transport protocols where SCSI initiator device port names are required, the SCSI initiator device port name; otherwise, the SCSI initiator device port identifier;
- b) indication of the SCSI target port through which the logical block encryption parameters were established;
- c) logical block encryption scope;
- d) encryption mode;
- e) decryption mode;
- f) logical block encryption key;
- g) supplemental decryption keys where supported;
- h) algorithm index;
- i) logical block encryption parameters key instance counter;
- j) CKOD;
- k) CKORL;
- l) CKORP;
- m) U-KAD;
- n) A-KAD;
- o) M-KAD;
- p) nonce;
- q) raw decryption mode disable where supported;
- r) check external encryption mode where supported; and

- s) KAD FORMAT where supported.

If a Set Data Encryption page is processed and the logical block encryption scope is LOCAL, then the device server shall associate this set of logical block encryption parameters with the I_T_L nexus through which the command is received.

A device entity may release a previously established set of logical block encryption parameters if a Set Data Encryption page is processed and there are not enough unused resources available. The method of choosing which set of logical block encryption parameters to release is vendor specific. If the device entity does release a previously established set of logical block encryption parameters to free the resources, the device server shall establish a unit attention condition for each affected I_T_L nexus (see 4.2.29.6) that has its registered for logical block encryption unit attentions state set to one (see 4.2.29.14). A device entity is not required to have separate resources to store logical block encryption parameters for every logical block encryption scope that is supported.

If resources to save a set of logical block encryption parameters are released, then the value of each parameter in the set of logical block encryption parameters shall be set to a vendor-specific value.

When a power on condition occurs, the logical block encryption parameters shall be set to vendor-specific values.

A device server that supports logical block encryption shall support a logical block encryption scope value of ALL I_T NEXUS and the device entity shall have resources to save one set of logical block encryption parameters with this logical block encryption scope.

If the device server supports an logical block encryption scope value of LOCAL, the device entity shall have resources to save one or more sets of logical block encryption parameters with this logical block encryption scope.

The logical block encryption parameters that shall be used for an I_T_L nexus shall be established by the following order of precedence:

- a) if the ITL_scope for the I_T_L nexus is set to LOCAL or ALL I_T NEXUS (see 4.2.29.14), the logical block encryption parameters set by the last Set Data Encryption page processed through that I_T_L nexus; or
- b) if the ITL_scope for the I_T_L nexus is set to PUBLIC:
 - 1) the logical block encryption parameters that have been saved by the device entity with a logical block encryption scope of ALL I_T NEXUS if any logical block encryption parameters have been saved with this logical block encryption scope; or
 - 2) the default logical block encryption parameters.

4.2.29.16 Effects of reservation loss on logical block encryption parameters

4.2.29.16.1 Effects of reservation loss on logical block encryption parameters overview

The CKORL bit (see table 215) specifies that logical block encryption parameters are cleared on a reservation loss (see 3.1.66). This subclause describes the effects of reservation loss on the different scopes of logical block encryption parameters.

4.2.29.16.2 Effects of reservation loss on logical block encryption parameters with a logical block encryption scope of LOCAL

If a reservation loss occurs and a device entity has a saved set of logical block encryption parameters with a logical block encryption scope set to LOCAL and the CKORL bit set to one, then the device server shall:

- 1) release the resources for that saved set of logical block encryption parameters;
- 2) establish:

- A) a unit attention with the additional sense code set to DATA ENCRYPTION PARAMETERS CHANGED BY VENDOR SPECIFIC EVENT for each I_T_L nexus that has its registered for logical block encryption unit attentions state set to one (see 4.2.29.14) and is affected by the release of that set of logical block encryption parameters; and
- B) any other unit attention conditions required by the reservation loss; and
- 3) set the ITL_scope value to PUBLIC for the affected I_T_L nexus.

If a reservation loss occurs and a device entity has a saved set of logical block encryption parameters with a logical block encryption scope set to LOCAL and the CKORL bit set to zero, then the logical block encryption parameters shall not be affected unless the reservation loss occurred as the result of a reservation preempt. If the reservation loss occurred as the result of a reservation preempt, then the behavior shall be as specified in 4.2.29.17.2.

4.2.29.16.3 Effects of reservation loss on logical block encryption parameters with a logical block encryption scope of ALL I_T NEXUS

If a reservation loss occurs and a device entity has a saved set of logical block encryption parameters with a logical block encryption scope set to ALL I_T NEXUS and the CKORL bit set to one, then the device server shall:

- 1) release the resources for that saved set of logical block encryption parameters;
- 2) establish:
 - A) a unit attention with the additional sense code set to DATA ENCRYPTION PARAMETERS CHANGED BY VENDOR SPECIFIC EVENT for each I_T_L nexus that has its registered for logical block encryption unit attentions state set to one (see 4.2.29.14) and is affected by the release of that set of logical block encryption parameters; and
 - B) any other unit attention conditions required by the reservation loss; and
- 3) set the ITL_scope value to PUBLIC for the affected I_T_L nexus.

If a reservation loss occurs and device entity has a saved set of logical block encryption parameters with a logical block encryption scope set to ALL I_T NEXUS and the CKORL bit set to zero, then the logical block encryption parameters shall not be affected unless the reservation loss occurred as the result of a reservation preempt. If the reservation loss occurred as the result of a reservation preempt, then the behavior shall be as specified in 4.2.29.17.3.

4.2.29.17 Effects of reservation preempt on logical block encryption parameters

4.2.29.17.1 Effects of reservation preempt on logical block encryption parameters overview

The CKORP bit (see table 215) specifies that logical block encryption parameters are cleared on a reservation preempt. This subclause describes the effects of reservation preempt on the different scopes of logical block encryption parameters.

4.2.29.17.2 Effects of reservation preempt on logical block encryption parameters with a logical block encryption scope of LOCAL

If a reservation preempt occurs and a device entity has a saved set of logical block encryption parameters with a logical block encryption scope set to LOCAL and the CKORL bit set to one, then the device server shall:

- 1) release the resources for that saved set of logical block encryption parameters;
- 2) establish:
 - A) a unit attention with the additional sense code set to DATA ENCRYPTION PARAMETERS CHANGED BY VENDOR SPECIFIC EVENT for each I_T_L nexus that has its registered for logical block encryption unit attentions state set to one (see 4.2.29.14) and is affected by the release of that set of logical block encryption parameters; and
 - B) any other unit attention conditions required by the reservation preempt; and

- 3) set the ITL_scope value to PUBLIC for the affected I_T_L nexus.

If a reservation preempt occurs and a device entity has a saved set of logical block encryption parameters with a logical block encryption scope set to LOCAL and the CKORP bit set to zero, then the logical block encryption parameters shall not be affected unless the reservation preempt causes a reservation loss. If the reservation preempt causes a reservation loss, then the behavior shall be as specified in 4.2.29.16.2.

4.2.29.17.3 Effects of reservation preempt on logical block encryption parameters with a logical block encryption scope of ALL I_T NEXUS

If a reservation preempt associated with the I_T_L nexus through which the logical block encryption parameters were set occurs or a reservation preempt associated with an I_T_L nexus that has an ITL_scope of PUBLIC occurs, and a device entity has a saved set of logical block encryption parameters with a logical block encryption scope set to ALL I_T NEXUS and the CKORP bit set to one, then the device server shall:

- 1) release the resources used for the set of logical block encryption parameters with a logical block encryption scope of ALL I_T NEXUS;
- 2) establish:
 - A) a unit attention with the additional sense code set to DATA ENCRYPTION PARAMETERS CHANGED BY VENDOR SPECIFIC EVENT for each I_T_L nexus that has its registered for logical block encryption unit attentions state set to one (see 4.2.29.14) and is affected by the release of that set of logical block encryption parameters; and
 - B) any other unit attention conditions required by the reservation preempt; and
- 3) set the ITL_scope value to PUBLIC for the affected I_T_L nexus.

If a reservation preempt occurs and a device entity has a saved set of logical block encryption parameters with a logical block encryption scope set to ALL I_T NEXUS and the CKORP bit set to zero, then the logical block encryption parameters shall not be affected unless the reservation preempt causes a reservation loss. If the reservation preempt causes a reservation loss, then the behavior shall be as specified in 4.2.29.16.3.

4.2.30 External data encryption control

4.2.30.1 External data encryption control overview

A device entity that supports logical block encryption may support external data encryption control and provide the ability for an entity that is not part of the device server to configure logical block encryption capabilities or logical block encryption parameters using an interface not specified by this standard (e.g., an ADC device server or a management interface).

4.2.30.2 External data encryption control of data encryption capabilities

4.2.30.2.1 External data encryption control of data encryption capabilities overview

If the device entity has a saved set of data encryption parameters associated with this device server or has a volume mounted, then the device entity shall not allow external data encryption control of logical block encryption capabilities (see 4.2.29.7). If the device entity does not have a set of logical block encryption parameters associated with this device server and does not have a volume mounted, then external data encryption control may be used to change the logical block encryption capabilities.

The device entity shall not allow external data encryption control to change the logical block encryption capabilities (e.g., change the logical block encryption parameters control policy, see ADC-3) if the device entity has a saved set of logical block encryption parameters associated with this device server, or:

- a) has a volume mounted; and

- b) has a primary port enabled (see ADC-3).

The device entity should allow external data encryption control to change the logical block encryption capabilities if the device entity does not have a saved set of logical block encryption parameters associated with this device server, and:

- a) does not have a volume mounted; or
- b) has no primary ports enabled.

If external data encryption control is used to change any of the logical block encryption capabilities of the device entity, then the device server shall establish a unit attention condition with the additional sense code of DATA ENCRYPTION CAPABILITIES CHANGED for all L_T nexus that have their registered for logical block encryption unit attentions state set to one (see 4.2.29.14).

4.2.30.2.2 External data encryption control detection

The SECURITY PROTOCOL IN command specifying the Tape Data Encryption security protocol and the Data Encryption Capabilities page may be used to determine whether the device server supports external data encryption control.

The SECURITY PROTOCOL IN command specifying the Tape Data Encryption security protocol and the Data Encryption Status page may be used to determine which device server has control of the logical block encryption parameters and whether external data encryption control has been used to establish or change a set of logical block encryption parameters.

4.2.30.2.3 External data encryption control of encryption algorithm support

External data encryption control may be used to change the device server encryption algorithm support by configuring the device entity to:

- a) disable a supported logical block encryption algorithm; or
- b) prevent device server control of logical block encryption parameters.

If a supported encryption algorithm has been disabled, then:

- a) the device entity shall not accept logical block encryption parameters specifying that algorithm; and
- b) the device server shall:
 - A) not report the disabled logical block encryption algorithm in the Data Encryption Capabilities page; or
 - B) report the disabled logical block encryption algorithm in the Data Encryption Capabilities page with the DECRYPT_C field set to no capability and the ENCRYPT_C field set to no capability.

If external data encryption control has been used to configure the device entity to prevent device server control of logical block encryption parameters (e.g., an ADC device server logical block encryption parameters control policy is set to ADC exclusive, see ADC-3), then the device server shall:

- a) terminate a SECURITY PROTOCOL OUT command that attempts to establish or clear a set of logical block encryption parameters with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to DATA ENCRYPTION CONFIGURATION PREVENTED; and
- b) set the CFG_C (see 8.7.2.4) field in the Data Encryption Capabilities page to 10b (i.e., the device entity is configured to not allow this device server to establish or change logical block encryption parameters) and:
 - A) not report any encryption algorithms in the Data Encryption Capabilities page; or

- B) report all of the supported logical block encryption algorithms in the Data Encryption Capabilities page with the DECRYPT_C field set to capable with external control and the ENCRYPT_C field set to capable with external control.

NOTE 14 - The SECURITY PROTOCOL IN command specifying the Tape Data Encryption security protocol and the Data Encryption Status page may be used to determine whether external data encryption control has been used to provide a set of logical block encryption parameters.

4.2.30.3 External data encryption control of logical block encryption parameters

4.2.30.3.1 External data encryption control of logical block encryption parameters overview

External data encryption control may be used to control logical block encryption parameters by using:

- a) a logical block encryption parameters request policy to set a logical block encryption parameters request indicator to TRUE;
- b) a logical block encryption parameters period to determine how long to wait for the logical block encryption parameters request indicator to be set to FALSE; and
- c) the set of logical block encryption parameters that have been set in the device entity.

A device entity that supports external data encryption control shall contain a logical block encryption parameters request policy (see 4.2.30.3.2) and a set of logical block encryption parameters request indicators (see 4.2.30.3.3).

4.2.30.3.2 Logical block encryption parameters request policy

The logical block encryption parameters request policy determines if the device entity shall request a set of logical block encryption parameters from an entity using external data encryption control. The logical block encryption parameters request policy shall contain a logical block encryption parameters for encryption request policy and a logical block encryption parameters for decryption request policy.

External data encryption control sets the logical block encryption parameters for encryption request policy (see table 25) and the logical block encryption parameters for decryption request policy (see table 26) to indicate to the device entity what events shall cause a logical block encryption parameters request indicator to be set to TRUE (see 4.2.30.3.3). If external data encryption control is not being used, then the logical block encryption parameters request policies shall be set to defaults.

The logical block encryption parameters for encryption request policies are specified in table 25.

Table 25 — Logical block encryption parameters for encryption request policies

Policy	Description
No logical block encryption parameter requests	The device entity shall not set the logical block encryption parameters for encryption request indicator to TRUE.
Request logical block encryption parameters every reposition	<p>The device entity shall set the logical block encryption parameters for encryption request indicator to TRUE upon the device server processing the first:</p> <ul style="list-style-type: none"> a) WRITE(6) command; b) WRITE(16) command; c) WRITE FILEMARKS(6)^a command with a non-zero FILEMARK COUNT field; or d) WRITE FILEMARKS(16)^a command with a non-zero FILEMARK COUNT field; <p>after:</p> <ul style="list-style-type: none"> a) an ERASE(6) command; b) an ERASE(16) command; c) a FORMAT MEDIUM command; d) a LOCATE(10) command; e) a LOCATE(16) command; f) a LOAD UNLOAD command; g) a REWIND command; h) a READ(6) command; i) a READ(16) command; j) a READ REVERSE(6) command; k) a READ REVERSE(16) command; l) a VERIFY(6) command; m) a VERIFY(16) command; n) a SPACE(6) command; or o) a SPACE(16) command.
Request logical block encryption parameters if not set	<p>The device entity shall set the logical block encryption parameters for encryption request indicator to TRUE before accepting any data into the buffer or adding any filemarks to the buffer if in buffered mode or to the medium if in unbuffered mode, upon the device server processing the first:</p> <ul style="list-style-type: none"> a) WRITE(6) command; b) WRITE(16) command; c) WRITE FILEMARKS(6)^a command with a non-zero FILEMARK COUNT field; or d) WRITE FILEMARKS(16)^a command with a non-zero FILEMARK COUNT field; <p>after:</p> <ul style="list-style-type: none"> a) there is not an established set of logical block encryption parameters; or b) an event that causes the logical block decryption parameters request indicator to be set to TRUE (see table 26).
	<p>a) The WRITE FILEMARKS command is included in the list of commands that cause the logical block encryption parameters for encryption request indicator to be set to TRUE to prevent an application client from writing a filemark as part of a new operation (e.g., a backup operation starting with a WRITE FILEMARKS command and followed by a series of WRITE commands) if the operation is not successful due to a failure to retrieve a set of logical block encryption parameters.</p>

The logical block encryption parameters for decryption request policies are specified in table 26.

Table 26 — Logical block encryption parameters for decryption request policies

Policy	Description
No logical block decryption parameter requests	The device entity shall not set the logical block encryption parameters for decryption request indicator to TRUE.
Request logical block decryption parameters as needed	The device entity shall set the logical block encryption parameters for decryption request indicator to TRUE if the device entity detects that the current set of logical block encryption parameters is not correct for a logical block being processed as a result of processing: <ul style="list-style-type: none"> a) a READ(6) command; b) a READ(16) command; c) a READ REVERSE(6) command; d) a READ REVERSE(16) command; e) a RECOVER BUFFERED DATA command; f) a VERIFY(6) command with the BYTCMP bit set to one; or g) a VERIFY(16) command with the BYTCMP bit set to one.

The logical block encryption parameters for encryption request policy and the logical block encryption parameters for decryption request policy settings shall be set to defaults upon:

- a) a hard reset condition; or
- b) other vendor specific events.

4.2.30.3.3 Logical block encryption parameters request indicators

The logical block encryption parameters request indicators indicate if the device entity requires a set of logical block encryption parameters from an entity using external data encryption control. The logical block encryption parameters request indicators shall contain a logical block encryption parameters for encryption request indicator and a logical block encryption parameters for decryption request indicator.

The logical block encryption parameters for encryption request indicator settings are specified in table 27.

Table 27 — Logical block encryption parameters for encryption request indicator settings

Setting	Description
TRUE	The device entity is waiting for the logical block encryption parameters for encryption request indicator to be set to FALSE (e.g., an ADC device server processes a SECURITY PROTOCOL OUT command with a DATA ENCRYPTION PARAMETERS COMPLETE page and the clear encryption parameters request (CEPR) bit set to one, see ADC-3) before continuing to process the task in the enabled task state.
FALSE	The device entity is not waiting for the logical block encryption parameters for encryption request indicator to be set to FALSE before continuing to process the task in the enabled task state. This is the default setting for the logical block encryption parameters for encryption request indicator.

The device entity shall not change the logical position while the logical block encryption parameters for encryption request indicator is set to TRUE.

If the logical block encryption parameters for encryption request indicator is set to FALSE, then the device server shall resume processing of the command that caused the logical block encryption parameters for encryption request indicator to be set to TRUE.

The logical block encryption parameters for decryption request indicator settings are specified in table 28.

Table 28 — Logical block encryption parameters for decryption request indicator settings

Setting	Description
TRUE	The device entity is waiting for the logical block encryption parameters for decryption request indicator to be set to FALSE (e.g., an ADC device server processes a SECURITY PROTOCOL OUT command with a DATA ENCRYPTION PARAMETERS COMPLETE page and the clear encryption parameters request (CEPR) bit set to one, see ADC-3) before continuing to process the task in the enabled task state.
FALSE	The device entity is not waiting for the logical block encryption parameters for decryption request indicator to be set to FALSE before continuing to process the task in the enabled task state. This is the default setting for the logical block encryption parameters for decryption request indicator.

The device entity shall not change the logical position while the logical block encryption parameters for decryption request indicator is set to TRUE.

If the logical block encryption parameters for decryption request indicator is set to FALSE, then the device server shall resume processing of the command that caused the logical block encryption parameters for decryption request indicator to be set to TRUE.

The logical block encryption parameters for encryption request indicator and the logical block encryption parameters for decryption request indicator shall be set to defaults:

- a) on a hard reset condition;
- b) if a volume is demounted;
- c) after a logical block encryption parameters request period timeout (see 4.2.30.3.4); or
- d) after successfully processing a task management request that terminates processing of the task.

If the logical block encryption parameters for decryption request indicator is set to FALSE or the logical block encryption parameters for encryption request indicator is set to FALSE, then the logical block encryption period timer shall be set to zero.

4.2.30.3.4 Logical block encryption parameters period settings

The logical block encryption parameters period settings contain values that:

- a) determine how long the device entity waits for the logical block encryption parameters request indicator to be set to FALSE;
- b) track how long the device entity has waited for a set of logical block encryption parameters after a logical block encryption parameters request indicator has been set to TRUE; and
- c) indicate if the time to wait for a set of logical block encryption parameters period has expired.

The logical block encryption parameters period settings (see 4.2.3) shall contain:

- a) a logical block encryption parameters period time;
- b) a logical block encryption period timer; and
- c) a logical block encryption parameters period expired indicator.

The logical block encryption parameters period time shall contain a value indicating the amount of time that the device entity shall wait for a set of logical block encryption parameters if:

- a) the logical block encryption parameters for encryption request indicator (see 4.2.30.3.3) is set to TRUE; or
- b) the logical block encryption parameters for decryption request indicator (see 4.2.30.3.3) is set to TRUE.

The logical block encryption period timer shall contain the time since:

- a) the logical block encryption parameters for encryption request indicator was set to TRUE; or
- b) the logical block encryption parameters for decryption request indicator was set to TRUE.

The logical block encryption period timer shall be set to zero if:

- a) the logical block encryption parameters for encryption request indicator is set to FALSE; or
- b) the logical block encryption parameters for decryption request indicator is set to FALSE.

The values of the logical block encryption period timer expired indicator are specified in table 29.

Table 29 — Logical block encryption period timer expired indicator

Setting	Description
TRUE	The logical block encryption period timer has expired.
FALSE	The logical block encryption period timer has not expired.

If the logical block encryption period timer reaches the logical block encryption period time, then the:

- a) logical block encryption period timer expired indicator shall be set to TRUE;
- b) logical block encryption parameters for encryption request indicator shall be set to FALSE;
- c) logical block encryption parameters for decryption request indicator shall be set to FALSE; and
- d) the device server shall terminate the command that caused a request indicator to be set to TRUE with CHECK CONDITION status, the sense key set to DATA PROTECT, and the additional sense code set to EXTERNAL DATA ENCRYPTION CONTROL TIMEOUT.

4.2.30.4 External data encryption control - logical block encryption parameters exclusive control

An entity outside the scope of this standard may configure the device entity for exclusive control of logical block encryption using external data encryption control. If control of logical block encryption parameters by this device server has been prevented by external data encryption control and the device server returns a Data Encryption Status page (see 8.7.2.7), then the PARAMETERS CONTROL field shall be set to 011b or 100b.

4.2.30.5 External data encryption control error conditions

If external data encryption control is being used to control the logical block encryption parameters and the external data encryption control logical block encryption parameters lookup process returns an error, then the device server shall terminate the command that initiated the logical block encryption parameters lookup process with CHECK CONDITION status, with the sense key set to DATA PROTECT, and the additional sense code set to the value of the external data encryption control additional sense code in the device entity, or to EXTERNAL DATA ENCRYPTION CONTROL ERROR if the external data encryption control additional sense code is set to NO ADDITIONAL SENSE INFORMATION.

An application client may use the DTD Status log page to get information about the error that occurred (see ADC-3).

4.2.31 Logical block encryption key protection

4.2.31.1 Logical block encryption key protection overview

A SCSI device that supports logical block encryption should protect logical block encryption keys from disclosure. The probability of logical block encryption key disclosure may be reduced by several countermeasures (e.g., key wrapping and/or securing the channel used to transmit the logical block encryption key).

4.2.31.2 Logical block encryption key protection using security associations

A security association (SA) (see SPC-4) may be used to protect logical block encryption keys and associated logical block encryption parameters from disclosure and modification. A device server that supports SAs as a way to protect logical block encryption keys may require that all logical block encryption key operations be protected using an SA.

4.2.31.3 Key wrapping using public key cryptography

A device server that supports public key cryptography for key wrapping, shall have a secret private key and a public key. The public key is used for wrapping key material sent to the device server. The private key is used by the device server to unwrap the logical block encryption keys that it receives. The entity wrapping the logical block encryption key is assured that only the device server that knows the private key corresponding to this public key is able to unwrap the logical block encryption key.

Verifying the key wrapper's signature allows a device server that supports public key cryptography for key wrapping to ensure the authenticity of the wrapped key. The key wrapping entity's secret private key is used to sign the wrapped key. The key signing entity's public key is used by the device server to verify the signature.

A device server that supports signature verification shall store the key wrappers public keys in an authorization white list. To prevent an attacker from having the ability to send a wrapped key, the device server shall maintain the authorization white list in a manner that prevents an attacker from modifying the white list.

A device server that supports signature verification should be able to store a minimum of four public keys for signature verification to allow for two key wrapping entities and the ability to replace these keys.

The method of adding signature verification public keys to the authorization white list is outside the scope of this standard.

4.2.31.4 Key wrapping using key manager specific methods

4.2.31.4.1 Overview of key wrapping using key manager specific methods

Some encryption algorithms allow or require the use of one or more WK-KADs. If use of a WK-KAD is required, then the EEDK from one or more WK-KADs shall be stored to the medium with the encrypted logical block(s). One or more WK-KADs are served to the device server with the logical block encryption key. If multiple WK-KADs are used, then the EEDK in each WK-KAD should contain the same logical block encryption key, but may use a different KEK.

4.2.31.4.2 EEDK content

The EEDK is a key storage unit constructed by a key manager. The EEDK is stored to the recording volume and returned to a key manager upon request. The format of the EEDK does not need to be known by the device server. The EEDK should include:

- a) the logical block encryption key protected by a KEK;

- b) identification information (e.g., key label, key manager index, etc.); and
- c) an integrity check.

4.2.31.4.3 Creating wrapped keys

If the device server signals that logical block encryption parameters are needed (see 4.2.30.3.3) and the desired key manager operation attribute (i.e., 0000h) (see 8.7.4.4.2) of the encryption management attributes is set to 0001h (i.e., create attribute), then the key manager (see 3.1.41) uses the logical block encryption key selection criteria attribute and the logical block encryption key wrapping attribute to create the key and wrap it. The key manager creates an EEDK for each KEKS returned using a KEK derived from each KEKS to wrap the logical block encryption key. The key manager sends the logical block encryption key to the device server using the Set Data Encryption page (see 8.7.3.2.1). The EEDK(s) are sent in the WK-KAD descriptor(s).

4.2.31.4.4 Resolving wrapped keys

If the device server signals that logical block encryption parameters are needed (see 4.2.30.3.3) and the desired key manager operation attribute of the encryption management attributes is set to 0002h (i.e., resolve attribute), then the key manager retrieves the EEDK(s) from the device server by requesting the Next Block Encryption Status page (see 8.7.2.8) to get the KAD descriptors and unwrap the logical block encryption key that is contained in one of the EEDKs from the WK-KAD(s). The key manager may not be able to unwrap each EEDK. The key manager unwraps the logical block encryption key from any one of the EEDKs that it is capable of unwrapping.

The key manager then provides the logical block encryption key and the WK-KAD(s) to the device server using the Set Data Encryption page (see 8.7.3.2.1) and the encryption process or decryption process proceeds.

4.2.31.5 Encryption management attributes

4.2.31.5.1 Encryption management attributes overview

Encryption management attributes are a set of attributes used to control how logical block encryption keys are:

- a) selected;
- b) protected;
- c) identified;
- d) created;
- e) modified; and
- f) used.

Encryption management attributes are specified in 8.7.4.4.

The entity that determines the device server's encryption management attributes is called a policy manager. The policy manager may set the encryption management attributes by acting as an application client using the Set Encryption Management Attributes page (see 8.7.3.4) or the policy manager may set encryption management attributes using vendor-specific methods. The device server stores the encryption management attributes for retrieval by a key manager. All encryption management attributes received in a Set Encryption Management Attributes page should be stored by the device server for return in the Get Encryption Management Attributes page (see 8.7.2.9). All encryption management attributes in the encryption management attributes that have a CRIT bit set to one shall be stored by the device server and returned in the Get Encryption Management Attributes page. The currently stored encryption management attributes are entirely replaced by the encryption management attributes provided in the Set Encryption Management Attributes page. Extending or modifying existing encryption management attributes is accomplished by first retrieving the encryption management attributes via the Get Encryption Management Attributes page and then returning the updated encryption management attributes via the Set Encryption Management Attributes page.

In common usage, a policy manager provides encryption management attributes to the device server prior to a recording volume being loaded in the device. The encryption management attributes are then retrieved by a key manager at the beginning of the processing of logical objects to or from the recording volume.

A key manager retrieves the encryption management attributes by using the Get Encryption Management Attributes page (see 8.7.2.9) if the device server signals that logical block encryption parameters are needed by using the logical block encryption parameters request indicators (see 4.2.30.3.3) or if the device server signals the key manager in a vendor-specific manner. The key manager examines the encryption management attributes to determine the desired response.

4.2.31.5.2 Managing encryption management attributes in the device

Encryption management attributes are common to all I_T_L nexus and are stored in volatile memory in the device entity.

A device server that supports key wrapping using key manager specific methods (see 4.2.31.4) shall support at least one of the encryption management attributes.

The device entity shall release the resources used to save a set of encryption management attributes under the following conditions:

- a) the CAOD bit is set to one in the saved encryption management attributes list and the volume is demounted;
- b) a microcode update is performed on the device; or
- c) a hard reset condition occurs.

The device entity may release the resources used to save encryption management attributes on vendor-specific events.

The key manager may be the device server or an application client. If the key manager is the device server, then the device server shall qualify encryption management attributes.

If the device server is qualifying encryption management attributes and the device server receives an encryption management attribute or descriptor in a Set Encryption Management Attributes page that it does not recognize with the critical (CRIT) bit set to one, then the device server shall terminate the command with CHECK CONDITION, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST. If the device server is qualifying encryption management attributes and receives an encryption management attribute or descriptor that is not recognized with the CRIT bit set to zero, then it may bypass processing of the unrecognized encryption management attribute or descriptor and continue processing at the next encryption management attribute (i.e., by using the ENCRYPTION ATTRIBUTES LENGTH field).

If the key manager is an application client, then the encryption management attributes are retrieved using the Get Encryption Management Attributes page. If the application client receives an encryption management attribute or descriptor that it does not recognize, with the critical (CRIT) bit set to one, then the application client should terminate the processing with an error (e.g., in an ADC device, send a Logical Block Encryption Parameters Complete page with the AUTOMATION COMPLETE RESULTS field set to 04h - EXTERNAL LOGICAL BLOCK ENCRYPTION KEY MANAGER ERROR, see ADC-3).

4.2.32 Appending data to a volume containing encrypted logical blocks

A recording volume contains no encrypted logical blocks, all encrypted logical blocks, or a mixture of encrypted logical blocks and unencrypted logical blocks.

A device server that supports encryption should be capable of determining if a mounted volume contains an encrypted logical block. The device server reports its capability of determining if a volume contains an encrypted

logical block using the VCELB_C bit in the logical block encryption algorithm descriptor (see 8.7.2.4). If the device server is capable of determining whether a mounted volume contains an encrypted logical block, it should support a value of one in the VCELBRE bit of the Device Configuration Extension mode page (see 8.5.8).

The device server shall terminate a command with CHECK CONDITION status, with the sense key set to DATA PROTECT, and the additional sense code set to ENCRYPTION PARAMETERS NOT USEABLE if the following are true:

- a) the VCELBRE bit in the Device Configuration Extension mode page (see 8.5.8) is set to one;
- b) the VCELB bit in the Data Encryption status page (see 8.7.2.7) is set to one;
- c) the encryption mode of the set of logical block encryption parameters in use by the I_T_L nexus on which the command arrived is set to DISABLE;
- d) the logical object identifier does not equal zero; and
- e) the command is a:
 - A) WRITE(6);
 - B) WRITE(16);
 - C) WRITE FILEMARKS(6); or
 - D) WRITE FILEMARKS(16).

5 Explicit address command descriptions for sequential-access devices

5.1 Summary of commands for explicit address mode

The explicit address command set for sequential-access devices shall be as shown in table 30. Commands specified as mandatory in table 30 shall be implemented if the explicit address command set is supported.

Refer to table 17 for a description of reservations.

The following operation codes are vendor specific: 02h, 06h, 07h, 09h, 0Ch, 0Dh, and 0Eh.

Table 30 — Explicit address command set for sequential-access devices (part 1 of 5)

Command Name	Type	Operation code ^g	Synchronize operation required ^a	Command type	DCC eligible ^h	Reference
ACCESS CONTROL IN	O	86h	No	G	No	SPC-4
ACCESS CONTROL OUT	O	87h	No	G	No	SPC-4
ALLOW OVERWRITE	O	82h	No	G-A	No	7.1
CHANGE ALIASES	O	A4h/0Bh ^c	No	G	No	SPC-4
ERASE(16)	M	93h	Yes	W-E	Yes	5.2
EXTENDED COPY	O	83h	No	W or R ^b	Yes	SPC-4
FORMAT MEDIUM	O	04h	No	W	Yes	7.2
GENERATE RECOMMENDED ACCESS ORDER	O	A4h/1Dh	Yes	G	No	7.3
INQUIRY	M	12h	No	G-A	No	SPC-4

Key: M = Command implementation is mandatory.

O = Command implementation is optional.

R = Read type command.

W = Write type command.

G = Generic type command.

E = Explicit command.

A = Allowed command while in write capable state.

- a) Refer to 4.2.14.
- b) This command has some specific actions that fall under write type commands and some that fall into read type commands.
- c) This command is defined by a combination of operation code and service action. The operation code value is shown preceding the slash and the service action value is shown after the slash.
- d) Some vendor MANAGEMENT PROTOCOL OUT commands may require a synchronize operation.
- e) Some vendor MANAGEMENT PROTOCOL OUT commands may be a read type or write type command.
- f) The following operation codes are obsolete: 16h, 17h, 18h, 39h, 3Ah, 40h, 56h, 57h, A5h, A7h, B4h, B8h.
- g) A complete summary of operation codes is available at <http://www.t10.org/lists/2op.htm>. The summary includes information about obsolete commands.
- h) Refer to 4.2.18.4.
- i) Some vendor MANAGEMENT PROTOCOL OUT commands may be DCC eligible.

Table 30 — Explicit address command set for sequential-access devices (part 2 of 5)

Command Name	Type	Operation code ^g	Synchronize operation required ^a	Command type	DCC eligible ^h	Reference
LOAD UNLOAD	O	1Bh	Yes	G	Yes	7.4
LOCATE(16)	M	92h	Yes	G-E	Yes	7.5
LOG SELECT	O	4Ch	No	G	No	SPC-4
LOG SENSE	O	4Dh	No	G-A	No	SPC-4
MANAGEMENT PROTOCOL IN	O	A3h/10h ^c	No	G	No	SPC-4
MANAGEMENT PROTOCOL OUT	O	A4h/10h ^c	No ^d	G ^e	No ⁱ	SPC-4
MODE SELECT(10)	O	55h	Yes ^a	W or R ^b	No	SPC-4
MODE SELECT(6)	M	15h	Yes ^a	W or R ^b	No	SPC-4
MODE SENSE(10)	O	5Ah	No	G	No	SPC-4
MODE SENSE(6)	M	1Ah	No	G	No	SPC-4
PERSISTENT RESERVE IN	M	5Eh	No	G	No	SPC-4
PERSISTENT RESERVE OUT	M	5Fh	No	G	No	SPC-4
PREVENT ALLOW MEDIUM REMOVAL	O	1Eh	No	G-A	No	7.6
READ ATTRIBUTE	O	8Ch	No	G	Yes	SPC-4
READ BLOCK LIMITS	M	05h	No	G-A	No	7.7
READ BUFFER	O	3Ch	Yes	G	No	SPC-4
READ DYNAMIC RUNTIME ATTRIBUTE	O	A3h/1Eh	No	G	No	7.8

Key: M = Command implementation is mandatory.

O = Command implementation is optional.

R = Read type command.

W = Write type command.

G = Generic type command.

E = Explicit command.

A = Allowed command while in write capable state.

- a) Refer to 4.2.14.
- b) This command has some specific actions that fall under write type commands and some that fall into read type commands.
- c) This command is defined by a combination of operation code and service action. The operation code value is shown preceding the slash and the service action value is shown after the slash.
- d) Some vendor MANAGEMENT PROTOCOL OUT commands may require a synchronize operation.
- e) Some vendor MANAGEMENT PROTOCOL OUT commands may be a read type or write type command.
- f) The following operation codes are obsolete: 16h, 17h, 18h, 39h, 3Ah, 40h, 56h, 57h, A5h, A7h, B4h, B8h.
- g) A complete summary of operation codes is available at <http://www.t10.org/lists/2op.htm>. The summary includes information about obsolete commands.
- h) Refer to 4.2.18.4.
- i) Some vendor MANAGEMENT PROTOCOL OUT commands may be DCC eligible.

Table 30 — Explicit address command set for sequential-access devices (part 3 of 5)

Command Name	Type	Operation code ^g	Synchronize operation required ^a	Command type	DCC eligible ^h	Reference
READ POSITION	M	34h	No	G-A	No	7.9
READ REVERSE(16)	O	81h	Yes	R-E	Yes	5.4
READ(16)	M	88h	Yes	R-E	Yes	5.3
READ DYNAMIC RUNTIME ATTRIBUTE	O	A3h/1Eh	No	G	No	7.8
RECEIVE COPY RESULTS	O	84h	No	G	Yes	SPC-4
RECEIVE DIAGNOSTIC RESULTS	O	1Ch	No	G	No	SPC-4
RECEIVE RECOMMENDED ACCESS ORDER	O	A4h/1Dh	No	G	No	7.10
RECOVER BUFFERED DATA	O	14h	May	R	Yes	7.11
REPORT ALIASES	O	A3h/0Bh ^c	No	G	No	SPC-4
REPORT DENSITY SUPPORT	M	44h	No	G-A	No	7.12
REPORT IDENTIFYING INFORMATION	O	A3h/05h ^c	No	G	No	SPC-4
REPORT LUNS	M	A0h	No	G-A	No	SPC-4
REPORT PRIORITY	O	A3h/0Eh ^c	No	G	No	SPC-4
REPORT SUPPORTED OPERATION CODES	O	A3h/0Ch ^c	No	G	No	SPC-4
REPORT SUPPORTED TASK MANAGEMENT FUNCTIONS	O	A3h/0Dh ^c	No	G	No	SPC-4
Key: M = Command implementation is mandatory. O = Command implementation is optional. R = Read type command. W = Write type command. G = Generic type command. E = Explicit command. A = Allowed command while in write capable state.						
a)	Refer to 4.2.14.					
b)	This command has some specific actions that fall under write type commands and some that fall into read type commands.					
c)	This command is defined by a combination of operation code and service action. The operation code value is shown preceding the slash and the service action value is shown after the slash.					
d)	Some vendor MANAGEMENT PROTOCOL OUT commands may require a synchronize operation.					
e)	Some vendor MANAGEMENT PROTOCOL OUT commands may be a read type or write type command.					
f)	The following operation codes are obsolete: 16h, 17h, 18h, 39h, 3Ah, 40h, 56h, 57h, A5h, A7h, B4h, B8h.					
g)	A complete summary of operation codes is available at http://www.t10.org/lists/2op.htm . The summary includes information about obsolete commands.					
h)	Refer to 4.2.18.4.					
i)	Some vendor MANAGEMENT PROTOCOL OUT commands may be DCC eligible.					

Table 30 — Explicit address command set for sequential-access devices (part 4 of 5)

Command Name	Type	Operation code ^g	Synchronize operation required ^a	Command type	DCC eligible ^h	Reference
REPORT TARGET PORT GROUPS	O	A3h/0Ah ^c	No	G	No	SPC-4
REPORT TIMESTAMP	M	A3h/0Fh ^c	No	G	No	SPC-4
REQUEST SENSE	M	03h	No	G	No	SPC-4
REWIND	M	01h	Yes	G	Yes	7.13
SECURITY PROTOCOL IN	O	A2h	No	G	No	SPC-4
SECURITY PROTOCOL OUT	O	B5h	No	G	Yes	SPC-4
SEND DIAGNOSTIC	M	1Dh	Yes ^a	W or R ^b	No	SPC-4
SET CAPACITY	O	0Bh	May	W	Yes	7.14
SET IDENTIFYING INFORMATION	O	A4h/06h ^c	No	G	No	SPC-4
SET PRIORITY	O	A4h/0Eh ^c	No	G	No	SPC-4
SET TARGET PORT GROUPS	O	A4h/0Ah ^c	No	G	No	SPC-4
SET TIMESTAMP	M	A4h/0Fh ^c	No	G	No	SPC-4
SPACE(16)	O	91h	May	G-E	Yes	7.15

Key: M = Command implementation is mandatory.

O = Command implementation is optional.

R = Read type command.

W = Write type command.

G = Generic type command.

E = Explicit command.

A = Allowed command while in write capable state.

- a) Refer to 4.2.14.
- b) This command has some specific actions that fall under write type commands and some that fall into read type commands.
- c) This command is defined by a combination of operation code and service action. The operation code value is shown preceding the slash and the service action value is shown after the slash.
- d) Some vendor MANAGEMENT PROTOCOL OUT commands may require a synchronize operation.
- e) Some vendor MANAGEMENT PROTOCOL OUT commands may be a read type or write type command.
- f) The following operation codes are obsolete: 16h, 17h, 18h, 39h, 3Ah, 40h, 56h, 57h, A5h, A7h, B4h, B8h.
- g) A complete summary of operation codes is available at <http://www.t10.org/lists/2op.htm>. The summary includes information about obsolete commands.
- h) Refer to 4.2.18.4.
- i) Some vendor MANAGEMENT PROTOCOL OUT commands may be DCC eligible.

Table 30 — Explicit address command set for sequential-access devices (part 5 of 5)

Command Name	Type	Operation code ^g	Synchronize operation required ^a	Command type	DCC eligible ^h	Reference
TEST UNIT READY	M	00h	No	G	Yes	SPC-4
VERIFY(16)	O	8Fh	Yes	R-E	Yes	5.5
WRITE ATTRIBUTE	O	8Dh	No	G	Yes	SPC-4
WRITE BUFFER	O	3Bh	Yes ^a	G	No	SPC-4
WRITE DYNAMIC RUNTIME ATTRIBUTE	O	A4h/1Eh	No	G	No	7.16
WRITE FILEMARKS(16)	M	80h	May	W-E	Yes	5.7
WRITE(16)	M	8Ah	No	W-E	Yes	5.6
Obsolete ^f						
Reserved		all others				

Key: M = Command implementation is mandatory.

O = Command implementation is optional.

R = Read type command.

W = Write type command.

G = Generic type command.

E = Explicit command.

A = Allowed command while in write capable state.

- a) Refer to 4.2.14.
- b) This command has some specific actions that fall under write type commands and some that fall into read type commands.
- c) This command is defined by a combination of operation code and service action. The operation code value is shown preceding the slash and the service action value is shown after the slash.
- d) Some vendor MANAGEMENT PROTOCOL OUT commands may require a synchronize operation.
- e) Some vendor MANAGEMENT PROTOCOL OUT commands may be a read type or write type command.
- f) The following operation codes are obsolete: 16h, 17h, 18h, 39h, 3Ah, 40h, 56h, 57h, A5h, A7h, B4h, B8h.
- g) A complete summary of operation codes is available at <http://www.t10.org/lists/2op.htm>. The summary includes information about obsolete commands.
- h) Refer to 4.2.18.4.
- i) Some vendor MANAGEMENT PROTOCOL OUT commands may be DCC eligible.

5.2 ERASE(16) command

The ERASE(16) command (see table 31) causes part or all of the partition to be erased beginning at the logical object identifier and partition specified in the command descriptor block. Prior to performing the erase operation, the device server shall perform a synchronize operation (see 4.2.14).

Table 31 — ERASE(16) command

Bit Byte	7	6	5	4	3	2	1	0				
0	OPERATION CODE (93h)											
1	Reserved			FCS	LCS	IMMED	LONG					
2	Reserved		METHOD	Reserved		SMD	VCM					
3	PARTITION											
4	(MSB)											
...	LOGICAL OBJECT IDENTIFIER											
11	(LSB)											
12												
...	Reserved											
14												
15	CONTROL											

See SPC-5 for the definition of the OPERATION CODE field. The OPERATION CODE field shall be set as shown in table 31.

A first command in sequence (FCS) bit set to one specifies this command is the first command in a write sequence. An FCS bit set to zero specifies this command is not the first command in a write sequence.

A last command in sequence (LCS) bit set to one specifies this command is the last command in a write sequence. An LCS bit set to zero specifies this command is not the last command in a write sequence.

An immediate (IMMED) bit set to zero specifies the device server shall not return status until the erase operation has completed. An IMMED bit set to one specifies the device server shall return status prior to performing the erase operation. If CHECK CONDITION status is returned for an ERASE(16) command with an IMMED bit set to one, then the erase operation shall not be performed.

NOTE 15 - If the METHOD field is set to 10b, then application clients should set the IMMED bit to zero to guarantee the operation has completed successfully. If the METHOD field is set to 10b, the duration of the processing may be extended (i.e., may be longer than just setting the LONG bit to one).

A LONG bit set to one specifies all remaining medium in the specified partition beginning at the specified logical object identifier shall be erased using the method specified by the METHOD field value (see table 32). If the format on the medium specifies a recorded indication of EOD (see 3.1.20), the erase operation shall establish an EOD indication at the specified location as part of the erase operation. The logical position following an ERASE(16) command with a LONG bit set to one is not specified by this standard.

NOTE 16 - Some logical units may reject an ERASE(16) command if the logical object identifier is not zero.

A LONG bit set to zero specifies the device server shall perform the action specified by the short erase mode field in the Device Configuration Extension mode page (see 8.5.8) at the logical object identifier and partition specified in

the command. The logical position following a ERASE(16) command with a LONG bit set to zero shall be at the specified logical object identifier and partition.

The METHOD field (see table 32) specifies the erase method that shall be used to erase data. If the LONG bit is set to zero, the METHOD field only applies to data outside the user data area(s).

Table 32 — METHOD field

Code	Description
00b	Vendor specific
01b	The device server shall erase or over-write the partition with a format-specific pattern. Upon successful processing of the command, the recording volume may contain fragments of data specified for erasure. The data specified for erasure shall not be recognizable as valid user data using normal recording volume processing methods.
10b ^a	The device server shall erase or over-write the partition with a format-specific pattern(s). Upon successful processing of the command, the recording volume shall not contain fragments of data specified for erasure.
11b	Reserved
a) The METHOD field set to a value of 10b is intended to support data sanitization (e.g., sanitization as specified in NIST SP 800-88).	

If the security metadata (SMD) bit is set to one, the device server shall alter the security metadata stored on the recording volume with the method specified by the METHOD field. If the SMD bit is set to zero, the device server handling of the security meta-data stored on the recording volume is vendor specific.

If the vendor-specific control metadata (VCM) bit is set to one, the device server shall alter the vendor-specific control metadata stored on the recording volume with the method specified by the METHOD field. If the VCM bit is set to zero, the device server handling of the vendor-specific control metadata stored on the recording volume is vendor specific.

The PARTITION and LOGICAL OBJECT IDENTIFIER fields specify the position at which the ERASE(16) command shall start. If the current position does not match the specified LOGICAL OBJECT IDENTIFIER and PARTITION fields, the device server shall perform a locate operation to the specified logical object identifier and partition prior to performing the erase operation. If the requested location is beyond LEOP, then the command shall be terminated as specified in 4.2.8. If the locate operation fails for any other reason, then the device server shall return CHECK CONDITION status and the additional sense code shall be set to LOCATE OPERATION FAILURE. The logical position is undefined following a locate operation failure with a LONG bit set to zero.

See 4.2.6 for a description of the effect of a PEWZ on the completion of the ERASE(16) command.

5.3 READ(16) command

The READ(16) command (see table 33) requests that the device server transfer one or more logical block(s) to the application client beginning at the logical object identifier and partition specified in the command descriptor block. Prior to performing the read operation, the device server shall perform a synchronize operation (see 4.2.14). See

4.2.28.5.3 for the effects of protection information on the transfer of logical blocks during a READ(16) command and see 4.2.28.3 for the format of a logical block if logical block protection is enabled.

Table 33 — READ(16) command

Bit Byte	7	6	5	4	3	2	1	0		
0	OPERATION CODE (88h)									
1	Reserved					SILI	FIXED			
2	Reserved									
3	PARTITION									
4	(MSB)									
...	LOGICAL OBJECT IDENTIFIER									
11						(LSB)				
12	(MSB)									
...	TRANSFER LENGTH									
14						(LSB)				
15	CONTROL									

See SPC-5 for the definition of the OPERATION CODE field. The OPERATION CODE field shall be set as shown in table 33.

If the suppress incorrect-length indicator (SILI) bit is one and the FIXED bit is zero, the device server shall:

- a) report CHECK CONDITION status for an incorrect-length condition only if the overlength condition exists and the BLOCK LENGTH field in the mode parameter block descriptor is nonzero (see SPC-4); or
- b) not report CHECK CONDITION status if the only error is the underlength condition, or if the only error is the overlength condition and the BLOCK LENGTH field of the mode parameters block descriptor is zero.

NOTE 17 - Since the residual information normally provided in the INFORMATION field of the sense data may not be available if the SILI bit is set to one, other methods for determining the actual block length should be used (e.g., including length information in the data).

If the SILI bit is one and the FIXED bit is one, the device server shall terminate the command with CHECK CONDITION status and the sense key shall be set to ILLEGAL REQUEST with an additional sense code of INVALID FIELD IN CDB.

If the SILI bit is zero and an incorrect-length logical block is read, CHECK CONDITION status shall be returned and the ILI and VALID bits shall be set to one in the sense data with an additional sense code of NO ADDITIONAL SENSE INFORMATION. Upon termination, the logical position shall be after the incorrect-length logical block (i.e., end-of-partition side). If the FIXED bit is one, the INFORMATION field shall be set to the requested transfer length minus the actual number of logical blocks read, not including the incorrect-length logical block. If the FIXED bit is zero, the INFORMATION field shall be set to the requested transfer length minus the actual logical block length. Logical units that do not support negative values shall set the INFORMATION field to zero if the overlength condition exists.

NOTE 18 - In the above case with the FIXED bit set to one, only the position of the incorrect-length logical block may be determined from the sense data. The actual length of the incorrect logical block is not reported. Other means may be used to determine its actual length (e.g., read it again with the fixed bit set to zero).

The FIXED bit specifies whether fixed-block transfers or variable-block transfers are to be used. Refer to the READ BLOCK LIMITS command (see 7.7) for additional information about fixed-block transfers and variable-block transfers.

If the FIXED bit is one, the TRANSFER LENGTH specifies the number of fixed-length blocks to be transferred, using the current block length reported in the mode parameters block descriptor (see SPC-4). If the FIXED bit is zero, a variable-length block is requested with the TRANSFER LENGTH specifying the maximum number of bytes allocated for the returned logical block.

A successful READ(16) command (i.e., the command returns GOOD status or returns a CHECK CONDITION status with the sense key set to RECOVERED ERROR or COMPLETED) with a FIXED bit set to one shall transfer the requested transfer length times the current block length in bytes to the application client. A successful READ(16) command with a FIXED bit set to zero shall transfer the requested transfer length in bytes to the application client. Upon completion, the logical position shall be after the last logical block transferred (end-of-partition side).

The LOGICAL OBJECT IDENTIFIER and PARTITION fields specify the position at which the READ(16) command shall start. If the TRANSFER LENGTH field is not set to zero and the current logical position does not match the specified LOGICAL OBJECT IDENTIFIER and PARTITION fields, the device server shall perform a locate operation to the specified logical object identifier and partition prior to performing the read operation. If the requested location is beyond LEOP, then the command shall be terminated as specified in 4.2.8. If the locate operation fails for any other reason, then the device server shall return CHECK CONDITION status and the additional sense code shall be set to LOCATE OPERATION FAILURE. The INFORMATION field in the sense data shall be set to the requested transfer length. Following a locate operation failure the logical position is undefined.

If the TRANSFER LENGTH field (see SPC-4) is set to zero, no logical block shall be transferred and the current logical position shall not be changed. This condition shall not be considered an error.

In the case of an unrecovered read error, if the FIXED bit is one, the sense data VALID bit shall be set to one and the INFORMATION field shall be set to the requested transfer length minus the actual number of logical blocks read (not including the unrecovered logical blocks). If the FIXED bit is zero, the sense data VALID bit shall be set to one and the INFORMATION field shall be set to the requested transfer length. Upon termination, the logical position shall be after the unrecovered logical block.

If the device server encounters a filemark during a READ(16) command, CHECK CONDITION status shall be returned and the FILEMARK and VALID bits shall be set to one in the sense data. The sense key shall be set to NO SENSE or RECOVERED ERROR, as appropriate, and the additional sense code shall be set to FILEMARK DETECTED. Upon termination, the logical position shall be after the filemark (i.e., end-of-partition side). If the FIXED bit is one, the INFORMATION field shall be set to the requested transfer length minus the actual number of logical blocks read. If the FIXED bit is zero, the INFORMATION field shall be set to the requested transfer length.

If the device server encounters early-warning during a READ(16) command and the REW bit is set to one in the Device Configuration mode page (see 8.5.3), CHECK CONDITION status shall be returned upon completion of the current logical block. The sense key shall be set to NO SENSE or RECOVERED ERROR, as appropriate, and the additional sense code shall be set to END-OF-PARTITION/MEDIUM DETECTED. The EOM and VALID bits shall be set to one in the sense data. Upon termination, the logical position shall be after the last logical block transferred (i.e., end-of-partition side). If the FIXED bit is one, the INFORMATION field in the sense data shall be set to the requested transfer length minus the actual number of logical blocks read. If the FIXED bit is zero, the INFORMATION field in the sense data shall be set to the requested transfer length minus the actual logical block length. The device server shall not return CHECK CONDITION status if early-warning is encountered and the REW bit is set to zero.

NOTE 19 - A REW bit set to one is not recommended for most applications since read data may be present after early-warning.

If the device server encounters end-of-data during a READ(16) command, CHECK CONDITION status shall be returned, the sense key shall be set to BLANK CHECK, and the VALID bit shall be set to one in the sense data. If end-of-data is encountered at or after early-warning, the EOM bit shall also be set to one. Upon termination, the logical position shall be immediately after the last recorded logical object (end-of-partition side). If the FIXED bit is one, the INFORMATION field in the sense data shall be set to the requested transfer length minus the actual number of logical blocks read. If the FIXED bit is zero, the INFORMATION field shall be set to the requested transfer length.

If the device server encounters end-of-partition during a READ(16) command, CHECK CONDITION status shall be returned, the sense key shall be set to MEDIUM ERROR, and the EOM and VALID bits shall be set to one in the sense data. The medium position following this condition is not defined. If the FIXED bit is one, the INFORMATION field shall be set to the requested transfer length minus the actual number of logical blocks read. If the FIXED bit is zero, the INFORMATION field in the sense data shall be set to the requested transfer length.

NOTE 20 - If a READ(16) command terminates with an error condition other than ILLEGAL REQUEST, and no data transfer has occurred, the logical position of the medium is undefined. The application client should issue a READ POSITION(16) command to determine the logical position.

If the device server encounters LEOP, then the command shall be terminated as specified in 4.2.8. The sense data VALID bit shall be set to one. If the FIXED bit is one, the INFORMATION field shall be set to the requested transfer length minus the actual number of logical blocks read. If the FIXED bit is zero, the INFORMATION field in the sense data shall be set to the requested transfer length.

5.4 READ REVERSE(16) command

The READ REVERSE(16) command (see table 34) requests that the device server transfer one or more logical block(s) to the application client beginning at the logical object identifier and partition specified in the command descriptor block. Prior to performing the read reverse operation, the device server shall perform a synchronize operation (see 4.2.14). See 4.2.28.5.3 for the effects of protection information on the transfer of logical blocks during a READ REVERSE(16) command and see 4.2.28.3 for the format of a logical block if logical block protection is enabled.

Table 34 — READ REVERSE(16) command

Bit Byte	7	6	5	4	3	2	1	0							
0	OPERATION CODE (81h)														
1	Reserved				BYTORD	SILI	FIXED								
2	Reserved														
3	PARTITION														
4	(MSB)	LOGICAL OBJECT IDENTIFIER					(LSB)								
...															
11															
12	(MSB)	TRANSFER LENGTH					(LSB)								
...															
14															
15	CONTROL														

This command is similar to the READ(16) command except that medium motion is in the reverse direction. Upon completion of a READ REVERSE(16) command, the logical position shall be before the last logical block transferred (i.e., beginning-of-partition side).

See SPC-5 for the definition of the OPERATION CODE field. The OPERATION CODE field shall be set as shown in table 34.

A byte order (BYTORD) bit set to zero specifies all logical block(s), and the byte(s) within the logical block(s), are transferred in the reverse order. The order of bits within each byte shall not be changed. A BYTORD bit set to one specifies all logical block(s) are transferred in the reverse order but the byte(s) within the logical block(s) are transferred in the same order as returned by the READ(16) command. Support for either value of the BYTORD bit is optional.

Refer to the READ(16) command (see 5.3) for a description of the FIXED bit, the SILI bit, the TRANSFER LENGTH field, and any conditions that may result from incorrect usage of these fields.

The LOGICAL OBJECT IDENTIFIER and PARTITION fields specify the position at which the READ REVERSE(16) command shall start. If the TRANSFER LENGTH field is not set to zero and the current logical position does not match the specified LOGICAL OBJECT IDENTIFIER and PARTITION fields, the device server shall perform a locate operation to the specified logical object identifier and partition prior to performing the read reverse operation. If the requested location is beyond LEOP, then the command shall be terminated as specified in 4.2.8. If the locate operation fails for any other reason, then the device server shall return CHECK CONDITION status and the additional sense code shall be set to LOCATE OPERATION FAILURE. The INFORMATION field in the sense data shall be set to the requested transfer length. Following a locate operation failure, the logical position is undefined.

Filemarks, incorrect-length logical blocks, and unrecovered read errors are handled the same as in the READ(16) command, except that upon termination the logical position shall be before the filemark, incorrect-length logical block, or unrecovered logical block (i.e., beginning-of-partition side).

If the device server encounters beginning-of-partition during a READ REVERSE(16) command, CHECK CONDITION status shall be returned and the EOM and VALID bits shall be set to one in the sense data. The sense key shall be set to NO SENSE or RECOVERED ERROR, as appropriate. If the FIXED bit is one, the INFORMATION field shall be set to the requested transfer length minus the actual number of logical blocks transferred. If the FIXED bit is zero, the INFORMATION field shall be set to the requested transfer length.

5.5 VERIFY(16) command

The VERIFY(16) command (see table 35) requests that the device server verify one or more logical block(s) or logical blocks in one or more logical file(s) (see 4.2.9) beginning at the logical object identifier and partition specified in the command descriptor block. Prior to performing the sequence of verify operation(s), the device server shall perform a synchronize operation (see 4.2.14). See 4.2.28.5 for the effects of protection information on the transfer of logical blocks during a VERIFY(16) command and see 4.2.28.3 for the format of a logical block if logical block protection is enabled. See 4.2.11.2 for a description of verify command processing.

Table 35 — VERIFY(16) command

Bit Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (8Fh)							
1	Reserved	VTE	VLBPM	VBF	IMMED	BYTCMP	FIXED	
2	Reserved							
3	PARTITION							
4	(MSB)	LOGICAL OBJECT IDENTIFIER						
...								(LSB)
11								
12	(MSB)	VERIFICATION LENGTH						
...								
14								(LSB)
15	CONTROL							

See SPC-5 for the definition of the OPERATION CODE field. The OPERATION CODE field shall be set as shown in table 35.

See the VERIFY(6) command (6.7) for a description of the VTE bit, VLBPM bit, VBF bit, IMMED bit, BYTCMP bit, FIXED bit and VERIFICATION LENGTH field.

The LOGICAL OBJECT IDENTIFIER and PARTITION fields specify the position where the VERIFY(16) command shall start. If the VERIFICATION LENGTH field is not zero and the current logical position does not match the specified LOGICAL OBJECT IDENTIFIER and PARTITION fields, the device server shall perform a locate operation to the specified logical object identifier and partition prior to performing the verify operation. If the requested location is beyond LEOP, then the command shall be terminated as specified in 4.2.8. If the locate operation fails for any other reason, then the device server shall return CHECK CONDITION status and the additional sense code shall be set to LOCATE OPERATION FAILURE. The INFORMATION field in the sense data shall be set to the requested verification length. Following a locate operation failure the logical position is undefined.

5.6 WRITE(16) command

The WRITE(16) command (see table 36) requests that the device server write the logical block that is transferred from the application client to the logical object identifier and partition specified in the command descriptor block. See 4.2.28.5 for the effects of protection information on the transfer of logical blocks during a WRITE(16) command and see 4.2.28.3 for the format of a logical block if logical block protection is enabled.

Table 36 — WRITE(16) command

Bit Byte	7	6	5	4	3	2	1	0		
0	OPERATION CODE (8Ah)									
1	Reserved			FCS	LCS	Rsvd	FIXED			
2	Reserved									
3	PARTITION									
4	(MSB)									
...	LOGICAL OBJECT IDENTIFIER									
11										
12	(MSB)									
...	TRANSFER LENGTH									
14										
15	CONTROL									

See SPC-5 for the definition of the OPERATION CODE field. The OPERATION CODE field shall be set as shown in table 36.

A first command in sequence (FCS) bit set to one specifies this command is the first command in a write sequence. An FCS bit set to zero specifies this command is not the first command in a write sequence.

A last command in sequence (LCS) bit set to one specifies this command is the last command in a write sequence. An LCS bit set to zero specifies this command is not the last command in a write sequence.

The FIXED bit specifies whether fixed-block transfers or variable-block transfers are to be used. See the READ BLOCK LIMITS command (see 7.7) for additional information about fixed-block transfers and variable-block transfers.

If the FIXED bit is one, the TRANSFER LENGTH value specifies the number of fixed-length blocks to be transferred, using the current block length reported in the mode parameter block descriptor (see 8.5). If the FIXED bit is zero, a single logical block is transferred with TRANSFER LENGTH specifying the logical block length in bytes.

The LOGICAL OBJECT IDENTIFIER and PARTITION fields specify the position where the WRITE(16) command shall start. If the TRANSFER LENGTH field is not set to zero and the current logical position does not match the specified LOGICAL OBJECT IDENTIFIER and PARTITION fields, the device server shall perform a locate operation to the specified logical object identifier and partition prior to performing the write operation. If the requested location is beyond LEOP, then the command shall be terminated as specified in 4.2.8. If the locate operation fails for any other reason, then the device server shall return CHECK CONDITION status and the additional sense code shall be set to LOCATE OPERATION FAILURE. The INFORMATION field in the sense data shall be set to the requested transfer length. Following a locate operation failure the logical position is undefined.

If the TRANSFER LENGTH field is zero, no data shall be transferred and the current logical position shall not be changed. This condition shall not be considered an error.

A WRITE(16) command may be buffered or unbuffered, as specified by the BUFFERED MODE field of the mode parameter header (see 8.5). If operating in unbuffered mode (see 3.1.88), the device server shall not return GOOD status until all logical block(s) are successfully written to the medium. If operating in buffered mode (see 3.1.9), the device server may return GOOD status as soon as all logical block(s) are successfully transferred to the logical unit's object buffer.

A WRITE FILEMARKS(16) command with the IMMED bit set to zero should be issued to perform a synchronize operation (see 4.2.14) upon completion of buffered write operations.

If the device server enables a WRITE(16) command while positioned between EW and LEOP, or encounters EW during the processing of a WRITE(16) command, an attempt to finish writing any data may be made as determined by the current settings of the REW and SEW bits in the Device Configuration mode page (see 8.5.3). The command shall terminate with CHECK CONDITION status and the additional sense code shall be set to END-OF-PARTITION/MEDIUM DETECTED. If all data that is to be written is successfully transferred to the medium, the sense key shall be set to NO SENSE or RECOVERED ERROR, as appropriate. If the device server is unable to transfer all the data to the medium, buffered or unbuffered, before end-of-partition is encountered, the sense key shall be set to VOLUME OVERFLOW. The EOM bit shall be set to one. If the SEW bit is set to one, then the VALID bit shall be set to one.

The INFORMATION field shall be set as follows:

- a) if the FIXED bit is set to one, the INFORMATION field shall be set to the requested transfer length minus the actual number of logical blocks transferred to the device server; or
- b) if the FIXED bit is set to zero, the INFORMATION field shall be set to the requested transfer length.

If the device server encounters LEOP, then the command shall be terminated as specified in 4.2.8.

The sense data VALID bit shall be set to one and the INFORMATION field shall be set as follows:

- a) if the FIXED bit is set to one, then the INFORMATION field shall be set to the requested transfer length minus the actual number of logical blocks transferred to the device server; or
- b) if the FIXED bit is set to zero, then the INFORMATION field shall be set to the requested transfer length.

The device server should perform a synchronize operation (see 4.2.14) after the first early-warning indication (see 4.2.5) has been returned to the application client.

NOTE 21 - For some application clients it is important to recognize an error if end-of-partition is encountered during the processing of a WRITE(16) command, without regard for whether all data that is to be written is successfully transferred to the medium. The VOLUME OVERFLOW sense key may be returned if end-of-partition is encountered while writing, and such usage is recommended. Reporting the MEDIUM ERROR sense key may cause confusion as to whether there was really defective medium encountered during the processing of the last WRITE(16) command.

If a WRITE(16) command is terminated early, an incomplete logical block (i.e., a logical block not completely transferred to the device server from the initiator) shall be discarded. The incomplete logical block may be accessible until a new logical block is written to the media at the same logical object identifier. The device server shall be logically positioned after the last logical block that was successfully transferred.

See 4.2.6 for a description of the effect of a PEWZ on the completion of the WRITE(16) command.

5.7 WRITE FILEMARKS(16) command

The WRITE FILEMARKS(16) command (see table 37) requests that the device server write the specified number of filemarks to the logical object identifier and partition specified in the command descriptor block.

Table 37 — WRITE FILEMARKS(16) command

Bit Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (80h)							
1	Reserved			FCS	LCS	Obsolete	IMMED	
2	Reserved							
3	PARTITION							
4	(MSB)	LOGICAL OBJECT IDENTIFIER						
11							(LSB)	
12	(MSB)	FILEMARK COUNT						
14							(LSB)	
15		CONTROL						

See SPC-5 for the definition of the OPERATION CODE field. The OPERATION CODE field shall be set as shown in table 37.

A first command in sequence (FCS) bit set to one specifies this command is the first command in a write sequence. An FCS bit set to zero specifies this command is not the first command in a write sequence.

A last command in sequence (LCS) bit set to one specifies this command is the last command in a write sequence. An LCS bit set to zero specifies this command is not the last command in a write sequence.

An immediate (IMMED) bit set to one specifies the device server shall return status as soon as the command descriptor block has been validated. An IMMED bit set to one is only valid if the device is operating in buffered mode (see 3.1.9). If the IMMED bit is set to one and the device is operating in unbuffered mode the command shall be terminated with CHECK CONDITION status. The sense key shall be set to ILLEGAL REQUEST and the additional sense code shall be set to INVALID FIELD IN CDB.

An IMMED bit set to zero specifies the device server shall not return status until the write operation has completed. Any buffered logical objects shall be written to the medium prior to completing the command.

NOTE 22 - Upon completion of any buffered write operation, the application client may issue a WRITE FILEMARKS(16) command with the IMMED bit set to zero and the FILEMARK COUNT field set to zero to perform a synchronize operation (see 4.2.14).

The LOGICAL OBJECT IDENTIFIER and PARTITION fields specify the position where the WRITE FILEMARKS(16) command shall start. If the FILEMARK COUNT field is not set to zero and the current logical position does not match the specified LOGICAL OBJECT IDENTIFIER and PARTITION fields, the device server shall perform a locate operation to the specified logical object identifier and partition prior to performing the write filemarks operation. If the requested location is beyond LEOP, then the command shall be terminated as specified in 4.2.8. If the locate operation fails for any other reason, then the device server shall return CHECK CONDITION status and the additional sense code shall be set to LOCATE OPERATION FAILURE. The INFORMATION field in the sense data shall be set to the requested filemark count. Following a locate operation failure the logical position is undefined.

The FILEMARK COUNT field specifies the number of filemarks to be written. If the FILEMARK COUNT field is set to zero, the current logical position shall not be changed. It shall not be considered an error if the FILEMARK COUNT field is set to zero.

If the device server enables a WRITE FILEMARKS(16) command while positioned between EW and EOP, or encounters EW during the processing of a WRITE FILEMARKS(16) command, an attempt to finish writing any buffered logical objects may be made, as determined by the current settings of the REW and SEW bits in the Device Configuration mode page (see 8.5.3). The command shall terminate with CHECK CONDITION status and the additional sense code shall be set to END-OF-PARTITION/MEDIUM DETECTED. If all buffered logical objects to be written are successfully transferred to the medium, the sense key shall be set to NO SENSE or RECOVERED ERROR, as appropriate. If the device server is unable to transfer all the buffered logical objects to the medium before end-of-partition is encountered, the sense key shall be set to VOLUME OVERFLOW. The EOM bit shall be set to one. If the SEW bit is set to one, then the VALID bit shall be set to one.

The INFORMATION field shall be set to the FILEMARK COUNT field value minus the actual number of filemarks:

- a) written to the object buffer if the IMMED bit is set to one; or
- b) written to the medium if the IMMED bit is set to zero.

The device server should perform a synchronize operation (see 4.2.14) after the first early-warning indication (see 4.2.5) has been returned to the application client.

See 4.2.6 for a description of the effect of a PEWZ on the completion of the WRITE FILEMARKS(16) command.

If the device server encounters LEOP, then the command shall be terminated as specified in 4.2.8.

The sense data VALID bit shall be set to one and the INFORMATION field shall be set to the FILEMARK COUNT field value minus the actual number of filemarks:

- a) written to the object buffer if the IMMED bit is set to one; or
- b) written to the medium if the IMMED bit is set to zero.

6 Implicit address command descriptions for sequential-access devices

6.1 Summary of commands for implicit address mode

The implicit address commands for sequential-access devices are shown in table 38. Commands specified as mandatory in table 38 shall be implemented if the implicit address command set is supported.

If a synchronize operation is required for a command, the synchronize operation shall be performed as specified in 4.2.14.

Refer to table 17 for a description of reservations.

The following operation codes are vendor specific: 02h, 06h, 07h, 09h, 0Ch, 0Dh, and 0Eh.

Table 38 — Implicit address command set for sequential-access devices (part 1 of 4)

Command name	Type	Operation code ^e	Synchronize operation required ^a	DCC eligible ^f	Reference
ACCESS CONTROL IN	O	86h	No	No	SPC-4
ACCESS CONTROL OUT	O	87h	No	No	SPC-4
ALLOW OVERWRITE	O	82h	No	No	7.1
CHANGE ALIASES	O	A4h/0Bh ^b	No	No	SPC-4
ERASE(6)	M	19h	Yes	Yes	6.2
EXTENDED COPY	O	83h	No	Yes	SPC-4
FORMAT MEDIUM	O	04h	No	Yes	7.2
GENERATE RECOMMENDED ACCESS ORDER	O	A4h/1Dh	Yes	G	7.3
INQUIRY	M	12h	No	No	SPC-4
LOAD UNLOAD	O	1Bh	Yes	Yes	7.4
LOCATE(10)	M	2Bh	Yes	Yes	6.3
LOCATE(16)	M	92h	Yes	Yes	7.5
LOG SELECT	O	4Ch	No	No	SPC-4
Key: M = Command implementation is mandatory. O = Command implementation is optional.					
a) Refer to 4.2.14. b) This command is defined by a combination of operation code and service action. The operation code value is shown preceding the slash and the service action value is shown after the slash. c) Some vendor MANAGEMENT PROTOCOL OUT commands may require a synchronize operation. d) The following operation codes are obsolete: 16h, 17h, 18h, 39h, 3Ah, 40h, 56h, 57h, A5h, A7h, B4h, B8h. e) A complete summary of operation codes is available at http://www.t10.org/lists/2op.htm . The summary includes information about obsolete commands. f) Refer to 4.2.18.4. g) Some vendor MANAGEMENT PROTOCOL OUT commands may be DCC eligible.					

Table 38 — Implicit address command set for sequential-access devices (part 2 of 4)

Command name	Type	Operation code ^e	Synchronize operation required ^a	DCC eligible ^f	Reference
LOG SENSE	O	4Dh	No	No	SPC-4
MANAGEMENT PROTOCOL IN	O	A3h/10h ^b	No	No	SPC-4
MANAGEMENT PROTOCOL OUT	O	A4h/10h ^b	No ^c	No ^g	SPC-4
MODE SELECT(10)	O	55h	Yes ^a	No	SPC-4
MODE SELECT(6)	M	15h	Yes ^a	No	SPC-4
MODE SENSE(10)	O	5Ah	No	No	SPC-4
MODE SENSE(6)	M	1Ah	No	No	SPC-4
PERSISTENT RESERVE IN	M	5Eh	No	No	SPC-4
PERSISTENT RESERVE OUT	M	5Fh	No	No	SPC-4
PREVENT ALLOW MEDIUM REMOVAL	O	1Eh	No	No	7.6
READ ATTRIBUTE	O	8Ch	No	Yes	SPC-4
READ BLOCK LIMITS	M	05h	No	No	7.7
READ BUFFER	O	3Ch	Yes	No	SPC-4
READ DYNAMIC RUNTIME ATTRIBUTE	O	A3h/1Eh	No	No	7.8
READ POSITION	M	34h	No	No	7.9
READ REVERSE(6)	O	0Fh	Yes	Yes	6.5
READ(6)	M	08h	Yes	Yes	6.4
RECEIVE COPY RESULTS	O	84h	No	Yes	SPC-4
RECEIVE DIAGNOSTIC RESULTS	O	1Ch	No	No	SPC-4
RECEIVE RECOMMENDED ACCESS ORDER	O	A4h/1Dh	No	G	7.10
RECOVER BUFFERED DATA	O	14h	May	Yes	7.11
REPORT ALIASES	O	A3h/0Bh ^b	No	No	SPC-4
REPORT DENSITY SUPPORT	M	44h	No	No	7.12

Key: M = Command implementation is mandatory.
O = Command implementation is optional.

a) Refer to 4.2.14.
 b) This command is defined by a combination of operation code and service action. The operation code value is shown preceding the slash and the service action value is shown after the slash.
 c) Some vendor MANAGEMENT PROTOCOL OUT commands may require a synchronize operation.
 d) The following operation codes are obsolete: 16h, 17h, 18h, 39h, 3Ah, 40h, 56h, 57h, A5h, A7h, B4h, B8h.
 e) A complete summary of operation codes is available at <http://www.t10.org/lists/2op.htm>. The summary includes information about obsolete commands.
 f) Refer to 4.2.18.4.
 g) Some vendor MANAGEMENT PROTOCOL OUT commands may be DCC eligible.

Table 38 — Implicit address command set for sequential-access devices (part 3 of 4)

Command name	Type	Operation code ^e	Synchronize operation required ^a	DCC eligible ^f	Reference
REPORT IDENTIFYING INFORMATION	O	A3h/05h ^b	No	No	SPC-4
REPORT LUNS	M	A0h	No	No	SPC-4
REPORT PRIORITY	O	A3h/0Eh ^b	No	No	SPC-4
REPORT SUPPORTED OPERATION CODES	O	A3h/0Ch ^b	No	No	SPC-4
REPORT SUPPORTED TASK MANAGEMENT FUNCTIONS	O	A3h/0Dh ^b	No	No	SPC-4
REPORT TARGET PORT GROUPS	O	A3h/0Ah ^b	No	No	SPC-4
REPORT TIMESTAMP	M	A3h/0Fh ^b	No	No	SPC-4
REQUEST SENSE	M	03h	No	No	SPC-4
REWIND	M	01h	Yes	Yes	7.13
SECURITY PROTOCOL IN	O	A2h	No	No	SPC-4
SECURITY PROTOCOL OUT	O	B5h	No	Yes	SPC-4
SEND DIAGNOSTIC	M	1Dh	Yes ^a	No	SPC-4
SET CAPACITY	O	0Bh	May	Yes	7.14
SET IDENTIFYING INFORMATION	O	A4h/06h ^b	No	No	SPC-4
SET PRIORITY	O	A4h/0Eh ^b	No	No	SPC-4
SET TARGET PORT GROUPS	O	A4h/0Ah ^b	No	No	SPC-4
SET TIMESTAMP	M	A4h/0Fh ^b	No	No	SPC-4
SPACE(16)	O	91h	May	Yes	7.15
SPACE(6)	M	11h	May	Yes	6.6
TEST UNIT READY	M	00h	No	Yes	SPC-4
VERIFY(6)	O	13h	Yes	Yes	6.7
WRITE ATTRIBUTE	O	8Dh	No	Yes	SPC-4
WRITE BUFFER	O	3Bh	Yes ^a	No	SPC-4
Key: M = Command implementation is mandatory. O = Command implementation is optional.					
a) Refer to 4.2.14. b) This command is defined by a combination of operation code and service action. The operation code value is shown preceding the slash and the service action value is shown after the slash. c) Some vendor MANAGEMENT PROTOCOL OUT commands may require a synchronize operation. d) The following operation codes are obsolete: 16h, 17h, 18h, 39h, 3Ah, 40h, 56h, 57h, A5h, A7h, B4h, B8h. e) A complete summary of operation codes is available at http://www.t10.org/lists/2op.htm . The summary includes information about obsolete commands. f) Refer to 4.2.18.4. g) Some vendor MANAGEMENT PROTOCOL OUT commands may be DCC eligible.					

Table 38 — Implicit address command set for sequential-access devices (part 4 of 4)

Command name	Type	Operation code ^e	Synchronize operation required ^a	DCC eligible ^f	Reference
WRITE DYNAMIC RUNTIME ATTRIBUTE	O	A4h/1Eh	No	No	7.16
WRITE FILEMARKS(6)	M	10h	May	Yes	6.9
WRITE(6)	M	0Ah	No	Yes	6.8
Obsolete ^d					
Reserved		all others			

Key: M = Command implementation is mandatory.
O = Command implementation is optional.

a) Refer to 4.2.14.
 b) This command is defined by a combination of operation code and service action. The operation code value is shown preceding the slash and the service action value is shown after the slash.
 c) Some vendor MANAGEMENT PROTOCOL OUT commands may require a synchronize operation.
 d) The following operation codes are obsolete: 16h, 17h, 18h, 39h, 3Ah, 40h, 56h, 57h, A5h, A7h, B4h, B8h.
 e) A complete summary of operation codes is available at <http://www.t10.org/lists/2op.htm>. The summary includes information about obsolete commands.
 f) Refer to 4.2.18.4.
 g) Some vendor MANAGEMENT PROTOCOL OUT commands may be DCC eligible.

6.2 ERASE(6) command

The ERASE(6) command (see table 39) causes part or all of the recording volume to be erased beginning at the current position. Prior to performing the erase operation, the device server shall perform a synchronize operation (see 4.2.14).

Table 39 — ERASE(6) command

Bit Byte	7	6	5	4	3	2	1	0				
0	OPERATION CODE (19h)											
1	Reserved					IMMED		LONG				
2	Reserved		METHOD		Reserved		SMD	VCM				
3	Reserved											
4	Reserved											
5	CONTROL											

See SPC-5 for the definition of the OPERATION CODE field. The OPERATION CODE field shall be set as shown in table 39.

An immediate (IMMED) bit set to zero specifies the device server shall not return status until the erase operation has completed. An IMMED bit set to one specifies the device server shall return status prior to performing the erase operation. If CHECK CONDITION status is returned for an ERASE(6) command with an IMMED bit set to one, then the erase operation shall not be performed.

NOTE 23 - If the METHOD field is set to 10b, then application clients should set the IMMED bit set to zero to guarantee the operation has completed successfully. If the METHOD field is set to 10b, the duration of the processing may be extended (i.e., may be longer than just setting the LONG bit to one).

A LONG bit set to one specifies all remaining medium in the current partition beginning at the current logical position shall be erased using the method specified by the METHOD field value (see table 32). If the format on the medium specifies a recorded indication of EOD (see 3.1.20), the erase operation shall establish an EOD indication at the current logical position as part of the erase operation. The logical position following an ERASE(6) command with a LONG bit set to one is not specified by this standard.

NOTE 24 - Some logical units may reject an ERASE(6) command if the logical unit is not at beginning-of-partition.

A LONG bit set to zero specifies the device server shall perform the action specified by the SHORT ERASE MODE field in the Device Configuration Extension mode page (see 8.5.8) at the current logical position. If the IMMED bit is one, the device server shall return status as soon as the command descriptor block has been validated.

The METHOD field, security metadata (SMD) bit, and the vendor-specific control metadata (VCM) bit are defined in 5.2.

See 4.2.6 for a description of the effect of a PEWZ on the completion of the ERASE(6) command.

6.3 LOCATE(10) command

The LOCATE(10) command (see table 40) causes the logical unit to position the medium to the specified logical object with a matching logical object identifier in the specified partition. Upon completion, the logical position shall be before the specified logical object. Prior to performing the locate operation, the device server shall perform a synchronize operation (see 4.2.14).

Table 40 — LOCATE(10) command

Bit Byte	7	6	5	4	3	2	1	0		
0	OPERATION CODE (2Bh)									
1	Reserved				BT		CP	IMMED		
2	Reserved									
3	(MSB)									
...	LOGICAL OBJECT IDENTIFIER									
6	(LSB)									
7	Reserved									
8	PARTITION									
9	CONTROL									

See SPC-5 for the definition of the OPERATION CODE field. The OPERATION CODE field shall be set as shown in table 40.

A block identifier type (BT) bit set to one specifies the value in the LOGICAL OBJECT IDENTIFIER field shall be interpreted as a vendor-specific value. A BT bit set to zero specifies the value in the LOGICAL OBJECT IDENTIFIER field shall be interpreted as a logical object identifier (see 3.1.49).

A change partition (CP) bit set to one specifies a change to the partition specified in the PARTITION field shall occur prior to positioning to the logical object specified in the LOGICAL OBJECT IDENTIFIER field. A CP bit set to zero specifies no partition change shall occur and the PARTITION field shall be ignored.

An immediate (IMMED) bit set to zero specifies the device server shall not return status until the locate operation has completed. If the IMMED bit is one, the device server shall return status as soon as all buffered logical objects have been written to the medium and the command descriptor block of the LOCATE(10) command has been validated. If CHECK CONDITION status is returned for a LOCATE(10) command with an IMMED bit set to one, the locate operation shall not be performed.

The LOGICAL OBJECT IDENTIFIER field specifies the logical object identifier to which the logical unit shall position the medium based on the current setting of the BT bit. If the requested location is beyond LEOP, then the command shall be terminated as specified in 4.2.8. An otherwise valid LOCATE(10) command to any position between beginning-of-data and the position immediately after the last logical block in the partition (i.e., position at end-of-data) shall not return a sense key of ILLEGAL REQUEST. A LOCATE(10) to a position past end-of-data shall return CHECK CONDITION status and the sense key shall be set to BLANK CHECK. Additionally, the sense data EOM bit shall be set to one if end-of-data is located at or after early-warning.

If end-of-partition is encountered, CHECK CONDITION status shall be returned, the sense key shall be set to MEDIUM ERROR, and the sense data EOM bit shall be set to one.

The PARTITION field specifies the partition to select if the CP bit is one. Refer to the sequential-access device model (see 4.2.7) and the Medium Partition mode page (see 8.5.4) for additional information about partitioning.

The logical unit position is undefined if a LOCATE(10) command fails with a sense key other than ILLEGAL REQUEST.

6.4 READ(6) command

The READ(6) command (see table 41) requests that the device server transfer one or more logical block(s) to the application client beginning with the next logical block. Prior to performing the read operation, the device server shall perform a synchronize operation (see 4.2.14). See 4.2.28.5.3 for the effects of protection information on the transfer of logical blocks during a READ(6) command and see 4.2.28.3 for the format of a logical block if logical block protection is enabled.

Table 41 — READ(6) command

Bit Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (08h)							
1	Reserved						SILI	FIXED
2	(MSB)							
...	TRANSFER LENGTH							
4	(LSB)							
5	CONTROL							

See SPC-5 for the definition of the OPERATION CODE field. The OPERATION CODE field shall be set as shown in table 41.

If the suppress incorrect-length indicator (SILI) bit is one and the FIXED bit is zero, the device server shall:

- a) report CHECK CONDITION status for an incorrect-length condition only if the overlength condition exists and the BLOCK LENGTH field in the mode parameter block descriptor is nonzero (see 8.5); or
- b) not report CHECK CONDITION status if the only error is the underlength condition, or if the only error is the overlength condition and the BLOCK LENGTH field of the mode parameters block descriptor is zero.

NOTE 25 - Since the residual information normally provided in the INFORMATION field of the sense data may not be available if the SILI bit is set, other methods for determining the actual block length should be used (e.g., including length information in the logical block).

If the SILI bit is one and the FIXED bit is one, the device server shall terminate the command with CHECK CONDITION status. The sense key shall be set to ILLEGAL REQUEST and the additional sense code shall be set to INVALID FIELD IN CDB.

If the SILI bit is zero and an incorrect-length logical block is read, CHECK CONDITION status shall be returned. The ILI and VALID bits shall be set to one in the sense data and the additional sense code shall be set to NO ADDITIONAL SENSE INFORMATION. Upon termination, the logical position shall be after the incorrect-length logical block (i.e., end-of-partition side). If the FIXED bit is one, the INFORMATION field shall be set to the requested transfer length minus the actual number of logical blocks read, not including the incorrect-length logical block. If the FIXED bit is zero, the INFORMATION field shall be set to the requested transfer length minus the actual logical block

length. Logical units that do not support negative values shall set the INFORMATION field to zero if the overlength condition exists.

NOTE 26 - In the above case with the FIXED bit set to one, only the position of the incorrect-length logical block may be determined from the sense data. The actual length of the incorrect logical block is not reported. Other means may be used to determine its actual length (e.g., read it again with the fixed bit set to zero).

The FIXED bit specifies whether fixed-block transfers or variable-block transfers are to be used. Refer to the READ BLOCK LIMITS command (see 7.7) for additional information about fixed-block transfers and variable-block transfers.

If the FIXED bit is one, the TRANSFER LENGTH specifies the number of fixed-length blocks to be transferred, using the current block length reported in the mode parameters block descriptor (see 8.5). If the FIXED bit is zero, a variable-length block is requested with the TRANSFER LENGTH specifying the maximum number of bytes allocated for the returned logical block.

A successful READ(6) command (i.e., the command returns GOOD status or returns a CHECK CONDITION status with the sense key set to RECOVERED ERROR or COMPLETED) with a FIXED bit set to one shall transfer the requested transfer length times the current block length in bytes to the application client. A successful READ(6) command with a FIXED bit set to zero shall transfer the requested transfer length in bytes to the application client. Upon completion, the logical position shall be after the last logical block transferred (i.e., end-of-partition side).

A TRANSFER LENGTH of zero specifies no logical block shall be transferred, and the logical position shall not be changed. This condition shall not be considered an error.

In the case of an unrecovered read error, if the FIXED bit is one, the sense data VALID bit shall be set to one and the INFORMATION field shall be set to the requested transfer length minus the actual number of logical blocks read, not including the unrecovered logical blocks. If the FIXED bit is zero, the sense data VALID bit shall be set to one and the INFORMATION field shall be set to the requested transfer length. Upon termination, the logical position shall be after the unrecovered logical block.

If the device server encounters a filemark during a READ(6) command, CHECK CONDITION status shall be returned and the FILEMARK and VALID bits shall be set to one in the sense data. The sense key shall be set to NO SENSE or RECOVERED ERROR, as appropriate, and the additional sense code shall be set to FILEMARK DETECTED. Upon termination, the logical position shall be after the filemark (i.e., end-of-partition side). If the FIXED bit is one, the INFORMATION field shall be set to the requested transfer length minus the actual number of logical blocks read. If the FIXED bit is zero, the INFORMATION field shall be set to the requested transfer length.

If the device server encounters early-warning during a READ(6) command and the REW bit is set to one in the Device Configuration mode page (see 8.5.3), CHECK CONDITION status shall be returned upon completion of the current logical block. The sense key shall be set to NO SENSE or RECOVERED ERROR, as appropriate, and the additional sense code shall be set to END-OF-PARTITION/MEDIUM DETECTED. The EOM and VALID bits shall be set to one in the sense data. Upon termination, the logical position shall be after the last logical block transferred (i.e., end-of-partition side). If the FIXED bit is one, the INFORMATION field in the sense data shall be set to the requested transfer length minus the actual number of logical blocks read. If the FIXED bit is zero, the INFORMATION field in the sense data shall be set to the requested transfer length minus the actual logical block length. The device server shall not return CHECK CONDITION status if early-warning is encountered and the REW bit is set to zero.

NOTE 27 - A REW bit set to one is not recommended for most applications since read data may be present after early-warning.

If the device server encounters end-of-data during a READ(6) command, CHECK CONDITION status shall be returned, the sense key shall be set to BLANK CHECK, and the VALID bit shall be set to one in the sense data. If end-of-data is encountered at or after early-warning, the EOM bit shall also be set to one. Upon termination, the

logical position shall be immediately after the last recorded logical object (i.e., end-of-partition side). If the FIXED bit is one, the INFORMATION field in the sense data shall be set to the requested transfer length minus the actual number of logical blocks read. If the FIXED bit is zero, the INFORMATION field shall be set to the requested transfer length.

If the device server encounters end-of-partition during a READ(6) command, CHECK CONDITION status shall be returned, the sense key shall be set to MEDIUM ERROR, and the EOM and VALID bits shall be set to one in the sense data. The medium position following this condition is not defined. If the FIXED bit is one, the INFORMATION field shall be set to the requested transfer length minus the actual number of logical blocks read. If the FIXED bit is zero, the INFORMATION field in the sense data shall be set to the requested transfer length.

If the device server encounters LEOP, then the command shall be terminated as specified in 4.2.8. The sense data VALID bit shall be set to one. If the FIXED bit is one, the INFORMATION field shall be set to the requested transfer length minus the actual number of logical blocks read. If the FIXED bit is zero, the INFORMATION field in the sense data shall be set to the requested transfer length.

6.5 READ REVERSE(6) command

The READ REVERSE(6) command (see table 42) requests that the device server transfer one or more logical block(s) to the application client beginning at the current logical position. Prior to performing the read reverse operation, the device server shall perform a synchronize operation (see 4.2.14). See 4.2.28.5.3 for the effects of protection information on the transfer of logical blocks during a READ REVERSE(6) command and see 4.2.28.3 for the format of a logical block if logical block protection is enabled.

Table 42 — READ REVERSE(6) command

Bit Byte	7	6	5	4	3	2	1	0				
0	OPERATION CODE (0Fh)											
1	Reserved			BYTORD		SILI	FIXED					
2	(MSB)											
...	TRANSFER LENGTH											
4	(LSB)											
5	CONTROL											

This command is similar to the READ(6) command except that medium motion is in the reverse direction. Upon completion of a READ REVERSE(6) command, the logical position shall be before the last logical block transferred (i.e., beginning-of-partition side).

See SPC-5 for the definition of the OPERATION CODE field. The OPERATION CODE field shall be set as shown in table 42.

A byte order (BYTORD) bit set to zero specifies all logical block(s), and the byte(s) within the logical block(s), are transferred in the reverse order. The order of bits within each byte shall not be changed. A BYTORD bit set to one specifies all logical block(s) are transferred in the reverse order but the byte(s) within the logical block(s) are transferred in the same order as returned by the READ(6) command. Support for either value of the BYTORD bit is optional.

Refer to the READ(6) command (see 6.4) for a description of the FIXED bit, the SILI bit, the TRANSFER LENGTH field, and any conditions that may result from incorrect usage of these fields.

Filemarks, incorrect-length logical blocks, and unrecovered read errors are handled the same as in the READ(6) command, except that upon termination the logical position shall be before the filemark, incorrect-length logical block, or unrecovered block (i.e., beginning-of-partition side).

If the device server encounters beginning-of-partition during a READ REVERSE(6) command, CHECK CONDITION status shall be returned and the EOM and VALID bits shall be set to one in the sense data. The sense key shall be set to NO SENSE or RECOVERED ERROR, as appropriate. If the FIXED bit is one, the INFORMATION field shall be set to the requested transfer length minus the actual number of logical blocks transferred. If the FIXED bit is zero, the INFORMATION field shall be set to the requested transfer length.

6.6 SPACE(6) command

The SPACE(6) command (see table 43) provides a variety of positioning functions that are determined by the CODE and COUNT fields. Both forward and reverse positioning are provided, although some logical units may only support a subset of this command. Prior to performing the space operation, except as stated in the description of the COUNT field, the device server shall perform a synchronize operation (see 4.2.14). If an application client requests an unsupported function, the command shall be terminated with CHECK CONDITION status. The sense key shall be set to ILLEGAL REQUEST and the additional sense code shall be set to INVALID FIELD IN CDB.

If a SPACE(6) command is terminated, then the INFORMATION field value shall be equal to the magnitude of the COUNT field minus the magnitude of the logical objects spaced over. A CHECK CONDITION caused by early termination of any SPACE(6) command shall not result in a negative INFORMATION field value.

Table 43 — SPACE(6) command

Bit Byte	7	6	5	4	3	2	1	0				
0	OPERATION CODE (11h)											
1	Reserved				CODE							
2	(MSB)											
...												
4												
5	COUNT (LSB)											
	CONTROL											

See SPC-5 for the definition of the OPERATION CODE field. The OPERATION CODE field shall be set as shown in table 43.

The CODE field (see table 44) specifies either the type of logical objects to be spaced over or the termination point of the operation.

Table 44 — CODE field

Code	Description	Support
0000b	Logical blocks	M
0001b	Filemarks	M
0010b	Sequential filemarks	O
0011b	End-of-data	O
0100b	Obsolete	

Table 44 — CODE field

Code	Description	Support
0101b	Obsolete	
0110b to 1111b	Reserved	

The COUNT field contains a value in two's complement notation specifying the number of logical objects (e.g., logical blocks if the CODE field is set to 0000b or filemarks if the CODE field is set to 0001b) to be spaced over and the direction of movement. If the CODE field is set to 0011b (i.e., end-of-data), then the COUNT field shall be ignored.

If spacing over logical objects, the magnitude of the value in the COUNT field specifies the number of logical objects to be spaced over in the current partition and the sign of the value in the COUNT field specifies the direction of movement.

If the COUNT field contains a positive value N and the CODE field does not contain 0011b (i.e., end-of-data), then the device server shall position in the forward direction (i.e., toward end-of-partition) over N logical objects ending on the end-of-partition side of the last logical object. If the device server reaches EOP before positioning over N logical objects, then it the device server shall end movement at EOP.

If the COUNT field contains a zero value and the CODE field does not contain 0011b (i.e., end-of-data), then the device server shall not change logical position.

If the COUNT field contains a negative value $-N$, in two's complement notation, and the CODE field does not contain 0011b (i.e., end-of-data) then the device server shall position in the reverse direction (i.e., toward beginning-of-partition) over N logical objects ending on the beginning-of-partition side of the last logical object. If the device server reaches beginning-of-partition before positioning over N logical objects, then it shall end movement at beginning-of-partition.

If the CODE field is 0011b (i.e., end-of-data), then the COUNT field shall be ignored and the device server shall perform a synchronize operation prior to positioning before EOD. If the COUNT field is zero and the CODE field is not 0011b (i.e., end-of-data), then a device server is not required to perform a synchronize operation. Support of spacing in the reverse direction is optional.

If a filemark is encountered while spacing over logical blocks, the command shall be terminated. CHECK CONDITION status shall be returned, and the FILEMARK and VALID bits shall be set to one in the sense data. The sense key shall be set to NO SENSE and the additional sense code shall be set to FILEMARK DETECTED. The INFORMATION field shall be set to the number of logical objects to be spaced over minus the actual number of logical objects spaced over not including the filemark. The logical position shall be on the end-of-partition side of the filemark if movement was in the forward direction and on the beginning-of-partition side of the filemark if movement was in the reverse direction.

If EW is encountered while spacing over logical objects and the REW bit is set to one in the Device Configuration mode page (see 8.5.3), CHECK CONDITION status shall be returned, the sense key shall be set to NO SENSE, and the EOM and VALID bits shall be set to one in the sense data. The additional sense code shall be set to END-OF-PARTITION/MEDIUM DETECTED. The INFORMATION field shall be set to the requested count minus the actual number of logical objects spaced over as defined by the CODE value. If the REW bit is zero, the device server shall not report CHECK CONDITION status at the early-warning point.

NOTE 28 - Setting the REW bit to one is not recommended for most applications since data may be present after early-warning.

If EOD is encountered while spacing over logical objects, CHECK CONDITION status shall be returned, the sense key shall be set to BLANK CHECK, and the sense data VALID bit shall be set to one in the sense data. The

additional sense code shall be set to END-OF-DATA DETECTED. The sense data EOM bit shall be set to one if end-of-data is encountered at or after early-warning. The INFORMATION field shall be set to the requested count minus the actual number of logical objects spaced over as defined by the CODE value. The medium shall be positioned after the last logical object.

If the EOP is encountered while spacing forward over logical objects, CHECK CONDITION status shall be returned, and the sense key shall be set to MEDIUM ERROR. The additional sense code shall be set to END-OF-PARTITION/MEDIUM DETECTED, and the sense data EOM and VALID bit shall be set to one. The INFORMATION field shall be set to the requested count minus the actual number of logical objects spaced over as defined by the CODE value. The medium position following this condition is not defined.

If LEOP is encountered while spacing forward over logical objects, then the command shall be terminated as specified in 4.2.8. The sense data VALID bit shall be set to one and the INFORMATION field shall be set to the requested count minus the actual number of logical objects spaced over as defined by the CODE field value.

If BOP is encountered while spacing over logical objects in the reverse direction, the device server shall return CHECK CONDITION status and shall set the sense key to NO SENSE. The additional sense code shall be set to BEGINNING-OF-PARTITION/MEDIUM DETECTED. The sense data EOM and VALID bits shall be set to one, and the INFORMATION field set to the total number of logical objects not spaced over as defined by the CODE value (i.e., the requested number of logical objects minus the actual number of logical objects spaced over as defined by the CODE value). The medium position following this condition is not defined. A successfully completed SPACE(6) command shall not set EOM to one at beginning-of-partition.

If spacing over sequential filemarks, the COUNT field is interpreted as follows:

- a) a positive value N shall cause forward movement to the first occurrence of N or more consecutive filemarks being logically positioned after the N^{th} filemark;
- b) a zero value shall cause no change in the logical position; or
- c) a negative value $-N$, in two's complement notation, shall cause reverse movement to the first occurrence of N or more consecutive filemarks being logically positioned on the beginning-of-partition side of the N^{th} filemark.

If EOP is encountered while spacing to sequential filemarks, CHECK CONDITION status shall be returned, and the sense key shall be set to MEDIUM ERROR. The additional sense code shall be set to END-OF-PARTITION/MEDIUM DETECTED, the EOM bit shall be set to one, and the VALID bit shall be set to zero in the sense data. The medium position following this condition is not defined.

If EOD is encountered while spacing to sequential filemarks, CHECK CONDITION status shall be returned, and the sense key shall be set to BLANK CHECK. The additional sense code shall be set to END-OF-DATA DETECTED, and the sense data VALID bit shall be set to zero. The medium shall be positioned after the last logical object. The sense data EOM bit shall be set to one if end-of-data is encountered at or after early-warning.

If spacing to EOD, the COUNT field is ignored. Upon successful completion, the medium shall be positioned after the last logical object.

If EOP is encountered while spacing to end-of-data, CHECK CONDITION status shall be returned, and the sense key shall be set to MEDIUM ERROR. The additional sense code shall be set to END-OF-PARTITION/MEDIUM DETECTED, the EOM bit shall be set to one, and the VALID bit shall be set to zero in the sense data. The medium position following this condition is not defined.

6.7 VERIFY(6) command

The VERIFY(6) command (see table 45) requests that the device server verify one or more logical block(s) or logical blocks in one or more logical file(s) (see 4.2.9) beginning at the current logical position. Prior to performing the sequence of verify operation(s), the device server shall perform a synchronize operation (see 4.2.14). See 4.2.28.5 for the effects of protection information on the transfer of logical blocks during a VERIFY(6) command and see 4.2.28.3 for the format of a logical block if logical block protection is enabled. See 4.2.11.2 for a description of verify command processing.

Table 45 — VERIFY(6) command

Bit Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (13h)							
1	Reserved	VTE	VLBPM	VBF	IMMED	BYTCMP	FIXED	
2	(MSB)							
...	VERIFICATION LENGTH							
4								
5	(LSB)							
	CONTROL							

See SPC-5 for the definition of the OPERATION CODE field. The OPERATION CODE field shall be set as shown in table 45.

If the verify to end-of-data (VTE) bit is set to zero, then a verify to EOD is not requested.

If the VTE bit is set to one, then

- a) the expected verification sequence termination condition is met when EOD is encountered;
- b) if a filemark is encountered during the sequence, processing continues;
- c) if the verify command fails, then the VALID bit and the INFORMATION field of the sense data shall be set to zero. See 4.2.12.3 for the settings of other fields in the sense data.;
- d) the BYTCMP bit and the VBF bit shall be set to zero; and
- e) the VERIFICATION LENGTH field shall be ignored.

If the verify logical block protection method (VLBPM) bit is set to zero, then the verification being performed does not require a verification that each logical block has protection information that is able to be validated using the logical block protection method specified in the Control Data Protection mode page (see 8.5.9).

If the VLBPM bit is set to one, then:

- a) the verification being performed shall also include a verification that each logical block has protection information that is able to be validated using the logical block protection method specified in the Control Data Protection mode page; and
- b) if the verification fails because a logical block did not have protection information that was able to be validated using the logical block protection method specified by the Control Data Protection mode page, then:
 - A) the device server shall terminate the verification sequence with the sense key set to MISCOMPARE;
 - B) the additional sense code shall be set to LOGICAL BLOCK PROTECTION METHOD ERROR; and
 - C) upon termination, the medium shall be positioned after the logical block on which the verification failed (i.e., end-of-partition side).

if the verify by filemarks (VBF) bit is set to zero, then a verify of n filemarks is not requested.

If the VBF bit is set to one, then:

- a) the expected verification sequence termination condition is met if the number of filemarks specified by the VERIFICATION LENGTH field have been traversed;
- b) if a filemark is encountered during the sequence before the number of filemarks specified by the VERIFICATION LENGTH field have been traversed, processing continues;
- c) if EOD is encountered:
 - A) the sense key shall be set to BLANK CHECK;
 - B) if the logical position is at or after early warning the EOM bit shall be set to one;
 - C) the additional sense code shall be set to END-OF-DATA DETECTED
- d) the BYTCMP bit and the VTE bit shall be set to zero; and
- e) If a verify operation fails, then:
 - A) the verification sequence shall terminate;
 - B) the VALID bit shall be set to one;
 - C) the INFORMATION field shall be set to the requested verification length minus the actual number of filemarks successfully traversed; and
 - D) see 4.2.12.3 for the settings of other fields in the sense data.

Following the completion of a verify with the VBF bit set to one, the application client should issue a READ POSITION command to determine the logical object identifier associated with the current logical position.

If the VTE bit is set to zero, the VBF bit is set to zero, and the FIXED bit is set to one, then:

- a) the expected verification sequence termination condition is met if the number of logical blocks specified in the VERIFICATION LENGTH field have been traversed;
- b) if the VERIFICATION LENGTH field is set to zero, then no logical objects shall be verified and the current logical position shall not be changed. This condition shall not be considered an error;
- c) if a filemark is encountered during the sequence, processing terminates with filemark encountered as specified in the READ(6) command (see 6.4);
- d) if EOD is encountered:
 - A) the sense key shall be set to BLANK CHECK;
 - B) if the logical position is at or after early warning the EOM bit shall be set to one; and
 - C) the additional sense code shall be set to END-OF-DATA DETECTED;
 - and
- e) if a verify operation fails, then:
 - A) the verification sequence shall terminate;
 - B) the VALID bit shall be set to one;
 - C) the information field shall be set to the requested verification length minus the actual number of logical blocks successfully traversed; and
 - D) see 4.2.12.3 for the settings of other fields in the sense data.

If the VTE bit is set to zero, the VBF bit is set to zero, and the FIXED bit is set to zero, then:

- a) the expected verification sequence termination condition is met when one logical block has been traversed;
- b) the length of the verified logical block shall be equal to the value specified in the VERIFICATION LENGTH field;
- c) if the VERIFICATION LENGTH field is set to zero, then no logical objects shall be verified and the current logical position shall not be changed. This condition shall not be considered an error;
- d) if a filemark is encountered during the sequence, processing terminates with filemark encountered as specified in the READ(6) command (see 6.4);
- e) if EOD is encountered:
 - A) the sense key shall be set to BLANK CHECK;

- B) if the logical position is at or after early warning the EOM bit shall be set to one; and
- C) the additional sense code shall be set to END-OF-DATA DETECTED; and
- f) if a verify operation fails, then:
 - A) the verification sequence shall terminate;
 - B) the VALID bit shall be set to one;
 - C) the INFORMATION field shall be set to the requested verification length minus the actual number of bytes successfully traversed; and
 - D) see 4.2.12.3 for the settings of other fields in the sense data.

An immediate (IMMED) bit set to zero specifies the command shall not return status until the verify sequence has completed.

An IMMED bit set to one specifies status shall be returned as soon as the command descriptor block has been validated, but after all verification data has been transferred from the application client to the device server, if the BYTCMP bit is one. Verification sequences that complete unsuccessfully shall generate deferred sense data indicating the reason for termination (e.g., an incorrect length logical block is encountered and the sense data is set to indicate an incorrect length block was encountered).

In order to ensure that no errors are lost, the application client should set the IMMED bit to zero on the last VERIFY(6) command of a series of VERIFY(6) commands.

A byte compare (BYTCMP) bit set to zero specifies the verification shall be a verification of logical blocks on the medium (e.g., is able to be read and the CRC and ECC validate). No data shall be transferred from the application client to the device server.

A BYTCMP bit set to one specifies the device server shall perform a byte-by-byte compare of the data on the medium and the data transferred from the application client. Data shall be transferred from the application client to the device server as in a WRITE(6) command (see 6.8). If the data does not compare:

- a) the verification sequence shall terminate with the sense data valid bit set to one, the sense key set to MISCOMPARE, and the additional sense code set to MISCOMPARE DURING VERIFY OPERATION;
- b) if the FIXED bit is set to one, the INFORMATION field shall be set to the requested verification length minus the actual number of logical blocks successfully verified;
- c) if the FIXED bit is set to zero, the INFORMATION field shall be set to the requested verification length minus the actual number of bytes successfully verified; and
- d) upon termination, the medium shall be positioned after the logical block containing the miscompare (i.e., end-of-partition side);

If the BYTCMP bit is set to one and the byte compare option is not supported, the device server shall terminate the command with CHECK CONDITION status, the sense key shall be set to ILLEGAL REQUEST, and the additional sense code shall be set to INVALID FIELD IN CDB.

A FIXED bit set to zero and either the VTE bit set to one or the VBF bit set to one specifies that the block length shall not be checked.

A FIXED bit set to one specifies that the length of verified logical blocks shall be equal to the current block length reported in the mode parameters block descriptor. Refer to the READ(6) command (see 6.4) for a description of the FIXED bit and any error conditions that may result from incorrect usage.

The VERIFICATION LENGTH field specifies the number of bytes, logical blocks, or filemarks to traverse during verification, as specified by the VBF bit and the FIXED bit. If the VTE bit is set to one, then the VERIFICATION LENGTH field shall be ignored. If the VERIFICATION LENGTH field is set to zero and the VTE bit is set to zero, then no logical

objects shall be verified and the current logical position shall not be changed. This condition shall not be considered an error.

6.8 WRITE(6) command

The WRITE command (see table 46) requests that the device server write the logical block that is transferred from the application client to the current logical position. See 4.2.28.5 for the effects of protection information on the transfer of logical blocks during a WRITE(6) command and see 4.2.28.3 for the format of a logical block if logical block protection is enabled.

Table 46 — WRITE(6) command

Bit Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (0Ah)							
1	Reserved							FIXED
2	(MSB)							
...	TRANSFER LENGTH							
4								
5	(LSB)							
	CONTROL							

See SPC-5 for the definition of the OPERATION CODE field. The OPERATION CODE field shall be set as shown in table 46.

The FIXED bit specifies whether fixed-block transfers or variable-block transfers are to be used. See the READ BLOCK LIMITS command (see 7.7) for additional information about fixed-block transfers and variable-block transfers.

If the FIXED bit is one, the TRANSFER LENGTH value specifies the number of fixed-length blocks to be transferred, using the current block length reported in the mode parameter block descriptor (see 8.5). If the FIXED bit is zero, a single logical block is transferred with TRANSFER LENGTH specifying the logical block length in bytes.

If TRANSFER LENGTH is zero, no data shall be transferred and the current position shall not be changed. This condition shall not be considered an error.

A WRITE(6) command may be buffered or unbuffered, as specified by the BUFFERED MODE field of the mode parameter header (see 8.5). If operating in unbuffered mode (see 3.1.88), the device server shall not return GOOD status until all logical block(s) are successfully written to the medium. If operating in buffered mode (see 3.1.9), the device server may return GOOD status as soon as all logical block(s) are successfully transferred to the logical unit's object buffer.

A WRITE FILEMARKS command with the IMMED bit set to zero should be issued to perform a synchronize operation (see 4.2.14) upon completion of buffered write operations.

If the device server enables a WRITE(6) command while positioned between EW and LEOP, or encounters EW during the processing of a WRITE(6) command, an attempt to finish writing any data may be made as determined by the current settings of the REW and SEW bits in the Device Configuration mode page (see 8.5.3). The command shall terminate with CHECK CONDITION status and the additional sense code shall be set to END-OF-PARTITION/MEDIUM DETECTED. If all data that is to be written is successfully transferred to the medium, the sense key shall be set to NO SENSE or RECOVERED ERROR, as appropriate. If the device server is

unable to transfer all the data to the medium, buffered or unbuffered, before end-of-partition is encountered, the sense key shall be set to VOLUME OVERFLOW. The EOM bit shall be set to one. If the SEW bit is set to one, then the VALID bit shall be set to one.

The INFORMATION field shall be set as follows:

- a) if the FIXED bit is set to one, the INFORMATION field shall be set to the requested transfer length minus the actual number of logical blocks transferred to the device server; or
- b) if the FIXED bit is set to zero, the INFORMATION field shall be set to the requested transfer length.

The device server should perform a synchronize operation (see 4.2.14) after the first early-warning indication (see 4.2.5) has been returned to the application client.

NOTE 29 - For some application clients, it is important to recognize an error if end-of-partition is encountered during the processing of a WRITE(6) command, without regard for whether all data that is to be written is successfully transferred to the medium. The VOLUME OVERFLOW sense key may be returned if end-of-partition is encountered while writing, and such usage is recommended. Reporting the MEDIUM ERROR sense key may cause confusion as to whether there was really defective medium encountered during the processing of the last WRITE(6) command.

If the device server encounters LEOP, then the command shall be terminated as specified in 4.2.8.

The sense data VALID bit shall be set to one and the INFORMATION field shall be set as follows:

- a) if the FIXED bit is set to one, then the INFORMATION field shall be set to the requested transfer length minus the actual number of logical blocks transferred to the device server; or
- b) if the FIXED bit is set to zero, then the INFORMATION field shall be set to the requested transfer length.

If a WRITE(6) command is terminated early, an incomplete logical block (i.e., a logical block not completely transferred to the device server from the initiator) shall be discarded. The incomplete logical block may be accessible until a new logical block is written to the media at the same logical object identifier. The device server shall be logically positioned after the last logical block that was successfully transferred.

See 4.2.6 for a description of the effect of a PEWZ on the completion of the WRITE(6) command.

6.9 WRITE FILEMARKS(6) command

The WRITE FILEMARKS(6) command (see table 47) requests that the device server write the specified number of filemarks to the current position.

Table 47 — WRITE FILEMARKS(6) command

Bit Byte	7	6	5	4	3	2	1	0			
0	OPERATION CODE (10h)										
1	Reserved				Obsolete	IMMED					
2	(MSB)										
...	FILEMARK COUNT										
4											
5	(LSB)										
	CONTROL										

See SPC-5 for the definition of the OPERATION CODE field. The OPERATION CODE field shall be set as shown in table 47.

An immediate (IMMED) bit set to one specifies the device server shall return status as soon as the command descriptor block has been validated. An IMMED bit set to one is only valid if the device is operating in buffered mode (see 3.1.9). If the IMMED bit is set to one and the device is operating in unbuffered mode the command shall be terminated with CHECK CONDITION status. The sense key shall be set to ILLEGAL REQUEST and the additional sense code shall be set to INVALID FIELD IN CDB.

An IMMED bit set to zero specifies the device server shall not return status until the write operation has completed. Any buffered logical objects shall be written to the medium prior to completing the command.

The FILEMARK COUNT field specifies the number of filemarks to be written. If the FILEMARK COUNT field is set to zero, the current logical position shall not be changed. It shall not be considered an error if the FILEMARK COUNT field is set to zero.

NOTE 30 - Upon completion of any buffered write operation, the application client may issue a WRITE FILEMARKS(6) command with the IMMED bit set to zero and the FILEMARK COUNT field set to zero to perform a synchronize operation (see 4.2.14).

If the device server enables a WRITE FILEMARKS(6) command while positioned between EW and LEOP, or encounters EW during the processing of a WRITE FILEMARKS(6) command, an attempt to finish writing any buffered logical objects may be made, as determined by the current settings of the REW and SEW bits in the Device Configuration mode page (see 8.5.3). The command shall terminate with CHECK CONDITION status and the additional sense code shall be set to END-OF-PARTITION/MEDIUM DETECTED. If all buffered logical objects to be written are successfully transferred to the medium, the sense key shall be set to NO SENSE or RECOVERED ERROR, as appropriate. If the device server is unable to transfer all the buffered logical objects to the medium before end-of-partition is encountered, the sense key shall be set to VOLUME OVERFLOW. The EOM bit shall be set to one. If the SEW bit is set to one, then the VALID bit shall be set to one.

The INFORMATION field shall be set to the FILEMARK COUNT field value minus the actual number of filemarks:

- a) written to the object buffer if the IMMED bit is set to one; or
- b) written to the medium if the IMMED bit is set to zero.

The device server should perform a synchronize operation (see 4.2.14) after the first early-warning indication (see 4.2.5) has been returned to the application client.

See 4.2.6 for a description of the effect of a PEWZ on the completion of the WRITE FILEMARKS(6) command.

If the device server encounters LEOP, then the command shall be terminated as specified in 4.2.8.

The sense data VALID bit shall be set to one and the INFORMATION field shall be set to the FILEMARK COUNT field value minus the actual number of filemarks:

- a) written to the object buffer if the IMMED bit is set to one; or
- b) written to the medium if the IMMED bit is set to zero.

7 Common command descriptions for sequential-access devices

7.1 ALLOW OVERWRITE command

If append-only mode is enabled, then an application client may use the ALLOW OVERWRITE command to enable overwrite of the medium at a non-append point. The processing of the ALLOW OVERWRITE command shall set the allow_overwrite and allow_overwrite_position variables as specified in 4.2.16.3.

Table 48 — ALLOW OVERWRITE command

Bit Byte	7	6	5	4	3	2	1	0				
0	OPERATION CODE (82h)											
1	Reserved											
2	Reserved				ALLOW OVERWRITE							
3	PARTITION											
4	(MSB)											
...	LOGICAL OBJECT IDENTIFIER											
11												
12												
...	Reserved											
14												
15	CONTROL											

See SPC-5 for the definition of the OPERATION CODE field. The OPERATION CODE field shall be set as shown in table 48.

The ALLOW OVERWRITE field (see table 49) specifies what type of overwrite is allowed (see 4.2.16.3).

Table 49 — ALLOW OVERWRITE field definition

Code	Description
0h	The allow_overwrite variable shall be set to Disabled.
1h	The allow_overwrite variable shall be set to Current Position.
2h	The allow_overwrite variable shall be set to Format.
3h to Fh	Reserved.

If the ALLOW OVERWRITE field is set to Current Position (i.e., 1h), then:

- a) the PARTITION field shall be set to the active partition; and
- b) the LOGICAL OBJECT IDENTIFIER field shall be set to the current position.

If the ALLOW OVERWRITE field is set to Current Position (i.e., 1h), then the allow_overwrite_position variable shall be set to the current position.

If the ALLOW OVERWRITE field is not set to Current Position (i.e., 1h), then the PARTITION field and LOGICAL OBJECT IDENTIFIER field shall be ignored.

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 SEQUENTIAL POSITIONING ERROR if:

- a) the ALLOW OVERWRITE field is set to Current Position (i.e., 1h); and
- b) the logical position of the medium is not at the location specified by the PARTITION field and LOGICAL OBJECT IDENTIFIER field.

If the device server terminates that command with any status other than GOOD, then the allow_overwrite variable shall be set to Disabled and the allow_overwrite_position variable shall be set to invalid.

7.2 FORMAT MEDIUM command

The FORMAT MEDIUM command (see table 50) is used to prepare the recording volume for use by the logical unit. A FORMAT MEDIUM command shall be terminated with CHECK CONDITION status, the sense key set to ILLEGAL REQUEST and the additional sense code set to POSITION PAST BEGINNING OF MEDIUM if:

- a) there are buffered logical objects to be written to the medium; or
- b) the medium is not at beginning-of-partition 0 (BOP 0).

Table 50 — FORMAT MEDIUM command

Bit Byte	7	6	5	4	3	2	1	0
0					OPERATION CODE (04h)			
1					Reserved		VERIFY	IMMED
2			Reserved			FORMAT		
3	(MSB)				TRANSFER LENGTH			
4								(LSB)
5					CONTROL			

See SPC-5 for the definition of the OPERATION CODE field. The OPERATION CODE field shall be set as shown in table 50.

At the successful completion of a FORMAT MEDIUM command, the medium shall be positioned at BOP 0.

During the format operation, the device server shall respond to commands as follows:

- a) in response to all commands except REQUEST SENSE, REPORT LUNS, and INQUIRY, the device server shall return CHECK CONDITION unless a reservation conflict exists. In that case, RESERVATION CONFLICT status shall be returned; or
- b) in response to the REQUEST SENSE command, assuming no error has occurred, the device server shall return a sense key of NOT READY and the additional sense code shall be set to LOGICAL UNIT NOT READY, FORMAT IN PROGRESS, with the sense key specific bytes set for progress indication (see SPC-4).

A VERIFY bit set to one specifies the logical unit shall format the recording volume and then verify that the format was successfully accomplished. The method used to verify success of the FORMAT MEDIUM command is vendor specific. If the verify operation determines that the format was not successfully accomplished, the device server

shall return a sense key of MEDIUM ERROR and the additional sense code shall be set to MEDIUM FORMAT CORRUPTED. If the VERIFY bit is zero, the logical unit shall not perform the verify check.

An immediate (IMMED) bit set to zero specifies the device server shall not return status until the FORMAT MEDIUM command has completed. An IMMED bit set to one specifies the device server shall return status as soon as the valid medium location has been verified and the command descriptor block of the FORMAT MEDIUM command has been validated. If CHECK CONDITION status is returned for a FORMAT MEDIUM command with an IMMED bit set to one, the format operation shall not be performed.

The FORMAT field is specified in table 51.

Table 51 — FORMAT field

Code	Description	Support
0h	Use default format	O
1h	Partition volume	O
2h	Default format then partition	O
3h to 7h	Reserved	
8h to Fh	Vendor specific	

If the FORMAT field is 0h, the logical unit shall determine the format method to use. A valid FORMAT MEDIUM command with 0h in the FORMAT field shall cause:

- a) all logical objects on the entire volume to become unavailable (e.g., inaccessible or destroyed);
- b) all host type MAM attributes to be destroyed;
- c) all device type MAM attributes other than those belonging to partition 0 to be destroyed; and
- d) other vendor-specific data to become unavailable.

If the FORMAT field is 1h, the logical unit shall partition the recording volume using the current mode data from the Medium Partition mode page (see 8.5.4). If none of the mode bits SDP, FDP, or IDP are set to one, the device server shall return CHECK CONDITION. The sense key shall be set to ILLEGAL REQUEST with the addition sense code set to PARAMETER VALUE INVALID. If conditions on the volume prohibit modifications to the partitions as specified, then the device server shall return CHECK CONDITION status. The sense key shall be set to ILLEGAL REQUEST with the additional sense code set to PARAMETER VALUE INVALID. A valid FORMAT MEDIUM command with 1h in the FORMAT field may cause:

- a) all logical objects on any or all partitions to become unavailable (e.g., inaccessible or destroyed);
- b) all host type MAM attributes to be destroyed on any partition where all logical objects become unavailable;
- c) all device type MAM attributes to be destroyed on any partition where all its logical objects become unavailable; or
- d) vendor-specific data to become unavailable.

A valid FORMAT MEDIUM command with 1h in the FORMAT field shall cause:

- a) all logical objects in partitions that are removed to become unavailable;
- b) all logical objects beyond the new end of partition in a partition that is truncated to become unavailable;
- c) all host type MAM attributes belonging to partitions that are removed to be destroyed; and
- d) all device type MAM attributes belonging to partitions that are removed to be destroyed.

If the FORMAT field is 2h, the logical unit shall perform the operations equivalent to a FORMAT field of 0h followed by a FORMAT field of 1h. A valid FORMAT MEDIUM command with 2h in the FORMAT field shall cause:

- a) all logical objects on the entire volume to become unavailable (e.g., inaccessible or destroyed);
- b) all host type MAM attributes to be destroyed;
- c) all device type MAM attributes other than those belonging to partition 0 to be destroyed; and
- d) other vendor-specific data to become unavailable.

If the FORMAT field contains 1h or 2h, some errors related to mode page field contents may not be detected until the FORMAT MEDIUM command is processed. Therefore, some error conditions described in 8.5.4 may be returned in response to a FORMAT MEDIUM command with 1h or 2h in the FORMAT field.

The TRANSFER LENGTH specifies the length in bytes of format information that shall be transferred from the initiator. A transfer length of zero specifies no format information shall be transferred. This condition shall not be considered an error. If the FORMAT field is 0h, 1h, or 2h, the TRANSFER LENGTH shall be zero. Use of format information is restricted to vendor-specific values in the FORMAT field and the contents of the format information is vendor specific.

7.3 GENERATE RECOMMENDED ACCESS ORDER command

The GENERATE RECOMMENDED ACCESS ORDER command is used by an application client to request the device server generate a recommended access order for the User Data Segments described by the parameter data. Prior to generating the RAO list, the device server shall perform a synchronize operation (see 4.2.14).

After a GENERATE RECOMMENDED ACCESS ORDER command completes, the RECEIVE RECOMMENDED ACCESS ORDER command (see 7.10) may be used to receive the results in the form of an RAO list. A previous RAO list is cleared when the device server begins processing the parameter list.

Table 52 — GENERATE RECOMMENDED ACCESS ORDER command

Bit Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (A4h)							
1	OPERATION CODE (1Dh)							
2	Reserved							
3	Reserved							
4	Reserved							
5	Reserved							
6	(MSB)							
...	PARAMETER LIST LENGTH							
9	(LSB)							
10	Reserved							
11	CONTROL							

See SPC-5 for the definition of the OPERATION CODE field and the OPERATION CODE field. The OPERATION CODE field and the OPERATION CODE field shall be set as shown in table 52.

The RAO PROCESS field specifies the method that shall be used to generate the contents of the RAO list as specified in 4.2.25.

The UDS_TYPE field (see table 53) specifies the format of the User Data Segment descriptor to be used in the RAO list.

Table 53 — UDS_TYPE field

Code	Description
000b	User data segment
001b	User data segment with additional information
010b-111b	Reserved

The PARAMETER LIST LENGTH field specifies the length in bytes of the parameter list that is transferred from the initiator to the target. A parameter list length value of zero indicates that no data is transferred and that the RAO list is to be cleared. This condition is not considered an error. If the parameter list length results in the truncation of one or more User Data Segments, then the device server shall return CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB. If the specified PARAMETER LIST LENGTH value results in the truncation of the parameter list header (i.e., the parameter list length is less than eight bytes), then the command is rejected with a CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.

7.3.1 GENERATE RECOMMENDED ACCESS ORDER command parameter data

The GENERATE RECOMMENDED ACCESS ORDER parameter data format is shown in table 54.

Table 54 — GENERATE RECOMMENDED ACCESS ORDER parameter data format

The UDS DESCRIPTOR LIST LENGTH field specifies the number of bytes to follow. If the UDS DESCRIPTOR LIST LENGTH field is set to zero, then the RAO list is cleared. This is not considered an error.

The GRAO user data segment descriptor list contains user data segments that the device server is being requested to process as specified in the RAO PROCESS field of the CDB. The GRAO - user data segment descriptor is defined in 7.3.2. 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 PARAMETER LIST and the recommended access order list shall be cleared, if:

- a) the number of user data segment descriptors sent is larger than the maximum number of user data segments supported by the device server;
- b) a user data segment specified by one of the descriptors does not exist (i.e., the BEGINNING LOGICAL OBJECT LOCATION and PARTITION NUMBER combination does not exist or the ENDING LOGICAL OBJECT LOCATION and PARTITION NUMBER combination does not exist); or
- c) a user data segment descriptor is malformed (e.g., the ENDING LOGICAL OBJECT LOCATION is located at an earlier logical object identifier than the BEGINNING LOGICAL OBJECT LOCATION).

7.3.2 GRAO - user data segment descriptor

The GRAO - user data segment descriptor format is shown in table 55.

Table 55 — GRAO - user data segment descriptor format

Bit Byte	7	6	5	4	3	2	1	0
0								
1								DESCRIPTOR LENGTH (1Eh)
2								
...								
4								Reserved
5								
...								
14								UDS NAME
15								PARTITION NUMBER
16	(MSB)							
...								BEGINNING LOGICAL OBJECT IDENTIFIER
23								(LSB)
24	(MSB)							
...								ENDING LOGICAL OBJECT IDENTIFIER
31								(LSB)

The DESCRIPTOR LENGTH field specifies the number of bytes to follow.

The UDS NAME field contains an application client specified name for this user data segment.

The PARTITION NUMBER field specifies the number of the partition in which this user data segment is located.

The BEGINNING LOGICAL OBJECT IDENTIFIER field specifies the logical object identifier of the beginning logical object of the user data segment.

The ENDING LOGICAL OBJECT IDENTIFIER field specifies the logical object identifier of the ending logical object of the user data segment.

7.4 LOAD UNLOAD command

The LOAD UNLOAD command (see table 56) requests that a volume be mounted or demounted. This command may also be used to request a retension function. Prior to performing the LOAD UNLOAD operation, the device server shall perform a synchronize operation (see 4.2.14). If the buffered mode is not 0h (see 8.5) and a previous command was terminated with CHECK CONDITION status and the device is unable to continue successfully writing, the logical unit shall discard any unwritten buffered logical objects prior to performing the LOAD UNLOAD operation. If operating in buffered mode 1h or 2h (see 8.5), the logical unit shall discard any unwritten buffered logical objects after the LOAD UNLOAD command is validated if the device is unable to continue successfully writing (e.g., the device reported CHECK CONDITION status to a previous command, reported a write type error, and the error has not been cleared or otherwise recovered).

Table 56 — LOAD UNLOAD command

Bit Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (1Bh)							
1	Reserved							IMMED
2	Reserved							
3	Reserved							
4	Reserved			HOLD	EOT	RETEN	LOAD	
5	CONTROL							

See SPC-5 for the definition of the OPERATION CODE field. The OPERATION CODE field shall be set as shown in table 56.

An immediate (IMMED) bit set to zero specifies the device server shall not return status until the mount or demount has completed. If the IMMED bit is one, the device server shall return status as soon as all buffered logical objects have been written to the medium and the command descriptor block of the LOAD UNLOAD command has been validated. If CHECK CONDITION status is returned for a LOAD UNLOAD command with an IMMED bit set to one, the mount or demount shall not be performed.

NOTE 31 - For compatibility with devices implemented prior to this version of the standard, a WRITE FILEMARKS command with an IMMED bit set to zero should be used to perform a synchronize operation (see 4.2.14) prior to issuing a LOAD UNLOAD command with an IMMED bit set to one.

A HOLD bit set to one and a LOAD bit set to one or a LOAD bit set to zero specifies MAM shall be accessible upon completion of the command but the volume shall not be positioned for access. A HOLD bit set to zero and a LOAD bit set to one specifies the volume shall be positioned for access. A HOLD bit set to zero and a LOAD bit set to zero specifies MAM shall not be accessible upon completion of the command.

An end-of-tape (EOT) bit set to one specifies a demount (i.e., LOAD bit set to zero) shall position the medium at end-of-medium for removal from the device. For devices that do not support unloading at end-of-medium, the device server shall terminate the command with CHECK CONDITION status. The sense key shall be set to ILLEGAL REQUEST and the additional sense code shall be set to INVALID FIELD IN CDB. An EOT bit set to zero specifies a demount shall position the medium at beginning-of-medium for removal from the device.

An EOT bit set to one and a LOAD bit set to one shall cause the device server to return CHECK CONDITION status and the sense key shall be set to ILLEGAL REQUEST and the additional sense code shall be set to INVALID FIELD IN CDB.

A retension (RETEN) bit set to one specifies the logical unit shall perform a retension function on the current medium if a volume is present and loadable. If no volume is present or the volume is not loadable, then the device server shall terminate the command with CHECK CONDITION status. The sense key shall be set to NOT READY and the additional sense code shall be set to MEDIUM NOT PRESENT. A RETEN bit set to zero specifies the logical unit shall not perform a retension function on the current medium. Implementation of the retension function is vendor specific.

A LOAD bit set to one and a HOLD bit set to zero specifies the volume in the logical unit shall be loaded and positioned to the beginning-of-partition zero. A LOAD bit set to zero and a HOLD bit set to zero specifies the volume in the logical unit shall be positioned for removal at the extreme position along the medium specified by the EOT bit. Following successful completion of a demount, the device server shall return CHECK CONDITION status with the sense key set to NOT READY for all subsequent medium access commands until a new volume is mounted.

A LOAD bit set to one and a HOLD bit set to one specifies the volume shall be moved in but not positioned for access. If the EOT bit is set to one or the RETEN bit is set to one, then the device server shall terminate the command with CHECK CONDITION status. The sense key shall be set to ILLEGAL REQUEST and the additional sense code shall be set to INVALID FIELD IN CDB. Following successful completion of a command requesting a load to the hold position, the device server shall return GOOD status. If the device server, volume, and medium support MAM, the device server shall establish a unit attention condition for all initiators and the additional sense code shall be set to MEDIUM AUXILIARY MEMORY ACCESSIBLE.

A LOAD bit set to zero and a HOLD bit set to one specifies if the volume is in the logical unit, the medium shall be positioned as specified by the RETEN and EOT bits or shall be unthreaded, whichever is appropriate for the medium type, but shall not be demounted. Following successful completion, the device server shall return GOOD status. If a volume is not present or the volume is not loadable, then the device server shall terminate the command with CHECK CONDITION status. The sense key shall be set to NOT READY and the additional sense code shall be set to MEDIUM NOT PRESENT.

A LOAD bit set to zero and a HOLD bit set to zero specifies if the volume is in the logical unit, the medium shall be positioned as specified by the RETEN and EOT bits or shall be unthreaded and the volume shall be demounted. Following successful completion, the device server shall return GOOD status. If a volume is not present, then the device server may terminate the command with CHECK CONDITION status with the sense key set to NOT READY and the additional sense code set to MEDIUM NOT PRESENT. If a volume is not present and the RETEN bit is set to zero and the EOT bit is set to zero, then the device server may return GOOD status.

If the device server does not support the bit combination in any state, then a CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB shall be returned.

If operating in buffered mode 1h or 2h (see 8.5), the logical unit shall discard any unwritten buffered data after the LOAD UNLOAD command is validated if the device is unable to continue successfully writing (e.g., the device reported CHECK CONDITION status to a previous command, reported a write type error, and the error has not been cleared or otherwise recovered).

7.5 LOCATE(16) command

The LOCATE(16) command (see table 57) causes the logical unit to position the medium to the logical object or logical file (see 4.2.9), as specified by the DEST_TYPE and LOGICAL IDENTIFIER fields. Upon completion, the logical

position shall be as specified in table 58. Prior to performing the locate operation, the device server shall perform a synchronize operation (see 4.2.14).

Table 57 — LOCATE(16) command

Bit Byte	7	6	5	4	3	2	1	0				
0	OPERATION CODE (92h)											
1	Reserved		DEST_TYPE		Rsvd	CP	IMMED					
2	Reserved											
3	PARTITION											
4	(MSB)											
...	LOGICAL IDENTIFIER											
11												
12												
...	Reserved											
14												
15	CONTROL											

See SPC-5 for the definition of the OPERATION CODE field. The OPERATION CODE field shall be set as shown in table 57.

The DEST_TYPE field (see table 58) shall be used in conjunction with the LOGICAL IDENTIFIER field to locate to the appropriate position of the medium. The DEST_TYPE field specifies whether the location specified is a logical object identifier, logical file identifier, or EOD (i.e., the LOGICAL IDENTIFIER field is not used).

Table 58 — DEST_TYPE field

Code	Description	Logical position upon successful completion	Support
000b	Logical object identifier	BOP side	M
001b	Logical file identifier	BOP side of the logical file	M
010b	Obsolete		
011b	End-of-data	EOD	M
100b to 111b	Reserved		

A change partition (CP) bit set to one specifies a change to the partition specified in the PARTITION field shall occur prior to positioning to the logical object or logical file, as specified in the LOGICAL IDENTIFIER field. A CP bit set to zero specifies no partition change shall occur and the PARTITION field shall be ignored.

An immediate (IMMED) bit set to zero specifies the device server shall not return status until the locate operation has completed. If the IMMED bit is one, the device server shall return status as soon as all buffered logical objects have been written to the medium and the command descriptor block of the LOCATE(16) command has been validated. If CHECK CONDITION status is returned for a LOCATE(16) command with an IMMED bit set to one, the locate operation shall not be performed.

A block address mode type (BAM) bit set to zero specifies the logical unit shall process this command as an implicit address command. A BAM bit set to one specifies the logical unit shall process this command as an explicit address command.

The LOGICAL IDENTIFIER field specifies the logical identifier to which the logical unit shall position the medium based on the current setting of the DEST_TYPE field. If the DEST_TYPE field is set to 11b (i.e., End-of-data), then the LOGICAL IDENTIFIER field shall be ignored. If the requested location is beyond LEOP, then the command shall be terminated as specified in 4.2.8. An otherwise valid LOCATE(16) command to any position between beginning-of-data and the position immediately after the last object in the partition (i.e., position at end-of-data) shall not return a sense key of ILLEGAL REQUEST. A LOCATE(16) to a position past end-of-data shall return CHECK CONDITION status and the sense key shall be set to BLANK CHECK. Additionally, the sense data EOM bit shall be set to one if end-of-data is located at or after early-warning.

If end-of-partition is encountered, CHECK CONDITION status shall be returned, the sense key shall be set to MEDIUM ERROR, and the sense data EOM bit shall be set to one.

The PARTITION field specifies the partition to select if the CP bit is one. See the sequential-access device model (see 4.2.7) and the Medium Partition mode page (see 8.5.4) for additional information about partitioning.

The logical unit position is undefined if a LOCATE(16) command fails with a sense key other than ILLEGAL REQUEST.

7.6 PREVENT ALLOW MEDIUM REMOVAL command

The PREVENT ALLOW MEDIUM REMOVAL command (see table 59) requests that the logical unit enable or disable the removal of the volume. The logical unit shall not allow volume removal if any initiator port currently has volume removal prevented.

Table 59 — PREVENT ALLOW MEDIUM REMOVAL command

Bit Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (1Eh)							
1								
...	Reserved							
3								
4	Reserved						PREVENT	
5	CONTROL							

See SPC-5 for the definition of the OPERATION CODE field. The OPERATION CODE field shall be set as shown in table 59.

Table 60 specifies the PREVENT field values.

Table 60 — PREVENT field

Code	Description
00b	Volume removal shall be allowed.
01b	Volume removal shall be prevented.
10b	Obsolete
11b	Obsolete

The prevention of volume removal shall begin upon the device server successfully processing a PREVENT ALLOW MEDIUM REMOVAL command with a PREVENT field of 01b (i.e., volume removal prevented). The prevention of volume removal for the logical unit shall terminate after:

- a) one of the following occurs for each I_T nexus that previously had volume removal prevented:
 - A) the device server successfully processes a PREVENT ALLOW MEDIUM REMOVAL command with a PREVENT field of 00b; or
 - B) an I_T nexus loss;
- b) a power on;
- c) a hard reset; or
- d) a logical unit reset.

If a persistent reservation or registration is being preempted by a PERSISTENT RESERVE OUT command with a service action of PREEMPT AND ABORT or CLEAR (see SPC-4), the equivalent of a PREVENT ALLOW MEDIUM REMOVAL command with the PREVENT field set to 00b shall be processed for each I_T nexus associated with the persistent reservation or registrations being preempted. This allows an application client to override the prevention of volume removal function for a SCSI initiator port that is no longer operating correctly.

While a prevention of volume removal condition is in effect, the logical unit shall inhibit mechanisms that normally allow removal of the volume by an operator.

7.7 READ BLOCK LIMITS command

7.7.1 READ BLOCK LIMITS command overview

The READ BLOCK LIMITS command (see table 61) requests that the READ BLOCK LIMITS DATA (see table 62) be returned. The READ BLOCK LIMITS block length data (see 7.7.2) specifies the block length limits capability of the device server. The READ BLOCK LIMITS maximum logical object identifier data (see 7.7.3) specifies the maximum value of the logical object identifier of the device server. The READ BLOCK LIMITS DATA reported shall not change without a hard reset. See 4.2.28.1 for restrictions placed on the MAXIMUM BLOCK LENGTH LIMIT if the device server supports protection information.

Table 61 — READ BLOCK LIMITS command

Bit Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (05h)							
1	Reserved							
2								
...	Reserved							
4								
5	CONTROL							

See SPC-5 for the definition of the OPERATION CODE field. The OPERATION CODE field shall be set as shown in table 61.

If the maximum logical object identifier (MLOI) bit is set to zero, then the data returned is the READ BLOCK LIMITS block length data (see 7.7.2). If the MLOI bit is set to one, then the data returned is the READ BLOCK LIMITS maximum logical object identifier data (see 7.7.3).

7.7.2 READ BLOCK LIMITS block length data

The READ BLOCK LIMITS block length data (see table 62) specifies the block length limits capability of the logical unit.

Table 62 — READ BLOCK LIMITS block length data

Bit Byte	7	6	5	4	3	2	1	0
0	Reserved		GRANULARITY					
1	(MSB)	MAXIMUM BLOCK LENGTH LIMIT						
...								(LSB)
3								
4	(MSB)	MINIMUM BLOCK LENGTH LIMIT						
5								(LSB)

The GRANULARITY field specifies the supported block size granularity. The logical unit shall support all block sizes n such that n minus the minimum block length limit is a multiple of $2^{\text{GRANULARITY}}$ and n is greater than or equal to the MINIMUM BLOCK LENGTH LIMIT and less than or equal to the MAXIMUM BLOCK LENGTH LIMIT.

The MAXIMUM BLOCK LENGTH LIMIT field specifies the maximum number of bytes of user data that may be recorded on the medium in a logical block. The MAXIMUM BLOCK LENGTH LIMIT field does not specify the maximum transfer length supported by the logical unit (i.e., additional bytes that are not user data may be transferred).

If the MAXIMUM BLOCK LENGTH LIMIT value equals the MINIMUM BLOCK LENGTH LIMIT value, the logical unit supports fixed-block transfers only, with the block length equal to the MINIMUM BLOCK LENGTH LIMIT value. In this case, READ and WRITE commands with the FIXED bit set to zero shall result in CHECK CONDITION status and the sense key shall be set to ILLEGAL REQUEST and the additional sense code shall be set to INVALID FIELD IN CDB.

For read type commands and write type commands with the FIXED bit set to one, block lengths are limited to multiples of four (see 8.5).

The MINIMUM BLOCK LENGTH LIMIT field specifies the minimum number of bytes of user data that may be recorded on the medium in a logical block. If the MAXIMUM BLOCK LENGTH LIMIT value is not equal to the MINIMUM BLOCK LENGTH LIMIT value, the logical unit supports fixed-block transfers or variable-block transfers, with the block length constrained between the given limits in either transfer mode. The transfer mode is controlled by the FIXED bit in the WRITE or READ commands. If the maximum block limit is zero, a maximum block length is not specified.

7.7.3 READ BLOCK LIMITS maximum logical object identifier data

The READ BLOCK LIMITS maximum logical object identifier data (see table 63) specifies the maximum value of the logical object identifier the logical unit supports.

Table 63 — READ BLOCK LIMITS maximum logical object identifier data

Bit Byte	7	6	5	4	3	2	1	0
0								
...								
11								
12	(MSB)							
...								
19								(LSB)

The MAXIMUM LOGICAL OBJECT IDENTIFIER field specifies the maximum value the device server supports in a logical object identifier field.

7.8 READ DYNAMIC RUNTIME ATTRIBUTE command

The READ DYNAMIC RUNTIME ATTRIBUTE command (see table 64) requests the device server to read dynamic attribute information from the device.

Table 64 — READ DYNAMIC RUNTIME ATTRIBUTE command

Bit Byte	7	6	5	4	3	2	1	0						
0	OPERATION CODE (A3h)													
1	ATTRIBUTE REPORT TYPE			SERVICE ACTION (1Eh)										
2	(MSB)	FIRST DYNAMIC RUNTIME ATTRIBUTE IDENTIFIER						(LSB)						
3														
4	(MSB)	LAST DYNAMIC RUNTIME ATTRIBUTE IDENTIFIER						(LSB)						
5														
6	(MSB)													
...		ALLOCATION LENGTH						(LSB)						
9														
10	Reserved													
11	Control													

See SPC-5 for the definition of the OPERATION CODE field and the SERVICE ACTION field. The OPERATION CODE field and the SERVICE ACTION field shall be set to the values shown in table 64.

The ATTRIBUTE REPORT TYPE field specifies an attribute report type (see 7.8.1) for the parameter data returned by the READ DYNAMIC RUNTIME ATTRIBUE command.

The FIRST DYNAMIC RUNTIME ATTRIBUTE IDENTIFIER field specifies the dynamic runtime attribute identifier of the first attribute to be returned. Only attributes with a dynamic runtime attribute identifier greater than or equal to the value specified in the FIRST DYNAMIC RUNTIME ATTRIBUTE IDENTIFIER field and that are not in the nonexistent or unsupported state shall be reported. It shall not be considered an error if the specified dynamic runtime attribute is in the unsupported or nonexistent state.

The LAST DYNAMIC RUNTIME ATTRIBUTE IDENTIFIER field specifies the dynamic runtime attribute identifier of the last attribute to be returned. Only attributes with a dynamic runtime attribute identifier less than or equal to the value specified in the LAST DYNAMIC RUNTIME ATTRIBUTE IDENTIFIER field and that are not in the nonexistent or unsupported state shall be reported. It shall not be considered an error if the specified dynamic runtime attribute is in the unsupported or nonexistent state. If the attribute identifier specified in the LAST DYNAMIC RUNTIME ATTRIBUTE IDENTIFIER field is less than the attribute identifier specified in the FIRST DYNAMIC RUNTIME ATTRIBUTE IDENTIFIER field, 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.

If the ALLOCATION LENGTH specified causes an attribute to be truncated this shall not be considered an error. All data up to the allocation length shall be returned.

The CONTROL byte is defined in SAM-5.

7.8.1 READ DYNAMIC RUNTIME ATTRIBUTE attribute report type

The attribute report types defined for the READ DYNAMIC RUNTIME ATTRIBUTE command are shown in table 65.

Table 65 — READ DYNAMIC RUNTIME ATTRIBUTE attribute report type codes

Code	Name	Description	Reference
000b	SUPPORTED ATTRIBUTES	Return a list of dynamic runtime attribute identifiers that the device server supports. No indication of attribute state is implied.	7.8.2
001b	ATTRIBUTE VALUES FOR THIS I_T NEXUS	Return values for: a) all device type attributes; b) all target type attributes associated with the I_T nexus through which the command was received; and c) all initiator type attributes associated with the I_T nexus through which the command was received.	7.8.3
010b	ATTRIBUTE VALUES FOR ALL I_T NEXUSES	Return values for: a) all device type attributes; b) all target type attributes associated with all I_T nexuses; and c) all initiator type attributes associated with all I_T nexuses.	7.8.4
011b - 111b	Reserved		

7.8.2 SUPPORTED ATTRIBUTES attribute report type

The READ DYNAMIC RUNTIME ATTRIBUTE command with SUPPORTED ATTRIBUTES attribute report type (ATTRIBUTE REPORT TYPE field set to 000b) returns parameter data containing the attribute identifiers that the device

server supports. The returned parameter data shall contain the requested attribute identifiers in ascending numerical order by attribute identifier and in the format shown in table 66.

Table 66 — READ DYNAMIC RUNTIME ATTRIBUTE with SUPPORTED ATTRIBUTES attribute report type parameter list format

Bit Byte	7	6	5	4	3	2	1	0
0	ATTRIBUTE REPORT TYPE						Reserved	
1								
...					Reserved			
3								
4	(MSB)							
...					AVAILABLE DATA (n-7)			
7								(LSB)
					Attribute Identifier list			
8	(MSB)				Attribute Identifier [first] (see 8.2.2)			
9								(LSB)
					⋮			
n-1	(MSB)				Attribute Identifier [last] (see 8.2.2)			
n								(LSB)

The ATTRIBUTE REPORT TYPE field shall contain the attribute report type specified in the READ DYNAMIC RUNTIME ATTRIBUTE CDB.

The AVAILABLE DATA field shall contain the number of bytes of attribute identifiers in the parameter list. The available data field shall not be adjusted by the ALLOCATION LENGTH field in the CDB.

An attribute identifier is returned for each attribute that the device server supports. No indication of the current state of the reported attributes shall be made. See 8.2.2 for a description of the attribute identifier.

7.8.3 ATTRIBUTE VALUES FOR THIS I_T NEXUS attribute report type

The READ DYNAMIC RUNTIME ATTRIBUTE command with ATTRIBUTE VALUES FOR THIS I_T NEXUS attribute report type (i.e., the ATTRIBUTE REPORT TYPE field is set to 001b) returns parameter data containing the attributes for the I_T_L Nexus through which this command was received starting with the FIRST ATTRIBUTE IDENTIFIER field in the CDB and ending with the LAST ATTRIBUTE IDENTIFIER FIELD in the CDB.

The returned parameter data shall contain the requested attributes in the format shown in table 67 and in ascending numerical order by I_T nexus index then by attribute identifier.

Table 67 — READ DYNAMIC RUNTIME ATTRIBUTE with ATTRIBUTE VALUES FOR THIS I_T NEXUS attribute report type parameter list format

Bit Byte	7	6	5	4	3	2	1	0
0	ATTRIBUTE REPORT TYPE						Reserved	
1								
...					Reserved			
3								
4	(MSB)							
...					AVAILABLE DATA (n-7)			
7								(LSB)
					Attribute list			
8	(MSB)							
9					Attribute [first] (see 8.2.1)			
								(LSB)
					⋮			
n-1	(MSB)							
n					Attribute [last] (see 8.2.1)			
								(LSB)

The ATTRIBUTE REPORT TYPE field shall contain the attribute report type specified in the READ DYNAMIC RUNTIME ATTRIBUTE CDB.

The AVAILABLE DATA field shall contain the number of bytes of attribute information in the parameter list. The value in the AVAILABLE DATA field shall not be adjusted by the ALLOCATION LENGTH field in the CDB.

The format of the attributes is described in 8.2.1.

7.8.4 ATTRIBUTE VALUES FOR ALL I_T NEXUSES attribute report type

The READ DYNAMIC RUNTIME ATTRIBUTE command with ATTRIBUTE VALUES FOR ALL I_T NEXUSES attribute report type (i.e., the ATTRIBUTE REPORT TYPE field set to 010b) returns parameter data containing the attributes for all known I_T nexuses starting with the first ATTRIBUTE IDENTIFIER field in the CDB and ending with the last ATTRIBUTE IDENTIFIER field in the CDB.

The returned parameter data shall contain the requested attribute values for all I_T nexuses in the format shown in table 68 and in ascending numerical order by I_T nexus index then by attribute identifier.

Table 68 — READ DYNAMIC RUNTIME ATTRIBUTE with ATTRIBUTE VALUES FOR ALL I_T NEXUSES attribute report type parameter list format

Bit Byte	7	6	5	4	3	2	1	0
0	ATTRIBUTE REPORT TYPE						Reserved	
1								
...					Reserved			
3								
4	(MSB)				AVAILABLE DATA (n-7)			
...								
7							(LSB)	
					Attribute list			
8	(MSB)				Attribute [first] (see 8.2.1)			
9							(LSB)	
					⋮			
n-1	(MSB)				Attribute [last] (see 8.2.1)			
n							(LSB)	

The ATTRIBUTE REPORT TYPE field shall contain the attribute report type specified in the READ DYNAMIC RUNTIME ATTRIBUTE CDB.

The AVAILABLE DATA field shall contain the number of bytes to follow. The value in the AVAILABLE DATA field shall not be adjusted by the CDB ALLOCATION LENGTH field.

The format of the attributes is described in 8.2.1.

7.9 READ POSITION command

7.9.1 READ POSITION command description

The READ POSITION command (see table 69) reports the current position and provides information about logical objects contained in the object buffer. Medium movement shall not occur as a result of responding to the command.

Table 69 — READ POSITION command

Bit Byte	7	6	5	4	3	2	1	0						
0	OPERATION CODE (34h)													
1	Reserved		SERVICE ACTION											
2														
...	Reserved													
6														
7	(MSB)		ALLOCATION LENGTH				(LSB)							
8														
9	CONTROL													

See SPC-5 for the definition of the OPERATION CODE field and the SERVICE ACTION field. The OPERATION CODE field and the SERVICE ACTION field shall be set to the values shown in table 69.

The service actions defined for the READ POSITION command are specified in table 70.

Table 70 — READ POSITION service action codes (part 1 of 2)

Code	Name	Description	Implementation requirements	Reference
00h	SHORT FORM -- BLOCK ID	Device server shall return 20 bytes of data with the FIRST LOGICAL OBJECT LOCATION and LAST LOGICAL OBJECT LOCATION fields as logical object identifier values (see 4.2.8.2), relative to a partition.	Mandatory	7.9.2
01h	SHORT FORM -- VENDOR SPECIFIC	Device server shall return 20 bytes of data with the FIRST LOGICAL OBJECT LOCATION and LAST LOGICAL OBJECT LOCATION fields as vendor-specific values.	Optional	7.9.2
02h	Reserved	Illegal request		
03h	Reserved	Illegal request		
04h	Reserved	Illegal request		
05h	Reserved	Illegal request		
06h	LONG FORM	Device server shall return 32 bytes of data.	Mandatory	7.9.3
07h	Reserved	Illegal request		

Table 70 — READ POSITION service action codes (part 2 of 2)

Code	Name	Description	Implementation requirements	Reference
08h	EXTENDED FORM	Device server shall return 32 bytes of data up to the maximum length specified by the ALLOCATION LENGTH field.	Optional	7.9.4
09h to 1Fh	Reserved	Illegal request		

If the device server does not implement the specified service action code, then the command shall be terminated with CHECK CONDITION status. The sense key shall be set to ILLEGAL REQUEST, and the additional sense code shall be set to INVALID FIELD IN CDB.

To maintain compatibility with earlier implementations, the service action codes 02h, 03h, 04h, 05h, and 07h shall not be implemented.

For service action codes of 00h, 01h, and 06h, if the ALLOCATION LENGTH field in the CDB is not set to zero, then the command shall be terminated with CHECK CONDITION status. The sense key shall be set to ILLEGAL REQUEST, and the additional sense code shall be set to INVALID FIELD IN CDB.

For a service action code of 08h, the ALLOCATION LENGTH field in the CDB specifies how much space has been allocated for the parameter data. If the length is not sufficient to contain the parameter data, the first portion of the parameter data shall be returned. This shall not be considered an error. If the remainder of the parameter data is required, the application client should send a new READ POSITION command with an ALLOCATION LENGTH field large enough to contain all the parameter data.

7.9.2 READ POSITION data format, short form

Table 71 specifies the READ POSITION data that shall be returned if the SERVICE ACTION field is 00h or 01h.

Table 71 — READ POSITION data format, short form

Bit Byte	7	6	5	4	3	2	1	0
0	BOP	EOP	LOCU	BYCU	Rsvd	LOLU	Obsolete	BPEW
1	PARTITION NUMBER							
2	Reserved							
3	Reserved							
4	(MSB)							
...	FIRST LOGICAL OBJECT LOCATION							
7								
8	(MSB)							
...	LAST LOGICAL OBJECT LOCATION							
11								
12	Reserved							
13	(MSB)							
...	NUMBER OF LOGICAL OBJECTS IN OBJECT BUFFER							
15								
16	(MSB)							
...	NUMBER OF BYTES IN OBJECT BUFFER							
19								

A beginning-of-partition (BOP) bit set to one specifies the logical unit is at the beginning-of-partition in the current partition. A BOP bit set to zero specifies the current logical position is not at the beginning-of-partition.

An end-of-partition (EOP) bit set to one specifies the logical unit is positioned between early-warning and end-of-partition in the current partition. An EOP bit set to zero specifies the current logical position is not between early-warning and end-of-partition.

A logical object count unknown (LOCU) bit set to one specifies the NUMBER OF LOGICAL OBJECTS IN OBJECT BUFFER field does not represent the actual number of logical objects in the object buffer. A LOCU bit set to zero specifies the NUMBER OF LOGICAL OBJECTS IN OBJECT BUFFER field is valid.

A byte count unknown (BYCU) bit set to one specifies the NUMBER OF BYTES IN OBJECT BUFFER field does not represent the actual number of bytes in the object buffer. A BYCU bit set to zero specifies the NUMBER OF BYTES IN OBJECT BUFFER field is valid.

A logical object location unknown (LOLU) bit set to one specifies the first logical object location, last logical object location, or partition number are not currently known or not otherwise obtainable. A LOLU bit set to zero specifies the FIRST LOGICAL OBJECT LOCATION, LAST LOGICAL OBJECT LOCATION, and PARTITION NUMBER fields contain valid position information.

If the Beyond Programmable Early Warning (BPEW) bit is set to one, then the logical object location is in a PEWZ or on the EOP side of EW. If the BPEW bit is set to zero, then the logical object location is not in a PEWZ and not on

the EOP side of EW. The BPEW bit shall be set to zero if the LOLU bit is set to one or if the PEWS field (see 8.5.8) is set to zero.

The PARTITION NUMBER field reports the partition number for the current logical position. If the logical unit only supports one partition for the recording volume, the PARTITION NUMBER field shall be set to zero.

The FIRST LOGICAL OBJECT LOCATION field specifies the logical object identifier associated with the current logical position. The value shall specify the logical object identifier of the next logical object to be transferred between an application client and the device server if a READ or WRITE command is enabled. If the FIRST LOGICAL OBJECT LOCATION field is not large enough to represent the number of logical objects recorded in the partition, then the device server shall return a CHECK CONDITION as specified in 4.2.8.2.

The LAST LOGICAL OBJECT LOCATION field specifies the logical object identifier associated with the next logical object to be transferred from the object buffer to the medium. If the object buffer does not contain a complete logical object or is empty, the value reported for the last logical object location shall be equal to the value reported for the first logical object location. If the LAST LOGICAL OBJECT LOCATION field is not large enough to represent the number of logical objects recorded in the partition, then the device server shall return a CHECK CONDITION as specified in 4.2.8.2.

NOTE 32 - The information provided by the FIRST LOGICAL OBJECT LOCATION and LAST LOGICAL OBJECT LOCATION fields may be used in conjunction with the LOCATE command to position the medium at the appropriate logical object on another device in the case of unrecoverable errors on the first device.

The NUMBER OF LOGICAL OBJECTS IN OBJECT BUFFER field specifies the number of logical objects in the object buffer of the logical unit that have not been written to the medium.

The NUMBER OF BYTES IN OBJECT BUFFER field specifies the total number of data bytes in the object buffer of the logical unit that have not been written to the medium.

7.9.3 READ POSITION data format, long form

Table 72 specifies the format of the READ POSITION data that shall be returned if the SERVICE ACTION field is 06h.

Table 72 — READ POSITION data format, long form

Bit Byte	7	6	5	4	3	2	1	0
0	BOP	EOP		Reserved	MPU	LONU	Rsvd	BPEW
1								
...				Reserved				
3								
4	(MSB)							
...				PARTITION NUMBER				
7								(LSB)
8	(MSB)							
...				LOGICAL OBJECT NUMBER				
15								(LSB)
16	(MSB)							
...				LOGICAL FILE IDENTIFIER				
23								(LSB)
24								
...				Obsolete				
31								

The BOP, EOP, and PARTITION NUMBER fields are as defined in the READ POSITION data format, short form (see table 71).

A mark position unknown (MPU) bit set to one specifies the logical file identifier is not known or accurate reporting is not currently available. A MPU bit set to zero specifies the LOGICAL FILE IDENTIFIER field contains valid position information.

A logical object number unknown (LONU) bit set to one specifies the logical object number or partition number are not known or accurate reporting is not currently available. A LONU bit set to zero specifies the LOGICAL OBJECT NUMBER and PARTITION NUMBER fields contain valid position information.

If the Beyond Programmable Early Warning (BPEW) bit is set to one, then the logical object location is in a PEWZ or on the EOP side of EW. If the BPEW bit is set to zero, then the logical object location is not in a PEWZ or on the EOP side of EW. The BPEW bit shall be set to zero if the LONU bit is set to one or if the PEWS field (see 8.5.8) is set to zero.

The PARTITION NUMBER field reports the partition number for the current logical position. If the logical unit only supports one partition for the volume, the PARTITION NUMBER field shall be set to zero.

The LOGICAL OBJECT NUMBER field specifies the number of logical objects between beginning-of-partition and the current logical position. A filemark counts as one logical object. If the LOGICAL OBJECT NUMBER field is not large enough to represent the number of logical objects recorded in the partition, then the device server shall return a CHECK CONDITION as specified in 4.2.8.2.

The LOGICAL FILE IDENTIFIER field specifies the number of filemarks between beginning-of-partition and the current logical position. This value is the current logical file identifier. If the LOGICAL FILE IDENTIFIER field is not large enough to represent the number of logical objects recorded in the partition, then the device server shall return a CHECK CONDITION as specified in 4.2.8.2.

7.9.4 READ POSITION data format, extended form

Table 73 specifies the format of the READ POSITION data that shall be returned if the SERVICE ACTION field is 08h.

Table 73 — READ POSITION data format, extended form

Bit Byte	7	6	5	4	3	2	1	0
0	BOP	EOP	LOCU	BYCU	Rsvd	LOLU	Obsolete	BPEW
1	PARTITION NUMBER							
2	(MSB)							
3	ADDITIONAL LENGTH (1Ch)							
4	Reserved							
5	(MSB)							
...	NUMBER OF LOGICAL OBJECTS IN OBJECT BUFFER							
7	(LSB)							
8	(MSB)							
...	FIRST LOGICAL OBJECT LOCATION							
15	(LSB)							
16	(MSB)							
...	LAST LOGICAL OBJECT LOCATION							
23	(LSB)							
24	(MSB)							
...	NUMBER OF BYTES IN OBJECT BUFFER							
31	(LSB)							

The fields are defined the same as for the corresponding fields in the READ POSITION data format, short form (see table 71).

The ADDITIONAL LENGTH field shall contain 1Ch. If the information transferred to the Data-In Buffer is truncated because of an insufficient ALLOCATION LENGTH value, the ADDITIONAL LENGTH field shall not be altered to reflect the truncation.

7.10 RECEIVE RECOMMENDED ACCESS ORDER command

The RECEIVE RECOMMENDED ACCESS ORDER command (see table 74) is used to retrieve a RAO list of user data segments (see 4.2.25).

After a GENERATE RECOMMENDED ACCESS ORDER command (see 7.3) completes successfully, the RECEIVE RECOMMENDED ACCESS ORDER command may be used to retrieve the results. The results shall not be cleared when read. See Annex F for the recommended usage of the RAO list returned in this command.

Table 74 — RECEIVED RECOMMENDED ACCESS ORDER command

Bit Byte	7	6	5	4	3	2	1	0						
0	OPERATION CODE (A3h)													
1	UDS_LIMITS	Reserved		SERVICE ACTION (1Dh)										
2	(MSB)													
...	RAO LIST OFFSET													
5														
6	(MSB)													
...	ALLOCATION LENGTH													
9														
10	Reserved				UDS_TYPE									
11	CONTROL													

See SPC-5 for the definition of the OPERATION CODE field and the SERVICE ACTION field. The OPERATION CODE field and the SERVICE ACTION field shall be set to the values shown in table 74.

The UDS_LIMITS bit set to zero specifies that the RAO list 7.10.1.2 shall be returned. The UDS_LIMITS bit set to one specifies that the UDS Limits page 7.10.1.1 shall be returned using the settings of the other fields in the CDB to determine the values supported. If the UDS_LIMITS bit is set to one and the RAO LIST OFFSET field is not set to zero, 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 RAO LIST OFFSET field specifies the offset into the RAO list from which to begin returning data. The RAO LIST OFFSET field shall be set to zero or a multiple of four.

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, if the RAO LIST OFFSET field:

- a) is not set to zero and is not a multiple of four; or
- b) is greater than or equal to the length of the RAO list 7.10.1.2.

The ALLOCATION LENGTH field specifies the maximum number of bytes that the device server is allowed to return.

The UDS_TYPE field (see table 53) specifies the format of the user data segment descriptor to use for calculating the maximum number of UDS descriptors supported, if the UDS_LIMITS bit is set to one. If the UDS_LIMITS bit set to zero, the UDS_TYPE field is ignored.

7.10.1 RECEIVE RECOMMENDED ACCESS ORDER command parameter data

7.10.1.1 UDS Limits page

The format of the parameter data returned if the UDS_LIMITS bit of the RECEIVE RECOMMENDED ACCESS ORDER command is set to 1b (i.e., return UDS Limits page) is shown in table 75.

Table 75 — UDS Limits page

Bit Byte	7	6	5	4	3	2	1	0
0	(MSB)							
1								MAXIMUM UDS SUPPORTED
2	(MSB)							
3								MAXIMUM UDS SIZE (LSB)

The MAXIMUM UDS SUPPORTED field specifies the maximum number of user data segments that the device server supports for the specified value of the UDS TYPE field.

The MAXIMUM UDS SIZE field specifies the maximum size of the RRAO - user data segment descriptor (see 7.10.1.3) for the specified value of the UDS TYPE field.

7.10.1.2 RAO list

The format of the parameter data returned if the UDS_LIMITS bit of the RECEIVE RECOMMENDED ACCESS ORDER command is set to 0b (i.e., return Recommended Access Order list) is shown in table 76.

Table 76 — RAO List

Bit Byte	7	6	5	4	3	2	1	0
0								RAO PROCESS
1								STATUS
2							Reserved	
3							Reserved	
4	(MSB)							
...							RAO DESCRIPTOR LIST LENGTH (n-7)	
7								(LSB)
							RAO user data segment descriptor list	
8							RRAO - user data segment descriptor [first] (see table 78)	
							⋮	
n							RRAO - user data segment descriptor [last] (see table 78)	

The RAO PROCESS field indicates the process (see table 16) that was used to generate the contents of the RAO list.

The STATUS field (see table 77) indicates the status of the RAO list.

The RAO DESCRIPTOR LIST LENGTH field specifies the number of bytes that follow. If the RAO list is invalid (i.e., the STATUS field is set to 000b), then the RAO DESCRIPTOR LIST LENGTH field shall be set to zero.

Table 77 — RAO list STATUS codes

Code	Description
000b	The RAO list is empty (e.g., an RAO list was never generated or the GENERATE RECOMMENDED ACCESS ORDER command invalidated the list) and the RAO DESCRIPTOR LIST LENGTH field is set to zero.
001b	The RAO list contains a user data segment descriptor for each UDS that was described in the GENERATE RECOMMENDED ACCESS ORDER command parameter list and the RAO list was generated using the process specified in the RAO PROCESS field.
010b	The device server was unable to successfully complete the process specified in the RAO PROCESS field on the GENERATE RECOMMENDED ACCESS ORDER command parameter list due to an inability to do so at this time (e.g., the tape directory is corrupted). The RAO list contains a user data segment descriptor for each UDS that was described in the GENERATE RECOMMENDED ACCESS ORDER command parameter list. The ESTIMATED LOCATE TIME TO UDS field and any additional information descriptors are present but may contain data that is not valid. The order of the user data segment descriptors in the RAO list is not specified.
all others	Reserved

Each RRAO - user data segment descriptor (see 7.10.1.3) specifies one of the UDSs from the associated GENERATE RECOMMENDED ACCESS ORDER command parameter data that was requested to be sorted into a Recommended Access Order list. The order of the User Data Segments is as specified in the STATUS field.

7.10.1.3 RRAO - user data segment descriptor

The RRAO - user data segment descriptor format is shown in table 78.

The uds DESCRIPTOR LENGTH field indicates the number of bytes to follow.

The ESTIMATED LOCATE TIME TO UDS field contents are specified by the RAO PROCESS field (see table 16). If the device server does not support ESTIMATED LOCATE TIME TO UDS then the ESTIMATED LOCATE TIME TO UDS field shall be set to zero. If the time to position to this UDS is more than FEh seconds, then the ESTIMATED LOCATE TIME TO UDS field shall be set to FFh. If the time to position to this UDS is unknown or cannot be estimated, then the STATUS field of the RAO list shall be set to 010b.

The UDS NAME field contains the name given to this User Data Segment in the GENERATE RECOMMENDED ACCESS ORDER command parameter list.

The PARTITION NUMBER field contains the number of the partition in which this user data segment is located.

The BEGINNING LOGICAL OBJECT IDENTIFIER field contains the logical object identifier of the beginning logical object of the user data segment.

The ENDING LOGICAL OBJECT IDENTIFIER field contains the logical object identifier of the ending logical object of the user data segment.

The additional information descriptors (see table 79), if any, describe additional information. Additional information descriptors are returned if the GENERATE RECOMMENDED ACCESS ORDER command UDS_TYPE field requested that those descriptors be generated as part of the RAO list generation.

The ADDITIONAL INFORMATION DESCRIPTOR LENGTH field contains the number of additional bytes that follow in the additional information descriptors.

Table 78 — RRAO - user data segment descriptor format

Bit Byte	7	6	5	4	3	2	1	0
0								
1								UDS DESCRIPTOR LENGTH (n-1)
2								Reserved
3								Reserved
4								ESTIMATED LOCATE TIME TO UDS
5								
...								UDS NAME
14								
15								PARTITION NUMBER
16	(MSB)							
...								BEGINNING LOGICAL OBJECT IDENTIFIER
23								(LSB)
24	(MSB)							
...								ENDING LOGICAL OBJECT IDENTIFIER
31								(LSB)
								Additional information descriptors
32								
								Additional information descriptor [first] (see table 79)
								⋮
n								Additional information descriptor [last] (see table 79)

Table 79 — Additional information descriptor format

Bit Byte	7	6	5	4	3	2	1	0
0	(MSB)							
1								ADDITIONAL INFORMATION DESCRIPTOR LENGTH (n-1)
2								(LSB)
3								ADDITIONAL INFORMATION TYPE
...								
n								Type specific data

The ADDITIONAL INFORMATION TYPE field (see table 80) indicates the type of additional information descriptor.

The contents of the type specific data field are defined by the additional information type specified in the ADDITIONAL INFORMATION TYPE field.

Table 80 — ADDITIONAL INFORMATION TYPE codes

Code	Description
all others	Reserved
F0h-FFh	Vendor specific

7.11 RECOVER BUFFERED DATA command

The RECOVER BUFFERED DATA command (see table 81) is used to recover data that has been transferred to the logical unit's object buffer but has not been successfully written to the medium. It is normally used to recover the buffered data after error or exception conditions make it impossible to write the buffered data to the medium. One or more RECOVER BUFFERED data commands may be required to recover all unwritten buffered data.

If the OBR bit in the Device Configuration mode page (see 8.5.3) is set to one, then this command is supported by the device server. If the OBR bit in the Device Configuration mode page is set to zero, then this command is not supported by the device server.

Table 81 — RECOVER BUFFERED data command

Bit Byte	7	6	5	4	3	2	1	0		
0	OPERATION CODE (14h)									
1	Reserved					SILI	FIXED			
2	(MSB)									
...	TRANSFER LENGTH									
4	(LSB)									
5	CONTROL									

See SPC-5 for the definition of the OPERATION CODE field. The OPERATION CODE field shall be set to the value shown in table 81.

The processing of this command is similar to the READ(6) command except that the data is transferred from the logical unit's object buffer instead of the medium. The order that logical block(s) (see 4.2.28.4) are transferred is defined by the ROBO bit in the Device Configuration mode page (see 8.5.3). If the ROBO bit is not implemented, then logical block(s) are transferred in the same order as if the ROBO bit is set to zero.

Refer to the READ(6) command (see 6.4) for a description of the FIXED bit, the SILI bit, the TRANSFER LENGTH field, and any conditions that may result from incorrect usage of these fields.

If the FIXED bit is zero, no more than the requested transfer length shall be transferred to the application client. If the requested transfer length is smaller than the actual length of the logical block to be recovered, only the requested transfer length shall be transferred to the application client and the remaining data for the current logical block shall be discarded.

NOTE 33 - During recovery operations involving unknown block sizes, the application client should set the FIXED bit to zero and select the maximum block length supported by the logical unit to ensure that all buffered data is transferred.

If a buffered filemark is encountered during a RECOVER BUFFERED DATA command, CHECK CONDITION status shall be returned, the sense key shall be set to NO SENSE, and the FILEMARK and VALID bits shall be set to one in the sense data. Upon termination, the logical position shall be after the filemark. If the FIXED bit is one, the INFORMATION field shall be set to the requested transfer length minus the actual number of logical blocks transferred. If the FIXED bit is zero, the INFORMATION field shall be set to the requested transfer length.

If an attempt is made to recover more logical blocks of data than are contained in the logical unit's object buffer, CHECK CONDITION status shall be returned, the sense key shall be set to NO SENSE. The additional sense code shall be set to END-OF-DATA DETECTED, and the EOM and VALID bits shall be set to one in the sense data. If the FIXED bit is one, the INFORMATION field shall be set to the requested transfer length minus the actual number of logical blocks transferred. If the FIXED bit is zero, the INFORMATION field shall be set to the requested transfer length.

7.12 REPORT DENSITY SUPPORT command

7.12.1 REPORT DENSITY SUPPORT command description

The REPORT DENSITY SUPPORT command (see table 82) requests that information regarding the supported densities or medium type for the logical unit be sent to the application client.

Table 82 — REPORT DENSITY SUPPORT command

Bit Byte	7	6	5	4	3	2	1	0							
0	OPERATION CODE (44h)														
1	Reserved					MEDIUM TYPE	MEDIA								
2															
...	Reserved														
6															
7	(MSB)	ALLOCATION LENGTH					(LSB)								
8															
9	CONTROL														

See SPC-5 for the definition of the OPERATION CODE field. The OPERATION CODE field shall be set to the value shown in table 82.

If the MEDIA bit is set to zero, the device server shall return descriptors for densities or medium types supported by the logical unit for any supported media. If the MEDIA bit is set to one, the device server shall return descriptors for densities or the medium type supported by the mounted volume. If the MEDIA bit is set to one and the logical unit either contains no volume or contains a volume but is unable to determine the medium density or the medium type, CHECK CONDITION status shall be returned. The sense key shall be set to NOT READY and the additional sense code shall specify the reason for NOT READY.

If the MEDIUM TYPE bit is set to zero, the device server shall return data as specified in 7.12.3. If the MEDIUM TYPE bit is set to one, the device server shall return data as specified in 7.12.4.

The ALLOCATION LENGTH field specifies the maximum number of bytes that the device server may return.

7.12.2 REPORT DENSITY SUPPORT header

The REPORT DENSITY SUPPORT header is specified in table 83.

Table 83 — REPORT DENSITY SUPPORT header

Bit Byte	7	6	5	4	3	2	1	0
0	(MSB)							
1								(LSB)
2								
3								
								Density support descriptor(s)
4								Density support descriptor [first] (see table 84 or table 85)
								⋮
n								Density support descriptor [last] (see table 84 or table 85)

See SPC-5 for the definition of the OPERATION CODE field. The OPERATION CODE field shall be set to the value shown in table 83.

The AVAILABLE DENSITY SUPPORT LENGTH field specifies the number of bytes in the following data that is available to be transferred. The available density support length does not include itself. If the parameter data is truncated due to insufficient allocation length, then the AVAILABLE DENSITY SUPPORT LENGTH field shall not be altered to reflect the truncation.

7.12.3 Density support report

The REPORT DENSITY SUPPORT command with a MEDIUM TYPE bit set to zero returns the REPORT DENSITY SUPPORT header (see table 83) followed by one or more density support data block descriptors (see table 84). The density support data block descriptors shall follow the REPORT DENSITY SUPPORT header. The density support data block descriptors shall be in numerical ascending order of the PRIMARY DENSITY CODE value.

Table 84 — Density support data block descriptor

Bit Byte	7	6	5	4	3	2	1	0											
0	PRIMARY DENSITY CODE																		
1	SECONDARY DENSITY CODE																		
2	WRTOK	DUP	DEFLT	Reserved				DLV											
3	(MSB)	Descriptor Length																	
4	(LSB)																		
5	(MSB)	Bits per mm																	
...	(LSB)																		
7	(LSB)																		
8	(MSB)	Media width																	
9	(LSB)																		
10	(MSB)	Tracks																	
11	(LSB)																		
12	(MSB)	Capacity																	
...	(LSB)																		
15	(LSB)																		
16																			
...																			
23	Assigning organization																		
24																			
...																			
31	Density name																		
32																			
...																			
51	Description																		

Density support data block descriptors shall be returned by ascending PRIMARY DENSITY CODE values. Multiple entries may exist for a given PRIMARY DENSITY CODE value. For all entries with equal PRIMARY DENSITY CODE values, all fields except for ASSIGNING ORGANIZATION, DENSITY NAME, and DESCRIPTION shall be identical. Density support data block descriptors with the same PRIMARY DENSITY CODE value should be ordered from most to least preferred ASSIGNING ORGANIZATION, DENSITY NAME, and DESCRIPTION.

NOTE 34 - By allowing multiple entries for a given primary and secondary density code set, multiple standard names may identify the same density code. This facilitates the remapping of density codes, if required.

The density support data block descriptor may represent a particular format in addition to giving physical density information. The information in a density support data block descriptor provides an application client with a detailed review of the recording technologies supported by a logical unit. By supplying the density code value returned in a density support data block descriptor in a MODE SELECT command (see 8.5), an application client selects the recording technology (density, format, etc.).

The PRIMARY DENSITY CODE field contains the value returned by a MODE SENSE command for the density described in the remainder of the density support data block descriptor. The device server shall accept a MODE SELECT command containing this value, for appropriate media. The value 7Fh shall be reserved. All other values are available for use. The value of 00h shall only be used for the default density of the logical unit.

If multiple density code values are assigned to the same recording technology (density, format, etc.), then the SECONDARY DENSITY CODE field shall contain the equivalent density code value. If the SECONDARY DENSITY CODE is used in the mode select header with a MODE SELECT command, then the device server shall accept this value as equivalent to the PRIMARY DENSITY CODE value. If no secondary density code exists, the device server shall return the PRIMARY DENSITY CODE value in this field.

A write to media ok (WRTOK) bit set to zero specifies the logical unit does not support writing to the medium with this density. A WRTOK bit set to one specifies the logical unit is capable of writing this density to either the currently mounted volume (i.e., MEDIA bit in command descriptor block set to one) or for some media (i.e., MEDIA bit in command descriptor block set to zero). All density code values returned by the REPORT DENSITY SUPPORT command shall be supported for read operations.

A duplicated (DUP) bit set to zero specifies this primary density code has exactly one density support data block descriptor. A DUP bit set to one specifies this primary density code is specified in more than one density support data block descriptor.

A default (DEFLT) bit set to zero specifies this density is not the default density of the drive. A DEFLT bit set to one specifies this density is the default density. If either the PRIMARY DENSITY CODE or the SECONDARY DENSITY CODE field is zero, the DEFLT bit shall be one. If neither the primary or secondary density code is zero and the DEFLT bit is one, the logical unit shall accept a MODE SELECT header with a density code of 00h as equivalent to the primary and secondary density codes.

NOTE 35 - The default density of the logical unit may vary depending on the currently mounted media. Multiple codes may return a DEFLT bit set to one if the MEDIA bit is zero, since more than one default may be possible.

If the descriptor length valid (DLV) bit is set to one, the DESCRIPTOR LENGTH field shall contain the length of the descriptor minus 5. If the DLV bit is set to zero, the DESCRIPTOR LENGTH field shall be set to zero and the descriptor is 52 bytes in length.

The BITS PER MM field specifies the number of bits per millimeter per track as recorded on the medium. The value in this field shall be rounded up if the fractional value of the actual value is greater than or equal to 0.5. A value of 00h specifies the number of bits per millimeter does not apply to this logical unit. Direct comparison of this value between different vendors (possibly products) is discouraged since the definition of bits may vary.

The MEDIA WIDTH field specifies the width of the medium supported by this density. This field has units of tenths of millimeters. The value in this field shall be rounded up if the fractional value of the actual value is greater than or equal to 0.5. The MEDIA WIDTH field may vary for a given density depending on the mounted volume. A value of 00h specifies the width of the medium does not apply to this logical unit.

The TRACKS field specifies the number of data tracks supported on the medium by this density. The TRACKS value may vary for a given density depending on the mounted volume. Direct comparison of this value between different vendors or products is discouraged since the definition of the number of tracks may vary. For recording formats that are neither parallel nor serpentine, the TRACKS field specifies the maximum number of data tracks that are read or recorded simultaneously.

If the MEDIA bit is zero, the CAPACITY field specifies the approximate capacity of the longest supported medium assuming recording in this density with one partition. If the MEDIA bit is one, the CAPACITY field should specify the approximate capacity of the current medium, assuming recording in this density with one partition. If the approximate capacity of the current medium is not available for the mounted volume, the longest supported

medium capacity shall be used. If a SET CAPACITY command has affected the capacity of the medium, this shall be reflected in the CAPACITY field. The capacity assumes that compression is disabled, if possible. If this density does not support an uncompressed format, the capacity assumes that compression is enabled using average data. The capacity also assumes that the media is in good condition, and that normal data and block sizes are used. This value is in units of megabytes (i.e., 10^6 bytes). The logical unit does not guarantee that this space is actually available in all cases. Direct comparison of this value between different vendors (possibly products) is discouraged since the length of media and the method used to measure maximum capacity may vary. The CAPACITY field is intended to be used by the application client to determine that the correct density is being used, particularly if a lower-density format is required for interchange.

The ASSIGNING ORGANIZATION field contains eight bytes of ASCII data identifying the organization responsible for the specifications defining the values in this density support data block descriptor. The data shall be left aligned within this field. The ASCII value for a space (20h) shall be used if padding is required. The ASSIGNING ORGANIZATION field should contain a value listed in the vendor identification list (see SPC-4). The use of a specific vendor identification, other than the one associated with the device is allowed.

NOTE 36 - If vendor X defines a density and format, another vendor may use X in the ASSIGNING ORGANIZATION field. If exactly the same density and format construction later becomes known by another name, both X and the new assigning organization may be used for the density code. This is one condition that may result in multiple density support data block descriptors for a single density code value.

NOTE 37 - It is intended that the ASSIGNING ORGANIZATION field contain a unique identification of the organization responsible for the information in a density support data block descriptor. In the absence of any formal registration procedure, T10 maintains a list of vendor and assigning organization identification codes in use. Vendors are requested to voluntarily submit their identification codes to prevent duplication of codes.

The DENSITY NAME field contains eight bytes of ASCII data identifying the document, or other identifying name, that is associated with this density support data block descriptor. The data shall be left aligned within this field. The ASCII value for a space (20h) shall be used if padding is required. Two physical densities, and possibly formats, shall not have identical ASSIGNING ORGANIZATION and DENSITY NAME fields. Assigning organizations are responsible for preventing duplicate usage of one density name for multiple different densities and/or formats.

NOTE 38 - It is suggested that any document that specifies a format and density for the media should contain the values to be used by a logical unit if reporting the density support. The values for the BITS PER MM, MEDIA WIDTH, and TRACKS should also be included in such a document to help maintain consistency.

The DESCRIPTION field contains twenty bytes of ASCII data describing the density. The data shall be left aligned within this field. The ASCII value for a space (20h) shall be used if padding is required.

7.12.4 Medium type support report

The REPORT DENSITY SUPPORT command with a MEDIUM TYPE bit set to one returns the REPORT DENSITY SUPPORT header (see table 83) followed by one or more medium type descriptors (see table 85). The medium

type descriptors shall follow the REPORT DENSITY SUPPORT header. The medium type descriptors shall be in numerical ascending order of the medium type value.

Table 85 — Medium type descriptor

Bit Byte	7	6	5	4	3	2	1	0
0								MEDIUM TYPE
1								Reserved
2	(MSB)							
3								DESCRIPTOR LENGTH (52) (LSB)
4								NUMBER OF DENSITY CODES
5	(MSB)							
...								PRIMARY DENSITY CODES (LSB)
13								
14	(MSB)							
15								MEDIA WIDTH (LSB)
16	(MSB)							
17								MEDIUM LENGTH (LSB)
18								Reserved
19								Reserved
20	(MSB)							
...								ASSIGNING ORGANIZATION (LSB)
27								
28	(MSB)							
...								MEDIUM TYPE NAME (LSB)
35								
36	(MSB)							
...								DESCRIPTION (LSB)
55								

The MEDIUM TYPE field contains the value returned by a MODE SENSE command for the medium type described in the remainder of the medium type descriptor. The device server shall accept a MODE SELECT command containing this value, for appropriate media.

The DESCRIPTOR LENGTH field contains the length of the descriptor minus 4.

The NUMBER OF DENSITY CODES field contains the number of valid density codes present in the PRIMARY DENSITY CODES value field.

The PRIMARY DENSITY CODES field contains a list of primary density code values supported by the drive for the medium type. The primary density code values shall be listed in ascending order. Any unused bytes in this field shall be set to zero.

The MEDIA WIDTH field specifies the width of the medium. This field has units of tenths of millimeters. The value in this field shall be rounded up if the fractional portion of the actual value is greater than or equal to 0.5.

The MEDIUM LENGTH field specifies the nominal length of the medium. This field has units of meters. The value in this field shall be rounded up if the fractional portion of the actual value is greater than or equal to 0.5.

The ASSIGNING ORGANIZATION field contains eight bytes of ASCII data identifying the organization responsible for the specifications defining the values in this medium type descriptor. The data shall be left aligned within this field. The ASCII value for a space (20h) shall be used if padding is required. The ASSIGNING ORGANIZATION field should contain a value listed in the vendor identification list (see SPC-4). The use of a vendor identification other than the one associated with the device is allowed.

NOTE 39 - It is intended that the ASSIGNING ORGANIZATION field contain a unique identification of the organization responsible for the information in a medium type descriptor. In the absence of any formal registration procedure, T10 maintains a list of vendor and assigning organization identification codes in use. Vendors are requested to voluntarily submit their identification codes to prevent duplication of codes.

The MEDIUM TYPE NAME field contains eight bytes of ASCII data identifying the document (or other identifying name) that is associated with this medium type descriptor. The data shall be left aligned within this field. The ASCII value for a space (20h) shall be used if padding is required. Two different medium types shall not have identical ASSIGNING ORGANIZATION and MEDIUM TYPE NAME fields. Assigning organizations are responsible for preventing duplicate usage of one medium type name for multiple different medium types.

NOTE 40 - It is suggested that any document that specifies characteristics for the media should contain the values to be used by a logical unit if reporting the density support. The values for the MEDIUM WIDTH and MEDIUM LENGTH should also be included in such a document to help maintain consistency.

The DESCRIPTION field contains twenty bytes of ASCII data describing the medium type. The data shall be left aligned within this field. The ASCII value for a space (20h) shall be used if padding is required.

7.13 REWIND command

The REWIND command (see table 86) causes the logical unit to position to the beginning-of-partition in the current partition. Prior to performing the rewind operation, the device server shall perform a synchronize operation (see 4.2.14). If the buffered mode is not 0h (see 8.5), a previous command was terminated with CHECK CONDITION status, and the device is unable to continue successfully writing, then the logical unit shall discard any unwritten buffered logical objects prior to performing the REWIND operation.

Table 86 — REWIND command

Bit Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (01h)							
1	Reserved							IMMED
2								
...	Reserved							
4								
5	CONTROL							

See SPC-5 for the definition of the OPERATION CODE field. The OPERATION CODE field shall be set to the value shown in table 86.

An immediate (IMMED) bit set to zero specifies the device server shall not return status until the rewind operation has completed. If the IMMED bit is one, the device server shall return status as soon as all buffered logical objects

have been written to the medium and the command descriptor block of the REWIND command has been validated. If CHECK CONDITION status is returned for a REWIND command with an IMMED bit set to one, the rewind operation shall not be performed.

NOTE 41 - For compatibility with devices implemented prior to this standard, it is suggested that a WRITE FILEMARKS command with an IMMED bit set to zero be used to perform a synchronize operation (see 4.2.14) before issuing a REWIND command with an IMMED bit set to one.

7.14 SET CAPACITY command

The SET CAPACITY command (see table 87) sets the available medium for the currently mounted recording volume to a proportion of the total medium for use. Any excess space shall be unavailable on the medium after successful completion of this command until changed by a new SET CAPACITY command. This change shall persist through power cycles, logical unit resets, L_T nexus losses, and unloading or reloading of the recording volume. Other vendor-specific actions such as physical erasure may change the total capacity of the recording volume. The method for recording the available medium for use and other marks needed to manage the resulting medium for use for volume interchange may be specified in a recording format standard or may be vendor specific.

Table 87 — SET CAPACITY command

Bit Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (0Bh)							
1	Reserved							IMMED
2	Reserved							
3	(MSB)	MEDIUM FOR USE PROPORTION VALUE						
4								
5	CONTROL							

See SPC-5 for the definition of the OPERATION CODE field. The OPERATION CODE field shall be set to the value shown in table 87.

If the device server does not contain a volume, then the command shall be terminated with CHECK CONDITION status. The sense key shall be set to NOT READY, and the additional sense code shall be set to MEDIUM NOT PRESENT.

The SET CAPACITY command shall be accepted only if the medium is at beginning-of-partition 0 (BOP 0). If the medium is logically at any other position, the command shall be rejected with CHECK CONDITION status. The sense key shall be ILLEGAL REQUEST and the additional sense code shall be set to POSITION PAST BEGINNING OF MEDIUM.

A valid SET CAPACITY command shall cause all data and partitioning information on the entire physical recording volume to be lost. If the partitioning information changes, the device server shall establish a unit attention condition for all initiators with the additional sense code set to MODE PARAMETERS CHANGED.

An immediate (IMMED) bit set to zero specifies the device server shall not return status until the set capacity operation has completed. An IMMED bit set to one specifies the device server shall return status as soon as the command descriptor block of the SET CAPACITY command has been validated. If CHECK CONDITION status is returned for a SET CAPACITY command with an IMMED bit set to one, the set capacity operation shall not be performed.

The MEDIUM FOR USE PROPORTION VALUE field specifies the portion of the total medium to be made available for use. The MEDIUM FOR USE PROPORTION VALUE field is the numerator to a fraction with a denominator of 65 535. The resulting available medium for use on the recording volume shall be equal to the total medium for use multiplied by this fraction. The device server may round up the medium for use to the next highest supported value. This rounding shall not be considered an error and shall not be reported.

NOTE 42 - The MEDIUM FOR USE PROPORTION VALUE relates to the physical proportion of the medium that is available for use to record logical objects and should not be limited by the maximum number of logical objects that the device server is capable of supporting. This maintains the orthogonality between the recording volume and the logical unit that exists due to recording volumes being transferred to logical units that may support a different number of logical objects.

NOTE 43 - The available medium for use affects the approximate capacity values. The available capacity may be affected by defects that reduce the actual available capacity of the recording volume. Other factors, such as partitioning, compression, and logical block packing may also affect available capacity.

7.15 SPACE(16) command

The SPACE(16) command (see table 88) operates identically to the SPACE(6) command (see 6.6), but allows specifying a COUNT field up to eight bytes in length and has parameter data out that specifies the logical object identifier on the medium. Following completion of a SPACE(16) command a READ POSITION command should be issued to obtain positioning information.

Table 88 — SPACE(16) command

Bit Byte	7	6	5	4	3	2	1	0				
0	OPERATION CODE (91h)											
1	Reserved				CODE							
2	Reserved											
3	Reserved											
4	(MSB)											
...	COUNT											
11	(LSB)											
12	(MSB)											
13	PARAMETER LENGTH											
14	(LSB)											
15	Reserved											
	CONTROL											

See SPC-5 for the definition of the OPERATION CODE field. The OPERATION CODE field shall be set to the value shown in table 88.

The CODE field specifies either the type of logical objects to be spaced over or the termination point of the operation. The CODE field is defined in table 44 (see 6.6).

The COUNT field contains a value in two's complement notation specifying the number of logical objects (e.g., logical blocks if the CODE field is set to 0000b or filemarks if the CODE field is set to 0001b) to be spaced over and the direction of movement. If the CODE field is set to 0011b (i.e., end-of-data), then the COUNT field shall be ignored.

If spacing over logical objects, the magnitude of the value in the COUNT field specifies the number of logical objects to be spaced over in the current partition and the sign of the value in the COUNT field specifies the direction of movement.

If the COUNT field contains a positive value N and the code field does not contain 0011b (i.e., end-of-data), then the device server shall position in the forward direction (i.e., toward end-of-partition) over N logical objects ending on the end-of-partition side of the last logical object. If the device server reaches end-of-partition before positioning over N logical objects, then it shall end movement at end-of-partition.

If the COUNT field contains a zero value and the CODE field does not contain 0011b (i.e., end-of-data), then the device server shall not change logical position.

If the COUNT field contains a negative value $-N$, in two's complement notation, and the CODE field does not contain 0011b (i.e., end-of-data) then the device server shall position in the reverse direction (i.e., toward beginning-of-partition) over N logical objects ending on the beginning-of-partition side of the last logical object. If the device server reaches beginning-of-partition before positioning over N logical objects, then it shall end movement at beginning-of-partition.

If the CODE field is 0011b (i.e., end-of-data), the COUNT field shall be ignored and the device server shall perform a synchronize operation before moving before the end-of-data position. If the COUNT field is zero and the CODE field is not 0011b (i.e., end-of-data), then a device server is not required to perform a synchronize operation. Support of spacing in the reverse direction is optional.

The PARAMETER LENGTH field is used to send parameter data space positioning information specifying the position on the medium from which to start the SPACE(16) command function. For an implicit block address mode command, the PARAMETER LENGTH field shall be set to 0. For an explicit block address mode command, the PARAMETER LENGTH field shall be set to 16. If the PARAMETER LENGTH field is set to any other value, the command shall be terminated with a CHECK CONDITION status. The sense key shall be set to ILLEGAL REQUEST and the additional sense code shall be set to INVALID FIELD IN CDB.

Space positioning information is specified in table 89.

Table 89 — Space positioning information

Bit Byte	7	6	5	4	3	2	1	0
0								
...								
2								
3								
4	(MSB)							
...								
11								
12								
...								
15								

The LOGICAL OBJECT IDENTIFIER and PARTITION fields specify the position where the SPACE(16) command shall start. If the current logical position does not match the specified LOGICAL OBJECT IDENTIFIER and PARTITION fields, the device server shall perform a locate operation to the specified logical object identifier and partition prior to performing the space operation. If the locate operation fails, the device server shall return CHECK CONDITION

status and the additional sense code shall be set to LOCATE OPERATION FAILURE. Following a locate operation failure the logical position is undefined.

NOTE 44 - Locating to the logical object identifier prior to performing the space operation is necessary for the space operation to function properly if filemarks are between the starting logical object identifier and the ending logical object identifier of the space operation.

If a filemark is encountered while spacing over logical blocks, the command shall be terminated. CHECK CONDITION status shall be returned, and the FILEMARK and VALID bits shall be set to one in the sense data. The sense key shall be set to NO SENSE and the additional sense code shall be set to FILEMARK DETECTED. The INFORMATION field shall be set to the number of logical objects to be spaced over minus the actual number of logical objects spaced over not including the filemark. The logical position shall be on the end-of-partition side of the filemark if movement was in the forward direction and on the beginning-of-partition side of the filemark if movement was in the reverse direction.

NOTE 45 - For some space operations using the SPACE(16) command, the INFORMATION field value may exceed the maximum value allowed in the fixed format sense data (see SPC-4). As such, the descriptor format sense data (see SPC-4) should be enabled (i.e., the D_SENSE bit is set to one in the Control mode page).

If early-warning is encountered while spacing over logical objects and the REW bit is set to one in the Device Configuration mode page (see 8.5.3), CHECK CONDITION status shall be returned, the sense key shall be set to NO SENSE, and the EOM and VALID bits shall be set to one in the sense data. The additional sense code shall be set to END-OF-PARTITION/MEDIUM DETECTED. The INFORMATION field shall be set to the requested count minus the actual number of logical objects spaced over as defined by the CODE value. If the REW bit is zero or the option is not supported by the logical unit, the device server shall not report CHECK CONDITION status at the early-warning point.

NOTE 46 - Setting the REW bit to one is not recommended for most applications since data may be present after early-warning.

If end-of-data is encountered while spacing over logical objects, CHECK CONDITION status shall be returned, the sense key shall be set to BLANK CHECK, and the sense data VALID bit shall be set to one in the sense data. The additional sense code shall be set to END-OF-DATA DETECTED. The sense data EOM bit shall be set to one if end-of-data is encountered at or after early-warning. The INFORMATION field shall be set to the requested count minus the actual number of logical objects spaced over as defined by the CODE value. The medium shall be positioned after the last logical object.

If the end-of-partition is encountered while spacing forward over logical objects, CHECK CONDITION status shall be returned, and the sense key shall be set to MEDIUM ERROR. The additional sense code shall be set to END-OF-PARTITION/MEDIUM DETECTED, and the sense data EOM and VALID bit shall be set to one. The INFORMATION field shall be set to the requested count minus the actual number of logical objects spaced over as defined by the CODE value. The medium position following this condition is not defined.

If beginning-of-partition is encountered while spacing over logical objects in the reverse direction, the device server shall return CHECK CONDITION status and shall set the sense key to NO SENSE. The additional sense code shall be set to BEGINNING-OF-PARTITION/MEDIUM DETECTED. The sense data EOM and VALID bits shall be set to one, and the INFORMATION field set to the total number of logical objects not spaced over as defined by the CODE value (i.e., the requested number of logical objects minus the actual number of logical objects spaced over as defined by the CODE value). The medium position following this condition is not defined. A successfully completed SPACE(16) command shall not set EOM to one at beginning-of-partition.

If spacing over sequential filemarks, the count field is interpreted as follows:

- a) a positive value N shall cause forward movement to the first occurrence of N or more consecutive filemarks being logically positioned after the N^{th} filemark;
- b) a zero value shall cause no change in the logical position; or
- c) a negative value $-N$, in two's complement notation, shall cause reverse movement to the first occurrence of N or more consecutive filemarks being logically positioned on the beginning-of-partition side of the N^{th} filemark.

If end-of-partition is encountered while spacing to sequential filemarks, CHECK CONDITION status shall be returned, and the sense key shall be set to MEDIUM ERROR. The additional sense code shall be set to END-OF-PARTITION/MEDIUM DETECTED, the EOM bit shall be set to one, and the VALID bit shall be set to zero in the sense data. The medium position following this condition is not defined.

If end-of-data is encountered while spacing to sequential filemarks, CHECK CONDITION status shall be returned, and the sense key shall be set to BLANK CHECK. The additional sense code shall be set to END-OF-DATA DETECTED, and the sense data VALID bit shall be set to zero. The medium shall be positioned after the last logical object. The sense data EOM bit shall be set to one if end-of-data is encountered at or after early-warning.

If spacing to end-of-data, the COUNT field is ignored. Upon successful completion, the medium shall be positioned after the last logical object.

If end-of-partition is encountered while spacing to end-of-data, CHECK CONDITION status shall be returned, and the sense key shall be set to MEDIUM ERROR. The additional sense code shall be set to END-OF-PARTITION/ MEDIUM DETECTED, the EOM bit shall be set to one, and the VALID bit shall be set to zero in the sense data. The medium position following this condition is not defined.

7.16 WRITE DYNAMIC RUNTIME ATTRIBUTE command

The WRITE DYNAMIC RUNTIME ATTRIBUTE command (see table 90) requests the device server to write dynamic attribute information to the device. The initiator type attributes listed in 8.2.2.4 may be changed by a WRITE DYNAMIC RUNTIME ATTRIBUTE command

Table 90 — WRITE DYNAMIC RUNTIME ATTRIBUTE command

Bit Byte	7	6	5	4	3	2	1	0				
0	OPERATION CODE (A4h)											
1	Reserved			SERVICE ACTION (1Eh)								
2												
...	Reserved											
5												
6	(MSB)											
...	PARAMETER LIST LENGTH											
9												
10	(LSB)											
11	Reserved											
	CONTROL											

See SPC-5 for the definition of the OPERATION CODE field, the SERVICE ACTION field, and the PARAMETER LIST LENGTH field. The OPERATION CODE field and the SERVICE ACTION field shall be set to the values shown in table 90.

The CONTROL byte is defined in SAM-5.

The WRITE DYNAMIC RUNTIME ATTRIBUTE parameter list format is shown in table 91.

Table 91 — WRITE DYNAMIC RUNTIME ATTRIBUTE parameter list format

Bit Byte	7	6	5	4	3	2	1	0
0								
...								
7								
Attribute list								
8	(MSB)							
9								(LSB)
:								
n-1	(MSB)							
n								
Attribute [first] (see 8.2.1)								
Attribute [last] (see 8.2.1)								
(LSB)								

If an attribute that is not an initiator type attribute is sent in the list of attributes, then this shall not be considered an error, the attribute shall be ignored, and the remaining attributes shall be processed normally.

Attributes shall be sent in ascending order by attribute identifier. If the attributes are not in order, then no attributes shall be changed and the WRITE DYNAMIC RUNTIME ATTRIBUTE 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 PARAMETER LIST. The device server shall process attributes in the order received.

If an attribute that requires creation of that attribute is sent in the list of attributes and there are no available resources to create that attribute, then all attributes for which there are resources shall be changed and the WRITE DYNAMIC RUNTIME ATTRIBUTE command shall be terminated with CHECK CONDITION status with the sense key set to RECOVERED ERROR, and the additional sense code set to ROUNDED PARAMETER.

Editors Note 2 - CCB: See Editors Note 1 - this text repeats the same instructions and doesn't describe what to do if no attributes are created.

If the WRITE DYNAMIC RUNTIME ATTRIBUTE command parameter data contains an attribute with an ATTRIBUTE LENGTH field (see 8.2.1) set to zero, then one of the following actions shall occur:

- a) if the attribute state is unsupported or read only (see 8.2.1), then the attribute shall not be changed and the WRITE DYNAMIC RUNTIME ATTRIBUTE command shall continue processing and this shall not be considered an error;
- b) if the attribute state is read/write, the attribute shall be changed to the nonexistent state. This attribute shall not be returned in response to a READ DYNAMIC RUNTIME ATTRIBUTE command; or
- c) if the attribute state is nonexistent, the attribute in the WRITE ATTRIBUTE command parameter list shall be ignored and this shall not be considered an error.

If the WRITE DYNAMIC RUNTIME ATTRIBUTE command parameter data contains an attribute with an ATTRIBUTE LENGTH set to a non-zero value that is outside the range of the value specified in 8.2.2.4 for that attribute, then one of the following actions shall occur:

- a) if the format is not ASCII, then the attribute shall be ignored; or
- b) if the format is ASCII, then the attribute shall be:
 - A) truncated to the length specified in 8.2.2.4; or
 - B) ignored.

If the parameter list length results in the truncation of an attribute, the truncated attribute shall be ignored.

8 Parameters for sequential-access devices

8.1 Diagnostic parameters

This subclause defines the descriptors and pages for diagnostic parameters used with sequential-access devices.

The diagnostic page codes for sequential-access devices are specified in table 92.

Table 92 — Diagnostic page codes

Page code	Description	Reference
00h	Supported diagnostic pages	SPC-4
01h to 3Fh	Reserved (for all device types)	
40h to 7Fh	Reserved	
80h to FFh	Vendor specific	

8.2 Dynamic runtime attributes

8.2.1 Attribute format

The dynamic runtime attribute format used for attributes in the parameter data for the WRITE DYNAMIC RUNTIME ATTRIBUTE command (see 7.16) and the READ DYNAMIC RUNTIME ATTRIBUTE command (see 7.8) is shown in table 93.

Table 93 — Dynamic runtime attribute format

Bit Byte	7	6	5	4	3	2	1	0
0	(MSB)							
1				I_T NEXUS INDEX				(LSB)
2	(MSB)							
3				ATTRIBUTE IDENTIFIER				(LSB)
4	READ ONLY			Reserved			FORMAT	
5	(MSB)							
...				ATTRIBUTE LENGTH				
8								(LSB)
9								
...				ATTRIBUTE VALUE				
n								

The I_T NEXUS INDEX field contains an index associated to the I_T_L nexus by the device server. The method for maintaining the I_T NEXUS INDEX is not specified by this standard. The I_T nexus index association may change from one command to another. On a read the value of 0000h in the I_T NEXUS INDEX field indicates that there is no I_T_L nexus associated with the attribute specified by the ATTRIBUTE IDENTIFIER field. The device server shall set the I_T NEXUS INDEX field to 0000h in the device type attributes. In the parameter data for a WRITE DYNAMIC

RUNTIME ATTRIBUTE command the I_T NEXUS INDEX field should be set to 0000h by the application client and the device server shall ignore the value in the I_T NEXUS INDEX field and shall set the I_T NEXUS INDEX field of the stored attribute to the I_T nexus index value associated with the I_T_L nexus through which the command was received. A value of FFFFh is reserved.

The ATTRIBUTE IDENTIFIER field contains a code value identifying the attribute (see 8.2.2).

The READ ONLY bit indicates whether the attribute is in the read only state (see 8.2.1). If an attribute is not in the non-existent state or the unsupported state and the READ ONLY bit is set to one, the attribute is in the Read Only state. If an attribute is not in the non-existent state or the unsupported state and the READ ONLY bit is set to zero, then the attribute is in the Read/Write state.

The FORMAT field (see table 94) specifies the format of the data in the ATTRIBUTE VALUE field.

Table 94 — Dynamic runtime attribute FORMAT field

Format	Name	Description
00b	BINARY	The ATTRIBUTE VALUE field contains binary data.
01b	ASCII	The ATTRIBUTE VALUE Field contains left-aligned ASCII data (see SPC-5).
10b - 11b	n/a	Reserved

The ATTRIBUTE LENGTH field specifies the length in bytes of the ATTRIBUTE VALUE field. If the ATTRIBUTE LENGTH field is set to zero, then there is no ATTRIBUTE VALUE field.

The ATTRIBUTE VALUE field contains the current value of the attribute (see 8.2.1), for the READ DYNAMIC RUNTIME ATTRIBUTE command (see 7.8), or intended value of the attribute, for the WRITE DYNAMIC RUNTIME ATTRIBUTE command (see 7.16).

8.2.2 Attribute identifier values

8.2.2.1 Attribute identifier values overview

The values in the ATTRIBUTE IDENTIFIER field (see 8.2.1) are assigned based on the attribute type as shown in table 95.

Table 95 — Dynamic runtime attribute identifier range assignments

Attribute Identifiers	Attribute Type	Reference
0000h to 07FFh	Device	8.2.2.2
1000h to 13FFh	Target	8.2.2.3
1800h to 1BFFh	Initiator	8.2.2.4
all others	Reserved	

Devices that support dynamic runtime attributes shall process a WRITE DYNAMIC RUNTIME ATTRIBUTE command containing initiator type attribute identifier values (i.e., 1800h to 1BFFh).

8.2.2.2 Device type attributes

Device type attributes (see table 96) shall be maintained and updated by the device server. All supported device type attributes shall have a status of read only (see 8.2).

Table 96 — Device type attributes

Attribute Identifier	Name	Attribute Length (in bytes)	Format	Reference
0000h	NUMBER OF I_T NEXUSES SUPPORTED BY DYNAMIC RUNTIME ATTRIBUTES	2	BINARY	8.2.2.2.2
0001h	TIMESTAMP WHEN PROCESSED	12	BINARY	8.2.2.2.3
0010h	RESERVATION INFORMATION	variable	BINARY	8.2.2.2.4
0011h	REGISTRATION INFORMATION	variable	BINARY	8.2.2.2.5
0012h	PREVENT ALLOW MEDIUM REMOVAL INFORMATION	variable	BINARY	8.2.2.2.6
0013h	LAST FAILED RESERVATION	variable	BINARY	8.2.2.2.7
all others	Reserved			

8.2.2.2.1 I_T_L nexus identifying information descriptor

Attributes may contain one or more I_T_L nexus identifying information descriptors. The format of the I_T_L nexus identifying information descriptor is shown in table 97.

Table 97 — I_T_L nexus identifying information format

The I_T_L NEXUS IDENTIFYING INFORMATION LENGTH field indicates the number of bytes to follow.

The TIMESTAMP PARAMETER DATA LENGTH field indicates the number of bytes of timestamp parameter data that follow.

The TIMESTAMP ORIGIN field is defined in SPC-4.

The ATTRIBUTE CREATION TIME field contains the timestamp maintained by the device server at the time the attribute was created.

The TARGET TYPE ATTRIBUTES LIST LENGTH field indicates the length of the target type attributes that follow.

The target type attributes shall be listed in order by I_T NEXUS INDEX field value and ATTRIBUTE IDENTIFIER field value. The I_T NEXUS INDEX in this list is from the time the attribute was created and may be a different value than the I_T NEXUS INDEX associated with this I_T nexus at the time the READ DYNAMIC RUNTIME ATTRIBUTE is processed.

The INITIATOR TYPE ATTRIBUTES LIST LENGTH field indicates the length of the initiator type attributes that follow.

The initiator type attributes shall be listed in order by I_T NEXUS INDEX field value and ATTRIBUTE IDENTIFIER field value. The I_T NEXUS INDEX in this list is from the time the attribute was created and may be a different value than the I_T NEXUS INDEX associated with this I_T nexus at the time the READ DYNAMIC RUNTIME ATTRIBUTE is processed.

8.2.2.2 NUMBER OF I_T NEXUSES SUPPORTED BY DYNAMIC RUNTIME ATTRIBUTES attribute

The NUMBER OF I_T NEXUSES SUPPORTED BY DYNAMIC RUNTIME ATTRIBUTES attribute contains the maximum number of instances of target type attributes and initiator type attributes. This attribute is persistent and is neither created nor destroyed.

8.2.2.3 TIMESTAMP WHEN PROCESSED attribute

The TIMESTAMP WHEN PROCESSED attribute contains the timestamp when the READ DYNAMIC RUNTIME ATTRIBUTE command that returns this attribute is processed. The DYNAMIC RUNTIME ATTRIBUTE VALUE field of the TIMESTAMP WHEN PROCESSED attribute format is the same as the REPORT TIMESTAMP parameter data format (see SPC-4). This attribute is created during the processing of the READ DYNAMIC RUNTIME ATTRIBUTE command and is destroyed upon completion of processing that command.

8.2.2.2.4 RESERVATION INFORMATION attribute

The DYNAMIC RUNTIME ATTRIBUTE VALUE field of the RESERVATION INFORMATION attribute is the list of I_T_L nexus identifying information for each I_T_L nexus that is a reservation holder. The DYNAMIC RUNTIME ATTRIBUTE VALUE field of the RESERVATION INFORMATION attribute is shown in table 98.

Table 98 — RESERVATION INFORMATION dynamic runtime attribute value format

Bit Byte	7	6	5	4	3	2	1	0
0	RESERVATION TYPE							
1	Reserved							
I_T_L nexus identifying information descriptor list								
2								
...	I_T_L nexus identifying information descriptor [first] (see 8.2.2.2.1)							
:								
...								
n	I_T_L nexus identifying information descriptor [last] (see 8.2.2.2.1)							

The RESERVATION TYPE field contains a reservation type (see table 99).

Table 99 — RESERVATION TYPE codes

Code	Reservation type
00h - 0Fh	Persistent reservation type (see SPC-4)
10h	SPC-2 reservation (see SPC-2)
11h - FFh	Reserved

Each I_T_L nexus identifying information descriptor is a copy of the I_T_L nexus identifying information for an I_T_L nexus when a reservation is created by that I_T_L nexus or when that I_T_L nexus joins the reservation as a reservation holder. The RESERVATION INFORMATION attribute is created and an I_T_L nexus identifying information descriptor is created and added to the list when an I_T_L nexus reserves the logical unit with a PERSISTENT RESERVE OUT command (see SPC-4) or a RESERVE command (see SPC-2). Other I_T_L nexus identifying information descriptors are created for each I_T_L nexus that is a reservation holder, if any (e.g., due to receipt of a PERSISTENT RESERVE OUT command or due to already registered I_T_L nexuses if an ALL REGISTRANTS type reservation is created). When an I_T_L nexus is no longer a reservation holder (e.g., a PERSISTENT RESERVE OUT command with a PREEMPT service action for that I_T_L nexus is processed), then the I_T_L nexus identifying information descriptor related to that I_T_L nexus shall be removed from the list. If the reservation is released the RESERVATION INFORMATION attribute shall be destroyed.

8.2.2.2.5 REGISTRATION INFORMATION attribute

The DYNAMIC RUNTIME ATTRIBUTE VALUE field of the REGISTRATION INFORMATION dynamic runtime attribute contains the list of I_T_L nexus identifying information descriptors for each I_T_L nexus that is registered for a

persistent reservation. The DYNAMIC RUNTIME ATTRIBUTE VALUE field of the REGISTRATION INFORMATION dynamic runtime attribute is shown in table 100.

Table 100 — REGISTRATION INFORMATION dynamic runtime attribute value format

Each I_T_L nexus identifying information descriptor is created and added to the list when an I_T_L nexus registers with a PERSISTENT RESERVE OUT command. When an I_T_L nexus is no longer registered either due to a PERSISTENT RESERVE OUT command to unregister or the registration is removed as a side effect of an event that occurs, the I_T_L nexus identifying information descriptor related to that I_T_L nexus shall be removed from the list. When the last I_T_L nexus identifying information descriptor has been removed the REGISTRATION INFORMATION attribute shall be destroyed.

8.2.2.2.6 PREVENT ALLOW MEDIUM REMOVAL INFORMATION attribute

The DYNAMIC RUNTIME ATTRIBUTE VALUE field of the PREVENT ALLOW MEDIUM REMOVAL INFORMATION attribute is the list of I_T_L nexus identifying information for each I_T_L nexus that has prevented media removal. The DYNAMIC RUNTIME ATTRIBUTE VALUE field of the PREVENT ALLOW MEDIUM REMOVAL INFORMATION attribute is shown in table 101.

Table 101 — PREVENT ALLOW MEDIUM REMOVAL INFORMATION dynamic runtime attribute value format

Each I_T_L nexus identifying information descriptor is created and added to the list when a volume's removal is prevented due to a PREVENT ALLOW MEDIUM REMOVAL command received from the I_T_L nexus associated with that I_T_L nexus identifying information (i.e., the ATTRIBUTE CREATION TIME field is set to the time the PREVENT ALLOW REMOVAL command with the PREVENT bit set to one is received through that I_T_L nexus). When an I_T_L nexus no longer prevents medium removal (e.g., a PREVENT ALLOW MEDIUM REMOVAL command with the PREVENT bit set to zero is received through that I_T_L nexus), then the I_T_L nexus identifying information descriptor related to that I_T_L nexus shall be removed from the list. When the last I_T_L nexus

identifying information descriptor has been removed from the list, the PREVENT ALLOW MEDIUM REMOVAL INFORMATION attribute shall be destroyed.

8.2.2.2.7 LAST FAILED RESERVATION attribute

The LAST FAILED RESERVATION attribute indicates the I_T_L nexus that last failed a command requesting a reservation with RESERVATION CONFLICT status. The DYNAMIC RUNTIME ATTRIBUTE VALUE field of the LAST FAILED RESERVATION attribute contains the I_T_L nexus identifying information (see 8.2.2.2.1) for the I_T_L nexus that last received a RESERVATION CONFLICT status to one of the following commands:

- a) PERSISTENT RESERVE OUT;
- b) PERSISTENT RESERVE IN;
- c) RESERVE (see SPC-2); and
- d) RELEASE (see SPC-2).

8.2.2.3 Target type attributes

Target type attributes (see table 102) shall be maintained and updated by the device server. All supported target type attributes shall have a status of read only (see 8.2).

Table 102 — Target type attributes

Attribute Identifier	Name	Attribute Length (in bytes)	Format	Reference
1000h	TRANSPORTID	24	BINARY	8.2.2.3.1
1001h	TARGET PORT ID	2	BINARY	8.2.2.3.2
1002h	LAST ACCESS TIME	12	BINARY	8.2.2.3.3
all others	Reserved			

8.2.2.3.1 TRANSPORTID attribute

The TRANSPORTID target type attribute indicates the TransportID (see SPC-4) of the initiator port associated with this I_T_L nexus.

8.2.2.3.2 TARGET PORT ID attribute

The TARGET PORT ID target type attribute indicates the relative target port identifier of the target port associated with this I_T_L nexus.

8.2.2.3.3 LAST ACCESS TIME attribute

The LAST ACCESS TIME target type attribute indicates the time of most recent command that affects the volume received through this I_T_L nexus (see table 103).

Table 103 — LAST ACCESS TIME target type attribute format

Bit Byte	7	6	5	4	3	2	1	0
0	(MSB)							
1								(LSB)
2							TIMESTAMP ORIGIN	
3								
4	(MSB)							
...								
9								(LSB)
10								
11								

The TIMESTAMP ORIGIN field contains the origin of the timestamp used in the ACCESS TIME field (see SPC-4).

The ACCESS TIME field contains the timestamp value when the most recent command was received through this I_T_L nexus that is not one of the following commands:

- a) INQUIRY;
- b) LOG SENSE;
- c) MODE SENSE;
- d) READ DYNAMIC RUNTIME ATTRIBUTE;
- e) PERSISTENT RESERVE IN;
- f) REPORT BLOCK LIMITS;
- g) REPORT LUNS;
- h) REQUEST SENSE;
- i) TEST UNIT READY; and
- j) other commands for vendor specific reasons.

8.2.2.4 Initiator type attributes

Application clients may use the WRITE DYNAMIC RUNTIME ATTRIBUTE command (see 7.16) and the READ DYNAMIC RUNTIME ATTRIBUTE command (see 7.8) to maintain initiator type attributes. All attributes, once created, shall exist until:

- a) destroyed by a WRITE DYNAMIC RUNTIME ATTRIBUTE command;
- b) a power on event; or
- c) an I_T nexus loss event.

Table 104 — Initiator type attributes

Attribute Identifier	Name	Attribute Length (in bytes)	Format	Reference
1800h	DEVICE SPECIAL FILE NAME	1..32	ASCII	8.2.2.4.1
1801h	PATH INDEX	1..4	ASCII	8.2.2.4.2
1802h	HOST NAME	1..32	ASCII	8.2.2.4.3
1803h	OPERATING SYSTEM	1..16	ASCII	8.2.2.4.4
1804h	OPERATING SYSTEM VERSION	1..32	ASCII	8.2.2.4.5
1805h	DEVICE DRIVER NAME	1..16	ASCII	8.2.2.4.6
1806h	DEVICE DRIVER VERSION	1..16	ASCII	8.2.2.4.7
1807h	PROCESS ID	1..8	ASCII	8.2.2.4.8
all others	Reserved			

8.2.2.4.1 DEVICE SPECIAL FILE NAME attribute

The DEVICE SPECIAL FILE NAME initiator type attribute indicates the device special file name used by the application client to identify the I_T_L nexus (e.g., “\\.\tape0”, “/dev/rmt0”, “/dev/sg0”).

8.2.2.4.2 PATH INDEX attribute

The PATH INDEX initiator type attribute contains an identifier that indicates the path being used by the device special file name.

8.2.2.4.3 HOST NAME attribute

The HOST NAME initiator type attribute indicates the host name of the server that contains the initiator port of the I_T_L nexus (e.g., “foobar”).

8.2.2.4.4 OPERATING SYSTEM attribute

The OPERATING SYSTEM initiator type attribute indicates the operating system being used by the application client.

8.2.2.4.5 OPERATING SYSTEM VERSION attribute

The OPERATING SYSTEM VERSION initiator type attribute indicates the version of the operating system specified in 8.2.2.4.4.

8.2.2.4.6 DEVICE DRIVER NAME attribute

The DEVICE DRIVER NAME initiator type attribute indicates the name of the operating system device driver.

8.2.2.4.7 DEVICE DRIVER VERSION attribute

The DEVICE DRIVER VERSION initiator type attribute indicates the version of the operating system device driver specified in 8.2.2.4.6.

8.2.2.4.8 PROCESS ID attribute

The PROCESS ID initiator type attribute indicates the process ID of the thread that is sending commands through this I_T nexus.

8.3 Log parameters

8.3.1 Log parameters overview

This subclause defines the descriptors and pages for log parameters used with sequential-access devices.

The log page codes for sequential-access devices are specified in table 105.

Table 105 — Log page codes (part 1 of 2)

Page code	Subpage code	Log page name	Support	Reference
00h	00h	Supported Log Pages	M	SPC-4
00h	FFh	Supported Log Pages and Subpages	O	SPC-4
01h to 3Fh	FFh	Supported Subpages	O	SPC-4
01h	00h	Buffer Overrun/Underrun	O	SPC-4
02h	00h	Write Error Counter	M	SPC-4
03h	00h	Read Error Counter (read)	M	SPC-4
04h	00h	Read Reverse Error Counter	O	SPC-4
05h	00h	Verify Error Counter	O	SPC-4
06h	00h	Non-Medium Error	O	SPC-4
07h	00h	Last <i>n</i> Error Events	O	SPC-4
08h to 0Ah	00h to FEh	Reserved	-	
0Bh	00h	Last <i>n</i> Deferred Error Events or Asynchronous Events	O	SPC-4
0Ch	00h	Sequential-Access Device	M	8.3.2
0Dh	00h	Temperature	O	SPC-4
0Eh	00h	Start-Stop Cycle Counter	O	SPC-4
0Fh	00h	Application Client	O	SPC-4
10h	00h	Self-Test Results	O	SPC-4
11h	00h	DT Device Status	O	ADC-3

Table 105 — Log page codes (part 2 of 2)

Page code	Subpage code	Log page name	Support	Reference
12h	00h	TapeAlert Response	O	ADC-3
13h	00h	Requested Recovery	O	8.3.7
14h	00h	Device Statistics	O	8.3.4
15h	00h to FEh	Reserved	-	
16h	00h	Tape Diagnostic Data	O	8.3.5
17h	00h	Volume Statistics	O	8.3.9
18h	00h to FEh	Protocol Specific Port	O	SPC-4
19h to 1Ah	00h to FEh	Reserved	-	
1Bh	00h	Data Compression	O	8.3.8
1Ch to 2Ch	00h to FEh	Reserved	-	
2Dh	00h	Current Service Information	O	8.3.6
2Eh	00h	TapeAlert	M	8.3.3
2Fh	00h	Informational Exceptions	O	SPC-4
30h to 3Fh	00h to FEh	Vendor specific (does not require page format)	-	

8.3.2 Sequential-Access Device log page

The Sequential-Access Device log page defines:

- a) data counters associated with data bytes transferred to and from the medium and to and from the application client;
- b) binary list parameters describing native capacities; and
- c) a binary list parameter related to cleaning.

The default value for parameters 0 through 3 shall be zero.

NOTE 47 - The data in parameters 0 and 1 are intended to provide an indication of the compression ratio for the written data. Parameters 2 and 3 are intended to provide an indication of the compression ratio for read data.

The parameter page codes for the Sequential-Access Devices log page are specified in table 106.

Table 106 — Parameter codes for Sequential-Access Device log page (part 1 of 2)

Parameter Code	Description	Support
0000h	Number of data bytes received from application clients during WRITE command operations.	M
0001h	Number of data bytes written to the media as a result of WRITE command operations, not counting ECC and formatting overhead.	M
0002h	Number of data bytes read from the media during READ command operations, not counting ECC and formatting overhead.	M
0003h	Number of data bytes transferred to the initiator(s) during READ command operations.	M

Table 106 — Parameter codes for Sequential-Access Device log page (part 2 of 2)

Parameter Code	Description	Support
0004h	Approximate native capacity in megabytes (i.e., 10^6) from BOP to EOD. This is not sensitive to the current position of the medium. The approximate native capacity between EOD and EW is the difference of parameter 0005h and this parameter. Conditions may occur that reduce the amount of data that is written before reaching EW. EOD may be beyond LEOP. A value of all bits set to one indicates that this information is invalid (e.g., no volume is mounted or EOD information needs to be rebuilt).	M
0005h	Approximate native capacity in megabytes (i.e., 10^6) between BOP and EW of the current partition. If no volume is mounted or this value is unknown the device server shall set all bits in this parameter to one.	M
0006h	Minimum native capacity in megabytes (i.e., 10^6) between EW and LEOP of the current partition. If no volume is mounted the device server shall set all bits in this parameter to one.	M
0007h	Approximate native capacity in megabytes (i.e., 10^6) from BOP to the current position of the medium. If no volume is mounted the device server shall set all bits in this parameter to one.	M
0008h	Maximum native capacity in megabytes (i.e., 10^6) that is currently allowed to be in the device object buffer. This value may change depending on the current position of the medium (e.g., available native capacity may decrease as the current position of the medium approaches LEOP).	M
0009h to 00FFh	Reserved.	-
0100h	Cleaning requested.	O
0101h to 7FFFh	Reserved.	-
8000h to FFFFh	Vendor-specific parameters.	-

NOTE 48 - If the current partition has a native capacity of 200 GB (i.e., 200×10^9) with EW at 1GB prior to LEOP and the medium is positioned at EOD which is at the point that is 75% of the native capacity between BOP and EW, then the device server uses the following to determine parameters 0004h, 0005h, and 0006h. Since 75% of native capacity is remaining, $(200 \text{ GB} - 1 \text{ GB}) \times 75\% = 149.25 \text{ GB}$. This equation results in parameter 0004h = 149 250 (02 4702h), parameter 0005h = 199 000 (03 0958h), and parameter 0006h = 1 000 (00 03E8h).

A non-zero value of the cleaning requested parameter indicates that the device has requested a read/write mechanism cleaning and a subsequent cleaning cycle has not been completed. A zero value of the cleaning requested parameter indicates that the device has not requested a read/write mechanism cleaning. The cleaning requested parameter value shall persist across I_T nexus losses, logical unit resets, and power cycles.

8.3.3 TapeAlert log page

The TapeAlert log page (see table 107) defines error and informational flags used for detailed device diagnostics and management (see 4.2.24 and Annex A).

Table 107 — TapeAlert log page

Bit Byte	7	6	5	4	3	2	1	0
0	Reserved				PAGE CODE (2Eh)			
1					SUBPAGE CODE (00h)			
2	(MSB)				PAGE LENGTH (n-3)			
3								(LSB)
					TapeAlert log parameter(s)			
4								
...					TapeAlert log parameter [first] (see table 108)			
8								
					⋮			
n-4								
...					TapeAlert log parameter [last] (see table 108)			
n								

See SPC-4 for a description of the PAGE CODE field, SUBPAGE CODE field, and PAGE LENGTH field.

Table 108 specifies the format of a TapeAlert log parameter.

Table 108 — TapeAlert parameter format

Bit Byte	7	6	5	4	3	2	1	0
0	(MSB)							
1					PARAMETER CODE			(LSB)
2	DU	Obsolete	TSD (1)	ETC	TMC		FORMAT AND LINKING (00b)	
3					PARAMETER LENGTH (01h)			
4					Reserved		FLAG	

The value in the PARAMETER CODE field shall range from 1 to 64.

See SPC-4 for a description of the DU bit, TSD bit, ETC bit, TMC field, and FORMAT AND LINKING field. The TSD bit and FORMAT AND LINKING field shall be set to the value specified in table 108.

An active TapeAlert flag has the FLAG bit set to one. An inactive TapeAlert flag has the FLAG bit set to zero.

If processing a LOG SELECT command, the device server shall terminate the command with CHECK CONDITION status, set the sense key to ILLEGAL REQUEST, and set the additional sense code to INVALID FIELD IN PARAMETER LIST if the application client sends parameter data for the TapeAlert log page with:

- a) the TSD bit set to zero;
- b) the DS bit set to zero;
- c) the LP bit set to one;
- d) the LBIN bit set to one;
- e) the FLAG bit set to one; or
- f) the PARAMETER LENGTH field set to a value other than 01h.

If the TASER bit is set to zero (see 8.5.8), the device server shall terminate the command with CHECK CONDITION stats, set the sense key to ILLEGAL REQUEST, and set the additional sense code to INVALID FIELD IN PARAMETER LIST upon processing a LOG SELECT command where the application client has sent parameter data for the TapeAlert log page with the ETC bit set to one.

8.3.4 Device Statistics log page

8.3.4.1 Device Statistics log page overview

The Device Statistics log page (see table 109) defines parameters associated with utilization of the tape device. A device server that implements the Device Statistics log page shall implement support for the defined parameters as specified in table 110.

Table 109 — Device Statistics log page

Bit Byte	7	6	5	4	3	2	1	0
0	DS	SPF (0b)	PAGE CODE (14h)					
1	SUBPAGE CODE (00h)							
2	(MSB)	PAGE LENGTH (n-3)						
3								(LSB)
Device Statistics log parameter(s)								
4								
...	Device Statistics log parameter [first] (see table 110)							
...								
n	Device Statistics log parameter [last] (see table 110)							

See SPC-4 for a description of the DS bit, SPF bit, PAGE CODE field, SUBPAGE CODE field, and PAGE LENGTH field.

The parameter page codes for the Device Statistics log page are specified in table 110.

Table 110 — Device Statistics log parameter codes (part 1 of 2)

Parameter Code	Description	Persist	Clear	Type	Support	Reference
0000h	Lifetime volume loads	P	N	C	M	8.3.4.4.1
0001h	Lifetime cleaning operations	P	N	C	M	8.3.4.4.2
0002h	Lifetime power on hours	P	N	C	M	8.3.4.4.3
0003h	Lifetime medium motion hours	P	N	C	M	8.3.4.4.4
0004h	Lifetime meters of tape processed	P	N	C	M	8.3.4.4.5
0005h	Lifetime medium motion hours at last incompatible volume load	P	N	C	O	8.3.4.4.6
0006h	Lifetime power on hours at the last temperature condition occurrence	P	N	C	O	8.3.4.4.7
0007h	Lifetime power on hours at the last power consumption condition occurrence	P	N	C	O	8.3.4.4.8
0008h	Medium motion hours since last successful cleaning operation	P	N	C	M	8.3.4.4.9
0009h	Medium motion hours since second to last successful cleaning operation	P	N	C	O	8.3.4.4.9
000Ah	Medium motion hours since third to last successful cleaning operation	P	N	C	O	8.3.4.4.9
000Bh	Lifetime power on hours at the last operator initiated forced reset and/or emergency eject occurrence	P	N	C	O	8.3.4.4.10
000Ch	Lifetime power cycles	P	N	C	M	8.3.4.4.11
000Dh	Volume loads since last parameter reset	P	Y	C	M	8.3.4.4.12
000Eh	Hard write errors	P	Y	C	M	8.3.4.4.13
000Fh	Hard read errors	P	Y	C	M	8.3.4.4.14
0010h	Duty cycle sample time	V	Y	C	M	8.3.4.4.15
0011h	Read duty cycle	D	Y	C	M	8.3.4.4.16
Persist key:	P=Parameter shall be Persistent across I_T nexus loss, logical unit reset, and power on. V=The persistence of this parameter varies and is inherited from the value being reported. D=The parameter is a duty cycle parameter and is cleared if parameter 0010h (i.e., duty cycle sample time) is cleared.					
Clear key:	N=Parameter shall not be cleared by use of a LOG SELECT command. Y=Parameter may be cleared by use of a LOG SELECT command.					
Type key:	C=Device Statistics data counter log parameter (see 8.3.4.2.1). M=Device Statistics medium type log parameter (see 8.3.4.2.2). S=Device Statistics string data log parameter (see 8.3.4.3).					
Support key:	M=Mandatory O=Optional					

Table 110 — Device Statistics log parameter codes (part 2 of 2)

Parameter Code	Description	Persist	Clear	Type	Support	Reference
0012h	Write duty cycle	D	Y	C	M	8.3.4.4.17
0013h	Activity duty cycle	D	Y	C	M	8.3.4.4.18
0014h	Volume not present duty cycle	D	Y	C	M	8.3.4.4.19
0015h	Ready duty cycle	D	Y	C	M	8.3.4.4.20
0016h	Megabytes transferred from application client in duty cycle sample time	D	Y	C	O	8.3.4.4.21
0017h	Megabytes transferred to application client in duty cycle sample time	D	Y	C	O	8.3.4.4.22
0018h to 003Fh	Reserved					
0040h	Drive manufacturer's serial number	P	N	S	M	8.3.4.4.23
0041h	Drive serial number	P	N	S	M	8.3.4.4.24
0042h to 007Fh	Reserved					
0080h	Medium removal prevented	V	N	C	M	8.3.4.4.25
0081h	Maximum recommended mechanism temperature exceeded	P	N	C	M	8.3.4.4.26
0082h to 0FFFh	Reserved					
1000h	Medium motion hours for each medium type	P	N	M	O	8.3.4.2.2
1001h to 7FFFh	Reserved					
8000h to FFFFh	Vendor-specific					
Persist key:	P=Parameter shall be Persistent across I_T nexus loss, logical unit reset, and power on. V=The persistence of this parameter varies and is inherited from the value being reported. D=The parameter is a duty cycle parameter and is cleared if parameter 0010h (i.e., duty cycle sample time) is cleared.					
Clear key:	N=Parameter shall not be cleared by use of a LOG SELECT command. Y=Parameter may be cleared by use of a LOG SELECT command.					
Type key:	C=Device Statistics data counter log parameter (see 8.3.4.2.1). M=Device Statistics medium type log parameter (see 8.3.4.2.2). S=Device Statistics string data log parameter (see 8.3.4.3).					
Support key:	M=Mandatory O=Optional					

Parameter codes corresponding to values of time shall be reported in hours and rounded up to the next whole hour.

8.3.4.2 Device statistics log parameter formats

8.3.4.2.1 Device statistics data counter log parameter format

The device statistics data counter log parameter is used for reporting parameters specified as device statistics data counter log parameters (see table 110).

The device statistics data counter log parameter format is specified in table 111.

Table 111 — Device statistics data counter log parameter format

Bit Byte	7	6	5	4	3	2	1	0
0	(MSB)							
1					PARAMETER CODE			(LSB)
2	DU	Obsolete	TSD (0b)	ETC	TMC			FORMAT AND LINKING (11b)
3					PARAMETER LENGTH (n-3)			
4	(MSB)							
...					DEVICE STATISTICS DATA COUNTER			
n								(LSB)

The PARAMETER CODE field is defined in table 110.

See SPC-4 for descriptions of the DU bit, TSD bit, ETC bit, TMC field and FORMAT AND LINKING field. The TSD bit and FORMAT AND LINKING field shall be set to the values specified in table 111.

The PARAMETER LENGTH field indicates the number of bytes in the DEVICE STATISTICS DATA COUNTER field that follows.

The DEVICE STATISTICS DATA COUNTER field is the value of the data counter associated with the parameter code.

8.3.4.2.2 Device statistics medium type log parameter

The device statistics medium type log parameter is used for reporting parameters specified as device statistics medium type log parameters (see table 110). The device statistics medium type log parameter format is specified in table 112.

Table 112 — Device statistics medium type log parameter format

Bit Byte	7	6	5	4	3	2	1	0
0	(MSB)							
1								(LSB)
2	DU (0b)	Obsolete	TSD (0b)	ETC (0b)		TMC (00b)		FORMAT AND LINKING (11b)
3						PARAMETER LENGTH (n-3)		
						Device statistics medium type descriptor(s)		
4								
...								
11								
n-7								
...								
n								

The PARAMETER CODE field shall be set to 1000h to indicate the Medium type log parameter.

See SPC-4 for descriptions of the DU bit, TSD bit, ETC bit, TMC field and FORMAT AND LINKING field. These fields shall be set to the values specified in table 112.

The PARAMETER LENGTH field indicates the number of bytes in the device statistics medium type descriptors that follow.

The device statistics medium type descriptor format is specified in table 113.

Table 113 — Device statistics medium type descriptor format

Bit Byte	7	6	5	4	3	2	1	0
0								Reserved
1								Reserved
2								DENSITY CODE
3								MEDIUM TYPE
4	(MSB)							
...								
7								(LSB)

The DENSITY CODE field contains the value returned in the general mode parameter block descriptor (see SPC-4).

The MEDIUM TYPE field contains the value returned in the mode parameter header (see SPC-4).

The value returned in the MEDIUM TYPE field is vendor specific for sequential-access devices.

The MEDIUM MOTION HOURS field contains the number of medium motion hours for the type of medium specified by the combination of the MEDIUM TYPE field and DENSITY CODE field.

8.3.4.3 Device statistics string data log parameter format

The device statistics string data log parameter is used for reporting parameters specified as device statistics string data log parameters (see table 110). The device statistics string data log parameter shall be a multiple of 4 bytes. The device statistics string data log parameter format is specified in table 114.

Table 114 — Device statistics string data log parameter format

Bit Byte	7	6	5	4	3	2	1	0
0	(MSB)							
1								(LSB)
2	DU (0b)	Obsolete	TSD (0b)	ETC (0b)	TMC (00b)			FORMAT AND LINKING (01b)
3								PARAMETER LENGTH (n-3)
4								
...								STRING DATA
n								

The PARAMETER CODE field is defined in table 110.

See SPC-4 for descriptions of the DU bit, TSD bit, ETC bit, TMC field and FORMAT AND LINKING field. These fields shall be set to the values specified in table 114

The PARAMETER LENGTH field indicates the number of bytes of data that follows.

The STRING DATA field contains an ASCII string describing the device statistics parameter specified by the PARAMETER CODE value. The STRING DATA field is an ASCII data field (see SPC-4).

8.3.4.4 Device Statistics log parameters

8.3.4.4.1 Lifetime volume loads

The lifetime volume loads parameter contains the total number of successful load operations.

8.3.4.4.2 Lifetime cleaning operations

The lifetime cleaning operations parameter contains the total number of successful and failed cleaning operations.

8.3.4.4.3 Lifetime power on hours

The lifetime power on hours parameter contains the total number of hours the device has been powered on. The value reported shall be rounded up to the next full hour.

8.3.4.4.4 Lifetime medium motion hours

The lifetime medium motion hours parameter contains the total number of hours that the device has spent processing commands that require medium motion. The value reported shall be rounded up to the next full hour.

8.3.4.4.5 Lifetime meters of tape processed

The lifetime meters of tape processed parameter contains the total number of meters of tape that have been processed by the drive mechanism in either direction.

8.3.4.4.6 Lifetime medium motion hours at last incompatible volume load

The lifetime medium motion hours at last incompatible volume load parameter contains the value from the lifetime medium motion hours parameter code (i.e., parameter code 0003h) at the time of the last incompatible volume load.

8.3.4.4.7 Lifetime power on hours at the last temperature condition occurrence

The lifetime power on hours at the last temperature condition occurrence parameter contains the lifetime power on hours at the time TapeAlert code 24h flag was last set.

8.3.4.4.8 Lifetime power on hours at the last power consumption condition occurrence

The lifetime power on hours at the last power consumption condition occurrence parameter contains the lifetime power on hours at the time TapeAlert code 1Ch flag was last set.

8.3.4.4.9 Medium motion hours since (last, 2nd to last, 3rd to last) successful cleaning operation

The medium motion hours since (last, 2nd to last, 3rd to last) cleaning operation parameters contain a count of hours that medium has been in motion in either direction since the corresponding successful cleaning operation. The value reported shall be rounded up to the next full hour.

8.3.4.4.10 Lifetime power on hours at the last operator initiated forced reset and/or emergency eject occurrence

The lifetime power on hours at the last operator initiated forced reset and/or emergency eject occurrence parameter contains the lifetime power on hours at the time of the last forced reset and/or emergency eject.

8.3.4.4.11 Lifetime power cycles

The lifetime power cycles parameter contains the total number of times the drive has detected a power on event.

8.3.4.4.12 Volume loads since last parameter reset

The volume loads since last parameter reset parameter contains a count of successful volume loads since the last time this parameter was reset to zero by use of a LOG SELECT command. This parameter should be retained across a power cycle.

8.3.4.4.13 Hard write errors

The hard write errors parameter contains the number of times that a write type command has terminated with a CHECK CONDITION status having the sense key set to MEDIUM ERROR or HARDWARE ERROR since the last time this parameter was reset to zero by the use of a LOG SELECT command.

8.3.4.4.14 Hard read errors

The hard read errors parameter shall contain the number of times that a read type command has terminated with a CHECK CONDITION status having the sense key set to MEDIUM ERROR or HARDWARE ERROR since the last time this parameter was reset to zero by the use of a LOG SELECT command.

8.3.4.4.15 Duty cycle sample time

The duty cycle sample time parameter shall contain the time in milliseconds since the last time this parameter was reset to zero by:

- a) the use of a LOG SELECT command;
- b) a hard reset condition; or
- c) a vendor specific method

8.3.4.4.16 Read duty cycle

The read duty cycle parameter contains the percentage (i.e., an integer between 0 and 100 representing a percentage) of duty cycle sample time (i.e., the value reported in parameter code 0010h) that the device was in the ready state and was processing read type commands. This parameter shall be set to zero if the duty cycle sample time parameter is set to zero.

8.3.4.4.17 Write duty cycle

The write duty cycle parameter contains the percentage of duty cycle sample time (i.e., the value reported in parameter code 0010h) that the device was in the ready state and was processing write type commands. This parameter shall be set to zero if the duty cycle sample time parameter is set to zero.

8.3.4.4.18 Activity duty cycle

The activity duty cycle parameter contains the percentage of duty cycle sample time (i.e., the value reported in parameter code 0010h) that the device was in the ready state and was processing write type commands, read type commands, and other commands that cause the medium to be moved. This parameter shall be set to zero if the duty cycle sample time parameter is set to zero.

8.3.4.4.19 Volume not present duty cycle

The volume not present duty cycle parameter contains the percentage of the duty cycle sample time (i.e., the value reported in parameter code 0010h) that the device server did not detect a volume present (e.g., the physical device attribute medium present was not set to true).

8.3.4.4.20 Ready duty cycle

The ready duty cycle parameter contains the percentage of duty cycle sample time (i.e., the value reported in parameter code 0010h) that the device was in the ready state.

8.3.4.4.21 Megabytes transferred from application client in duty cycle sample time

The megabytes transferred from application client in duty cycle sample time parameter contains the number of megabytes (i.e., 10^6 bytes), rounded up to the next megabyte, contained in logical blocks that have been transferred from an application client during the duty cycle reported in the duty cycle sample time parameter (i.e., parameter code 0010h).

8.3.4.4.22 Megabytes transferred to application client in duty cycle sample time

The megabytes transferred to application client in duty cycle sample time parameter contains the number of megabytes (i.e., 10^6 bytes), rounded up to the next megabyte, contained in logical blocks that have been transferred to an application client during the duty cycle reported in the duty cycle sample time parameter (i.e., parameter code 0010h).

8.3.4.4.23 Drive manufacturer's serial number

The drive manufacturer's serial number parameter contains the value that is reported in the MANUFACTURER-ASSIGNED SERIAL NUMBER field of the Manufacturer-assigned serial number VPD page (B1h).

8.3.4.4.24 Drive serial number

The drive serial number parameter contains the value that is reported in the PRODUCT SERIAL NUMBER field of the Unit Serial Number VPD page (80h).

8.3.4.4.25 Medium removal prevented

The medium removal prevented parameter indicates whether volume removal has been prevented by:

- a) a prevention of volume removal condition in the device server,
- b) a configuration setting, or
- c) a vendor specific means.

The least significant byte in the medium removal prevented parameter data counter value set to 01h and all other bytes set to 00h in the medium removal prevented parameter data counter value indicates that volume removal has been prevented. A zero value in the medium removal prevented parameter data counter value indicates that volume removal has not been prevented.

An error condition preventing removal of the volume shall not cause this parameter to be set to a nonzero value.

8.3.4.4.26 Maximum recommended mechanism temperature exceeded

The maximum recommended mechanism temperature exceeded parameter indicates whether the device has detected at any point in the past that the maximum recommended mechanism temperature has been exceeded.

The maximum recommended mechanism temperature, temperature sampling frequency, and temperature resolution is vendor specific. This parameter shall not be cleared by the use of a LOG SELECT command.

The maximum recommended mechanism temperature exceeded code values are specified in table 115.

Table 115 — Maximum recommended mechanism temperature exceeded codes

Code	Example	Description
All bytes set to 00h	0000h	The device has not detected at any point in the past that the maximum recommended mechanism temperature has been exceeded.
Least significant byte set to 01h and all other bytes set to 00h	0001h	The device has detected at some point in the past that the maximum recommended mechanism temperature has been exceeded.
All bytes set to FFh	FFFFh	It is not known if at any point in the past that the maximum recommended mechanism temperature has been exceeded.
All other values	1234h	Reserved

8.3.5 Tape Diagnostic Data log page

The Tape Diagnostic Data log page (see table 116) provides for a number of error-event records using the list parameter format. Each error-event record contains diagnostic information for a single error type encountered by the device including data counters associated with the error event, sense data, operation code/service action and medium type with associated media motion hours, etc. The Tape Diagnostic Data log page may be used to aid in field analysis and repair.

The Tape Diagnostic Data log page shall only include parameter entries for commands that terminated with a CHECK CONDITION status having the sense key set to MEDIUM ERROR, HARDWARE ERROR or ABORTED COMMAND.

The parameter code value associated with an error-event indicates the relative time at which a command terminated with a CHECK CONDITION status. A lower parameter code indicates that the command terminated with a CHECK CONDITION status at a more recent time. The parameter code values returned shall be numbered consecutively from 0000h (i.e., the most recent) up to n, where n is the number of current parameter entries. The number of supported parameter entries, n, is vendor specific.

In each parameter (see table 117), if the REPEAT bit is set to zero, then the parameter represents only one event. If the REPEAT bit is set to one, then the parameter represents more than one consecutive events that had the identical values for the MEDIUM ID NUMBER field, SENSE KEY field, ADDITIONAL SENSE CODE field and ADDITIONAL SENSE CODE QUALIFIER field in the parameter. If the REPEAT bit is set to one in the parameter, then other fields in the parameter shall be set to the values for the first of the consecutive events that had the identical values for the MEDIUM ID NUMBER field, SENSE KEY field, ADDITIONAL SENSE CODE field and ADDITIONAL SENSE CODE QUALIFIER field.

The most recent parameter codes shall be persistent across I_T nexus losses, logical unit resets, and power-on. The parameter entries shall not be set to zero or changed with the use of a LOG SELECT command.

Table 116 — Tape Diagnostic Data log page

Bit Byte	7	6	5	4	3	2	1	0
0	DS	SPF (0b)	PAGE CODE (16h)					
1	SUBPAGE CODE (00h)							
2	(MSB)	PAGE LENGTH (n-3)						
3								(LSB)
Tape diagnostic data log parameter(s)								
4		Tape diagnostic data log parameter [first] (see table 117)						
...								
		⋮						
...		Tape diagnostic data log parameter [last] (see table 117)						
n								

See SPC-4 for a description of the DS bit, SPF bit, PAGE CODE field, SUBPAGE CODE field and PAGE LENGTH field.

The tape diagnostic data log parameter format is specified in table 117.

Table 117 — Tape diagnostic data log parameter format

Bit Byte	7	6	5	4	3	2	1	0			
0	(MSB)	PARAMETER CODE									
1								(LSB)			
2	DU (0b)	Obsolete	TSD (0b)	ETC (0b)	TMC (00b)	FORMAT AND LINKING (11b)					
3	PARAMETER LENGTH (n-3)										
4	Reserved										
5	Reserved										
6	DENSITY CODE										
7	MEDIUM TYPE										
8	(MSB)	LIFETIME MEDIA MOTION HOURS									
...											
11								(LSB)			
12	Reserved										
13	REPEAT	Reserved			SENSE KEY						
14	ADDITIONAL SENSE CODE										

Table 117 — Tape diagnostic data log parameter format (Continued)

Bit Byte	7	6	5	4	3	2	1	0
15								ADDITIONAL SENSE CODE QUALIFIER
16	(MSB)							
...								VENDOR-SPECIFIC CODE QUALIFIER
19								(LSB)
20								
...								PRODUCT REVISION LEVEL
23								
24	(MSB)							
...								HOURS SINCE LAST CLEAN
27								(LSB)
28								OPERATION CODE
29	Reserved							SERVICE ACTION
30								Reserved
31								Reserved
32	(MSB)							
...								MEDIUM ID NUMBER
63								(LSB)
64		Reserved						TIMESTAMP ORIGIN
65								Reserved
66	(MSB)							
...								TIMESTAMP
71								(LSB)
72								
...								VENDOR SPECIFIC
n								

See SPC-4 for descriptions of the DU bit, TSD bit, ETC bit, TMC field and FORMAT AND LINKING field. These fields shall be set to the values specified in table 117.

The PARAMETER LENGTH field indicates the number of bytes in the tape diagnostic data log parameter data that follows.

The DENSITY CODE field contains the density code of the volume loaded at the time the command terminated with the CHECK CONDITION status. The DENSITY CODE field is the same value as returned in the general mode parameter block descriptor (see SPC-4). If no volume was loaded at the time the command terminated with the CHECK CONDITION status, then the DENSITY CODE field shall be set to 00h.

The MEDIUM TYPE field contains the type of volume loaded at the time the command terminated with the CHECK CONDITION status. The MEDIUM TYPE field is the same value as returned in the mode parameter header (see SPC-4). If no volume was loaded at the time the command terminated with the CHECK CONDITION status, then the MEDIUM TYPE field shall be set to 00h.

The LIFETIME MEDIA MOTION HOURS field contains the number of media motion hours at the time the command terminated with the CHECK CONDITION status. The LIFETIME MEDIA MOTION HOURS field is equivalent to the value contained in the Device Statistics log page with a parameter code value of 0003h at the time the command terminated with the CHECK CONDITION status.

The REPEAT bit set to one indicates this parameter represents more than one consecutive events that had identical values for the MEDIUM ID NUMBER field, SENSE KEY field, ADDITIONAL SENSE CODE field, and ADDITIONAL SENSE CODE QUALIFIER field. The REPEAT bit set to zero indicates this parameter represents a single event.

See SPC-4 for descriptions of the SENSE KEY field, ADDITIONAL SENSE CODE field, and ADDITIONAL SENSE CODE QUALIFIER field. The SENSE KEY field, ADDITIONAL SENSE CODE field, and ADDITIONAL SENSE CODE QUALIFIER field shall contain the sense key and additional sense code values of the command that terminated with the CHECK CONDITION status.

The VENDOR-SPECIFIC CODE QUALIFIER field is vendor specific. The VENDOR-SPECIFIC CODE QUALIFIER may provide additional diagnostics information related to the command that terminated with the CHECK CONDITION status.

See SPC-4 for the descriptions of the PRODUCT REVISION LEVEL field. The PRODUCT REVISION LEVEL field shall contain the product revision level at the time the command terminated with the CHECK CONDITION status.

The HOURS SINCE LAST CLEAN field contains the time in media motion hours since the last successful cleaning at the time the command terminated with the CHECK CONDITION status. The HOURS SINCE LAST CLEAN field is equivalent to the value contained in the Device Statistics log page with a parameter code of 0008h at the time the command terminated with the CHECK CONDITION status.

See SPC-4 for descriptions of the OPERATION CODE field and SERVICE ACTION field. The OPERATION CODE field and SERVICE ACTION field, if applicable, contain the operation code and service action of the command that terminated with the CHECK CONDITION status.

If a volume was present at the time the command terminated with the CHECK CONDITION status, then the MEDIUM ID NUMBER field shall contain:

- 1) the BARCODE field value contained in the medium auxiliary memory (see SPC-4);
- 2) the MEDIUM SERIAL NUMBER field value contained in the medium auxiliary memory (see SPC-4); or
- 3) a vendor-specific value associated with the mounted volume.

If no volume was present at the time the command terminated with the CHECK CONDITION status, the MEDIUM ID NUMBER field shall be filled with 20h (i.e., ASCII space).

See SPC-4 for descriptions of the TIMESTAMP ORIGIN and TIMESTAMP fields. The TIMESTAMP ORIGIN field and TIMESTAMP field contain the timestamp origin and timestamp maintained by the device server at the time the command terminated with the CHECK CONDITION status. If a timestamp is not supported by the device server, the TIMESTAMP ORIGIN and TIMESTAMP fields shall be set to zero.

8.3.6 Current Service Information log page

8.3.6.1 Current Service Information log page overview

The Current Service Information log page (see table 118) specifies information used for detailed device diagnostics and management.

Table 118 — Current Service Information log page

Bit Byte	7	6	5	4	3	2	1	0
0	DS	SPF (0b)			PAGE CODE (2Dh)			
1					SUBPAGE CODE (00h)			
2	(MSB)				PAGE LENGTH (n-3)			
3								(LSB)
					Service information log parameter(s)			
4								
...					Service information log parameter [first] (see table 119)			
					⋮			
...					Service information log parameter [last] (see table 119)			
n								

See SPC-4 for a description of the DS bit, SPF bit, PAGE CODE field, SUBPAGE CODE field and PAGE LENGTH field.

The service information log parameter format is specified in table 119.

Table 119 — Service information log parameter format

See SPC-4 for descriptions of the DU bit, TSD bit, ETC bit, TMC field and FORMAT AND LINKING field. These fields shall be set to the values specified in table 119.

The PARAMETER LENGTH field indicates the number of bytes that follow.

The value in the PARAMETER CODE field shall be set to the flag number (see table 12) of the TapeAlert flag for which the information applies. If a TapeAlert flag is activated, then the parameter in this log page relating to that TapeAlert flag is created. This parameter shall continue to be reported until overwritten by the next activation of the associated TapeAlert flag or until cleared by a LOG SELECT command. The act of returning a parameter shall not clear that parameter and shall not cause deactivation of the TapeAlert flag.

The Timestamp descriptor is defined by the REPORT TIMESTAMP command parameter data format (see SPC-4) with values indicating the time the TapeAlert flag specified by the PARAMETER CODE field was activated.

Service information descriptors are returned and provide specific information about the TapeAlert flag. At least one service information descriptor shall be returned. The format of service information descriptors is specified in table 120.

Table 120 — Service information descriptor

Bit Byte	7	6	5	4	3	2	1	0
0								SERVICE INFORMATION DESCRIPTOR TYPE
1								SERVICE INFORMATION DESCRIPTOR LENGTH (n-1)
2								
...								Service information descriptor specific information (see table 121)
n								

Only one service information descriptor shall be returned for a specific SERVICE INFORMATION DESCRIPTOR TYPE (see table 121) field value.

Table 121 — SERVICE INFORMATION DESCRIPTOR TYPE field

Code	Service Information Descriptor Type	Reference
00h	Vendor-specific service information	8.3.6.2
01h	Device information	8.3.6.3
02h	Volume information	8.3.6.4
03h	TapeAlert flag specific information	8.3.6.5
04h to FEh	Reserved	

The SERVICE INFORMATION DESCRIPTOR LENGTH field specifies the number of bytes that follow.

8.3.6.2 Vendor-specific service information descriptor

Table 122 specifies the vendor-specific service information descriptor format.

Table 122 — Vendor-specific service information descriptor

Bit Byte	7	6	5	4	3	2	1	0
0								SERVICE INFORMATION DESCRIPTOR TYPE (00h)
1								SERVICE INFORMATION DESCRIPTOR LENGTH (n-1)
2								
...								Vendor-specific information
n								

The SERVICE INFORMATION DESCRIPTOR TYPE field shall be set as specified in table 122.

The SERVICE INFORMATION DESCRIPTOR LENGTH field specifies the number of bytes that follow.

8.3.6.3 Device information descriptor

Table 123 specifies the device information descriptor format. The device information descriptor is returned if the cause of the TapeAlert flag corresponding to the parameter may be related to the device. There shall be only one device information descriptor returned per service information log parameter.

Table 123 — Device information descriptor

Bit Byte	7	6	5	4	3	2	1	0
0								SERVICE INFORMATION DESCRIPTOR TYPE (01h)
1								SERVICE INFORMATION DESCRIPTOR LENGTH (n-1)
2								DEVICE SEVERITY CODE
3								DEC
4								DECQ
5								DECT LENGTH
6								
...								DECT
x								
x+1								NUMBER OF DEVICE REQUESTED RECOVERIES
x+2								DEVICE REQUESTED RECOVERY [first] (see table 125)
								⋮
n								DEVICE REQUESTED RECOVERY [last] (see table 125)

The SERVICE INFORMATION DESCRIPTOR TYPE field shall be set as specified in table 123.

The SERVICE INFORMATION DESCRIPTOR LENGTH field specifies the number of bytes that follow.

The DEVICE SEVERITY CODE field contains a severity code (see table 11).

The device element code (DEC) field is specified in table 124.

Table 124 — DEC field

Code	Description
00h	No message
10h	Device data path
20h	Mechanical
30h	Primary interface
40h	Automation interface
50h	Diagnostic interface
60h	Electronic elements
70h	Microcode
F0 to FFh	Reserved

The device element code qualifier (DECQ) field is a vendor-specific value providing more detailed information about the element specified by the DEC field.

The DECT LENGTH field specifies the length of the DECT field.

The device element code text (DECT) field is null-terminated and contains a description of what caused the TapeAlert flag to be activated.

The NUMBER OF DEVICE REQUESTED RECOVERIES field specifies the number of DEVICE REQUESTED RECOVERY fields.

The DEVICE REQUESTED RECOVERY field values (see table 125) shall be returned in prioritized order.

Table 125 — DEVICE REQUESTED RECOVERY field

Code	Description
00h	No recovery requested
01h	Retrieve device debug logs
02h	Clean device
03h	Update microcode
04h	Power off device and call service
05h	Leave the device in current state and call service
06h	Remove power from the device then apply power
07h to FFh	Reserved

8.3.6.4 Volume information descriptor

Table 126 specifies the volume information descriptor format. The volume information descriptor is returned if the cause of the TapeAlert flag corresponding to the parameter may be related to the volume.

Table 126 — Volume information descriptor

Bit Byte	7	6	5	4	3	2	1	0
0								SERVICE INFORMATION DESCRIPTOR TYPE (02h)
1								SERVICE INFORMATION DESCRIPTOR LENGTH (n-1)
2								VOLUME SEVERITY CODE
3								VIC
4								VICQ
								Volume identification descriptor(s)
5								
...								Volume identification descriptor [first] _____
								⋮
...								Volume identification descriptor [last] _____
n								

The SERVICE INFORMATION DESCRIPTOR TYPE field shall be set as specified in table 126.

The SERVICE INFORMATION DESCRIPTOR LENGTH field specifies the number of bytes that follow.

The VOLUME SEVERITY CODE field contains a severity code (see table 11).

The volume information code (VIC) field is specified in table 127.

Table 127 — vic field

Code	Description
00h	No message
01h	Good WORM volume
06h	Good encrypted volume
0Bh	Good data volume
10h	Good cleaning volume
15h	Good microcode update volume
1Ah	Bad WORM volume
1Fh	Bad encrypted volume
25h	Bad data volume
2Ah	Bad cleaning volume

Table 127 — vic field

Code	Description
2Fh	Bad microcode update volume
all others	Reserved

The volume information code qualifier (VICQ) field is specified in table 128.

Table 128 — VICQ field

Code	Description
00h	No message
01h	Read only permitted at this logical position
06h	Logical block encryption key required
0Bh	Read only permitted for the entire volume
10h	Rewrite volume if possible
15h	Tape directory invalid, re-read volume if possible
1Ah	Unable to read or write
1Fh	Replace volume
25h	Auxiliary memory error
all others	Reserved

The volume identification descriptor format is the same as the MAM ATTRIBUTE format for medium auxiliary memory (see SPC-4). If a volume information descriptor is returned and:

- 1) if a MAM attribute exists for the volume identifier parameter of the device type attributes (i.e., set by the SMC device), then this attribute shall be returned as a volume identification descriptor;
 - 2) if a MAM attribute exists for the barcode parameter of the host type attributes (i.e., set by an application client), then this attribute shall be returned as a volume identification descriptor; and
 - 3) if a MAM attribute exists for the medium serial number parameter of the medium type attributes (i.e., set by the manufacturer), then this attribute shall be returned as a volume identification descriptor.

8.3.6.5 TapeAlert flag specific information

Table 129 specifies the TapeAlert flag specific information descriptor format. Table 12 specifies which flags are returned for this descriptor.

Table 129 — TapeAlert flag specific information descriptor

The SERVICE INFORMATION DESCRIPTOR TYPE field shall be set as specified in table 129.

The SERVICE INFORMATION DESCRIPTOR LENGTH field specifies the number of bytes that follow and shall be set as specified in table 129.

The CURRENT PERCENTAGE field specifies a signed percentage multiplied by 16 384 indicating how close to operating limits the item is.

The device is inside the operating specifications if:

$$-100\% \leq \text{CURRENT PERCENTAGE field value} \leq 100\%$$

The device is outside the operating specifications if:

$$-100\% > \text{CURRENT PERCENTAGE field value}, \text{ or}$$

$$100\% < \text{CURRENT PERCENTAGE field value}$$

If the item has upper and lower operating limits, then the equation to calculate the CURRENT PERCENTAGE field value is:

$$\frac{\text{measuredValue} - \left[\frac{\langle \text{upperLimit} - \text{lowerLimit} \rangle}{2} + \text{lowerLimit} \right]}{\text{upperLimit} - \left[\frac{\langle \text{upperLimit} - \text{lowerLimit} \rangle}{2} + \text{lowerLimit} \right]} \times 16\ 384$$

Example 1: If the power specification states the operation range is between 4.78 volts and 5.32 volts and the measured voltage is 4.70 volts, then the value returned by the equation is:

$$\frac{4.7 - \left[\frac{\langle 5.32 - 4.78 \rangle}{2} + 4.78 \right]}{5.32 - \left[\frac{\langle 5.32 - 4.78 \rangle}{2} + 4.78 \right]} \times 16\ 384 = -21\ 239 = -AD09h$$

Example 2: If the media life is specified to be 260 full backups and the media has had 234 full backups performed, then the value returned by the equation is:

$$\frac{234 - \left[\frac{\langle 260 - 0 \rangle}{2} + 0 \right]}{260 - \left[\frac{\langle 260 - 0 \rangle}{2} + 0 \right]} \times 16\ 384 = 13\ 107 = 3333h$$

8.3.7 Requested Recovery log page

8.3.7.1 Requested Recovery log page overview

Table 130 specifies the Requested Recovery log page. If the device is unable to complete an action (e.g., a volume load or a volume unload) the device server may set the RRQST bit to one in the very high frequency data log

parameter (see ADC-3) to request that the application client perform a recovery action. The application client is able to obtain a list of suggested recovery actions by reading the Requested Recovery log page.

Table 130 — Requested Recovery log page

Bit Byte	7	6	5	4	3	2	1	0
0	DS	SPF (0b)			PAGE CODE (13h)			
1					SUBPAGE CODE (00h)			
2	(MSB)				PAGE LENGTH (n-3)			
3								(LSB)
4								
...					Requested recovery log parameters			
n								

See SPC-4 for a description of the PAGE CODE field, the DS bit, then SPF bit, the SUBPAGE CODE field, and the PAGE LENGTH field.

Table 131 specifies the requested recovery log parameter codes.

Table 131 — Requested recovery log parameter codes

Parameter Code	Description	Reference
0000h	Recovery procedures	8.3.7.2
0001h to 7FFFh	Reserved	
8000h to FFFFh	Vendor specific	

8.3.7.2 Recovery procedures log parameter

The recovery procedures log parameter format is specified in table 132.

Table 132 — Requested recovery log parameter format

Bit Byte	7	6	5	4	3	2	1	0
0	(MSB)							
1					PARAMETER CODE (0000h)			(LSB)
2	DU (1b)	Obsolete	TSD (1b)	ETC (0b)	TMC (00b)	FORMAT AND LINKING (11b)		
3					PARAMETER LENGTH (n-3)			
					Recovery procedures list			
4					Recovery procedure [first] (see table 133)			
					⋮			
n					Recovery procedure [last] (see table 133)			

See SPC-4 for descriptions of the DU bit, TSD bit, ETC bit, TMC field, and FORMAT AND LINKING field. These bits and fields shall be set to the values specified in table 132.

The PARAMETER LENGTH field specifies the number of recovery procedure bytes that follow.

The PARAMETER CODE field shall be set to 0000h to specify the recovery procedures log parameter.

The recovery procedures list contains recovery procedures (see table 133) listed in order from the most preferred (i.e., in byte 4) to the least preferred procedure.

Each recovery procedure consists of one or more actions to be performed. If the INTXN bit in the VHF data descriptor of the DT Device Status log page is set to one, the parameter shall report only code 00h (i.e., Recovery not requested). If a failure occurs in performing one of the actions in a procedure, an appropriate list of requested recovery procedures may be reported.

Recovery procedures do not persist across a power cycle.

Table 133 — Recovery procedures

Recovery procedure	Description
00h	Recovery not requested.
01h	Recovery requested, no recovery procedure defined.
02h	Instruct operator to push volume.
03h	Instruct operator to remove and re-insert volume.
04h	Issue UNLOAD command. Instruct operator to remove and re-insert volume.
05h	Instruct operator to power cycle target device.
06h	Issue LOAD command.
07h	Issue UNLOAD command.
08h	Issue LOGICAL UNIT RESET task management function.
09h	No recovery procedure specified. Contact service organization.
0Ah	Issue UNLOAD command. Instruct operator to remove and quarantine volume.
0Bh	Instruct operator to not insert a volume. Contact service organization.
0Ch	Issue UNLOAD command. Instruct operator to remove volume. Contact service organization.
0Dh	Request creation of a target device error log.
0Eh	Retrieve a target device error log.
0Fh	Modify configuration to allow microcode update and instruct operator to re-insert volume.
10h to 7Fh	Reserved
80h to FFh	Vendor-specific procedures.

If the Requested Recovery log page is requested and the RRQST bit in the VHF data descriptor of the DT Device Status log page is set to zero, then a recovery procedure of 00h (i.e., recovery not requested) shall be reported. If the requested recovery procedure is 09h (i.e., no recovery procedure defined. Contact service organization), then the device may not be able to unload the medium via subsequent LOAD UNLOAD commands or operator panel

requests until the issue is resolved (i.e., attempting to do so may damage the mechanism of the device, the medium, or both).

If the requested recovery procedure is 0Ah (i.e., Issue UNLOAD command. Instruct operator to remove and quarantine volume), then the volume should be removed from service.

If the requested recovery procedure is 0Bh (i.e., Instruct operator to not insert a volume. Contact service organization), then a nonrecoverable error has occurred and insertion of a volume may cause damage. If the 0Bh recovery procedure is requested, then the RAA bit in the VHF data descriptor of the DT Device Status log page shall be set to zero, and no other recovery procedures other than 0Dh and 0Eh shall be reported.

If the requested recovery procedure is 0Ch (i.e., Issue UNLOAD command. Instruct operator to remove volume; Contact service organization), then a nonrecoverable error has occurred and insertion of a new volume may cause damage. If recovery procedure 0Ch is requested and the volume has been removed, then the RAA bit in the VHF data descriptor of the DT Device Status log page shall be set to zero, and no other recovery procedures other than 0Dh and 0Eh shall be reported.

8.3.8 Data Compression log page

8.3.8.1 Data Compression log page overview

The Data Compression log page (see table 134) defines data counters associated with the data compression operation for the most recent volume mount.

The device server may:

- a) retain data compression log parameters across hard resets; or
- b) reset all data compression log parameter data fields to 00h after a hard reset.

Table 134 — Data Compression log page

Bit Byte	7	6	5	4	3	2	1	0					
0	DS	SPF (0b)	PAGE CODE (1Bh)										
1	SUBPAGE CODE (00h)												
2	(MSB)												
3	PAGE LENGTH (n-3) (LSB)												
Data compression log parameter(s)													
4													
...	Data compression log parameter [first] (see table 135)												
⋮													
...													
n	Data compression log parameter [last] (see table 135)												

See SPC-4 for a description of the DS bit, SPF bit, PAGE CODE field, SUBPAGE CODE field, and PAGE LENGTH field.

Table 135 specifies the Data Compression log page parameter codes. All of the parameters are data compression counter log parameters (see 8.3.8.2). Parameters may be reset to zero but shall not be changed with the use of a LOG SELECT command. Parameters shall be set to zero at the start of a medium load.

Parameters 0002h through 0009h are pairs that represent a large number of bytes transferred. The first four-byte parameter represents the number of whole megabytes (i.e., 10^6 bytes) transferred, rounded to the nearest megabyte. The second four-byte parameter represents the difference between this number of megabytes and the actual number of bytes. This is a signed number and may be negative, in two's complement notation.

Parameters associated with data transferred from an application client indicate values prior to compression processing. Parameters associated with data transferred to the medium indicate values after compression processing. Compression processing may or may not compress logical blocks.

Table 135 — Data Compression log parameter codes

Parameter Code	Description	Support	Reference
0000h	Read compression ratio	M	8.3.8.3.1
0001h	Write compression ratio	M	8.3.8.3.2
0002h	Megabytes transferred to application client	M	8.3.8.3.3
0003h	Bytes transferred to application client	M	8.3.8.3.4
0004h	Megabytes read from medium	M	8.3.8.3.5
0005h	Bytes read from medium	M	8.3.8.3.6
0006h	Megabytes transferred from application client	M	8.3.8.3.7
0007h	Bytes transferred from application client	M	8.3.8.3.8
0008h	Megabytes written to medium	M	8.3.8.3.9
0009h	Bytes written to medium	M	8.3.8.3.10
000Ah to 00FFh	Reserved		
0100h	Data compression enabled	M	8.3.8.3.11
0101h to EFFFH	Reserved		
F000h to FFFFh	Vendor specific		

8.3.8.2 Data compression counter log parameter format

The data compression counter log parameter format is used for reporting parameters specified as data compression counter log parameters specified in table 135. The data compression counter log parameter format is specified in table 136.

Table 136 — Data compression counter log parameter format

Bit Byte	7	6	5	4	3	2	1	0
0	(MSB)							
1								(LSB)
2	DU	Obsolete	TSD (0b)	ETC	TMC			FORMAT AND LINKING (11b)
3						PARAMETER LENGTH (n-3)		
4	(MSB)							
...								
n								(LSB)

See SPC-4 for a description of the DS bit, TSD bit, ETC bit, TMC field, and FORMAT AND LINKING field. The TSD bit and FORMAT AND LINKING field shall be set to the values specified in table 136.

The PARAMETER LENGTH field indicates the number of bytes in the DATA COMPRESSION COUNTER field that follows.

The DATA COMPRESSION COUNTER field is the value of the data counter associated with the parameter code.

8.3.8.3 Data Compression log parameters

8.3.8.3.1 Read compression ratio

The read compression ratio parameter contains the average data compression ratio multiplied by 100 for all user data read from the medium since the last time this parameter was reset to zero. This parameter shall be set to zero at the start of a mount. This parameter may include user data that was read from the medium but not returned to the application client (e.g., data that was read as part of a read-ahead operation). This parameter shall not be set to zero during the processing of a demount. See 8.3.9.3.20 for the read compression ratio calculation.

8.3.8.3.2 Write compression ratio

The write compression ratio parameter contains the average data compression ratio multiplied by 100 for all user data written by the device since the last time this parameter was reset to zero. This parameter shall be set to zero at the start of a mount. This parameter shall not be set to zero during the processing of a demount. See 8.3.9.3.19 for the write compression ratio calculation.

8.3.8.3.3 Megabytes transferred to an application client

The megabytes transferred to application client parameter contains a count of the number of megabytes (i.e., 10^6 bytes) of logical blocks that have been transferred to an application client after compression processing at the time this parameter is requested. The value reported shall be rounded to the nearest megabyte.

This parameter shall be set to zero at the start of a mount. This parameter shall not be set to zero during the processing on a demount.

8.3.8.3.4 Bytes transferred to application client

The bytes transferred to application client parameter contains the difference in bytes between the actual number of bytes of logical blocks transferred to the application client and the value reported in parameter 0002h (i.e., megabytes transferred to application client). The value reported in this parameter is a signed number and may be negative, in two's complement notation.

This parameter shall be set to zero at the start of a mount. This parameter shall not be set to zero during the processing on a demount.

8.3.8.3.5 Megabytes read from medium

The megabytes read from medium parameter contains a count of the number of megabytes (i.e., 10^6 bytes) of logical blocks that have been read from the medium before compression processing at the time this parameter is requested. The value reported shall be rounded to the nearest megabyte.

This parameter shall be set to zero at the start of a mount. This parameter shall not be set to zero during the processing of an unload command.

8.3.8.3.6 Bytes read from medium

The bytes read from medium parameter contains the difference in bytes between the actual number of bytes of logical blocks read from the medium before decompression at the time this parameter is requested and the value reported in parameter 0004h (i.e., megabytes read from medium). The value reported in this parameter is a signed number and may be negative, in two's complement notation.

This parameter shall be set to zero at the start of a mount. This parameter shall not be set to zero during the processing on a demount.

8.3.8.3.7 Megabytes transferred from an application client

The megabytes transferred from an application client parameter contains a count of the number of megabytes (i.e., 10^6 bytes) of logical blocks that have been transferred from an application client before compression processing at the time this parameter is requested. The value reported shall be rounded to the nearest megabyte.

This parameter shall be set to zero at the start of a mount. This parameter shall not be set to zero during the processing of an unload command.

8.3.8.3.8 Bytes transferred from an application client

The bytes transferred from an application client parameter contains the difference in bytes between the actual number of bytes of logical blocks transferred from an application client before compression processing at the time this parameter is requested and the value reported in parameter 0006h (i.e., megabytes transferred from application client). The value reported in this parameter is a signed number and may be negative, in two's complement notation.

This parameter shall be set to zero at the start of a mount. This parameter shall not be set to zero during the processing on a demount.

8.3.8.3.9 Megabytes written to medium

The megabytes written to medium parameter contains a count of the number of megabytes (i.e., 10^6 bytes) of logical blocks that have written to the medium after compression processing at the time this parameter is requested. The value reported shall be rounded to the nearest megabyte.

This parameter shall be set to zero at the start of a mount. This parameter shall not be set to zero during the processing of an unload command.

8.3.8.3.10 Bytes written to medium

The bytes written to medium parameter contains the difference in bytes between the actual number of bytes of logical blocks written to the medium in logical blocks at the time this parameter is requested and the value reported in parameter 0008h (i.e., megabytes written to medium). The value reported in this parameter is a signed number and may be negative, in two's complement notation.

This parameter shall be set to zero at the start of a mount. This parameter shall not be set to zero during the processing of a demount.

8.3.8.3.11 Data Compression enabled

The data compression enabled parameter indicates whether logical blocks are requested to be compressed before they are written to the medium. The value reported indicates the current state of the DT device and does not indicate that logical blocks previously written to the medium were compressed. The least significant byte in the data compression enabled parameter data counter value set to 01h and all other bytes set to 00h indicates that logical blocks are requested to be compressed before being written to the medium during write type commands. A zero value in the data compression parameter data counter value indicates that logical blocks are not compressed before being written to the medium during write type commands.

8.3.9 Volume Statistics log page

8.3.9.1 Volume Statistics log page overview

The Volume Statistics log page (see table 137) defines data parameters associated with utilization of the tape volume and the medium within the volume. Volume statistics for the most recent mounted volume are reported in the Volume Statistics log page parameters. The most recent mounted volume is the mounted volume or the last volume that was mounted if no volume is mounted. Volume statistics for previously mounted volumes may be reported in Volume Statistics log subpages (see table 138). A device server that implements the Volume Statistics log page shall implement support for the most recent mounted volume and the defined parameters as specified in table 139. The parameters shall not be set to zero or changed with the use of a LOG SELECT command.

If a supported log subpage is requested for a mount that has not occurred, then all bytes in the parameter data fields shall be set to 00h.

The device server may:

- a) retain volume statistics log parameters across hard resets; or
- b) reset all volume statistics log parameter data fields to 00h after a hard reset.

NOTE 49 - An application client may detect if parameter values are valid by testing for parameter value 0000h, (i.e., page valid) set to zero.

Table 137 — Volume Statistics log page

Bit Byte	7	6	5	4	3	2	1	0
0	DS	SPF			PAGE CODE (17h)			
1					SUBPAGE CODE			
2	(MSB)				PAGE LENGTH (n-3)			
3								(LSB)
					Volume Statistics log parameter(s)			
4								
...					Volume Statistics log parameter [first] (see table 139)			
					⋮			
...					Volume Statistics log parameter [last] (see table 139)			
n								

See SPC-4 for a description of the DS bit, SPF bit, and PAGE CODE field.

The SUBPAGE CODE field is specified in table 138.

Table 138 — Volume Statistics log subpage codes

Code	Description
00h	Reserved
01h to 0Fh	Volume statistics for previously mounted volumes 1 through 15.
10h to FEh	Reserved
FFh	Supported subpages log page (see SPC-4)

See SPC-4 for a description of the PAGE LENGTH field.

Table 139 specifies the Volume Statistics log page parameter codes.

Table 139 — Volume Statistics log parameter codes (part 1 of 2)

Parameter Code	Description	Type	Support	Reference
0000h	Page valid	C	M	8.3.9.3.1
0001h	Thread count	C	M	8.3.9.3.2
0002h	Total data sets written	C	M	8.3.9.3.3
0003h	Total write retries	C	M	8.3.9.3.4
0004h	Total unrecovered write errors	C	M	8.3.9.3.5
0005h	Total suspended writes	C	M	8.3.9.3.6
0006h	Total fatal suspended writes	C	M	8.3.9.3.7
0007h	Total data sets read	C	M	8.3.9.3.8
0008h	Total read retries	C	M	8.3.9.3.9
0009h	Total unrecovered read errors	C	M	8.3.9.3.10
000Ah	Total suspended reads	C	O	8.3.9.3.11
000Bh	Total fatal suspended reads	C	O	8.3.9.3.12
000Ch	Last mount unrecovered write errors	C	M	8.3.9.3.13
000Dh	Last mount unrecovered read errors	C	M	8.3.9.3.14
000Eh	Last mount megabytes written	C	M	8.3.9.3.15
000Fh	Last mount megabytes read	C	M	8.3.9.3.16
0010h	Lifetime megabytes written	C	M	8.3.9.3.17
0011h	Lifetime megabytes read	C	M	8.3.9.3.18
0012h	Last load write compression ratio	C	M	8.3.9.3.19
0013h	Last load read compression ratio	C	M	8.3.9.3.20
0014h	Medium mount time	C	M	8.3.9.3.21
0015h	Medium ready time	C	M	8.3.9.3.22
0016h	Total native capacity	C	M	8.3.9.3.23
0017h	Total used native capacity	C	M	8.3.9.3.24
0018h to 003Fh	Reserved			
0040h	Volume serial number	S	M	8.3.9.3.25
0041h	Tape lot identifier	S	M	8.3.9.3.26
0042h	Volume barcode	S	M	8.3.9.3.27
0043h	Volume manufacturer	S	M	8.3.9.3.28
0044h	Volume license code	S	M	8.3.9.3.29
Type key: C=Volume statistics counter log parameter (see 8.3.9.2.1) S=Volume statistics string data log parameter (see 8.3.9.2.2) P=Volume statistics partition record log parameter (see 8.3.9.2.3)				

Table 139 — Volume Statistics log parameter codes (part 2 of 2)

Parameter Code	Description	Type	Support	Reference
0045h	Volume personality	S	M	8.3.9.3.30
0046h	Volume manufacture date	S	O	8.3.9.3.32
0046h to 7Fh	Reserved			
0080h	Write protect	C	M	8.3.9.3.32
0081h	WORM	C	M	8.3.9.3.33
0082h	Maximum recommended tape path temperature exceeded	C	M	8.3.9.3.34
0083h - 00FFh	Reserved			
0100h	Volume write mounts	C	O	8.3.9.3.35
0101h	Beginning of medium passes	C	M	8.3.9.3.36
0102h	Middle of tape passes	C	M	8.3.9.3.37
0103h to 01FFh	Reserved			
0200h	Logical position of first encrypted logical object	P	M	8.3.9.3.38
0201h	Logical position of first unencrypted logical object after the first encrypted logical object	P	M	8.3.9.3.39
0202h	Native capacity of partition(s)	P	M	8.3.9.3.40
0203h	Used native capacity of partition(s)	P	M	8.3.9.3.41
0204h	Remaining native capacity of partition(s)	P	O	8.3.9.3.42
0205h to 2FFFh	Reserved			
0300h	Mount history	-	O	8.3.9.3.43
0301h to EFFFh	Reserved			
F000h to FFFFh	Vendor specific			
Type key: C=Volume statistics counter log parameter (see 8.3.9.2.1) S=Volume statistics string data log parameter (see 8.3.9.2.2) P=Volume statistics partition record log parameter (see 8.3.9.2.3)				

8.3.9.2 Volume statistics log parameter formats

8.3.9.2.1 Volume statistics data counter log parameter format

The volume statistics data counter log parameter is used for reporting parameters specified as volume statistics data counter log parameters (see table 139). The volume statistics data counter log parameter format is specified in table 140.

Table 140 — Volume statistics data counter log parameter format

Bit Byte	7	6	5	4	3	2	1	0
0	(MSB)							
1								(LSB)
2	DU	Obsolete	TSD (0b)	ETC	TMC			FORMAT AND LINKING (11b)
3								PARAMETER LENGTH (n-3)
4	(MSB)							
...								VOLUME STATISTICS DATA COUNTER
n								(LSB)

The PARAMETER CODE field is defined in table 139.

See SPC-4 for descriptions of the DU bit, TSD bit, ETC bit, TMC field and FORMAT AND LINKING field. The TSD bit and FORMAT AND LINKING field shall be set to the values specified in table 140.

The PARAMETER LENGTH field indicates the number of bytes in the VOLUME STATISTICS DATA COUNTER field that follows.

The VOLUME STATISTICS DATA COUNTER field is the value of the data counter associated with the parameter code.

8.3.9.2.2 Volume statistics string data log parameter format

The volume statistics string data log parameter is used for reporting parameters specified as volume statistics string data log parameters (see table 139). The volume statistics string data log parameter shall be a multiple of 4 bytes. The volume statistics string data log parameter format is specified in table 141.

Table 141 — Volume statistics string data log parameter format

Bit Byte	7	6	5	4	3	2	1	0
0	(MSB)							
1								(LSB)
2	DU (0b)	Obsolete	TSD (0b)	ETC (0b)	TMC (00b)			FORMAT AND LINKING (01b)
3								PARAMETER LENGTH (n-3)
4								
...								STRING DATA
n								

The PARAMETER CODE field is defined in table 139.

See SPC-4 for descriptions of the DU bit, TSD bit, ETC bit, TMC field and FORMAT AND LINKING field. These fields shall be set as specified in table 141.

The PARAMETER LENGTH field indicates the number of bytes that follow.

The STRING DATA field contains an ASCII string describing the volume statistics parameter specified by the PARAMETER CODE field value. The STRING DATA field is an ASCII data field (see SPC-4).

8.3.9.2.3 Volume statistics partition record log parameter format

The volume statistics partition record log parameter is used for reporting parameters specified as volume statistics partition record log parameters (see table 139). The volume statistics partition record log parameter format is specified in table 142.

Table 142 — Volume statistics partition record log parameter format

Bit Byte	7	6	5	4	3	2	1	0
0	(MSB)							
1								(LSB)
2	DU	Obsolete	TSD (0b)	ETC	TMC			FORMAT AND LINKING (11b)
3						PARAMETER LENGTH (n-3)		
						Volume statistics partition record descriptor(s)		
4								
...						Volume statistics partition record descriptor [first] (see table 143)		
						⋮		
...						Volume statistics partition record descriptor [last] (see table 143)		
n								

The PARAMETER CODE field is defined in table 139.

See SPC-4 for descriptions of the DU bit, TSD bit, ETC bit, TMC field and FORMAT AND LINKING field. The TSD bit and FORMAT AND LINKING field shall be set to the values specified in table 142.

The PARAMETER LENGTH field indicates the number of bytes in the volume statistics partition record descriptors.

The volume statistics partition record descriptor format is specified in table 143.

Table 143 — Volume statistics partition record descriptor format

Bit Byte	7	6	5	4	3	2	1	0
0	PARTITION RECORD DESCRIPTOR LENGTH (n)							
1	Reserved							
2	(MSB)	PARTITION NUMBER						(LSB)
3								
4	(MSB)	PARTITION RECORD DATA COUNTER						
...								
n								(LSB)

The PARTITION RECORD DESCRIPTOR LENGTH field specifies the number of bytes that follow.

The PARTITION NUMBER field indicates the number of the partition that the following counter is associated with.

The PARTITION RECORD DATA COUNTER field is the value of the data counter associated with the parameter code and associated with the specified partition.

8.3.9.3 Volume statistics log parameters

8.3.9.3.1 Page valid

The page valid parameter indicates whether the values reported in the parameters to follow are valid.

The least significant byte in the page valid parameter data counter value set to 01h and all other bytes set to 00h indicates that the values reported in the parameters to follow are valid. A zero value in the page valid parameter data counter value indicates that the values reported in the parameters to follow are not valid.

8.3.9.3.2 Thread count

The thread count parameter contains the total number of times the medium has been threaded over the lifetime of the volume. The thread count parameter shall be incremented each time the medium is successfully threaded.

8.3.9.3.3 Total data sets written

The total data sets written parameter contains the total number of data sets written to the medium in the volume over the lifetime of the volume. The data set size is a format or vendor specific value.

8.3.9.3.4 Total write retries

The total write retries parameter contains the total number of write retries for the lifetime of the volume.

The algorithm by which the device determines that a write retry is required is vendor specific.

8.3.9.3.5 Total unrecovered write errors

The total unrecovered write errors parameter contains the total number of times that a write type command was terminated with CHECK CONDITION status and a sense key of MEDIUM ERROR or HARDWARE ERROR over the lifetime of the volume.

8.3.9.3.6 Total suspended writes

The total suspended writes parameter contains the total number of times that the device suspended a write due to detection of a condition for which there is a possibility that a track be incorrectly written, and attempted to write the data at a different location. If a logical block is interrupted more than once this parameter shall only be incremented once.

8.3.9.3.7 Total fatal suspended writes

The total fatal suspended writes parameter contains the total number of times that the device suspended a write due to detection of a condition for which there is a possibility that a track be incorrectly written, and was unable to write the data at a different location.

8.3.9.3.8 Total data sets read

The total data sets read parameter contains the total number of data sets read from the medium in the volume over the lifetime of the volume. The data set size is a format or vendor specific value.

8.3.9.3.9 Total read retries

The total read retries parameter contains the total number of read retries for the lifetime of the volume.

The algorithm by which the device determines that a read retry is required is vendor specific.

8.3.9.3.10 Total unrecovered read errors

The total unrecovered read errors parameter contains the total number of times that a write type command was terminated with CHECK CONDITION status and a sense key of MEDIUM ERROR or HARDWARE ERROR over the lifetime of the volume.

8.3.9.3.11 Total suspended reads

The total suspended reads parameter contains the total number of times that the device suspended a read due to detection of a condition for which there was a possibility of an incorrectly read track. If a logical block is interrupted more than once this parameter shall only be incremented once.

8.3.9.3.12 Total fatal suspended reads

The total fatal suspended reads parameter contains the total number of times that the device suspended a read due to detection of a condition for which there was a possibility of an incorrectly read track, and was unable to successfully complete the read.

8.3.9.3.13 Last mount unrecovered write errors

The last mount unrecovered write errors parameter contains the count of the number times a write type command was terminated with status of CHECK CONDITION and a sense key of HARDWARE ERROR, or MEDIA ERROR during the last mount.

8.3.9.3.14 Last mount unrecovered read errors

The last mount unrecovered read errors parameter contains the count of the number times a read type command was terminated with status of CHECK CONDITION and a sense key of HARDWARE ERROR, or MEDIA ERROR during the last mount.

8.3.9.3.15 Last mount megabytes written

The last mount megabytes written parameter contains a count of the number of megabytes (i.e., 10^6 bytes) of logical objects that were written to the medium after compression during the last mount. The value reported shall be rounded up to the next megabyte. The value reported contains bytes written as part of the process of writing a filemark.

8.3.9.3.16 Last mount megabytes read

The last mount megabytes read parameter contains a count of the number of megabytes (i.e., 10^6 bytes) of logical objects that were read from the medium before decompression during the last mount. The value reported shall be rounded up to the next megabyte. The value reported contains bytes read as part of a filemark.

8.3.9.3.17 Lifetime megabytes written

The lifetime megabytes written parameter contains a count of the number of megabytes (i.e., 10^6 bytes) of logical objects that have been written to the medium after compression during the lifetime of the volume. The value reported shall be rounded up to the next megabyte. The value reported contains bytes written as part of the process of writing a filemark.

8.3.9.3.18 Lifetime megabytes read

The lifetime megabytes read parameter contains a count of the number of megabytes (i.e., 10^6 bytes) that have been read from the medium before decompression during the lifetime of the volume. The value reported shall be rounded up to the next megabyte. The value reported contains bytes read as part of a filemark.

8.3.9.3.19 Last load write compression ratio

The last load write compression ratio parameter contains a value calculated by the following formula:

(number of bytes transferred from an application client into the logical object buffer ÷ the number of bytes in logical objects written to the medium) x 100

8.3.9.3.20 Last load read compression ratio

The last load read compression ratio parameter contains a value calculated by the following formula:

(number of bytes transferred out of the logical object buffer to an application client ÷ the number of bytes in logical objects read from the medium) x 100

8.3.9.3.21 Medium mount time

The medium mount time parameter contains the time in milliseconds from the completion of a volume mount to the time that the device server no longer detects that the volume is present.

8.3.9.3.22 Medium ready time

The medium ready time parameter contains the time in milliseconds that the device server was in the ready state (see 4.2.2.1).

8.3.9.3.23 Total native capacity

The total native capacity parameter contains the sum of the total native capacity of all partitions in megabytes (i.e., 10^6 bytes) from BOP to EOP. A data counter value with all bytes set to FFh in the PARTITION RECORD DATA COUNTER field (see 8.3.9.2.3) indicates that the total native capacity is unknown.

8.3.9.3.24 Total used native capacity

The total used native capacity parameter contains the sum of the used native capacity of all partitions in megabytes (i.e., 10^6 bytes) from BOP to EOD. A data counter value with all bytes set to FFh in the PARTITION RECORD DATA COUNTER field (see 8.3.9.2.3) indicates that the total used native capacity is unknown.

8.3.9.3.25 Volume serial number

The volume serial number parameter contains the serial number assigned to the volume.

8.3.9.3.26 Tape lot identifier

The tape lot identifier field contains a lot identifier for the medium.

8.3.9.3.27 Volume barcode

The volume barcode parameter contains the value from MAM attribute 0806h (i.e., BARCODE, see SPC-4).

8.3.9.3.28 Volume manufacturer

The volume manufacturer parameter contains a T10 vendor ID format identifier for the volume manufacturer. If a T10 vendor ID is assigned for the volume manufacturer, then that value shall be reported in this parameter. If a T10 vendor ID has not been assigned to the volume manufacturer a string that conforms to the T10 vendor ID format and does not conflict with assigned T10 vendor ID's shall be reported.

8.3.9.3.29 Volume license code

The volume license code contains a vendor specified ASCII code to represent the license under which this volume was manufactured.

8.3.9.3.30 Volume personality

The volume personality parameter contains a vendor specified ASCII string to identify the combination of physical volume type and density formatted on the medium. The intent is to provide a designator sufficient for successful volume interchange (e.g., "FooXDG7" to designate a type X data volume compatible with Foo drives and with a generation 7 format or "FooYWG3" to designate a type Y WORM volume compatible with Foo drives and with a generation 3 format).

8.3.9.3.31 Volume manufacture date

The volume manufacture date parameter contains the date of manufacture of the volume. The format is YYYYMMDD (i.e., four numeric ASCII characters for the year followed by two numeric ASCII characters for the month followed by two numeric ASCII characters for the day with no intervening spaces).

8.3.9.3.32 Write protect

The write protect parameter indicates whether any persistent write protection conditions are set.

Persistent write protection conditions include physical write protection (e.g., mechanical switches). The least significant byte in the write protect parameter data counter value set to 01h and all other bytes set to 00h indicates that a write protection state that is persistent with this volume is set or was set at the time the volume was unloaded. A zero value in the write protect parameter data counter value indicates that no write protection state that is persistent with this volume is set or was set at the time the volume was unloaded.

All bytes in the write protect parameter data counter value set to FFh indicates that it is unknown if there is a persistent write protection condition set or it is unknown if there was a persistent write protection condition set at the time the volume was unloaded.

8.3.9.3.33 WORM

The WORM parameter indicates whether the volume is a WORM volume. The least significant byte in the WORM parameter data counter value set to 01h and all other bytes set to 00h indicates that the volume is a WORM volume. A zero value in the WORM parameter data counter value indicates that the volume is not a WORM volume. All bytes in the WORM parameter data counter value set to FFh indicates that it is unknown if the volume is a WORM volume.

8.3.9.3.34 Maximum recommended tape path temperature exceeded

The maximum recommended tape path temperature exceeded parameter indicates whether the device has detected that the maximum recommended tape path temperature has been exceeded while this volume was in a drive.

The maximum recommended tape path temperature exceeded codes are specified in table 144.

Table 144 — Maximum recommended tape path temperature exceeded codes

Code	Description
All bytes set to 00h	The device has not detected at any point in the past that the maximum recommended tape path temperature has been exceeded.
Least significant byte set to 01h and all other bytes set to 00h	The device has detected at some point in the past that the maximum recommended tape path temperature has been exceeded.
All bytes set to FFh	It is not known if at any point in the past that the maximum recommended tape path temperature has been exceeded.
All other values	Reserved

8.3.9.3.35 Volume write mounts

The volume write mounts parameter contains a count of the number of mounts in which logical objects (see 3.1.48) on the medium were written or modified.

8.3.9.3.36 Beginning of medium passes

The beginning of medium passes parameter contains a count of the total number of times the beginning of medium position has passed over the read/write mechanism.

8.3.9.3.37 Middle of medium passes

The middle of medium passes parameter contains a count of the total number of times that the physical middle of the user data region on the tape has passed over the read/write mechanism (e.g., $(\text{EOM} - \text{BOM}) \div 2$).

8.3.9.3.38 Logical position of first encrypted logical object

The logical position of first encrypted logical object parameter reports partition record descriptors containing the logical object identifier for the first logical object which has been encrypted in each partition. Each byte in the PARTITION RECORD DATA COUNTER field(s) shall be set to FFh if there are no encrypted logical objects in the specified partition. The least significant byte in the PARTITION RECORD DATA COUNTER field (see 8.3.9.2.3) shall be set to FEh and all other bytes shall be set to FFh if it is not known if there are encrypted logical objects in the specified partition.

8.3.9.3.39 Logical position of first unencrypted logical object after the first encrypted logical object

The logical position of first unencrypted logical object after the first encrypted logical object parameter indicates the logical object identifier of the first logical object in the partition that contains unencrypted data and follows the first logical object in the partition that contains encrypted data.

Each byte in the PARTITION RECORD DATA COUNTER field (see 8.3.9.2.3) shall be set to FFh if:

- a) each byte of the logical position of first encrypted logical object is set to FFh; or
- b) no logical object in the partition after the first encrypted logical object contains unencrypted data and the logical position of first encrypted logical object is set to any value other than:
 - A) a value with each byte set to FFh; or
 - B) a value with the least significant byte set to FEh and all other bytes set to FFh.

The least significant byte in the PARTITION RECORD DATA COUNTER field (see 8.3.9.2.3) shall be set to FEh and all other bytes in the PARTITION RECORD DATA COUNTER field shall be set to FFh if:

- a) the least significant byte of the logical position of first encrypted logical object is set to FEh and all other bytes in the logical position of first encrypted logical object are set to FFh; or
- b) it is unknown whether any logical object in the partition after the first encrypted logical object contains unencrypted data and the logical position of first encrypted object is set to any value other than:
 - A) a value with each byte set to FFh; or
 - B) a value with the least significant byte set to FEh and all other bytes set to FFh.

8.3.9.3.40 Native capacity of partition(s)

The native capacity of partition(s) parameter contains the native capacity of the partition(s) in megabytes (i.e., 10^6 bytes) from BOP to EOP. A data counter value with all bytes set to FFh in the PARTITION RECORD DATA COUNTER field (see 8.3.9.2.3) indicates that the native capacity of the partition is unknown.

8.3.9.3.41 Used native capacity of partition(s)

The used native capacity of partition(s) parameter contains the used native capacity of the partition in megabytes (i.e., 10^6 bytes) from BOP to EOD. A data counter value with all bytes set to FFh in the PARTITION RECORD DATA COUNTER field (see 8.3.9.2.3) indicates that the used native capacity of the partition is unknown.

8.3.9.3.42 Approximate remaining native capacity to early warning of partition(s)

The approximate remaining native capacity to early warning of partition(s) parameter contains the approximate remaining native capacity of the partition(s) in megabytes (i.e., 10^6 bytes) that is less than or equal to the native

capacity from EOD to EW. The value reported in this parameter shall be zero once EOD is at or beyond EW. A data counter value with all bytes set to FFh in the PARTITION RECORD DATA COUNTER field (see 8.3.9.2.3) indicates that the remaining native capacity of the partition is unknown.

8.3.9.3.43 Mount history

The mount history parameter contains the T10 vendor identification and unit serial number of the most recent devices into which this volume was mounted. The mount history log parameter format is specified in table 145.

Table 145 — Mount history log parameter format

The PARAMETER CODE field is defined in Table 3, Volume statistics log parameters.

See SPC-4 for descriptions of the DU bit, TSD bit, ETC bit, TMC field and FORMAT AND LINKING field. The TSD bit and FORMAT AND LINKING field shall be set to the values specified in table 110.

The PARAMETER LENGTH field indicates the number of bytes in the mount history log descriptors.

The mount history descriptor format is specified in table 146.

Table 146 — Mount history descriptor format

Bit Byte	7	6	5	4	3	2	1	0
0	MOUNT HISTORY DESCRIPTOR LENGTH (n)							
1	Reserved							
2	(MSB)	MOUNT HISTORY INDEX						
3								(LSB)
4	(MSB)	MOUNT HISTORY VENDOR ID						
...								(LSB)
11								
12	(MSB)	MOUNT HISTORY UNIT SERIAL NUMBER						
...								
n								(LSB)

The MOUNT HISTORY DESCRIPTOR LENGTH field specifies the number of bytes that follow.

The MOUNT HISTORY INDEX field indicates which previous mount was to the device specified in this descriptor. The value is incrementing starting at zero (e.g., the current device, or most recent device if not currently mounted, is numbered 0000h, the previous device is numbered 0001h).

The MOUNT HISTORY VENDOR ID field indicates the value returned in the T10 VENDOR IDENTIFICATION field of the standard inquiry data associated with this mount history index.

The MOUNT HISTORY UNIT SERIAL NUMBER field indicates the value of the PRODUCT SERIAL NUMBER field of the Unit Serial Number vital product data page associated with this mount history index. The MOUNT HISTORY UNIT SERIAL NUMBER field:

- a) shall be right-aligned;
- b) shall have all unused bytes filled with ASCII space characters (i.e., 20h); and
- c) shall be a multiple of two bytes in length.

8.4 Medium auxiliary memory attributes

8.4.1 Medium auxiliary memory device type attributes

Medium auxiliary memory device type attributes are specified in table 147.

Table 147 — Medium auxiliary memory device type attributes

ID	Attribute name	Number of bytes	Format
0002h	TAPEALERT FLAGS	8	Binary
0005h	ASSIGNING ORGANIZATION	8	ASCII
0006h	FORMATTED DENSITY CODE	1	Binary
0009h	VOLUME CHANGE REFERENCE	Not specified ^a	Binary

a) A fixed length shall be defined by the volume format.

The TAPEALERT FLAGS attribute provides a means of reporting the state of the TapeAlert flags for the previous load of the volume. Each TapeAlert flag occupies one bit (Flag 1 = MSB, byte 1; Flag 64 = LSB, byte 8). The bits specify all the TapeAlert flags that were set to one during the previous load, (i.e., the bits remain set to one for the duration of the load).

The ASSIGNING ORGANIZATION attribute identifies the organization responsible for the specifications defining the values in the FORMATTED DENSITY CODE attribute. The ASSIGNING ORGANIZATION attribute should contain a vendor identification. A vendor identification other than the one associated with the device may be used.

NOTE 50 - It is intended that the ASSIGNING ORGANIZATION attribute provide a unique vendor identification of the FORMATTED DENSITY CODE attribute. In the absence of a formal registration procedure, T10 maintains a list of known vendor identification codes for use in the Standard INQUIRY data (see SPC-4). Vendors are requested to voluntarily submit their identification codes to T10 to prevent duplication of codes (see SPC-4).

If the device server formats the medium using a format other than the one specified in the MEDIUM DENSITY CODE attribute (see 8.4.2) (e.g., for compatibility with a previous generation format), then the FORMATTED DENSITY CODE attribute specifies the DENSITY CODE of the format chosen (see SPC-4). Otherwise this attribute shall be the same as the MEDIUM DENSITY CODE attribute.

The VOLUME CHANGE REFERENCE attribute indicates changes in the state of the medium related to logical objects or format-specific symbols of the currently mounted recording volume. There is one value for the volume change reference and the VOLUME CHANGE REFERENCE attribute for each partition shall use the same value. The VOLUME CHANGE REFERENCE attribute value shall:

- a) be written to non-volatile medium auxiliary memory before the change on medium is valid for reading; and
- b) change in a non-repeating fashion (i.e., never repeat for the life of the recording volume) in a consistent manner as defined by the volume format.

The VOLUME CHANGE REFERENCE attribute value shall change if:

- a) the first logical object for each mount is written on the medium in any partition;
- b) the first logical object is written after GOOD status has been returned for a READ ATTRIBUTE command with the SERVICE ACTION field set to ATTRIBUTE VALUES (i.e., 00h) and the FIRST ATTRIBUTE IDENTIFIER field set to VOLUME CHANGE REFERENCE (i.e., 0009h);
- c) any logical object on the medium (i.e., in any partition) is overwritten; or
- d) the medium is formatted.

The VOLUME CHANGE REFERENCE attribute may change at other times if the contents on the medium change.

The VOLUME CHANGE REFERENCE attribute should not change if the logical objects on the medium do not change.

A value of zero in the VOLUME CHANGE REFERENCE attribute indicates that the medium has not had any logical objects written to it (i.e., the recording volume is blank and has never been written to) or the value is unknown.

A value of all ones (e.g., 0xFFFFh) in the VOLUME CHANGE REFERENCE attribute indicates that all values have been used. This value indicates the VOLUME CHANGE REFERENCE value is no longer able to indicate changes to the recording volume. The device server may or may not allow further modifications of the medium.

If adding or modifying logical objects, then the VOLUME CHANGE REFERENCE attribute should only be read after all writing to the recording volume has completed and been synchronized.

8.4.2 Medium auxiliary memory medium type attributes

Medium auxiliary memory medium type attributes are specified in table 148.

Table 148 — Medium auxiliary memory medium type attributes

ID	Attribute name	Number of bytes	Format
0402h	MEDIUM LENGTH	4	Binary
0403h	MEDIUM WIDTH	4	Binary
0404h	ASSIGNING ORGANIZATION	8	ASCII
0405h	MEDIUM DENSITY CODE	1	Binary

The MEDIUM LENGTH attribute specifies the length of the medium in meters. A value of 0h specifies that the length of the medium is undefined.

The MEDIUM WIDTH attribute specifies the width of the medium supported by this density. This attribute has units of tenths of millimeters. The value in this attribute shall be rounded up if the fractional value of the actual value is greater than or equal to 0.5. The MEDIUM WIDTH attribute may vary for a given density depending on the mounted volume. A value of 0h specifies the width of the medium is undefined.

The ASSIGNING ORGANIZATION attribute identifies the organization responsible for the specifications defining the values in the MEDIUM DENSITY CODE attribute. The ASSIGNING ORGANIZATION attribute should contain a vendor identification.

NOTE 51 - It is intended that the ASSIGNING ORGANIZATION attribute provide a unique vendor identification of the MEDIUM DENSITY CODE attribute. In the absence of a formal registration procedure, T10 maintains a list of known vendor identification codes for use in the Standard INQUIRY data (see SPC-4). Vendors are requested to voluntarily submit their identification codes to T10 to prevent duplication of codes (see SPC-4).

The MEDIUM DENSITY CODE attribute specifies the DENSITY CODE (see SPC-4) that is used as the default for the volume.

8.4.3 Medium auxiliary memory host type attributes

Medium auxiliary memory host type attributes are specified in table 149

Table 149 — Medium auxiliary memory host type attributes

ID	Attribute name	Number of bytes	Format
080Ch	VOLUME COHERENCY INFORMATION	Not specified	Binary

The VOLUME COHERENCY INFORMATION attribute contains information used to maintain coherency of information on a recording volume (see annex C). The VOLUME COHERENCY INFORMATION attribute ATTRIBUTE VALUE field is defined in table 150.

Table 150 — VOLUME COHERENCY INFORMATION attribute format

Bit Byte	7	6	5	4	3	2	1	0
0	VOLUME CHANGE REFERENCE VALUE LENGTH (n)							
1	VOLUME CHANGE REFERENCE VALUE							
n	VOLUME COHERENCY COUNT							
n+ 1	VOLUME COHERENCY SET IDENTIFIER							
n+ 8	APPLICATION CLIENT SPECIFIC INFORMATION LENGTH (y-n-18)							
n+ 9	APPLICATION CLIENT SPECIFIC INFORMATION							
n+16								
n+17								
n+18								
n+19								
y								

The contents of the VOLUME CHANGE REFERENCE VALUE field, the VOLUME COHERENCY COUNT field, the VOLUME COHERENCY SET IDENTIFIER field, and the APPLICATION CLIENT SPECIFIC INFORMATION field are specified in (see annex C).

8.5 Mode parameters

8.5.1 Mode parameters overview

This subclause defines the descriptors and pages for mode parameters used with sequential-access devices.

The mode parameter list, including the mode parameter header and mode block descriptor, are described in SPC-4.

The MEDIUM TYPE field in the mode parameter header is vendor specific for sequential-access devices.

The value of the BLOCK LENGTH field in the mode parameter block descriptor shall be a multiple of four.

NOTE 52 - The block length field is limited to multiples of four to ensure data integrity is maintained if fixed-block transfers are performed using transports such as Fibre Channel.

The DEVICE-SPECIFIC PARAMETER field in the mode parameter header is defined in table 151 for sequential-access devices.

Table 151 — Device-specific parameter

Bit Byte	7	6	5	4	3	2	1	0
0	WP		BUFFERED MODE				SPEED	

If used with the MODE SENSE command, a write protection (WP) bit set to zero specifies the volume is write enabled. A WP bit set to one specifies the volume is currently in the write protected state. If used with the MODE SELECT command, this field is ignored.

NOTE 53 - The write protected state may be due to logical unit internal restrictions, software write protection, or physical write protection.

The BUFFERED MODE field code values are specified in table 152.

Table 152 — Buffered modes

Code	Description
0h	The device server shall not report GOOD status on WRITE commands until the logical blocks are actually written on the medium.
1h	The device server may report GOOD status on WRITE commands as soon as all the data specified in the WRITE command has been transferred to the logical unit's object buffer. One or more logical blocks may be buffered prior to writing the logical block(s) to the medium.
2h	The device server may report GOOD status on WRITE commands as soon as: <ul style="list-style-type: none"> a) all the data specified in the WRITE command has been successfully transferred to the logical unit's object buffer; and b) all buffered logical objects from different initiators has been successfully written to the medium.
3h to 7h	Reserved

The SPEED field code values are specified in table 153.

Table 153 — SPEED field

Code	Description
0h	Default (use the device's default speed).
1h	Use the device's lowest speed.
2h to Fh	Use increasing device speeds.

For the MODE SELECT command, the DENSITY CODE field of the sequential-access device block descriptor (see SPC-4) specifies the density selected by the application client for use in subsequent read and write operations. For logical units capable of automatic density recognition, the density code selected by the application client may be overridden by the logical unit for a subsequent read operation if the selected value does not match the current

recorded density of the medium. If the MODE SELECT command specifies the default density code the logical unit selects the actual density code to be used in a vendor-specific manner. The value should be the principal density code (or an optimal density code).

For the MODE SENSE command, the DENSITY CODE field reflects the current operating density of the logical unit. If a current operating density has not been selected, either because no volume is mounted or because the density of the installed volume has not been determined, the DENSITY CODE field should be set to the principal density code value (see 3.1.63). For some logical units, the principal density code value returned in response to a MODE SENSE command may change dynamically to match the most recently detected density. The DENSITY CODE value returned in response to a MODE SENSE command shall be determined as follows:

- a) following a logical unit reset, if the logical unit is in the not ready state, the device server shall report the principal density;
- b) following a unit attention condition for a not-ready-to-ready transition or an unsuccessful read operation, the device server shall:
 - A) report the principal density if no attempt has been made by the logical unit to determine the density;
 - B) report the principal density if the logical unit is unable to automatically determine the density from the volume; or
 - C) report the current medium density if the logical unit has determined the density from the volume.
- c) following a successful read operation, the device server shall report a density code value reflecting the recorded density of the medium. For some implementations, the logical unit may automatically determine this value from the volume. For devices not capable of automatic density determination, the principal density is reported if the density code value is not provided by the preceding MODE SELECT command;
- d) following a successful write operation, the device server shall report a density code value reflecting the most recently recorded density of the medium;
- e) following an unsuccessful read operation or an unsuccessful write operation, while at beginning-of-partition, the device server shall report a density code value as described for item (b);
- f) following a successful demount, the device server shall report the most recent density code value as determined by items (b) through (e) above; or
- g) following a logical unit reset, if the logical unit is in the ready state, the device server shall retain knowledge of the density code as determined by items (b) through (e) above.

For a MODE SELECT command, a density code of 7Fh specifies the application client is not selecting a density. The value 7Fh shall not be returned by a MODE SENSE command. Table 154 specifies the sequential-access device density codes.

Table 154 — Sequential-access density codes

Code	Description
00h ^a	Default density.
01h to 7Eh	Density code from REPORT DENSITY SUPPORT command.
7Fh ^b	No change from previous density (NO-OP).
80h to FFh	Density code from REPORT DENSITY SUPPORT command.
	<ul style="list-style-type: none"> a) Only reported by MODE SENSE commands if primary density code for the density. b) This density code value is defined for the MODE SELECT command and shall not be returned by the MODE SENSE command.

The mode page codes and subpage codes for sequential-access devices are specified in table 155. All page code and subpage code combinations not shown in table 155 are reserved.

Table 155 — Mode page codes and subpage codes

Page code	Subpage code	Mode page name	Support	Reference
00h	not applicable	Vendor specific (does not require page format)	-	
01h	00h	Read-Write Error Recovery	M	8.5.5
02h	00h	Disconnect-Reconnect	M	SPC-4
03h to 08h	00h - FEh	Reserved	-	
09h	00h	Obsolete	-	3.4
0Ah	00h	Control	M	SPC-4
0Ah	01h	Control Extension	O	SPC-4
0Ah	F0h	Control Data Protection	O	8.5.9
0Bh to 0Eh	00h - FEh	Reserved	-	
0Fh	00h	Data Compression	M	8.5.2
10h	00h	Device Configuration	M	8.5.3
10h	01h	Device Configuration Extension	O	8.5.8
11h	00h	Medium Partition	O	8.5.4
12h	00h	Obsolete	-	3.4
13h	00h	Obsolete	-	3.4
14h	00h	Obsolete	-	3.4
15h	00h	Extended	O	SPC-4
16h	00h	Extended Device-Type Specific	O	SPC-4
17h	00h - FEh	Reserved	-	
18h	00h	Protocol Specific Logical Unit	M ^b	SPC-4
18h	01h to FEh		O	SPC-4
19h	00h	Protocol Specific Port	M ^c	SPC-4
a) Valid only for MODE SENSE command. b) Mandatory only if explicit command set is supported. c) Mandatory only if supported by the SCSI transport protocol. d) See SPC-4 for support requirements.				

Table 155 — Mode page codes and subpage codes (Continued)

Page code	Subpage code	Mode page name	Support	Reference
19h	01h to FEh		O	SPC-4
1Ah	00h	Power Condition	O	SPC-4
1Bh	00h - FEh	Reserved	-	
1Ch	00h	Informational Exceptions Control	M	8.5.6
1Dh	00h	Medium Configuration	M	8.5.7
1Eh to 1Fh	00h - FEh	Reserved	-	
20h to 3Eh	00h - FEh	Vendor specific (does not require page format)	-	
3Fh	00h	Return all pages ^a	^c d	SPC-4
3Fh	FFh	Return all pages and subpages ^a	^c d	SPC-4
00h to 3Eh	FFh	Return all subpages ^a	^c d	SPC-4

a) Valid only for MODE SENSE command.
 b) Mandatory only if explicit command set is supported.
 c) Mandatory only if supported by the SCSI transport protocol.
 d) See SPC-4 for support requirements.

8.5.2 Data Compression mode page

The Data Compression mode page (see table 156) specifies the parameters for the control of data compression in a sequential-access device.

Table 156 — Data Compression mode page

Bit Byte	7	6	5	4	3	2	1	0
0	PS	SPF(0)			PAGE CODE (0Fh)			
1					PAGE LENGTH (0Eh)			
2	DCE	DCC			Reserved			
3	DDE		RED		Reserved			
4	(MSB)				COMPRESSION ALGORITHM			
...								
7								(LSB)
8	(MSB)				DECOMPRESSION ALGORITHM			
...								
11								(LSB)
12					Reserved			
...								
15								

See SPC-4 for a description of the PS bit, SPF bit, PAGE CODE field, and PAGE LENGTH field.

A data compression enable (DCE) bit set to one requests that data compression be enabled. A DCE bit set to zero requests that data compression be disabled. See 4.2.17 for the interaction between the DCE bit, the COMPRESSION

ALGORITHM field, and the SELECT DATA COMPRESSION ALGORITHM field in the Data Configuration mode page (see 8.5.3).

A data compression capable (DCC) bit set to one specifies the device supports data compression and is capable of processing data sent to it for transferal to the medium using the selected compression algorithm. A DCC bit set to zero specifies the device does not support data compression. This shall be a nonchangeable bit.

A data decompression enable (DDE) bit set to one specifies data decompression is enabled. A DDE bit set to zero specifies data decompression is disabled. Uncompressed data shall be unaffected by the setting of the DDE bit.

The report exception on decompression (RED) field specifies the response to certain boundaries detected in the data on the medium. There are a number of boundaries that may occur on the medium between compressed and uncompressed data. These boundaries are shown in table 157. Only boundaries shown in table 157 may generate a CHECK CONDITION status.

Table 157 — Possible boundaries and resulting sense keys due to data compression

Prior data	Current data	Sense Key ^{a,b}		
		RED field value		
		00b	01b	10b
Uncompressed	Compressed (unsupported algorithm)	MEDIUM ERROR	MEDIUM ERROR	MEDIUM ERROR
Uncompressed	Compressed (supported algorithm)	[none]	[none]	RECOVERED ERROR
Compressed (supported algorithm)	Uncompressed	[none]	[none]	NO SENSE
Compressed (supported algorithm)	Compressed (unsupported algorithm)	MEDIUM ERROR	MEDIUM ERROR	MEDIUM ERROR
Compressed (supported algorithm A)	Compressed (supported algorithm B)	[none]	[none]	RECOVERED ERROR
Compressed (unsupported algorithm)	Uncompressed	[none]	NO SENSE	NO SENSE
Compressed (unsupported algorithm)	Compressed (supported algorithm)	[none]	RECOVERED ERROR	RECOVERED ERROR
Compressed (unsupported algorithm A)	Compressed (unsupported algorithm B)	MEDIUM ERROR	MEDIUM ERROR	MEDIUM ERROR
All other combinations		[none]	[none]	[none]
a) [none] specifies no CHECK CONDITION status is returned given the data boundary condition and the current value of the RED field. b) The appropriate additional sense code is specified following this table in this subclause.				

If a CHECK CONDITION status is returned and the current data is compressed, the additional sense code shall be set to either DECOMPRESSION EXCEPTION SHORT ALGORITHM ID OF NN with the additional sense code qualifier set to the algorithm id or DECOMPRESSION EXCEPTION LONG ALGORITHM with no additional sense code qualifier.

If a CHECK CONDITION status is returned and the current data is uncompressed, the additional sense code shall be set to DECOMPRESSION EXCEPTION SHORT ALGORITHM ID OF NN with the additional sense code qualifier set to zero.

A RED field set to 00b specifies the device shall return a CHECK CONDITION status if data is encountered on the medium during a read operation that the device is unable to decompress. Data boundaries in table 157 marked other than [none] in the column for RED field values of zero shall generate CHECK CONDITION status with the specified sense key if the RED field is zero.

A RED field set to 01b specifies the device shall return a CHECK CONDITION status if data is encountered on the medium during a read operation that requires different handling by the application client than the data most recently encountered during a prior read operation. At each of these boundaries, the data that is sent to the application client is of a fundamentally different nature from that which was previously sent. Data boundaries in table 157 marked other than [none] in the column for RED field values of one shall generate CHECK CONDITION status with the specified sense key if the RED field is one.

A RED field set 10b specifies the device shall return a CHECK CONDITION status if data is encountered on the medium during a read operation that has been processed using a different algorithm from that data most recently encountered during a prior read operation. Data boundaries in table 157 marked other than [none] in the column for RED field values of two shall generate CHECK CONDITION status with the specified sense key if the RED field is two.

A RED field set to 11b is reserved. If a mode page containing a RED field set to 11b is received, the MODE SELECT command shall be terminated with CHECK CONDITION status, the sense key shall be set to ILLEGAL REQUEST, and the additional sense code shall be set to INVALID FIELD IN PARAMETER LIST.

Upon detection of any of the boundary conditions described in table 157 that results in a CHECK CONDITION status, the additional sense code shall be set to either DECOMPRESSION EXCEPTION SHORT ALGORITHM ID OF NN (if the algorithm identifier is less than or equal to 255) or DECOMPRESSION EXCEPTION LONG ALGORITHM ID. The device shall, in both cases, set the DECOMPRESSION ALGORITHM field to the algorithm identifier of the compression algorithm used to process the encountered data. The logical position shall be on the EOP side of the encountered data, and the INFORMATION field in the sense data shall contain a count of the number of logical blocks contained within the encountered data.

If compressed data is encountered on the medium that the device server is unable to decompress, then the device server shall return a CHECK CONDITION status. The sense key shall be set to MEDIUM ERROR and the additional sense code shall be set to CANNOT DECOMPRESS USING DECLARED ALGORITHM. Undecompressed data may be returned to the application client as a single variable length logical block with the ILI bit and INFORMATION fields set accordingly. The logical position is vendor specific following this condition.

NOTE 54 - The undecompressed data may contain more than one logical object. As such, the application client should issue a READ POSITION command following this condition to re-establish positioning.

The COMPRESSION ALGORITHM field specifies the currently selected compression algorithm. The default value of the COMPRESSION ALGORITHM field shall specify the default compression algorithm for the device. The field specifies the compression algorithm the device shall use to process data sent to it by the application client if compression is enabled.

If the application client selects an algorithm that the device does not support, then the device shall return a CHECK CONDITION status. The sense key shall be set to ILLEGAL REQUEST and the additional sense code shall be set to INVALID FIELD IN PARAMETER LIST. Algorithm identifiers are shown in table 158. See 4.2.17 for the interaction between the DCE bit, the COMPRESSION ALGORITHM field, and the SELECT DATA COMPRESSION ALGORITHM field in the Data Configuration mode page (see 8.5.3).

For the MODE SELECT command, the DECOMPRESSION ALGORITHM field specifies the decompression algorithm selected by the application client for use in subsequent decompression of data encountered on the medium. For devices capable of the automatic recognition of the compression algorithm used to process data encountered on the medium, the decompression algorithm selected by the application client may be ignored, or overridden by the logical unit for a subsequent read operation if the selected value does not match the compression algorithm that was used to process the data encountered on the medium.

For the MODE SENSE command, the DECOMPRESSION ALGORITHM field reflects the algorithm selected by the application client. For some devices, the DECOMPRESSION ALGORITHM value returned in response to a MODE SENSE command may change dynamically to match the compression algorithm, detected by the device, that was used to process the data most recently encountered on the medium, during a read operation. A value of zero specifies the data encountered on the medium during the most recent read operation was uncompressed. Compression algorithm identifiers are shown in table 158.

Table 158 — Compression algorithm identifiers

Algorithm identifier	Description
0000 0000h	No algorithm selected (i.e., identifies uncompressed data).
0000 0001h	Set with MODE SELECT to select the default algorithm. MODE SENSE shall return the actual compression algorithm that was selected.
0000 0002h	Reserved
0000 0003h	IBM ALDC ^a data compression algorithm with 512 byte buffer.
0000 0004h	IBM ALDC ^a data compression algorithm with 1024 byte buffer.
0000 0005h	IBM ALDC ^a data compression algorithm with 2048 byte buffer.
0000 0006h to 0000 000Fh	Reserved
0000 0010h	IBM IDRC ^b data compaction algorithm.
0000 0011h to 0000 001Fh	Reserved
0000 0020h	DCLZ ^c data compression algorithm.
0000 0021h to 0000 007Fh	Reserved
0000 0080h to 0000 00FFh	Unregistered algorithm.
0000 0100h to FFFFFFFFh	Reserved

a) Adaptive Lossless Data Compression (see ISO/IEC 15200:1996).
 b) Improved Data Recording Capability
 c) Data Compression according to Lempel and Ziv (see ISO/IEC 11558:1992).

8.5.3 Device Configuration mode page

The Device Configuration mode page (see table 159) is used to specify the appropriate sequential-access device configuration.

Table 159 — Device Configuration mode page

Bit Byte	7	6	5	4	3	2	1	0
0	PS	SPF(0)			PAGE CODE (10h)			
1				PAGE LENGTH (0Eh)				
2	Rsvd	Obsolete	CAF		ACTIVE FORMAT			
3				ACTIVE PARTITION				
4				WRITE OBJECT BUFFER FULL RATIO				
5				READ OBJECT BUFFER EMPTY RATIO				
6	(MSB)				WRITE DELAY TIME			
7								(LSB)
8	OBR	LOIS	Obsolete	AVC	SOCF	ROBO	REW	
9				Obsolete				
10		EOD DEFINED		EEG	SEW	SWP	BAML	BAM
11	(MSB)							
...				OBJECT BUFFER SIZE AT EARLY WARNING				
13								(LSB)
14				SELECT DATA COMPRESSION ALGORITHM				
15	WTRE		OIR	REWIND ON RESET	ASOCWP	PERSWP	PRMW	

See SPC-4 for a description of the PS bit, SPF bit, PAGE CODE field, and PAGE LENGTH field.

NOTE 55 - The change active partition (CAP) bit (byte 2, bit 6 in the Device Configuration mode page) has been obsoleted. To change active partitions refer to the LOCATE command.

A change active format (CAF) bit set to one specifies the active format is to be changed to the value specified in the ACTIVE FORMAT field. A CAF bit set to zero specifies no active format change is specified. For some devices, the format may only be changed if the logical unit is at beginning-of-partition.

The ACTIVE FORMAT field specifies the recording format that is in use for the selected density code while reading or writing data on a logical unit. The value of the ACTIVE FORMAT field is vendor specific.

The ACTIVE PARTITION field specifies the current logical partition number in use on the recording volume. This shall be a non-changeable field.

The WRITE OBJECT BUFFER FULL RATIO field, on WRITE commands, specifies to the device server how full the object buffer shall be before writing data to the medium. A value of zero specifies the value is not specified.

The READ OBJECT BUFFER EMPTY RATIO field, on READ commands, specifies to the device server how empty the object buffer shall be before retrieving additional data from the medium. A value of zero specifies the value is not specified.

The WRITE DELAY TIME field specifies the maximum time, in 100 ms increments, that the device server should wait before any buffered data that is to be written, is forced to the medium after the last buffered WRITE command that did not cause the object buffer to exceed the write object buffer full ratio. A value of zero specifies the device server shall never force buffered data to the medium under these conditions.

An object buffer recovery (OBR) bit set to one specifies the logical unit supports object buffer recovery using the RECOVER BUFFERED DATA command. An OBR bit set to zero specifies the logical unit does not support object buffer recovery. Most device servers consider this bit to be not changeable.

A logical object identifiers supported (LOIS) bit set to zero specifies logical object identifiers are not supported in the format written on the medium. A LOIS bit set to one specifies the format on the medium has recorded information about the logical object identifiers relative to a partition. Most device servers consider this bit to be not changeable.

The automatic velocity control (AVC) bit set to one, specifies the device shall select the speed (if the device supports more than one speed) based on the data transfer rate that should optimize streaming activity and minimize medium repositioning. An AVC bit set to zero specifies the speed chosen shall be defined by the SPEED field in the mode parameter header.

A stop on consecutive filemarks (SOCF) field of 00b specifies the device server shall pre-read data from the medium to the limits of the object buffer capacity without regard for filemarks. Values 01b, 10b, and 11b specify that the device server shall terminate the pre-read operation if one, two, or three consecutive filemarks are detected, respectively.

A recover object buffer order (ROBO) bit set to one specifies logical blocks shall be returned from the object buffer of the logical unit on a RECOVER BUFFERED DATA command in LIFO order (last-in-first-out) from that they were written to the object buffer. A ROBO bit set to zero specifies logical blocks shall be returned in FIFO (first-in-first-out) order.

A report early-warning (REW) bit set to zero specifies the device server shall not report the early-warning condition for read operations and it shall report early-warning at or before any medium-defined early-warning position during write operations. Application clients should set the REW bit to zero.

A REW bit set to one specifies the device server shall return CHECK CONDITION status with the additional sense code set to END-OF-PARTITION/MEDIUM DETECTED, and the EOM bit set to one in the sense data if early-warning position is encountered during read and write operations. If the REW bit is one and the SEW bit is zero, the device server shall return CHECK CONDITION status with the sense key set to VOLUME OVERFLOW if early-warning is encountered during write operations.

NOTE 56 - A REW bit set to one is intended for compatibility with application clients using legacy formats that require an early-warning indication during read operations.

The EOD DEFINED field (see table 160) specifies the format type that the logical unit shall use to detect and generate the EOD area.

Table 160 — EOD DEFINED field

Code	Description
000b	Logical unit's default EOD definition
001b	Format-defined erased area of medium
010b	As specified in the SOCF field
011b	EOD recognition and generation is not supported
100b to 111b	Reserved

An enable EOD generation (EEG) bit set to one specifies the logical unit shall generate the appropriate EOD area, as determined by the EOD field. A value of zero specifies EOD generation is disabled.

NOTE 57 - Some logical units may not generate EOD at the completion of any write type operation.

A synchronize at early-warning (SEW) bit set to one specifies the logical unit shall cause any buffered logical objects to be transferred to the medium prior to returning status if positioned between early-warning and EOP. A SEW bit set to zero specifies the logical unit may retain unwritten buffered logical objects in the object buffer if positioned between early-warning and EOP (see 5.6, 5.7, 6.8, and 6.9).

A software write protection (SWP) bit set to one specifies the device server shall perform a synchronize operation and then enter the write-protected state (see 4.2.20 and 4.2.20.3). If the SWP bit is set to one, all commands requiring eventual writes to the medium shall return CHECK CONDITION status. The sense key shall be set to DATA PROTECT and the additional sense code should be set to LOGICAL UNIT SOFTWARE WRITE PROTECTED (see 4.2.20.2). A SWP bit set to zero specifies the device server may inhibit writing to the medium, dependent on other write inhibits.

A block address mode lock (BAML) bit set to zero specifies the selection of the block address mode shall be determined based on the first block address mode unique command that is received after a successful mount or a successful completion of a command that positions the medium to BOP. A BAML bit set to one specifies the selection of the block address mode shall be determined based on the setting of the BAM bit. See 4.2.23 for a description of block address mode selection.

The block address mode (BAM) bit is valid only if the BAML bit is set to one. If the BAML bit is set to zero, the BAM bit shall be ignored. If the BAML bit is set to one and the BAM bit is set to zero, the logical unit shall operate using implicit address mode. If the BAML bit is set to one and the BAM bit is set to one, the logical unit shall operate using explicit address mode. See 4.2.23 for a description of block address mode selection.

The OBJECT BUFFER SIZE AT EARLY WARNING field specifies the value, in bytes, that the logical unit shall reduce its logical object buffer size to if writing in a position between its early-warning and end-of-partition. A value of zero specifies the implementation of this function is vendor specific.

NOTE 58 - The intent is to prevent the loss of data by limiting the size of the object buffer if nearing the end-of-partition.

The SELECT DATA COMPRESSION ALGORITHM field set to 00h requests that the logical unit not use a compression algorithm on any data sent to it prior to writing the data to the medium. A value of 01h requests that data to be written be compressed using the logical unit's default compression algorithm. Values 02h through 7Fh are reserved. Values 80h through FFh are vendor specific. See 4.2.17 for the interaction between the DCE bit, the COMPRESSION ALGORITHM field, and the SELECT DATA COMPRESSION ALGORITHM field.

New implementations should use the Data Compression mode page (see 8.5.2) for specifying data compression behavior.

The WORM Tamper Read Enable (WTRE) field specifies how the device server responds to detection of compromised integrity of a WORM volume if processing a locate, read, read reverse, space, or verify operation.

The WTRE field (see table 161) shall have no effect on the processing of a locate, read, read reverse, space, or verify operation if the device contains a non-WORM volume.

Table 161 — WTRE field

Code	Description
00b	The device server shall respond in a vendor specific manner.
01b	Detection of compromised integrity on a WORM volume shall not affect processing of a task.
10b	The device server shall return CHECK CONDITION status. The sense key shall be set to MEDIUM ERROR and the additional sense code shall be set to WORM MEDIUM - INTEGRITY CHECK. The position of the medium may have changed,
11b	Reserved. The device server shall return CHECK CONDITION status for a MODE SELECT command with the WTRE field set to 11b. The sense key shall be set to ILLEGAL REQUEST and the additional sense code shall be set to INVALID FIELD IN PARAMETER LIST.

NOTE 59 - An application client should set the WTRE field to 01b only for the recovery of data from a WORM volume where the integrity of the stored data has been compromised.

If the only if reserved (OIR) bit is set to one, the device server shall process a command only if a reservation (see SPC-2) or persistent reservation (see SPC-4) exists that allows access via the I_T nexus from which the command was received. If the OIR bit is set to one and a command is received from an I_T nexus for which no reservation exists, the device server shall not process the command. If the OIR bit is set to one and a command is received from an I_T nexus for a logical unit upon which no reservation or persistent reservation exists, the device server shall terminate the command with CHECK CONDITION status. The sense key shall be set to ILLEGAL REQUEST and the additional sense code shall be set to NOT RESERVED.

Commands that shall not be affected by the OIR bit set to one are defined:

- a) in SPC-2 as Allowed in the presence of reservations;
- b) in table 17 as Allowed in the presence of persistent reservations; or
- c) in the table in SPC-4 that defines the commands that are allowed in the presence of various reservations as Allowed.

Commands that shall be affected by the OIR bit set to one are defined:

- a) in SPC-2 as Conflict in the presence of reservations, except for the RESERVE, RELEASE, PERSISTENT RESERVATION IN and PERSISTENT RESERVATION OUT command;
- b) in table 17 as Conflict in the presence of persistent reservations; or
- c) in the table in SPC-4 that defines the commands that are allowed in the presence of various reservations as Conflict, except for the RESERVE, RELEASE, PERSISTENT RESERVATION IN and PERSISTENT RESERVATION OUT command.

An OIR bit set to zero specifies the device server shall process commands normally.

The REWIND ON RESET field (see table 162), if implemented, shall be persistent across logical unit resets.

Table 162 — REWIND ON RESET field

Code	Description
00b	Vendor specific
01b	The logical unit shall position to the beginning of the default data partition (BOP 0) on logical unit reset.
10b	The logical unit shall maintain its position on logical unit reset.
11b	Reserved

An associated write protection (ASOCWP) bit set to one specifies the logical unit shall inhibit all writing to the medium after performing a synchronize operation (see 4.2.20 and 4.2.20.4). If the ASOCWP bit is set to one, the currently mounted volume is logically write protected until the volume is demounted (see 4.2.20 and 4.2.20.4). If the ASOCWP bit is set to one, all commands requiring eventual writes to the medium shall return CHECK CONDITION status. The sense key shall be set to DATA PROTECT and the additional sense code should be set to ASSOCIATED WRITE PROTECT (see 4.2.20.2). An ASOCWP bit set to zero specifies the currently mounted volume is not write protected by the associated write protection. The ASOCWP bit shall be set to zero by the device server at the time the volume is demounted. This change of state shall not cause a unit attention condition. If the application client sets the ASOCWP bit to one while no volume is mounted, the device server shall terminate the MODE SELECT command with CHECK CONDITION status. The sense key shall be set to NOT READY and the additional sense code shall be set to MEDIUM NOT PRESENT. If the Device Configuration mode page is savable, the ASOCWP bit shall be saved as zero, regardless of the current setting.

A persistent write protection (PERSWP) bit set to one specifies the currently mounted volume is logically write protected (see 4.2.20 and 4.2.20.5). If the PERSWP bit is set to one, all commands requiring eventual writes to the medium shall return CHECK CONDITION status. The sense key shall be set to DATA PROTECT and the additional sense code should be set to PERSISTENT WRITE PROTECT (see 4.2.20.2). A PERSWP bit set to zero specifies the currently mounted volume is not write protected by the persistent write protection. The PERSWP bit shall be set to zero by the device server at the time the volume is demounted or at the time a volume is mounted with persistent write protection disabled. The PERSWP bit shall be set to one by the device server at the time a volume is mounted with persistent write protection enabled. These changes of state shall not cause a unit attention condition. If the application client sets the PERSWP bit to one while no volume is mounted, the device server shall terminate the MODE SELECT command with CHECK CONDITION status. The sense key shall be set to NOT READY and the additional sense code shall be set to MEDIUM NOT PRESENT. If the application client sets the PERSWP bit to one while the logical position is not at BOP 0, then the device server shall return CHECK CONDITION status. The sense key shall be set to ILLEGAL REQUEST and the additional sense code shall be set to POSITION PAST BEGINNING OF MEDIUM. If the Device Configuration mode page is savable, the PERSWP bit shall be saved as zero, regardless of the current setting.

A permanent write protection (PRMWP) bit set to one specifies the currently mounted volume is logically write protected (see 4.2.20 and 4.2.20.6). If the PRMWP bit is set to one, all commands requiring eventual writes to the medium shall return CHECK CONDITION status. The sense key shall be set to DATA PROTECT and the additional sense code should be set to PERMANENT WRITE PROTECT (see 4.2.20.2). A PRMWP bit set to zero specifies the currently mounted volume is not write protected by the permanent write protection. The PRMWP bit shall be set to zero by the device server at the time the volume is demounted or at the time a volume is mounted with permanent write protection disabled. The PRMWP bit shall be set to one by the device server at the time a volume is mounted with permanent write protection enabled. These changes of state shall not cause a unit attention condition. If the application client sets the PRMWP bit to one while no volume is mounted, the device server shall terminate the MODE SELECT command with CHECK CONDITION status. The sense key shall be set to NOT READY and the additional sense code shall be set to MEDIUM NOT PRESENT. If the application client sets the PRMWP bit to one while the logical position is not at BOP 0, then the device server shall return CHECK CONDITION

status. The sense key shall be set to ILLEGAL REQUEST and the additional sense code shall be set to POSITION PAST BEGINNING OF MEDIUM. If the application client attempts to change the PRMWP bit from one to zero, the device server shall terminate the MODE SELECT command with CHECK CONDITION status. The sense key shall be set to DATA PROTECT and the additional sense code shall be set to PERMANENT WRITE PROTECT. If the Device Configuration mode page is savable, the PRMWP bit shall be saved as zero, regardless of the current setting.

8.5.4 Medium Partition mode page

The Medium Partition mode page (see table 163) is used to specify the group of recording volume partitions. Fields in the Medium Partition mode page indicating the current state of the partitions for the recording volume shall be changed by the device server to the current volume state on a not ready to ready transition at the time the volume state changes from demounted to mounted. The physical placement and order of recording volume partitions are not specified by this standard.

NOTE 60 - Since defining partitions may require reformatting the recording volume for some implementations, an implicit write to the medium may occur as a result of a MODE SELECT command that supplies these parameters.

Table 163 — Medium Partition mode page

Bit Byte	7	6	5	4	3	2	1	0						
0	PS	SPF(0)	PAGE CODE (11h)											
1	PAGE LENGTH													
2	MAXIMUM ADDITIONAL PARTITIONS													
3	ADDITIONAL PARTITIONS DEFINED													
4	FDP	SDP	IDP	PSUM		POFM	CLEAR	ADDP						
5	MEDIUM FORMAT RECOGNITION													
6	PARTITIONING TYPE				PARTITION UNITS									
7	Reserved													
Partition size descriptor(s)														
8	(MSB)	PARTITION SIZE [FIRST]						(LSB)						
9														
:														
n-1	(MSB)	PARTITION SIZE [LAST]						(LSB)						
n														

See SPC-4 for a description of the PS bit, SPF bit, PAGE CODE field, and PAGE LENGTH field.

The MAXIMUM ADDITIONAL PARTITIONS field is a logical unit-defined value indicating the maximum number of additional partitions supported by the logical unit. A value of zero returned by the MODE SENSE command specifies no additional partitions are present or allowed.

The ADDITIONAL PARTITIONS DEFINED field specifies the number of additional partitions to be defined for a recording volume if the SDP bit is set to one or the IDP bit is set to one. The maximum value allowed is the value returned in the MAXIMUM ADDITIONAL PARTITIONS field. The ADDITIONAL PARTITIONS DEFINED value returned by the MODE SENSE

command shall report one less than the number of partitions on the media if the logical unit is in the ready state. If the logical unit is in the not ready state, then the ADDITIONAL PARTITIONS DEFINED field is undefined.

A fixed data partitions (FDP) bit set to one specifies the logical unit shall partition the recording volume based on its fixed definition of partitions. Setting the FDP bit to one if POFM is set to zero may only be valid at beginning-of-partition. If the FDP bit is set to one, then the SDP bit and the IDP bit shall be set to zero. The partition size descriptors are ignored by the MODE SELECT command if the FDP bit is set to one. The logical unit may assign any number of partitions from 1 to (MAXIMUM ADDITIONAL PARTITIONS + 1).

NOTE 61 - It is recommended that the partition size descriptors be present in MODE SENSE data regardless of the settings of the FDP, SDP or IDP fields to give an estimate of the size of each partition.

A select data partitions (SDP) bit set to one specifies the logical unit shall partition the recording volume into the number of partitions as specified by the ADDITIONAL PARTITIONS DEFINED field (n) using partition sizes defined by the device. The logical unit shall partition the recording volume into $n+1$ partitions numbered 0 through n . Setting the SDP bit to one if POFM is set to zero may only be valid at beginning-of-partition. If the SDP bit is set to one, then the FDP bit and the IDP bit shall be set to zero. The partition size descriptors are ignored by the MODE SELECT command if the SDP bit is set to one.

An initiator-defined partitions (IDP) bit set to one specifies the logical unit shall partition the recording volume as defined by the ADDITIONAL PARTITIONS DEFINED field and the partition size descriptors. Setting the IDP bit to one if POFM is set to zero may only be valid at beginning-of-partition. If the IDP bit is set to one, then the FDP bit and the SDP bit shall be set to zero. The number of non-zero partition size descriptors received in the Medium Partition mode page shall be one more than the ADDITIONAL PARTITIONS DEFINED value. The size of partition 0 shall be non-zero.

A logical unit is not required to retain the method used to partition the recording volume. The device server shall set only one of the IDP, FDP or SDP fields in the MODE SENSE data. If a recording volume was previously partitioned through a MODE SELECT command with FDP or SDP set to one, a device server may set IDP to one in subsequent MODE SENSE data since the recording volume has been initiator partitioned.

NOTE 62 - Since defining partitions may require reformatting the recording volume for some implementations, an implicit write to the medium may occur as a result of a MODE SELECT command that has any of the fields FDP, SDP, or IDP set to one and has a value of zero in the POFM field.

The partition size unit of measure (PSUM) field (see table 164) defines the units of the partition size descriptors. A logical unit is not required to retain the partition size unit of measure used to partition the recording volume.

Table 164 — PSUM field

Code	Description	Support
00b	bytes (unit of one)	O
01b	kilobytes (10^3 bytes)	O
10b	megabytes (10^6 bytes)	O
11b	$10^{(\text{PARTITION UNITS})}$ bytes	O

The PARTITION UNITS field defines the size of the partition size descriptors if the PSUM field is set to 11b. A value of n in the PARTITION UNITS field shall define the units of the partition size descriptors as 10^n bytes. If the PARTITION UNITS field is supported, all possible values shall be supported. A logical unit is not required to retain the partition units used to partition the volume. If PSUM is not equal to 11b, the PARTITION UNITS field is undefined. Some values of the PARTITION UNITS field may result in no legal non-zero partition size descriptors.

A PARTITION ON FORMAT MEDIUM (POFM) bit set to one specifies the MODE SELECT command shall not cause changes to the partition sizes or user data, either recorded or buffered, and that the actual medium partitioning occurs at the time the device server receives a subsequent FORMAT MEDIUM command (see 7.2). Some field checking may be performed by the MODE SELECT command. Invalid or unsupported settings in this page may cause an error on a subsequent FORMAT MEDIUM command.

At the time the FORMAT MEDIUM command partitions the medium, it shall do so based on the contents of the mode data for the Medium Partition mode page. If the POFM bit is set to one, then the field values established by a MODE SELECT command for the Medium Partition mode page shall not be changed by the device server before:

- a) the volume is unloaded;
- b) a logical unit reset occurs; or
- c) a command is processed that affects the format of the medium (e.g., a SET CAPACITY command or a FORMAT MEDIUM command).

A POFM bit set to zero specifies the MODE SELECT command shall alter the partition information for the recording volume if any of the SDP, FDP, or IDP bits are set to one.

A CLEAR bit set to zero and an ADDP bit set to zero specifies the logical unit may logically erase any or all partitions if one of the IDP, FDP, or SDP fields are set to one by a MODE SELECT command.

A CLEAR bit set to one and an ADDP bit set to zero specifies the logical unit shall logically erase every partition if one of the IDP, FDP, or SDP fields is set to one. No formatting of the recording volume is implied.

An additional partitions (ADDP) bit set to one and a CLEAR bit set to zero specifies the logical unit shall not logically erase any existing partitions, even if the size of the partition is changed. If the MODE SELECT command partition size descriptor and the current partition size differ, the logical unit shall truncate or extend the partition, whichever is appropriate. If the MODE SELECT command partition size is zero and the current partition size is non-zero, the partition shall be logically removed from the recording volume, resulting in the loss of all data in that partition. If the MODE SELECT command partition size is equivalent to the current partition size, no change in the partition size shall result. If the logical unit is unable to perform the operation or an operation causes loss of valid data in any partition that exists both before and after the MODE SELECT or FORMAT MEDIUM command, the device server shall return CHECK CONDITION status. The sense key shall be set to ILLEGAL REQUEST with the addition sense code set to PARAMETER VALUE INVALID. If the ADDP bit is set to one and either ADDP is not supported or the FDP field is set to one the device server shall return CHECK CONDITION status. The sense key shall be set to ILLEGAL REQUEST and the additional sense code shall be set to INVALID FIELD IN PARAMETER LIST. If both the ADDP and SDP fields are set to one, the logical unit shall add or remove partitions such that the resulting partition count on the recording volume is equal to the ADDITIONAL PARTITIONS DEFINED value plus one.

If both the ADDP and CLEAR fields are set to one, the logical unit shall logically erase all partitions that differ in size from the corresponding partition size descriptor in the MODE SELECT data. Partitions with the same size as the MODE SELECT data size shall retain all existing data. If the logical unit is incapable of supporting the changes requested without loss of data, the device server shall return CHECK CONDITION status. The sense key shall be set to ILLEGAL REQUEST and the additional sense code shall be set to PARAMETER VALUE INVALID. If setting both ADDP and CLEAR to one is not supported, the sense key shall be set to ILLEGAL REQUEST and the additional sense code shall be set to INVALID FIELD IN PARAMETER LIST.

A MODE SELECT command partition size descriptor has the equivalent (same) size as the current partition size if:

- a) the mode select PARTITION SIZE, PSUM, and PARTITION UNITS fields are exactly the same as those returned by MODE SENSE command;
- b) the mode select PARTITION SIZE field value is within plus or minus one of the current size expressed in the units of the mode select PSUM or PARTITION UNITS field; or

- c) the mode select PARTITION SIZE is FFFFh and the current size expressed in the units of the mode select PSUM or PARTITION UNITS is FFFFh.

The MEDIUM FORMAT RECOGNITION field (see table 165) specifies the logical unit's capability to automatically identify the volume format and volume partition information while reading a recording volume. The value in this field may be different following a volume change.

Table 165 — MEDIUM FORMAT RECOGNITION field

Code	Description
00h	Logical unit is incapable of format or partition recognition.
01h	Logical unit is capable of format recognition.
02h	Logical unit is capable of partition recognition.
03h	Logical unit is capable of format and partition recognition.
04h to FFh	Reserved

NOTE 63 - If a logical unit specifies it is not capable of volume format recognition, then the application client should supply all necessary parameters for the device to identify the specific format.

Some logical units may support more than one type of partitioning. The PARTITIONING TYPE field (see table 166) specifies the criteria used to create the partitions.

Table 166 — PARTITIONING TYPE field

Code	Description
0h	The type of partitioning is vendor-specific or unknown.
1h	The type of partitioning is optimized for streaming performance.
2h	The type of partitioning reduces the total native capacity of the volume as part of optimizing for streaming performance.
3h to Eh	Logical unit is capable of format and partition recognition.
Fh	For a MODE SENSE command this value indicates there is more than one type of partitioning in use on the volume. For a MODE SELECT command this value is reserved.

PARTITION SIZE fields within the partition size descriptor list define the approximate size of the respective partitions in the units specified in the PSUM and PARTITION UNITS fields. Partitions are numbered by their relative position in the partition size descriptor list, starting at 0. Only partition numbers in the range of 0 to n where n is less than or equal to 63 may have size descriptors in this mode page. Partition n , if present, shall be described by the partition size descriptor at mode page offsets $8+(2*n)$ and $9+(2*n)$. Partition 0 shall be the default partition. Partition size descriptor 0, shall contain the size of the default partition. The size of partition 0 shall be greater than 0. Up to 64 partitions may be defined using this mode page. Partitions not assigned shall have a partition size descriptor of 0. The logical unit may support more partitions than partition size descriptors. A logical unit may support more partition size descriptors than supported by the recording volume. All partition size descriptors representing a partition number greater than the maximum additional partition count shall be 0. The partition size descriptors are undefined if the logical unit is in the not ready state. A MODE SELECT command partition size descriptor of FFFFh requests that the logical unit allocate all remaining partition space to that partition. A MODE SENSE command shall return a partition size descriptor of FFFFh if the partition size, in units of PSUM or PARTITION UNITS, is greater than or equal to FFFFh. If insufficient space exists on the recording volume for the requested partition sizes or if multiple partition size descriptors are set to FFFFh, the device server shall return CHECK CONDITION status. The sense key shall be set to ILLEGAL REQUEST and the additional sense code shall be set to INVALID FIELD IN

PARAMETER LIST. A device server may round, as described by the MODE SELECT command in SPC-4, any partition size to the nearest valid partition size.

NOTE 64 - It is recommended, but not required, that the number of partition size descriptors available through the Medium Partition mode page equal at least the number of maximum additional partitions + 1. This provides a mechanism for the device server to disclose the current partition sizes.

8.5.5 Read-Write Error Recovery mode page

The Read-Write Error Recovery mode page (see table 167) specifies the error recovery and reporting parameters that the device server shall use while transferring data between the device and the medium. These parameters do not affect protocol-level recovery procedures or positioning error recovery procedures.

Table 167 — Read-Write Error Recovery mode page

Bit Byte	7	6	5	4	3	2	1	0
0	PS	SPF(0)	PAGE CODE (01h)					
1	PAGE LENGTH (0Ah)							
2	Reserved	TB	Rsvd	EER	PER	DTE		DCR
3	READ RETRY COUNT							
4								
...	Reserved							
7								
8	WRITE RETRY COUNT							
9								
...	Reserved							
11								

NOTE 65 - The parameters in the Read-Write Error Recovery mode page also apply to verify operations.

See SPC-4 for a description of the PS bit, SPF bit, PAGE CODE field, and PAGE LENGTH field.

A transfer block (TB) bit set to one specifies the device server shall use its best effort to transfer a logical block that is unable to be read successfully within the specified read recovery limits to the application client before CHECK CONDITION status is returned. A TB bit set to zero specifies an unrecoverable logical block shall not be transferred to the application client. Logical blocks that are recoverable within the recovery limits are always transferred, regardless of the value of the TB bit.

An enable early recovery (EER) bit set to one specifies the logical unit shall use the most expedient error recovery algorithm (e.g., attempt error correction prior to retries). An EER bit set to zero specifies the logical unit shall use the most deliberate error recovery algorithm, within the limits established by the other error recovery parameters (e.g., attempt to recover the logical block error-free prior to using error correction).

A post error (PER) bit set to one specifies the device server shall return CHECK CONDITION status to report recovered errors. A PER bit set to zero specifies the device server shall not report errors recovered within the limits established by the error recovery parameters. If this bit is zero, the DTE bit shall also be set to zero.

A disable transfer on error (DTE) bit set to one specifies the device server shall terminate the data transfer after a recovered read or write error occurs. All data from the recovered logical block shall be transferred prior to

terminating the read or write operation. A DTE bit set to zero specifies the device server shall not terminate the transfer for errors recovered within the limits established by the read-write error recovery parameters.

A disable correction (DCR) bit set to one specifies the logical unit shall not use error correction codes during error recovery. A DCR bit set to zero allows the use of error correction codes for error recovery.

The READ RETRY COUNT field specifies the number of times that the logical unit should attempt its recovery algorithm during a read operation before an unrecoverable error is reported. A READ RETRY COUNT of zero specifies the logical unit shall not use its recovery algorithm during read operations.

The WRITE RETRY COUNT field specifies the number of times that the logical unit should attempt its recovery algorithm during a write operation before an unrecoverable error is reported. A WRITE RETRY COUNT of zero specifies the logical unit shall not use its recovery algorithm during write operations.

8.5.6 Informational Exceptions Control mode page

The Informational Exceptions Control mode page (see table 168) defines the methods used by the device server to control the processing and reporting of informational exception conditions including TapeAlert specific informational exception conditions. Informational exception conditions are defined as any event that the device server reports or logs as failure predictions (i.e., with the ADDITIONAL SENSE CODE field set to 5Dh (e.g., FAILURE PREDICTION THRESHOLD EXCEEDED)) or warnings (i.e., with the ADDITIONAL SENSE CODE field set to 0Bh (e.g., WARNING)).

An informational exception condition may occur at any time (e.g., the condition may be asynchronous to any commands issued by an application client).

The mode page policy for this mode page shall be shared or per I_T nexus (see SPC-4).

Table 168 — Informational Exceptions Control mode page

Bit Byte	7	6	5	4	3	2	1	0
0	PS	SPF (0)			PAGE CODE (1Ch)			
1				PAGE LENGTH (0Ah)				
2	PERF	Rsvd	EBF	EWASC	DEXCPT	TEST	EBACKERR	LOGERR
3		Reserved			MRIE			
4	(MSB)							
...				INTERVAL TIMER				
7								(LSB)
8	(MSB)							
...				REPORT COUNT/TEST FLAG NUMBER				
11								(LSB)

See SPC-4 for a description of the PS bit, the SPF bit, the PAGE CODE field, and the PAGE LENGTH field.

The SPF bit, the PAGE CODE field, and the PAGE LENGTH field shall be set to the values shown in table 168.

A performance (PERF) bit set to one specifies that the device server shall not process informational exception conditions that cause delays in processing other operations. A PERF bit set to one may cause the device server to

disable some or all of the processing of informational exception conditions, thereby limiting the reporting of informational exception conditions. A PERF bit set to zero specifies that the device server may process informational exception conditions that cause delays in processing other operations (e.g., processing a command).

If device specific background functions (see SPC-4) are implemented by the logical unit, and the enable background function (EBF) bit is set to one, then the device server shall enable device specific background functions. If the EBF bit is set to zero, then the device server shall disable device specific background functions. Background functions with separate enable control bits are not controlled by the EBF bit.

An enable warning (EWASC) bit set to one specifies that the device server enables reporting of warnings. An EWASC bit set to zero specifies that the device server disables reporting of warnings.

The disable exception control (DEXCPT) bit specifies if the device server enables reporting of failure predictions (see table 169). The interaction between the DEXCPT bit and the TASER bit in the Device Configuration Extension mode page is shown in table 169.

A TEST bit set to one specifies that the device server creates a test device failure prediction (see table 171). If the TEST bit is set to one and the DEXCPT bit is set to one, then the device server shall terminate the MODE SELECT command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN PARAMETER LIST. A TEST bit set to zero specifies that the device server does not create a test device failure prediction.

In response to a MODE SENSE command reporting parameters from the Informational Exceptions Control mode page, the device server shall set the value of the TEST bit to zero.

If an informational exception condition occurs that is not the result of the logical unit processing a background self-test (see SPC-4) or device specific background function (see SPC-4), then the device server:

- a) shall ignore the EWASC bit, the DEXCPT bit, and the TASER bit if the MRIE field is set to 0h;
- b) shall use the value of the EWASC bit, value of the TEST bit, and the definitions for the combination of the values in the DEXCPT bit, and the TASER bit shown in table 169 for processing informational exception conditions if the MRIE field is set from 2h to 6h; and

- c) may use the value of the EWASC bit, value of the TEST bit, and the definitions for the combination of the values in the DEXCPT bit, and the TASER bit shown in table 169 for processing informational exception conditions if the MRIE field is set from Ch to Fh.

Table 169 — Definitions for the combinations of values in DEXCPT bit and TASER bit

DEXCPT	TASER ^a	Description
0	0	The device server shall process informational exception conditions as follows: a) failure predication processing for non-TapeAlert informational exceptions shall be disabled ^b ; and b) failure predication processing for TapeAlert informational exceptions shall be enabled ^b .
0	1	The device server shall process informational exception conditions as follows: a) failure predication processing for non-TapeAlert informational exceptions shall be enabled ^b ; and b) failure predication processing for TapeAlert informational exceptions shall be disabled ^b .
1	0	The device server shall disable failure predication processing for all informational exceptions ^b .
1	1	

a) The TASER bit in the Device Configuration Extension mode page.
b) If applicable, based on the value in the MRIE field (e.g., 2h to 6h), then the values in the LOGERR bit, the INTERVAL TIMER field, and the REPORT COUNT/TEST FLAG NUMBER field determine how the informational exception condition is processed.

If an informational exception condition occurs while the logical unit is processing a background self-test (see SPC-4) or background function (see SPC-4), then the enable background error (EBACKERR) bit determines how the device server processes the informational exception as defined in the following:

- a) if the EBACKERR bit is set to zero, then the device server shall disable reporting of informational exception conditions that occur during the processing of background self-tests and background functions;
- b) if the EBACKERR bit is set to one, then, for informational exception conditions that occur during the processing of background self-tests and background functions, the device server shall:
 - A) enable reporting of the informational exception conditions;
 - B) use the method for reporting the informational exception conditions as determined by contents of the MRIE field; and
 - C) report the informational exception conditions as soon as the method specified in the MRIE field occurs (i.e., the INTERVAL TIMER field and REPORT COUNT/TEST FLAG NUMBER field do not apply for background self-test errors and errors that occur during background functions); and
- c) logging by the device server of informational exception conditions is determined by the value in the LOGERR bit.

The log errors (LOGERR) bit determines how a device server logs informational exception conditions if the MRIE field is set to any value supported by the device server. If the LOGERR bit is set to one, then the device server shall log informational exception conditions in the Informational Exceptions log page. If the LOGERR bit is set to zero, then the device server may log any informational exception conditions in the Informational Exceptions log page (see SPC-4).

The method of reporting informational exceptions (MRIE) field (see table 170) defines the method that shall be used by the device server to report:

- a) non-TapeAlert informational exception conditions if the TASER bit in the Device Configuration Extension mode page is set to one and the specified code value is supported by the device server;
- b) TapeAlert informational exception conditions if the TASER bit in the Device Configuration Extension mode page is set to zero and the specified code value is supported by the device server; and
- c) background self-test errors and device specific background function errors with the ADDITIONAL SENSE CODE field set to 0Bh or 5Dh if the EBACKERR bit is set to one, and the specified code value is supported by the device server.

A device server that supports the Informational Exceptions Control mode page shall support at least one code value in the MRIE field. For MRIE codes 02h to 06h, if the TASER bit in the Device Configuration Extension mode page is set to zero, then an additional sense code of FAILURE PREDICTION THRESHOLD exceeded indicates that a TapeAlert event has occurred on the device and that the TapeAlert log page indicates which TapeAlert event occurred. If an event occurs that causes multiple TapeAlert activation conditions, then the device server shall report a single informational exception condition for that event.

The priority of reporting multiple informational exceptions is vendor specific.

Table 170 — Method of reporting informational exceptions (MRIE) field (part 1 of 2)

Code	Description
0h	No reporting of informational exception condition: The device server shall not report information exception conditions.
1h	Obsolete
2h	Establish unit attention condition: The device server shall report informational exception conditions by establishing a unit attention condition (see SAM-5) with the additional sense code set to indicate the cause of the informational exception condition: <ul style="list-style-type: none"> a) for this I_T nexus, if the mode page policy is per I_T nexus; or b) for the initiator port associated with every I_T nexus, if the mode page policy is shared.
3h	Conditionally generate recovered error: The device server shall report informational exception conditions, if the reporting of recovered errors is allowed ^a , by modifying the completion of the next command: <ul style="list-style-type: none"> a) received through this I_T nexus and processed without encountering any errors, if the mode page policy is per I_T nexus; or b) received through any I_T nexus and processed without encountering any errors, if the mode page policy is shared. The modification shall be to change the completion status of the command to CHECK CONDITION status with the sense key set to RECOVERED ERROR and the additional sense code set to indicate the cause of the informational exception condition.
4h	Unconditionally generate recovered error: The device server shall report informational exception conditions, regardless of whether the reporting of recovered errors is allowed ^a , by modifying the completion of the next command: <ul style="list-style-type: none"> a) received through this I_T nexus and processed without encountering any errors, if the mode page policy is per I_T nexus; or b) received through any I_T nexus and processed without encountering any errors, if the mode page policy is shared. The modification shall be to change the completion status of the command to CHECK CONDITION status with the sense key set to RECOVERED ERROR and the additional sense code set to indicate the cause of the informational exception condition.
a) This is controlled by the PER bit in the Read-Write Error Recovery mode page (see 8.5.5).	

Table 170 — Method of reporting informational exceptions (MRIE) field (part 2 of 2)

Code	Description
5h	Generate no sense: The device server shall report informational exception conditions by modifying the completion of the next command processed without encountering any errors, regardless of the I_T nexus on which the command was received. The modification shall be to terminate the command with CHECK CONDITION status with the sense key set to NO SENSE and the additional sense code set to indicate the cause of the informational exception condition.
6h	Only report informational exception condition on request: The device server shall provide informational exception(s) information with the sense key set to NO SENSE and the additional sense code set to indicate the cause of the informational exception condition for return in response to a REQUEST SENSE command as described in SPC-4. To find out about information exception conditions, the application client polls the device server by issuing a REQUEST SENSE command.
7h to Bh	Reserved
Ch to Fh	Vendor specific
a) This is controlled by the PER bit in the Read-Write Error Recovery mode page (see 8.5.5).	

The INTERVAL TIMER field specifies the period in 100 millisecond increments that the device server shall use for reporting that an informational exception condition has occurred (see table 172). After an informational exception condition has been reported, the interval timer shall be started. An INTERVAL TIMER field set to zero or FFFF_FFFFh specifies that the period for reporting an informational exception condition is vendor specific.

If the TEST bit is set to zero, then the REPORT COUNT/TEST FLAG NUMBER field specifies the maximum number of times the device server may report an informational exception condition to the application client and a REPORT COUNT/TEST FLAG NUMBER field set to zero specifies that there is no limit on the number of times the device server may report an informational exception condition.

If the TEST bit is set to one, then the REPORT COUNT/TEST FLAG NUMBER field is defined in table 171.

Table 171 — REPORT COUNT/TEST FLAG NUMBER field definition if the TEST bit is set to one

REPORT COUNT/TEST FLAG NUMBER	Description
0000 0000h (i.e., 0)	The device server shall not activate or deactivate any TapeAlert flag and the device server shall report an informational exception condition with the additional sense code set to FAILURE PREDICTION THRESHOLD EXCEEDED (FALSE) ^a .
0000 0001h to 0000 0040h (i.e., 1 to 64)	The device server shall activate the TapeAlert flag specified by the REPORT COUNT/TEST FLAG NUMBER field and report the informational exception condition for the TapeAlert flag with an additional sense code of FAILOVER PREDICTION THRESHOLD EXCEEDED (FALSE) ^a .
FFFF FFFFh to FFFF FFC0h (i.e., -1 to -64)	The device server shall deactivate the TapeAlert flag specified by the absolute value of the REPORT COUNT/TEST FLAG NUMBER field. Deactivating the flag in this way is equivalent to performing the specified corrective action for that flag.
0000 7FFFh (i.e., 32767)	The device server: <ul style="list-style-type: none"> a) shall activate all supported TapeAlert flags; b) may activate all TapeAlert flags; and c) shall report the informational exception condition for the TapeAlert flags with an additional sense code of FAILURE PREDICTION THRESHOLD EXCEEDED (FALSE)^a.
FFFF 7FFFh (i.e., -32768)	The device server shall deactivate all TapeAlert flags. Deactivating the flag in this way is equivalent to performing the specified corrective action for that flag.
all others	The device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

- a) If applicable, based on the value in the MRIE field (e.g., 2h to 6h), then the values in the LOGERR bit, the INTERVAL TIMER field, and the REPORT COUNT/TEST FLAG NUMBER field determine how the informational exception condition is processed.

The device server shall use the values in the INTERVAL TIMER field and the REPORT COUNT/TEST FLAG NUMBER field based on the value in the MREI field as shown in table 172.

Table 172 — Use of the INTERVAL TIMER field and the REPORT COUNT/TEST FLAG NUMBER field based on the MREI field

MREI ^a	Description
2h to 6h	If processing of an informational exception condition is enabled (see table 170), then the device server shall: <ol style="list-style-type: none"> 1) report an informational exception condition at the time the condition is first detected; and 2) if the TEST bit is set to zero and the value in the REPORT COUNT/TEST FLAG NUMBER field is not equal to one, then: <ol style="list-style-type: none"> 1) if the INTERVAL TIMER field is not set to zero or FFFF FFFFh, then wait the time specified in the INTERVAL TIMER field, and, if that informational exception condition still exists, report the informational exception again; and 2) while the informational exception condition exists, continue to report the informational exception condition after waiting the time specified in the INTERVAL TIMER field until the condition has been reported the number of times specified by the REPORT COUNT/TEST FLAG NUMBER field.
Ch to Fh	The device server may use or may ignore the values in the INTERVAL TIMER field and the REPORT COUNT/TEST FLAG NUMBER field to report the informational exception condition, based on the device specific implementation.
a For values in the MREI field (see table 170) not shown in this table, the INTERVAL TIMER field and the REPORT COUNT/TEST FLAG NUMBER field shall be ignored.	

Maintaining the interval timer and the report counter across power cycles, hard resets, logical unit resets, and I_T nexus losses by the device server is vendor specific.

8.5.7 Medium Configuration mode page

The Medium Configuration mode page (see table 173) specifies access methods the device server uses if performing an operation that accesses the medium.

Table 173 — Medium Configuration mode page

Bit Byte	7	6	5	4	3	2	1	0
0	PS	SPF(0)			PAGE CODE (1Dh)			
1				PAGE LENGTH (1Eh)				
2			Reserved					WORMM
3			Reserved					
4				WORM VOLUME LABEL RESTRICTIONS				
5					WORM VOLUME FILEMARK RESTRICTIONS			
6								
...				Reserved				
31								

See SPC-4 for a description of the PS bit, SPF bit, PAGE CODE field, and PAGE LENGTH field.

The WORM mounted (WORMM) bit shall be set to one if the mounted volume is a WORM volume. The WORMM bit shall be set to zero if the mounted volume is not a WORM volume or if there is no volume mounted. If a MODE SELECT command is processed that attempts to change the setting of the WORMM bit, then the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

The WORM VOLUME LABEL RESTRICTIONS field (see table 174) specifies the exceptions to the prevention of logical object modification on WORM volumes for overwriting format labels (see 3.1.30) on a WORM volume.

Table 174 — WORM VOLUME LABEL RESTRICTIONS field

Code	Description
00h	The device server shall not allow any logical blocks to be overwritten.
01h	The device server shall allow some types of format labels to be overwritten.
02h	The device server shall allow all format labels to be overwritten.
03h to FFh	Reserved

The WORM VOLUME FILEMARKS RESTRICTIONS field (see table 175) specifies the exceptions to the prevention of logical object modification on WORM volumes for overwriting one or more contiguous filemarks immediately preceding EOD if operating on a WORM volume.

Table 175 — WORM VOLUME FILEMARKS RESTRICTIONS field

Code	Description
00h to 01h	Reserved
02h	The device server shall allow one or more contiguous filemarks immediately preceding EOD to be overwritten except the filemark closest to BOP unless there is no logical block between that filemark and BOP, in which case, the filemark closest to BOP is also allowed to be overwritten.
03h	The device server shall allow one or more contiguous filemarks immediately preceding EOD to be overwritten.
04h to FFh	Reserved

8.5.8 Device Configuration Extension mode page

The Device Configuration Extension mode page (see table 176), a subpage of the Device Configuration mode page (see 8.5.3), provides control of the SCSI features specific to sequential-access devices. If a device server

supports the Device Configuration Extension mode page, the device server shall provide access to the mode page using the shared mode page policy (see SPC-4).

Table 176 — Device Configuration Extension mode page

Bit Byte	7	6	5	4	3	2	1	0
0	PS	SPF(1b)			PAGE CODE (10h)			
1					SUBPAGE CODE (01h)			
2	(MSB)				PAGE LENGTH (1Ch)			
3								(LSB)
4		Reserved		TARPF	TASER	TARPC	TAPLSD	
5		WRITE MODE			SHORT ERASE MODE			
6	(MSB)			PEWS				
7								(LSB)
8			Reserved				VCELBRE	
9								
...			Reserved					
31								

See SPC-4 for a description of the PS bit, SPF bit, PAGE CODE field, and PAGE LENGTH field.

A TapeAlert prevent LOG SENSE deactivation (TAPLSD) bit set to one specifies that the device server shall not alter the value of implemented TapeAlert FLAG parameters (see 8.3.3) due to processing of a LOG SENSE command with the PAGE CODE field set to 2Eh. A TAPLSD bit set to zero specifies that, as part of the processing of a LOG SENSE command with the PAGE CODE field set to 2Eh, the device server shall deactivate every supported TapeAlert flag.

A TapeAlert respect page control (TARPC) bit set to one specifies that the device server shall select the type of parameter values for the TapeAlert log page (see 8.3.3) using the value of the PC field in a LOG SELECT or LOG SENSE command (see SPC-4). A TARPC bit set to zero specifies that the device server shall select cumulative parameter values for the TapeAlert log page regardless of the value of the PC field in a LOG SELECT or LOG SENSE command.

NOTE 66 - An application client using the TapeAlert threshold usage model (see 4.2.24.2.4) should set the TARPC bit to one.

A TapeAlert Select exception reporting (TASER) bit set to one specifies that:

- a) activation of a TapeAlert flag shall not result in an informational exception condition (see 8.5.6); and
- b) activation or deactivation of a TapeAlert flag may result in the generation of a unit attention condition with the additional sense code set to THRESHOLD CONDITION MET depending on the value of the ETC bit in each parameter in the TapeAlert log page.

A TASER bit set to zero specifies that:

- a) activation of a TapeAlert flag shall result in an informational exception condition (see 8.5.6);
- b) activation or deactivation of a TapeAlert flag shall not result in the generation of a unit attention condition with the additional sense code set to THRESHOLD CONDITION MET; and

- c) the device server shall set the ETC bit in each parameter of the TapeAlert log page to zero.

The device server may provide independent sets of TapeAlert log parameters for each I_T nexus (see SPC-4). If the device server does not support independent sets of TapeAlert log parameters and the processing of a MODE SELECT command with the TASER bit set to zero results in a change in the value of the ETC bit in a TapeAlert log parameter, then the device server shall establish a unit attention condition (see SAM-5) for the initiator port associated with every I_T nexus, except the I_T nexus on which the MODE SELECT command was received, with the additional sense code set to LOG PARAMETERS CHANGED.

A TapeAlert respect parameter fields (TARPF) bit set to one specifies that the device server shall select the parameters of the TapeAlert log page (see 8.3.3) to report in response to a LOG SENSE command using the values of the PPC bit and the PARAMETER POINTER field (see SPC-4). A TARPF bit set to zero specifies that the device server shall report all TapeAlert log page parameters regardless of the values of the PPC bit and the PARAMETER POINTER field.

The SHORT ERASE MODE field (see table 177) specifies the action to be taken by the device server if an ERASE (16) or ERASE (6) command with the LONG bit set to zero is processed. A device server shall support at least one of the specified values.

Table 177 — SHORT ERASE MODE field

Code	Description
0h	The erase operation shall be performed as specified in SSC-2.
1h	The erase operation shall have no effect on the medium.
2h	The device server shall record an EOD indication at the specified location on the medium.
3h to Fh	Reserved

The WRITE MODE field (see table 178) specifies the write mode (see 4.2.16) in which to place the device server. A device server shall support at least one of the specified values. If a volume is mounted, the logical position is not at BOP 0, and an attempt is made to change the WRITE MODE to 01h (i.e., append-only) from another mode, then CHECK CONDITION status shall be returned with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN PARAMETER LIST. If a volume is mounted and an attempt is made to change the WRITE MODE to 00h (i.e., overwrite-allowed) from another mode, then CHECK CONDITION status shall be returned with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

Table 178 — WRITE MODE field

Code	Description	Reference
0h	Overwrite-allowed mode	4.2.16.2
1h	Append-only mode	4.2.16.3
2h to Dh	Reserved	
Eh to Fh	Vendor specific	

The programmable early warning size (PEWS) field specifies the number of megabytes native capacity to use in establishing a PEWZ (see 3.1.64). A device server should allow the PEWS field to be set to any value. If the value of the PEWS field is not on a supported boundary, then the value in the PEWS field shall be rounded up to the next supported boundary and the device server shall return CHECK CONDITION status, with the sense key set to RECOVERED ERROR, and the additional sense code set to ROUNDED PARAMETER. The value of the PEWS

field should not be rounded down (e.g., a PEWS field value larger than the volume capacity is not rounded down). See 4.2.6 for a description of programmable early warning.

If the volume containing encrypted logical blocks requires encryption (VCELBRE) bit is set to one and the VCELB bit in the Data Encryption Status page is set to one, then the device server shall require that any logical blocks written to the medium are encrypted (see 4.2.29). If the VCELBRE bit is set to zero, then the device server does not use the VCELB bit in the Data Encryption Status page to determine if encryption is required for writing logical blocks.

8.5.9 Control Data Protection mode page

The Control Data Protection mode page (see table 179), a subpage of the Control mode page (see SPC-4), provides controls that allow selective use of logical block protection. The mode page policy of this page shall be per I_T nexus.

Table 179 — Control Data Protection mode page

Bit Byte	7	6	5	4	3	2	1	0
0	PS	SPF(1b)			PAGE CODE (0Ah)			
1					SUBPAGE CODE (F0h)			
2	(MSB)				PAGE LENGTH (n-3)			
3								(LSB)
4					LOGICAL BLOCK PROTECTION METHOD			
5	Reserved				LOGICAL BLOCK PROTECTION INFORMATION LENGTH			
6	LBP_W	LBP_R	RBDP		Reserved			
7								
...					Reserved			
n								

See SPC-4 for a description of the PS bit, SPF bit, PAGE CODE field, and PAGE LENGTH field.

The LOGICAL BLOCK PROTECTION METHOD field contains a code specifying a logical block protection method (see table 180).

Table 180 — Logical block protection methods

Code	Description	Length in bytes
00h	Do not use logical block protection.	00h
01h	Use the Reed-Solomon CRC as specified in ECMA-319 as the logical block protection method.	04h
02h	Use the CRC32C (Castagnoli) polynomial as the logical block protection method. This is the same polynomial and usage defined in RFC 7143 and RFC 3385 except logical blocks are not padded to a 4-byte boundary. The CRC immediately follows the logical block data with no padding between the logical block data and the CRC.	04h
03h to EFh	Reserved	n/a
F0h to FFh	Vendor specific	VS

The LOGICAL BLOCK PROTECTION INFORMATION LENGTH field specifies the length in bytes of the logical block protection information (see table 180).

The logical blocks protected during write (LBP_W) bit set to one indicates that the protection information is included with logical blocks transferred during commands specified in 4.2.28.5. The LBP_W bit set to zero indicates that the protection information is not included with logical blocks transferred while writing. If the LOGICAL BLOCK PROTECTION METHOD field is set to zero, the LBP_W bit shall be set to zero.

The logical block protected during read (LBP_R) bit set to one indicates that the protection information is included with logical blocks transferred during the commands specified in 4.2.28.5.3. The LBP_R bit set to zero indicates that the protection information is not included with logical blocks transferred while reading. If the LOGICAL BLOCK PROTECTION METHOD field is set to zero, the LBP_R bit shall be set to zero.

The recover buffered data protected (RBDP) bit set to one indicates that the protection information is transferred with the logical blocks transferred by the RECOVER BUFFERED DATA command. The RBDP bit set to zero indicates that the protection information is not transferred with the logical blocks transferred by the RECOVER BUFFERED DATA command. If the LOGICAL BLOCK PROTECTION METHOD field is set to zero, the RBDP bit shall be set to zero.

8.6 Vital product data (VPD) parameters

8.6.1 VPD parameters overview and page codes

This subclause defines the VPD pages used with sequential-access device types. See SPC-4 for VPD pages used with all device types. The VPD page codes specific to sequential-access devices are specified in table 181.

Table 181 — Sequential-access device VPD page codes

Page code	VPD page name	Reference	Support requirements
B0h	Sequential-access Device Capabilities VPD page	8.6.2	Optional
B1h	Manufacturer-assigned Serial Number VPD page	8.6.3	Optional
B2h	TapeAlert Supported Flags VPD page	8.6.4	Optional
B3h	Automation Device Serial Number VPD page	8.6.5	Optional
B4h	Data Transfer Device Element Address VPD page	8.6.6	Optional
B5h	Logical Block Protection VPD page	8.6.7	Optional
B6h to BFh	Reserved for this device type		

8.6.2 Sequential-access Device Capabilities VPD page

Table 182 specifies the Sequential-access Device Capabilities VPD page. This page provides the application client with the means to determine if the features specified in this page are supported by the device server.

Table 182 — Sequential-access Device Capabilities VPD page

Bit Byte	7	6	5	4	3	2	1	0
0	PERIPHERAL QUALIFIER				PERIPHERAL DEVICE TYPE			
1				PAGE CODE (B0h)				
2	(MSB)				PAGE LENGTH (n-3)			
3								(LSB)
4			Reserved					WORM
5								
...			Reserved					
n								

The PERIPHERAL QUALIFIER field and the PERIPHERAL DEVICE TYPE field are defined in SPC-4.

The PAGE LENGTH field specifies the length of the following VPD page data. If the allocation length value in the INQUIRY command descriptor block is too small to transfer all of the VPD page data, the page length shall not be adjusted to reflect the truncation.

If the write once read many (WORM) bit is set to one, then the device server supports WORM volumes (see 4.2.2.4). If the WORM bit is set to zero, then the device server does not support WORM volumes.

8.6.3 Manufacturer-assigned Serial Number VPD page

Table 183 specifies the Manufacturer-assigned Serial Number VPD page.

Table 183 — Manufacturer-assigned Serial Number VPD page

Bit Byte	7	6	5	4	3	2	1	0
0	PERIPHERAL QUALIFIER				PERIPHERAL DEVICE TYPE			
1				PAGE CODE (B1h)				
2	(MSB)				PAGE LENGTH (n-3)			
3								(LSB)
4	(MSB)				MANUFACTURER-ASSIGNED SERIAL NUMBER			
...								
n								(LSB)

See SPC-4 for a description of the PERIPHERAL QUALIFIER field, PERIPHERAL DEVICE TYPE field, and PAGE LENGTH field. The PAGE CODE field shall be set to the value specified in table 183.

The MANUFACTURER-ASSIGNED SERIAL NUMBER field contains right-aligned ASCII data (see SPC-4) that is the manufacturer-assigned serial number. If the manufacturer-assigned serial number is not available, the device server shall return ASCII spaces (20h) in this field. If the manufacturer-assigned serial number differs from the

value in the PRODUCT SERIAL NUMBER field (see SPC-4), the value in the PRODUCT SERIAL NUMBER field shall be used in building the T10 vendor ID descriptor (see SPC-4).

8.6.4 TapeAlert Supported Flags VPD page

Table 184 specifies the TapeAlert Supported Flags VPD page. The TapeAlert Supported Flags VPD page provides the application client with the means to determine the TapeAlert flags supported by the device server.

Table 184 — TapeAlert Supported Flags VPD page

Bit Byte	7	6	5	4	3	2	1	0
0	PERIPHERAL QUALIFIER							PERIPHERAL DEVICE TYPE
1					PAGE CODE (B2h)			
2	(MSB)				PAGE LENGTH (08h)			
3								(LSB)
4	FLAG01h	FLAG02h	FLAG03h	FLAG04h	FLAG05h	FLAG06h	FLAG07h	FLAG08h
5	FLAG09h	FLAG0Ah	FLAG0Bh	FLAG0Ch	FLAG0Dh	FLAG0Eh	FLAG0Fh	FLAG10h
6	FLAG11h	FLAG12h	FLAG13h	FLAG14h	FLAG15h	FLAG16h	FLAG17h	FLAG18h
7	FLAG19h	FLAG1Ah	FLAG1Bh	FLAG1Ch	FLAG1Dh	FLAG1Eh	FLAG1Fh	FLAG20h
8	FLAG21h	FLAG22h	FLAG23h	FLAG24h	FLAG25h	FLAG26h	FLAG27h	FLAG28h
9	FLAG29h	FLAG2Ah	FLAG2Bh	FLAG2Ch	FLAG2Dh	FLAG2Eh	FLAG2Fh	FLAG30h
10	FLAG31h	FLAG32h	FLAG33h	FLAG34h	FLAG35h	FLAG36h	FLAG37h	FLAG38h
11	FLAG39h	FLAG3Ah	FLAG3Bh	FLAG3Ch	FLAG3Dh	FLAG3Eh	FLAG3Fh	FLAG40h

The PERIPHERAL QUALIFIER field and the PERIPHERAL DEVICE TYPE field are defined in SPC-4.

The PAGE LENGTH field specifies the length of the following VPD page data. If the allocation length value in the INQUIRY command descriptor block is too small to transfer all of the VPD page data, the page length shall not be adjusted to reflect the truncation.

Each FLAGXX bit indicates whether the device server supports the corresponding TapeAlert flag or not. A supported TapeAlert flag has the corresponding FLAGXX bit set to one. A TapeAlert flag that the device server does not support has the corresponding FLAGXX bit set to zero.

8.6.5 Automation Device Serial Number VPD page

Table 185 specifies the Automation Device Serial Number VPD page.

Table 185 — Automation Device Serial Number VPD page

Bit Byte	7	6	5	4	3	2	1	0
0	PERIPHERAL QUALIFIER				PERIPHERAL DEVICE TYPE			
1				PAGE CODE (B3h)				
2	(MSB)				PAGE LENGTH (n-3)			
3								(LSB)
4	(MSB)				AUTOMATION DEVICE SERIAL NUMBER			
...								
n								(LSB)

See SPC-4 for a description of the PERIPHERAL QUALIFIER field, PERIPHERAL DEVICE TYPE field, and PAGE LENGTH field. The PAGE CODE field shall be set to the value specified in table 185.

The AUTOMATION DEVICE SERIAL NUMBER field contains the automation device serial number contained in the device entity. If configured, the automation device serial number shall be the product serial number (see SPC-4) of the media changer containing the device entity under control of the device server (see 4.2.3). If no automation device serial number has been configured, then the device server shall return ASCII spaces (20h) in this field.

8.6.6 Data Transfer Device Element Address VPD page

Table 186 specifies the Data Transfer Device Element Address VPD page.

Table 186 — Data Transfer Device Element Address VPD page

Bit Byte	7	6	5	4	3	2	1	0
0	PERIPHERAL QUALIFIER				PERIPHERAL DEVICE TYPE			
1				PAGE CODE (B4h)				
2	(MSB)				PAGE LENGTH (n-3)			
3								(LSB)
4	(MSB)				DATA TRANSFER DEVICE ELEMENT			
...								
n								(LSB)

See SPC-4 for a description of the PERIPHERAL QUALIFIER field, PERIPHERAL DEVICE TYPE field, and PAGE LENGTH field. The PAGE CODE field shall be set to the value specified in table 186.

The DATA TRANSFER DEVICE ELEMENT field contains the data transfer device element address from the device entity. If no data transfer device element address has been configured, then the device server shall set the DATA TRANSFER DEVICE ELEMENT field to FFFFFFFFh.

NOTE 67 - The data transfer device element address is the element address within the media changer (see SMC-3) at which this data transfer device is found.

8.6.7 Logical Block Protection VPD page

Table 187 specifies the Logical Block Protection VPD page.

Table 187 — Logical Block Protection VPD page

Bit Byte	7	6	5	4	3	2	1	0
0	PERIPHERAL QUALIFIER							PERIPHERAL DEVICE TYPE
1								PAGE CODE (B5h)
2	(MSB)							PAGE LENGTH
3								(LSB)
4								
...								Reserved
7								
Logical block protection method descriptor list								
8								
...								Logical block protection method descriptor [first] (see table 188)
								⋮
...								Logical block protection method descriptor [last] (see table 188)
n								

See SPC-4 for a description of the PERIPHERAL QUALIFIER field, PERIPHERAL DEVICE TYPE field, and PAGE LENGTH field. The PAGE CODE field shall be set to the value specified in table 187.

The logical block protection method descriptor list contains a list of the protection information methods supported by the device server reported in ascending order by logical block protection method code (see 8.5.9). If logical block protection method zero (i.e., no protection information) is supported, then the device server shall report a logical block protection method descriptor for logical block protection method zero. The logical block protection method descriptor format is shown in table 188.

Table 188 — Logical block protection method descriptor format

Bit Byte	7	6	5	4	3	2	1	0
0								LOGICAL BLOCK PROTECTION METHOD DESCRIPTOR LENGTH (n)
1								LOGICAL BLOCK PROTECTION METHOD
2	Reserved							LOGICAL BLOCK PROTECTION INFORMATION LENGTH
3	LBP_W_C	LBP_R_C	RBDP_C					Reserved
4								
...								Reserved
n								

The LOGICAL BLOCK PROTECTION METHOD DESCRIPTOR LENGTH field indicates the number of available bytes to follow. The value of the LOGICAL BLOCK PROTECTION METHOD DESCRIPTOR LENGTH field shall be at least 7 and shall be one less than a multiple of four (i.e., the logical block protection method descriptor including the LOGICAL BLOCK PROTECTION METHOD DESCRIPTOR LENGTH field shall be a length that is a multiple of four).

The LOGICAL BLOCK PROTECTION METHOD field contains a logical block protection method code (see table 180) specifying a supported logical block protection method.

The LOGICAL BLOCK PROTECTION INFORMATION LENGTH field specifies the length in bytes of the logical block protection information (see table 180) for the specified logical block protection method.

Editors Note 3 - CCB: The text for the LOGICAL BLOCK PROTECTION INFORMATION LENGTH field was missing from the original proposal and was added by the editor - working group to review. The LOGICAL BLOCK PROTECTION METHOD field text was also revised to better align with the text in 8.4.9 by table 180.

The LBP_W_C bit shall be set to one if the device server supports the LBP_W bit in the Control Data Protection mode page (see 8.5.9) set to one for the specified logical block protection method. The LBP_W_C bit shall be set to zero if the device server does not support the LBP_W bit in the Control Data Protection mode page set to one for the specified logical block protection method.

The LBP_R_C bit shall be set to one if the device server supports the LBP_R bit in the Control Data Protection mode page set to one for the specified logical block protection method. The LBP_R_C bit shall be set to zero if the device server does not support the LBP_R bit in the Control Data Protection mode page set to one for the specified logical block protection method.

The RBDP_C bit shall be set to one if the device server supports the RBDP bit in the Control Data Protection mode page set to one for the specified logical block protection method. The RBDP_C bit shall be set to zero if the device server does not support the RBDP bit in the Control Data Protection mode page set to one for the specified logical block protection method.

8.7 Security protocol parameters

8.7.1 Security protocol overview

This subclause describes the protocols, pages, and descriptors used by sequential-access devices with the SECURITY PROTOCOL IN and SECURITY PROTOCOL OUT commands (see SPC-4).

8.7.2 SECURITY PROTOCOL IN command specifying Tape Data Encryption security protocol

8.7.2.1 SECURITY PROTOCOL IN command specifying Tape Data Encryption security protocol overview

The SECURITY PROTOCOL IN command (see SPC-4) specifying Tape Data Encryption security protocol (i.e., 20h) requests the device server to return information about the logical block security methods in the device entity and on the medium. The command supports a series of pages that are requested individually. An application client requests a page by using a SECURITY PROTOCOL IN command with the SECURITY PROTOCOL field set to 20h (i.e., Tape Data Encryption) (see SPC-4) and the SECURITY PROTOCOL SPECIFIC field set to the page code requested.

A device server that supports the Tape Data Encryption protocol in the SECURITY PROTOCOL OUT command shall also support a SECURITY PROTOCOL IN command specifying the Tape Data Encryption protocol.

The SECURITY PROTOCOL SPECIFIC field (see table 189) specifies the type of report that the application client is requesting.

Table 189 — SECURITY PROTOCOL SPECIFIC field

Code	Description	Support	Reference
0000h	Tape Data Encryption In Support page	M	8.7.2.2
0001h	Tape Data Encryption Out Support page	M	8.7.2.3
0002h to 000Fh	Reserved		
0010h	Data Encryption Capabilities page	E	8.7.2.4
0011h	Supported Key Formats page	E	8.7.2.5
0012h	Data Encryption Management Capabilities page	E	8.7.2.6
0013h to 001Fh	Reserved		
0020h	Data Encryption Status page	E	8.7.2.7
0021h	Next Block Encryption Status page	E	8.7.2.8
0022h	Get Encryption Management Attributes page	O	8.7.2.9
0023h to 002Fh	Reserved		
0030h	Random Number page	O	8.7.2.10
0031h	Device Server Key Wrapping Public Key page	O	8.7.2.11
0032h to FEFFh	Reserved		
FF00h to FFFFh	Vendor specific		
Support key: M - mandatory for a device server that supports the Tape Data Encryption protocol E - mandatory for a device server that supports the Set Data Encryption page (see 8.7.3.2) O - optional for a device server that supports the Tape Data Encryption protocol			

The ALLOCATION LENGTH field specifies the maximum number of bytes that the device server may return (see SPC-4).

8.7.2.2 Tape Data Encryption In Support page

Table 190 specifies the format of the Tape Data Encryption In Support page.

Table 190 — Tape Data Encryption In Support page

Bit Byte	7	6	5	4	3	2	1	0
0	(MSB)				PAGE CODE (0000h)			
1								(LSB)
2	(MSB)				PAGE LENGTH (n-3)			
3								(LSB)
Tape Data Encryption In Support page code list								
4	(MSB)				Tape Data Encryption In Support page code [first]			
5								(LSB)
:								
n-1	(MSB)				Tape Data Encryption In Support page code [last]			
n								(LSB)

The Tape Data Encryption In Support page code list shall contain a list of all of the pages that the device server supports for the SECURITY PROTOCOL IN command specifying the Tape Data Encryption security protocol in ascending order beginning with page code 0000h.

8.7.2.3 Tape Data Encryption Out Support page

Table 191 specifies the format of the Tape Data Encryption Out Support page.

Table 191 — Tape Data Encryption Out Support page

Bit Byte	7	6	5	4	3	2	1	0
0	(MSB)				PAGE CODE (0001h)			
1								(LSB)
2	(MSB)				PAGE LENGTH (n-3)			
3								(LSB)
Tape Data Encryption Out Support page code list								
4	(MSB)				Tape Data Encryption Out Support page code [first]			
5								(LSB)
:								
n-1	(MSB)				Tape Data Encryption Out Support page code [last]			
n								(LSB)

The Tape Data Encryption Out Support page code list shall contain a list of all of the pages that the device server supports for the SECURITY PROTOCOL OUT command specifying the Tape Data Encryption security protocol in ascending order.

8.7.2.4 Data Encryption Capabilities page

Table 192 specifies the format of the Data Encryption Capabilities page. The Data Encryption Capabilities page reports information on the set of logical block encryption algorithms supported by this device server. If external data encryption control is supported, then the set of logical block encryption algorithms reported by the device server may not include all of the algorithms in the set of logical block encryption algorithms supported by the device entity.

Table 192 — Data Encryption Capabilities page

Bit Byte	7	6	5	4	3	2	1	0
0	(MSB)							
1					PAGE CODE (0010h)			
2	(MSB)							(LSB)
3					PAGE LENGTH (n-3)			(LSB)
4		Reserved				EXTDECC		CFG_P
5								
...					Reserved			
19								
Logical block encryption algorithm descriptor list								
20								
...					Logical block encryption algorithm descriptor [first] (see table 195)			
43								
:								
n-23								
...					Logical block encryption algorithm descriptor [last] (see table 195)			
n								

See SPC-4 for a description of the PAGE CODE field and the PAGE LENGTH field. The PAGE CODE field shall be set to the value specified in table 192.

The external data encryption control capable (EXTDECC) field (see table 193) specifies the external data encryption control capability of the device entity.

Table 193 — EXTDECC field

Code	Description
00b	The external data encryption control capability is not reported.
01b	The device entity is not external data encryption control capable.
10b	The device entity is external data encryption control capable.
11b	Reserved

The configuration prevented (CFG_P) field (see table 194) specifies the logical block encryption parameters configuration capabilities for the algorithms reported in the logical block encryption algorithm descriptors.

Table 194 — CFG_P field

Code	Description
00b	The logical block encryption configuration capabilities are not reported.
01b	The device entity is configured to allow this device server to establish or change logical block encryption parameters.
10b	The device entity is configured to not allow this device server to establish or change logical block encryption parameters.
11b	Reserved

Each logical block encryption algorithm descriptor (see table 195) contains information about a logical block encryption algorithm supported by the device server. If more than one descriptor is included, they shall be sorted in ascending order of the value in the ALGORITHM INDEX field.

Table 195 — Logical block encryption algorithm descriptor

Bit Byte	7	6	5	4	3	2	1	0
0								ALGORITHM INDEX
1								Reserved
2	(MSB)							
3								DESCRIPTOR LENGTH (20) (LSB)
4	AVFMV	SDK_C	MAC_C	DELB_C	DECRYPT_C			ENCRYPT_C
5	AVFCLP		NONCE_C		KADF_C	VCELB_C	UKADF	AKADF
6	(MSB)							MAXIMUM UNAUTHENTICATED KEY-ASSOCIATED DATA BYTES (LSB)
7								
8	(MSB)							MAXIMUM AUTHENTICATED KEY-ASSOCIATED DATA BYTES (LSB)
9								
10	(MSB)							LOGICAL BLOCK ENCRYPTION KEY SIZE (LSB)
11								
12	DKAD_C		EEMC_C		RDMC_C			EAREM
13			Reserved					MAXIMUM NUMBER OF EEDKS
14	(MSB)				MSDK_COUNT			
15								(LSB)
16	(MSB)							
17								MAXIMUM EEDK SIZE (LSB)
18								
19					Reserved			
20	(MSB)							
...					SECURITY ALGORITHM CODE			
24								(LSB)

The ALGORITHM INDEX field is a device server assigned value associated with the algorithm that is being described. The value in the ALGORITHM INDEX field is used by the SECURITY PROTOCOL OUT command Set Data Encryption page to select this algorithm.

The device server shall assign a different algorithm index for:

- a) each unique set of values in the logical block encryption algorithm descriptor;
- b) each unique encoding scheme (e.g., each unique combination of M-KAD format and format of the encrypted logical block(s)); and
- c) vendor-specific reasons.

The algorithm valid for mounted volume (AVFMV) bit shall be set to one if there is a volume currently mounted and the encryption algorithm being described is valid for that volume. The AVFMV bit shall be set to zero if there is no volume mounted or the algorithm is not valid for the currently mounted volume.

The supplemental decryption key capable (SDK_C) bit shall be set to one if the device entity is capable of supporting one or more supplemental decryption keys. The supplemental decryption keys shall be used for decryption only. The SDK_C bit shall be set to zero if the device entity is not capable of supporting supplemental decryption keys.

The distinguish encrypted logical block capable (DELB_C) bit shall be set to one if the device entity is capable of distinguishing encrypted logical blocks from unencrypted logical blocks while reading from the medium. The DELB_C bit shall be set to zero if the device entity is not capable of distinguishing encrypted logical blocks from unencrypted logical blocks while reading from the medium. If the ability to distinguish encrypted logical blocks from unencrypted logical blocks is format-specific and a volume is mounted, the DELB_C bit shall be set based on the current format of the volume. If no volume is mounted, the DELB_C bit shall be set to one if the device entity is capable of distinguishing encrypted logical blocks from unencrypted logical blocks in one or more of the formats that the device server supports.

The message authentication code capable (MAC_C) bit shall be set to one if the algorithm includes a message authentication code added to encrypted logical blocks. The MAC_C bit shall be set to zero if the algorithm does not include a message authentication code added to encrypted logical blocks. If the inclusion of a message authentication code is format-specific and a volume is mounted, the MAC_C bit shall be set based on the current format of the volume. If no volume is mounted, the MAC_C bit shall be set to one if the device server adds a message authentication code to data encrypted with this algorithm in one or more of the formats that the device server supports.

The DECRYPT_C field (see table 196) specifies the decryption capabilities of the device entity.

Table 196 — DECRYPT_C field

Code	Name	Description
00b	No capability	The device entity has no logical block decryption capability using this algorithm. This value shall be returned if the specified algorithm is disabled.
01b	Software capable	The device entity has the ability to decrypt logical blocks using this algorithm in software.
10b	Hardware capable	The device entity has the ability to decrypt logical blocks using this algorithm in hardware.
11b	Capable with external control	The device entity has the capability to decrypt logical blocks using this algorithm, but control of the logical block encryption parameters by this device server is prevented.

The ENCRYPT_C field (see table 197) specifies the encryption capabilities of the device entity.

Table 197 — ENCRYPT_C field

Code	Name	Description
00b	No capability	The device entity has no logical block encryption capability using this algorithm. This value shall be returned if the specified algorithm is disabled.
01b	Software capable	The device entity has the ability to encrypt logical blocks using this algorithm in software.
10b	Hardware capable	The device entity has the ability to encrypt logical blocks using this algorithm in hardware.
11b	Capable with external control	The device entity has the capability to encrypt logical blocks using this algorithm, but control of the logical block encryption parameters by this device server is prevented.

The algorithm valid for current logical position (AVFCLP) field (see table 198) specifies if the encryption algorithm being specified is valid for writing to the mounted volume at the current logical position.

Table 198 — AVFCLP field

Code	Description
00b	Current logical position is not applicable to the encryption algorithm validity or no volume is loaded.
01b	The encryption algorithm being specified is not valid for writing to the mounted volume at the current logical position.
10b	The encryption algorithm being specified is valid for writing to the mounted volume at the current logical position.
11b	Reserved

Table 199 specifies the values for the NONCE_C field.

Table 199 — NONCE_C field

Code	Description
00b	This algorithm does not require a nonce value.
01b	The device entity generates the nonce value.
10b	The device entity requires all or part of the nonce value to be provided by the application client.
11b	The device entity supports all or part of the nonce value provided by the application client. If the Set Data Encryption page that enables encryption does not include a nonce value descriptor, the device entity generates the nonce value.

A KAD format capable (KADF_C) bit set to one indicates that the device server supports the:

- a) ENCRYPTION PARAMETERS KAD FORMAT field in the Data Encryption Status page (see 8.7.2.7);
- b) NEXT BLOCK KAD FORMAT field in the Next Block Encryption Status page (see 8.7.2.8); and
- c) KAD FORMAT field in the Set Data Encryption page (see 8.7.3.2).

A KADF_C bit set to zero indicates that the device server does not support the:

- a) ENCRYPTION PARAMETERS KAD FORMAT field in the Data Encryption Status page (see 8.7.2.7);
- b) NEXT BLOCK KAD FORMAT field in the Next Block Encryption Status page (see 8.7.2.8); or
- c) KAD FORMAT field in the Set Data Encryption page (see 8.7.3.2).

If the volume contains encrypted logical blocks capable (VCELB_C) bit is set to one, then the device server is capable of determining that a volume contains logical blocks encrypted using this algorithm at the time the volume is mounted. If the VCELB_C is set to zero, then the device server is not capable of determining that a volume contains logical blocks encrypted using this algorithm at the time the volume is mounted. If the capability of determining that a volume contains logical blocks encrypted using this algorithm is format-specific and a volume is mounted, then the VCELB_C bit is set based on the current format of the volume. If no volume is mounted, the VCELB_C bit is set to one if for at least one algorithm that the device server supports the device server is capable of determining that a volume contains logical blocks encrypted using that algorithm.

The U-KAD fixed (UKADF) bit shall be set to one if the device server requires the length of U-KAD in the parameter data for a SECURITY PROTOCOL OUT command to equal the value in the MAXIMUM UNAUTHENTICATED KEY-ASSOCIATED BYTES field. If the UKADF bit is set to one, then the MAXIMUM UNAUTHENTICATED KEY-ASSOCIATED BYTES field shall contain a non-zero value. If the UKADF bit is set to zero and the value in the MAXIMUM UNAUTHENTICATED KEY-ASSOCIATED BYTES field is non-zero, then the length of the U-KAD, if present in the parameter data for a SECURITY PROTOCOL OUT command, shall be a value between one and the value in the MAXIMUM UNAUTHENTICATED KEY-ASSOCIATED BYTES field.

The A-KAD fixed (AKADF) bit shall be set to one if the device server requires the length of A-KAD in the parameter data for a SECURITY PROTOCOL OUT command to equal the value in the MAXIMUM AUTHENTICATED KEY-ASSOCIATED BYTES field. If the AKADF bit is set to one, then the MAXIMUM AUTHENTICATED KEY-ASSOCIATED BYTES field shall contain a non-zero value. If the AKADF bit is set to zero and the value in the MAXIMUM AUTHENTICATED KEY-ASSOCIATED BYTES field is non-zero, then the length of the A-KAD, if present in the parameter data for a SECURITY PROTOCOL OUT command, shall be a value between one and the value in the MAXIMUM AUTHENTICATED KEY-ASSOCIATED BYTES field.

The MAXIMUM UNAUTHENTICATED KEY-ASSOCIATED DATA BYTES field indicates the maximum size of the unauthenticated key-associated data (see 4.2.29.11) supported by the device server for this algorithm.

The MAXIMUM AUTHENTICATED KEY-ASSOCIATED DATA BYTES field indicates the maximum size of the authenticated key-associated data (see 4.2.29.11) supported by the device server for this algorithm.

The LOGICAL BLOCK ENCRYPTION KEY SIZE field indicates the size in bytes of the logical block encryption key required by the algorithm.

The DKAD_C field (see table 200) indicates the decryption capabilities if the DECRYPTION MODE field of the Set Data Encryption page (see table 215) is set to DECRYPT or MIXED.

Table 200 — DKAD_C field

Code	Name	Description
00b	not specified	No capabilities are specified.
01b	KAD Required	The device entity requires a U-KAD or A-KAD to be provided by the application client with the Set Data Encryption page. If a U-KAD or A-KAD is not provided, 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 INCOMPLETE KEY-ASSOCIATED DATA SET.
10b	KAD Not Allowed	The device entity does not allow a U-KAD or A-KAD to be provided by the application client with the Set Data Encryption page. If a U-KAD or A-KAD is provided, the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN PARAMETER LIST.
11b	KAD Capable	The device entity has the capability to accept a U-KAD or A-KAD to be provided by the application client with the Set Data Encryption page, but one is not required.

The external encryption mode control capabilities (EEMC_C) field (see table 201) indicates the capabilities the encryption algorithm provides to the application client to control write operations that transfer encrypted logical blocks while the encryption mode is set to EXTERNAL.

Table 201 — EEMC_C field

Code	Description
0h	No capabilities are specified.
1h	The encryption algorithm does not allow write operations in EXTERNAL encryption mode. The device server does not act as a KCCLU (see 4.2.29.5) for this encryption algorithm.
2h	The encryption algorithm allows write operations in EXTERNAL encryption mode. The device server does act as a KCCLU (see 4.2.29.5) for this encryption algorithm.
3h	Reserved

The raw decryption mode control capabilities (RDMC_C) field (see table 202) indicates the capabilities the encryption algorithm provides to the application client to control read operations that access encrypted logical blocks while the decryption mode is set to RAW.

Table 202 — RDMC_C field

Code	Description
0h	No capabilities are specified.
1h	The encryption algorithm does not allow read operations in RAW decryption mode. The device server does not act as a KCSLU (see 4.2.29.5) for this encryption algorithm.
2h to 3h	Reserved
4h	The encryption algorithm disables read operations in RAW mode by default and allows the application client to control RAW reads via the RDMC field in the Set Data Encryption page (see 8.7.3.2). The device server acts as a KCSLU (see 4.2.29.5) for this encryption algorithm.
5h	The encryption algorithm enables read operations in RAW mode by default and allows the application client to control RAW reads via the RDMC field in the Set Data Encryption page (see 8.7.3.2). The device server acts as a KCSLU (see 4.2.29.5) for this encryption algorithm.
6h	The encryption algorithm disables read operations in RAW mode by default and does not allow the application client to control RAW reads via the RDMC field in the Set Data Encryption page (see 8.7.3.2). The device server does not act as a KCSLU (see 4.2.29.5) for this encryption algorithm.
7h	The encryption algorithm enables read operations in RAW mode by default and does not allow the application client to control RAW reads via the RDMC field in the Set Data Encryption page (see 8.7.3.2). The device server acts as a KCSLU (see 4.2.29.5) for this encryption algorithm.

The encryption algorithm records encryption mode (EAREM) bit shall be set to one if the encryption mode is recorded with each encrypted logical block. The EAREM bit shall be set to zero if the encryption mode is not recorded with each encrypted logical block.

The MAXIMUM NUMBER OF EEDKS field specifies the number of EEDKs (see 4.2.31.4.2) that the encryption algorithm supports. A MAXIMUM NUMBER OF EEDKS field set to zero indicates the encryption algorithm does not support EEDKs.

The maximum supplemental decryption key count (MSDK_COUNT) field contains the maximum number of supplemental decryption keys that the device server supports with this algorithm. If the SDK_C bit is set to one, then the MSDK_COUNT field shall be set to non-zero. If the SDK_C bit is set to zero, then the MSDK_COUNT field shall be set to zero.

The MAXIMUM EEDK SIZE field indicates the maximum size in bytes of any EEDK supported by the device server for this algorithm.

The device server shall support the number of EEDK(s) specified in the MAXIMUM NUMBER OF EEDKS field of the size specified in the MAXIMUM EEDK SIZE field for this algorithm.

The SECURITY ALGORITHM CODE field contains a security algorithm code (see SPC-4).

8.7.2.5 Supported Key Formats page

Table 203 specifies the format of the Supported Key Formats page.

Table 203 — Supported Key Formats page

Bit Byte	7	6	5	4	3	2	1	0
0	(MSB)				PAGE CODE (0011h)			
1								(LSB)
2	(MSB)				PAGE LENGTH (n-3)			
3								(LSB)
4					SUPPORTED KEY FORMATS LIST			
...								
n								

The SUPPORTED KEY FORMATS LIST field shall contain a list of all of the key formats (see table 221) that the device server supports for the Set Data Encryption page (see 8.7.3.2) in ascending order.

8.7.2.6 Data Encryption Management Capabilities page

Table 204 specifies the format of the Data Encryption Management Capabilities page.

Table 204 — Data Encryption Management Capabilities page

Bit Byte	7	6	5	4	3	2	1	0
0	(MSB)				PAGE CODE (0012h)			
1								(LSB)
2	(MSB)				PAGE LENGTH (0Ch)			
3								(LSB)
4					Reserved			LOCK_C
5					Reserved	CKOD_C	CKORP_C	CKORL_C
6					Reserved			
7					Reserved	AITN_C	LOCAL_C	PUBLIC_C
8								
...					Reserved			
15								

The LOCK_C bit shall be set to one if the device server supports the LOCK bit in the Set Data Encryption page (see 8.7.3.2). The LOCK_C bit shall be set to zero if the device server does not support the LOCK bit in the Set Data Encryption page.

The CKOD_C bit shall be set to one if the device server supports the CKOD bit in the Set Data Encryption page. The CKOD_C bit shall be set to zero if the device server does not support the CKOD bit in the Set Data Encryption page.

The CKORP_C bit shall be set to one if the device server supports the CKORP bit in the Set Data Encryption page. The CKORP_C bit shall be set to zero if the device server does not support the CKORP bit in the Set Data Encryption page.

The CKORL_C bit shall be set to one if the device server supports the CKORL bit in the Set Data Encryption page. The CKORL_C bit shall be set to zero if the device server does not support the CKORL bit in the Set Data Encryption page.

The AITN_C bit shall be set to one if the device server supports a logical block encryption scope of ALL I_T NEXUS. The AITN_C bit shall be set to zero if the device server does not support a logical block encryption scope of ALL I_T NEXUS.

The LOCAL_C bit shall be set to one if the device server supports a logical block encryption scope of LOCAL. The LOCAL_C bit shall be set to zero if the device server does not support a logical block encryption scope of LOCAL.

The PUBLIC_C bit shall be set to one if the device server supports a logical block encryption scope of PUBLIC. The PUBLIC_C bit shall be set to zero if the device server does not support a logical block encryption scope of PUBLIC.

8.7.2.7 Data Encryption Status page

Table 205 specifies the format of the Data Encryption Status page.

Table 205 — Data Encryption Status page

The I_T_NEXUS SCOPE field shall contain the value from the logical block encryption scope saved for the I_T_L nexus through which this command was received (see 4.2.29.14).

The LOGICAL BLOCK ENCRYPTION SCOPE field shall contain the value from the logical block encryption scope in the saved logical block encryption parameters currently associated with the I_T_L nexus on which this command was received (see 4.2.29.15).

The ENCRYPTION MODE field shall contain the value from the encryption mode in the saved logical block encryption parameters currently associated with the L_T_L nexus on which this command was received (see 4.2.29.15).

The DECRYPTION MODE field shall contain the value from the decryption mode in the saved logical block encryption parameters currently associated with the I_T_L nexus on which this command was received (see 4.2.29.15).

The ALGORITHM INDEX field shall contain the value from the algorithm index in the saved logical block encryption parameters currently associated with the I_T_L nexus on which this command was received (see 4.2.29.15). If the ENCRYPTION MODE field and the DECRYPTION MODE field are both set to DISABLE, the value in the ALGORITHM INDEX field is undefined.

The KEY INSTANCE COUNTER field contains the value of the logical block encryption parameters key instance counter (see 4.2.29.8) assigned to the logical block encryption key indicated by the LOGICAL BLOCK ENCRYPTION SCOPE field value.

The PARAMETERS CONTROL field (see table 206) specifies information on how the logical block encryption parameters are controlled.

Table 206 — PARAMETERS CONTROL field

Code	Description
000b	Logical block encryption parameters control is not reported.
001b	Logical block encryption parameters are not exclusively controlled by external data encryption control.
010b	Logical block encryption parameters are exclusively controlled by the sequential-access device server.
011b	Logical block encryption parameters are exclusively controlled by the automation/drive interface device server.
100b	Logical block encryption parameters are exclusively controlled by a management interface.
101b to 111b	Reserved

A volume contains encrypted logical blocks (VCELB) bit set to one indicates that the mounted volume contains an encrypted logical block. A VCELB bit set to zero indicates that either:

- a) the mounted volume does not contain any encrypted logical blocks;
- b) there is no volume mounted; or
- c) the VCELB_C bit in the Data Encryption Capabilities page is set to zero.

The raw decryption mode disabled (RDMD) bit shall be set to one if the device entity is configured to mark each encrypted record as disabled for raw read operations based on the RDMC_C value and the raw decryption mode disable parameter in the saved logical block encryption parameters currently associated with the I_T_L nexus on which the command was received (see 4.2.29.14).

The check external encryption mode status (CEEMS) field shall contain the value from the check external encryption mode parameter in the saved logical block encryption parameters currently associated with the I_T_L nexus on which the command was received (see 4.2.29.14).

The ENCRYPTION PARAMETERS KAD FORMAT field shall contain the value from the KAD FORMAT in the saved logical block encryption parameters (see 4.2.29.14) currently associated with the I_T_L nexus on which this command was received. If the encryption algorithm specified in the ALGORITHM INDEX field reports a KADF_C bit set to zero, then the ENCRYPTION PARAMETERS KAD FORMAT field shall be set to zero.

The available supplemental decryption key count (ASDK_COUNT) field contains the current number of additional supplemental decryption keys that may be loaded for this algorithm by an application client. If the device server is

not capable of supporting supplemental decryption keys, then the ASDK_COUNT field shall be set to zero. The current number of supplemental decryption keys loaded for this algorithm by application clients is the difference between the MSDK_COUNT field in the logical block encryption algorithm descriptor (see 8.7.2.4) and this ASDK_COUNT field.

If the ENCRYPTION MODE field and the DECRYPTION MODE field are both set to DISABLE, the key-associated data descriptors list shall not be included in the page.

If either the ENCRYPTION MODE field or the DECRYPTION MODE field is set to a value other than DISABLE, the key-associated data descriptors list shall include Tape Data Encryption descriptors (see 8.7.4) describing attributes assigned to the logical block encryption key defined by the I_T NEXUS SCOPE and LOGICAL BLOCK ENCRYPTION SCOPE fields at the time the logical block encryption key was established in the device entity. If more than one key-associated data descriptor is included, they shall be in increasing numeric order of the value in the KEY DESCRIPTOR TYPE field (see 8.7.4.2). Descriptors shall be included as defined by the following paragraphs.

An unauthenticated key-associated data descriptor (see 8.7.4.2.2) shall be included if an unauthenticated key-associated data descriptor was included at the time the logical block encryption key was established in the device entity. The AUTHENTICATED field is reserved. The KEY DESCRIPTOR field (see 8.7.4.2) shall contain the U-KAD value associated with the logical block encryption key.

An authenticated key-associated data descriptor (see 8.7.4.2.3) shall be included if an authenticated key-associated data descriptor was included at the time the logical block encryption key was established in the device entity. The AUTHENTICATED field is reserved. The KEY DESCRIPTOR field (see 8.7.4.2) shall contain the A-KAD value associated with the logical block encryption key.

A nonce value descriptor (see 8.7.4.2.4) shall be included if a nonce value descriptor was included at the time the logical block encryption key was established in the device entity. The AUTHENTICATED field is reserved. The KEY DESCRIPTOR field (see 8.7.4.2) shall contain the nonce value associated with the logical block encryption key. A nonce value descriptor may be included if no nonce value descriptor was included at the time the logical block encryption key was established in the device entity. In this case, the KEY DESCRIPTOR field (see 8.7.4.2) shall be set to the nonce value established by the device entity for use with the selected logical block encryption key.

A metadata key-associated data descriptor (see 8.7.4.2.5) shall be included if the metadata key-associated data descriptor was included at the time the logical block encryption parameters were established. The KEY DESCRIPTOR field (see 8.7.4.2) shall contain the M-KAD value associated with the logical block encryption key.

8.7.2.8 Next Block Encryption Status page

Table 207 specifies the format of the Next Block Encryption Status page.

Table 207 — Next Block Encryption Status page

Bit Byte	7	6	5	4	3	2	1	0
0	(MSB)				PAGE CODE (0021h)			
1								(LSB)
2	(MSB)				PAGE LENGTH (n-3)			
3								(LSB)
4	(MSB)				LOGICAL OBJECT NUMBER			
...								
11								(LSB)
12	COMPRESSION STATUS				ENCRYPTION STATUS			
13				ALGORITHM INDEX				
14			Reserved			EMES		RDMDS
15				NEXT BLOCK KAD FORMAT				
					Key-associated data descriptor list			
16								
...					Key-associated data descriptor [first]			
				⋮				
...					Key-associated data descriptor [last]			
n								

The LOGICAL OBJECT NUMBER field contains the logical object identifier of the next logical object (see 4.2.8.2).

The COMPRESSION STATUS field values are specified in table 208.

Table 208 — COMPRESSION STATUS field

Code	Description
0h	The device entity is incapable of determining if the logical object referenced by the LOGICAL OBJECT NUMBER field has been compressed.
1h	The device entity is capable of determining if the logical object referenced by the LOGICAL OBJECT NUMBER field has been compressed, but is not able to at this time. Possible reasons are: a) the next logical block has not yet been read into the buffer; b) there was an error reading the next logical block; or c) there are no more logical blocks (i.e., end-of-logical block).
2h	The device entity has determined that the logical object referenced by the LOGICAL OBJECT NUMBER field is not a logical block.
3h	The device entity has determined that the logical object referenced by the LOGICAL OBJECT NUMBER field is not compressed.
4h	The device entity has determined that the logical object referenced by the LOGICAL OBJECT NUMBER field is compressed.
5h to Fh	Reserved

The ENCRYPTION STATUS field values are specified in table 209.

Table 209 — ENCRYPTION STATUS field

Code	Description
0h	The device entity is incapable of determining if the logical object referenced by the LOGICAL OBJECT NUMBER field has been encrypted.
1h	The device entity is capable of determining if the logical object referenced by the LOGICAL OBJECT NUMBER field has been encrypted, but is not able to at this time. Possible reasons are: <ul style="list-style-type: none"> a) the next logical block has not yet been read into the buffer; b) there was an error reading the next logical block; or c) there are no more logical blocks (i.e., end-of-data).
2h	The device entity has determined that the logical object referenced by the LOGICAL OBJECT NUMBER field is not a logical block.
3h	The device entity has determined that the logical object referenced by the LOGICAL OBJECT NUMBER field is not encrypted.
4h	The device entity has determined that the logical object referenced by the LOGICAL OBJECT NUMBER field is encrypted by an algorithm that is not supported by this device server. The values in the key-associated data descriptors list contain information pertaining to the encrypted logical block.
5h	The device entity has determined that the logical object referenced by the LOGICAL OBJECT NUMBER field is encrypted by an algorithm that is supported by this device server. The values in the ALGORITHM INDEX field and key-associated data descriptors list contain information pertaining to the encrypted logical block.
6h	The device entity has determined that the logical object referenced by the LOGICAL OBJECT NUMBER field is encrypted by an algorithm that is supported by this device server, but the device entity is either not enabled to decrypt or does not have the correct logical block encryption key or nonce value to decrypt the encrypted logical block.
7h to Fh	Reserved

The ALGORITHM INDEX field indicates which of the encryption algorithms reported by the SECURITY PROTOCOL IN command Data Encryption Capabilities page was used to encrypt the logical block. For values in the ENCRYPTION STATUS field (see table 209) that do not indicate the ALGORITHM INDEX field is valid, the algorithm index is undefined.

The encryption mode external status (EMES) bit shall be set to one if:

- a) the ENCRYPTION STATUS field is set to either 5h or 6h;
- b) the EAREM bit in the algorithm descriptor (see 8.7.2.4) for the algorithm specified by the ALGORITHM INDEX field is set to one; and
- c) the next logical block is marked as having been written to the medium while the encryption mode was set to EXTERNAL.

The EMES bit shall be set to zero if:

- a) the ENCRYPTION STATUS field is set to a value other than 5h or 6h;
- b) the EAREM bit in the algorithm descriptor (see 8.7.2.4) for the algorithm specified by the ALGORITHM INDEX field is set to zero; or

- c) the next logical block is marked as having been written to the medium while the encryption mode was set to ENCRYPT.

The raw decryption mode disabled status (RDMDS) bit shall be set to one if the device server supports raw decryption mode, the ENCRYPTION STATUS field is set to either 5h or 6h, and the next logical block is marked as disabled for raw decryption mode operations (see 4.2.29). The RDMDS bit shall be set to zero if the device server does not support raw decryption mode, the ENCRYPTION STATUS field is set to a value other than 5h or 6h, or the next logical block is not marked as disabled for raw decryption mode operations.

If the value in the ENCRYPTION STATUS field indicates that the next logical block is encrypted by a supported algorithm, then the NEXT BLOCK KAD FORMAT field shall contain the KAD FORMAT logical block encryption parameters (see 4.2.29.14) associated with the encrypted logical block. If the value in the ENCRYPTION STATUS field does not indicate that the next logical object is an encrypted logical block, then the NEXT BLOCK KAD FORMAT field shall be ignored. If the encryption algorithm specified in the ALGORITHM INDEX field reports a KADF_C bit set to zero, then the NEXT BLOCK KAD FORMAT field shall be set to zero.

If the value in the ENCRYPTION STATUS field indicates that the next logical block is encrypted by a supported algorithm, then the device server shall include in the key-associated data descriptor list all key-associated data that is associated with the encrypted logical block. If more than one key-associated data descriptor is included in the Next Block Encryption Status page, then they shall be in increasing numeric order of the value in the KEY DESCRIPTOR TYPE field (see 8.7.4.2).

An unauthenticated key-associated data descriptor (see 8.7.4.2.2) shall be included if any unauthenticated key-associated data is associated with the next logical block. The AUTHENTICATED field shall be set to 1. The KEY DESCRIPTOR field shall contain the U-KAD value associated with the encrypted logical block.

An authenticated key-associated data descriptor (see 8.7.4.2.3) shall be included if any authenticated key-associated data is associated with the next logical block. The AUTHENTICATED field shall indicate the status of the authentication done by the device entity (see table 233). The KEY DESCRIPTOR field shall contain the A-KAD value associated with the encrypted logical block.

The Next Block Encryption Status page may include a nonce value descriptor (see 8.7.4.2.4). If a nonce value descriptor is included, then the AUTHENTICATED field shall indicate the status of the authentication done by the device entity (see table 233). The KEY DESCRIPTOR field shall contain the nonce value associated with the encrypted logical block.

A metadata key-associated data descriptor (see 8.7.4.2.5) shall be included if any M-KAD is associated with the next logical block and the decryption mode is set to RAW in the saved logical block encryption parameters currently associated with the I_T_L nexus on which this command was received. The KEY DESCRIPTOR field shall contain the M-KAD value associated with the encrypted logical block.

8.7.2.9 Get Encryption Management Attributes page

Table 210 specifies the format of the Get Encryption Management Attributes page.

Table 210 — Get Encryption Management Attributes page

Bit Byte	7	6	5	4	3	2	1	0
0	(MSB)				PAGE CODE (0022h)			
1								(LSB)
2								
...					Reserved			
4								
5					Reserved			CAOD
6	(MSB)				PAGE LENGTH (n-7)			
7								(LSB)
Encryption management attributes descriptor list								
8								
...					Encryption management attributes descriptor [first]			
:								
...					Encryption management attributes descriptor [last]			
n								

See SPC-4 for a description of the PAGE CODE field and the PAGE LENGTH field. The PAGE CODE field shall be set to the value specified in table 210.

A clear attributes on demount (CAOD) bit set to one indicates that the device entity sets the encryption management attributes list to default values upon completion of a volume demount. A CAOD bit set to zero indicates that the demounting of a volume does not affect the encryption management attributes.

The encryption management attributes descriptor is defined in 8.7.4.4.

8.7.2.10 Random Number page

Table 211 specifies the format of the Random Number page.

Table 211 — Random Number page

Bit Byte	7	6	5	4	3	2	1	0
0	(MSB)							
1					PAGE CODE (0030h)			(LSB)
2	(MSB)							
3					PAGE LENGTH (20h)			(LSB)
4	(MSB)							
...					RANDOM NUMBER			
35								(LSB)

The RANDOM NUMBER field contains a secure random number (see SPC-4), suitable for use as a random nonce (see SPC-4), that is generated by the device server using a source of entropy available within the device. Each request for the Random Number page shall generate a new secure random number for the RANDOM NUMBER field.

8.7.2.11 Device Server Key Wrapping Public Key page

8.7.2.11.1 Device Server Key Wrapping Public Key page overview

The Device Server Key Wrapping Public Key page returns the device server's key wrapping public key. The format of the Device Server Key Wrapping Public Key page is specified in table 212.

Table 212 — Device Service Key Wrapping Public Key page

Bit Byte	7	6	5	4	3	2	1	0
0	(MSB)							
1					PAGE CODE (0031h)			(LSB)
2	(MSB)							
3					PAGE LENGTH (n-3)			(LSB)
4	(MSB)							
...					PUBLIC KEY TYPE			
7								(LSB)
8	(MSB)							
...					PUBLIC KEY FORMAT			
11								(LSB)
12	(MSB)							
13					PUBLIC KEY LENGTH (n-13)			(LSB)
14	(MSB)							
...					PUBLIC KEY			
n								(LSB)

The PUBLIC KEY TYPE field (see table 213) specifies the type of public key in the PUBLIC KEY field.

Table 213 — PUBLIC KEY TYPE field

Code	Description	Reference
00000000h	RSA 2048	8.7.2.11.2
00000001h to 0000000Fh	Reserved	
00000010h	ECC 521	8.7.2.11.3
00000011h to FFFF BFFFh	Reserved	
FFFFC000h to FFFFFFFFh	Vendor specific	

The PUBLIC KEY FORMAT, PUBLIC KEY LENGTH, and PUBLIC KEY field values depend on the PUBLIC KEY TYPE field.

8.7.2.11.2 RSA 2048 public keys

The PUBLIC KEY FORMAT field shall be set to 00000000h. All other values for the PUBLIC KEY FORMAT field are reserved. The PUBLIC KEY LENGTH field shall be set to 512. Bytes 0 through 255 of the PUBLIC KEY field shall contain the modulus n. Bytes 256 through 511 of the PUBLIC KEY field shall contain the public exponent e (see PKCS #1 V2.1).

8.7.2.11.3 ECC 521 public keys

The PUBLIC KEY FORMAT field shall be set to 00000010h. All other values for the PUBLIC KEY FORMAT field are reserved. The PUBLIC KEY LENGTH field shall be set to 133. The PUBLIC KEY field shall be set to the ECC 521 public key as converted by the algorithm specified in ANSI X9.63, section 4.3.6, using the uncompressed form.

8.7.3 SECURITY PROTOCOL OUT command specifying Tape Data Encryption security protocol

8.7.3.1 SECURITY PROTOCOL OUT command specifying Tape Data Encryption security protocol overview

The SECURITY PROTOCOL OUT command specifying the Tape Data Encryption security protocol (i.e., 20h) (see SPC-4) is used to configure the logical block security methods in the device entity and on the medium. The command supports a series of pages that are sent individually. An application client requests to send a page by using a SECURITY PROTOCOL OUT command with the SECURITY PROTOCOL field set to Tape Data Encryption security protocol and the SECURITY PROTOCOL SPECIFIC field set to the page code requested.

The SECURITY PROTOCOL SPECIFIC field (see table 214) specifies the type of page that the application client is sending.

Table 214 — SECURITY PROTOCOL SPECIFIC field

Code	Description	Reference
0000h to 000Fh	Reserved	
0010h	Set Data Encryption page	8.7.3.2
0011h	SA Encapsulation page	8.7.3.3
0012h to 0021h	Reserved	
0022h	Set Encryption Management Attributes page	8.7.3.4
0023h to 002Fh	Reserved	
0030h to 003Fh	Restricted	ADC-3
0040h to FEFFh	Reserved	
FF00h to FFFFh	Vendor specific	

If the SECURITY PROTOCOL SPECIFIC field is set to a reserved or unsupported value, 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.

8.7.3.2 Set Data Encryption page

8.7.3.2.1 Set Data Encryption page overview

Table 215 specifies the format of the Set Data Encryption page.

Table 215 — Set Data Encryption page

Bit Byte	7	6	5	4	3	2	1	0				
0	(MSB)				PAGE CODE (0010h)							
1								(LSB)				
2	(MSB)				PAGE LENGTH (m-3)							
3								(LSB)				
4	SCOPE		Reserved				LOCK					
5	CEEM	RDMC	SDK	CKOD	CKORP	CKORL						
6	ENCRYPTION MODE											
7	DECRYPTION MODE											
8	ALGORITHM INDEX											
9	LOGICAL BLOCK ENCRYPTION KEY FORMAT											
10	KAD FORMAT											
11												
...	Reserved											
17												
18	(MSB)	LOGICAL BLOCK ENCRYPTION KEY LENGTH (n-19)										
19								(LSB)				
20	(MSB)	LOGICAL BLOCK ENCRYPTION KEY										
...												
n								(LSB)				
Key-associated data descriptor list												
n+1												
...	Key-associated data descriptor [first]											
	:											
...												
m	Key-associated data descriptor [last]											

The PAGE LENGTH field specifies the number of bytes of parameter data to follow. If the page length value results in the truncation of any field, the device server shall terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

The SCOPE field (see table 216) specifies the logical block encryption scope of the logical block encryption parameters.

Table 216 — SCOPE field

Code	Name	Description	Support
000b	PUBLIC	All fields other than the SCOPE field and LOCK bit shall be ignored. The I_T_L nexus shall use logical block encryption parameters that are shared by other I_T_L nexus. If no I_T_L nexus are sharing logical block encryption parameters, the device server shall use default logical block encryption parameters.	M
001b	LOCAL	The logical block encryption parameters are unique to the I_T_L nexus associated with the SECURITY PROTOCOL OUT command and shall not be shared with other I_T_L nexus.	O
010b	ALL I_T NEXUS	The logical block encryption parameters shall be shared with all I_T_L nexus that are presented on logical units that implement logical block encryption (see 4.2.29).	M
011b to 111b		Reserved	
Support key: M - mandatory for a device server that supports the Set Data Encryption page O - optional for a device server that supports the Set Data Encryption protocol			

See 4.2.29.9 for a description of the LOCK bit.

The values for the check external encryption mode (CEEM) field are specified in table 217.

Table 217 — CEEM field

Code	Description
00b	Vendor specific
01b	Do not check the encryption mode that was in use at the time the logical block was written to the medium.
10b	On read and verify commands, check the encryption mode that was in use at the time the logical block was written to the medium. Report an error if the logical block was written in EXTERNAL mode (see 4.2.29.5).
11b	On read and verify commands, check the encryption mode that was in use at the time the logical block was written to the medium. Report an error if the logical block was written in ENCRYPT mode (see 4.2.29.5).

The device server shall terminate the SECURITY PROTOCOL OUT command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER DATA if the CEEM field is set to either 10b or 11b, and:

- a) the DECRYPTION MODE field is set to DISABLE; or
- b) the EAREM bit in the algorithm descriptor (see 8.7.2.4) for the algorithm specified by the ALGORITHM INDEX field is set to zero.

The raw decryption mode control (RDMC) field specifies if the device entity shall mark each encrypted logical block written to the medium as disabled for read operations in raw mode (i.e., read operations with the decryption mode

set to RAW). The RDMC field shall be ignored if the ENCRYPTION MODE field is not set to ENCRYPT. Table 218 specifies the values for the RDMC field if the ENCRYPTION MODE field is set to ENCRYPT.

Table 218 — RDMC field

Code	Description
00b	The device entity shall mark each encrypted logical block per the default setting for the algorithm (see table 202).
01b	Reserved
10b	The device entity shall mark each encrypted logical block written to the medium in a format specific manner as enabled for raw decryption mode operations.
11b	The device entity shall mark each encrypted logical block written to the medium in a format specific manner as disabled for raw decryption mode operations.

The device server shall terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST and the additional sense key set to INVALID FIELD IN PARAMETER DATA if:

- a) the ENCRYPTION MODE field is set to ENCRYPT;
- b) the RDMC field is set to 10b or 11b; and
- c) the RDMC_C field in the algorithm descriptor for the encryption algorithm selected by the value in the ALGORITHM INDEX field is set to 1h, 6h, or 7h.

If the supplemental decryption key (SDK) bit is set to one, the logical block encryption key sent in this page shall be added to the set of logical block encryption parameters used by the device entity for the selected logical block encryption scope. The logical block encryption parameters key instance counter shall not be incremented for supplemental decryption keys. The ENCRYPTION MODE and LOCK fields shall be ignored and the DECRYPTION MODE field shall match the current setting for this logical block encryption scope. If the DECRYPTION MODE field does not match the current settings for this logical block encryption scope, the device server shall terminate the command with CHECK CONDITION status and set the sense key to ILLEGAL REQUEST and the additional sense code to INVALID FIELD IN PARAMETER LIST.

If the SDK bit is set to one and the device entity does not currently have a saved set of logical block encryption parameters associated with the I_T_L nexus that sent the Set Data Encryption page or the logical block encryption scope or decryption mode values do not match the values in that set of saved logical block encryption parameters, then the device server shall terminate the command with CHECK CONDITION status and set the sense key to ILLEGAL REQUEST and the additional sense code to INVALID FIELD IN PARAMETER LIST.

If the SDK bit is set to one and the SDK_C field is set to zero in the logical block encryption algorithm descriptor that matches the ALGORITHM INDEX field in the Data Encryption Capabilities page, then the device server shall terminate the command with CHECK CONDITION status and set the sense key set ILLEGAL REQUEST and the additional sense code to INVALID FIELD IN PARAMETER LIST.

If the device server is processing a Set Data Encryption page with the SDK bit set to one and does not have the resource available to store this supplemental decryption key, then the device server shall terminate the command with CHECK CONDITION status and set the sense key to ILLEGAL REQUEST and the additional sense code to MAXIMUM NUMBER OF SUPPLEMENTAL DECRYPTION KEYS EXCEEDED. Any previously saved supplemental decryption keys shall not be affected by this error.

If the SDK bit is set to zero, the logical block encryption key sent in this page shall be the logical block encryption key used for both encryption and decryption. Any logical block encryption key and supplemental decryption keys that have been previously stored by the device server shall be removed from memory. See 4.2.29.6.

If the clear key on demount (CKOD) bit is set to one, the device entity shall set the logical block encryption parameters to default values upon completion of a volume demount. If the CKOD bit is set to zero, the demounting of a volume shall not affect the logical block encryption parameters. If the CKOD bit is set to one and there is no volume mounted, the device server shall terminate the command with CHECK CONDITION status and set the sense key to ILLEGAL REQUEST and the additional sense code to INVALID FIELD IN PARAMETER DATA.

The behavior related to the clear key on reservation preempt (CKORP) bit is specified in 4.2.29.17. If the CKORP bit is set to one and there is no persistent reservation in effect for the I_T_L nexus associated with the SECURITY PROTOCOL OUT command, the device server shall terminate the command with CHECK CONDITION status and set the sense key to ILLEGAL REQUEST and the additional sense code to INVALID FIELD IN PARAMETER DATA.

The behavior related to the clear key on reservation loss (CKORL) bit is specified in 4.2.29.16. If the CKORL bit is set to one and there is no reservation in effect for the I_T_L nexus associated with the SECURITY PROTOCOL OUT command, the device server shall terminate the command with CHECK CONDITION status and set the sense key to ILLEGAL REQUEST and the additional sense code to INVALID FIELD IN PARAMETER DATA.

The values for the ENCRYPTION MODE field are specified in table 219.

Table 219 — ENCRYPTION MODE field

Code	Name	Description
0h	DISABLE	Logical block encryption is disabled.
1h	EXTERNAL	The logical blocks associated with the WRITE(6) and WRITE(16) commands has been encrypted by a system that is compatible with the algorithm specified by the ALGORITHM INDEX field.
2h	ENCRYPT	The device entity shall encrypt all logical blocks that it receives for a WRITE(6) or WRITE(16) command using the algorithm specified in the ALGORITHM INDEX field and the logical block encryption key specified in the LOGICAL BLOCK ENCRYPTION KEY field.
3h to Fh		Reserved

The values for the DECRYPTION MODE field are specified in table 220. See 4.2.29.3 for configuration and exception condition requirements.

Table 220 — DECRYPTION MODE field

Code	Name	Description
0h	DISABLE	Logical block decryption is disabled. If the device entity encounters an encrypted logical block while reading, it shall not allow access to the logical block.
1h	RAW	Logical block decryption is disabled. If the device entity encounters an encrypted logical block while reading, it shall pass the encrypted logical block to the host without decrypting it. The encrypted logical block may contain data that is not user data.
2h	DECRYPT	The device entity shall decrypt all logical blocks that are read from the medium if processing a READ(6), READ(16), READ REVERSE(6), READ REVERSE(16), or RECOVER BUFFERED DATA command or verified if processing a VERIFY(6) or VERIFY(16) command. The logical blocks shall be decrypted using the algorithm specified in the ALGORITHM INDEX field and the logical block encryption key specified in the LOGICAL BLOCK ENCRYPTION KEY field.
3h	MIXED	The device entity shall decrypt all logical blocks that are read from the medium that the device entity determines was encrypted if processing a READ(6), READ(16), READ REVERSE(6), READ REVERSE(16), or RECOVER BUFFERED DATA command or verified if processing a VERIFY(6) or VERIFY(16) command. The logical blocks shall be decrypted using the algorithm specified in the ALGORITHM INDEX field and the logical block encryption key specified in the LOGICAL BLOCK ENCRYPTION KEY field. If the device entity encounters unencrypted logical blocks while processing a READ(6), READ(16), READ REVERSE(6), READ REVERSE(16), RECOVER BUFFERED DATA, VERIFY(6), or VERIFY(16) command, the logical blocks shall be processed without decrypting.
4h to Fh		Reserved

If the device entity is not capable of distinguishing encrypted logical blocks from unencrypted logical blocks using the algorithm specified in the ALGORITHM INDEX field and the DECRYPTION MODE field is set to MIXED, the device server shall terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER DATA.

If the ENCRYPTION MODE field is set to ENCRYPT and the LOGICAL BLOCK ENCRYPTION KEY LENGTH field is set to zero, the device server shall terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER DATA.

If the DECRYPTION MODE field is set to DECRYPT or MIXED and the LOGICAL BLOCK ENCRYPTION KEY LENGTH field is set to zero, the device server shall terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER DATA.

The ALGORITHM INDEX field indicates which of the encryption algorithms reported by the SECURITY PROTOCOL IN command Data Encryption Capabilities pages shall be used to encrypt and decrypt logical blocks. If the algorithm specified in the ALGORITHM INDEX field is disabled, then the device server shall terminate the command with CHECK

CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to ENCRYPTION ALGORITHM DISABLED.

If a volume is mounted and the combination of the ENCRYPTION MODE, DECRYPTION MODE, and ALGORITHM INDEX fields is not valid for the mounted volume or current logical position, the device server shall terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the addition sense code set to INVALID FIELD IN PARAMETER DATA.

The LOGICAL BLOCK ENCRYPTION KEY FORMAT field (see table 221) specifies the format of the value in the LOGICAL BLOCK ENCRYPTION KEY field.

Table 221 — LOGICAL BLOCK ENCRYPTION KEY FORMAT field

Code	Description	Reference
00h	The LOGICAL BLOCK ENCRYPTION KEY field contains a plain-text key.	8.7.3.2.2
01h	The LOGICAL BLOCK ENCRYPTION KEY field contains a vendor-specific key reference.	8.7.3.2.3
02h	The LOGICAL BLOCK ENCRYPTION KEY field contains a key wrapped by device server public key.	8.7.3.2.4
03h	The LOGICAL BLOCK ENCRYPTION KEY field contains a key encrypted using ESP-SCSI.	8.7.3.2.5
04h to BFh	Reserved	
C0h to FFh	Vendor specific	

If the algorithm in the ALGORITHM INDEX field reports a KADF_C bit set to one, then the KAD FORMAT field shall contain a KAD format code (see table 222) to identify the format of the data stored in the U-KAD, A-KAD, or both (see 4.2.29.11). If the KAD format code indicates that the U-KAD, A-KAD, or both contains a logical block encryption key name (e.g., code 01h or any other code defined as a logical block encryption key name), then:

- a) only an A-KAD descriptor is provided and the authenticated key-associated data is the key name,
- b) only a U-KAD descriptor is provided and the unauthenticated key-associated data is the key name, or
- c) both an A-KAD descriptor and a U-KAD descriptor is provided and the key name is formed by the authenticated key-associated data followed by the unauthenticated key-associated data.

If the algorithm in the ALGORITHM INDEX field reports a KADF_C bit set to one, then the device server shall save the value in the KAD FORMAT field and associate it with every logical block that is encrypted with these logical block encryption parameters. If the algorithm specified in the ALGORITHM INDEX field reports a KADF_C bit set to zero, then the device server shall terminate a command attempting to set the KAD FORMAT to a value other than zero with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER DATA.

KAD format code values are specified in table 222.

Table 222 — KAD format code

Code	Description
00h	Unspecified
01h	Binary logical block encryption key name
02h	ASCII logical block encryption key name
03h to FFh	Reserved

The LOGICAL BLOCK ENCRYPTION KEY LENGTH field specifies the length of the LOGICAL BLOCK ENCRYPTION KEY field in bytes.

If the ENCRYPTION MODE field is set to ENCRYPT the device entity shall save the key-associated descriptors in the key-associated data descriptor list and associate them with every logical block that is encrypted with this logical block encryption key by the device entity.

If the ENCRYPTION MODE field is set to EXTERNAL the device entity shall save the key-associated descriptors in the key-associated data descriptor list and associate them with every logical block that is written using the logical block encryption parameters established by this command.

If more than one key-associated data descriptor is specified in the Set Data Encryption page, they shall be in increasing numeric order of the value in the KEY DESCRIPTOR TYPE field (see 8.7.4.2).

The device server shall terminate the command with CHECK CONDITION, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST if:

- a) key-associated descriptors are included in the key-associated data descriptor list;
- b) DECRYPTION MODE field is not set to RAW; and
- c) the ENCRYPTION MODE field is not set to:
 - A) EXTERNAL; or
 - B) ENCRYPT.

An unauthenticated key-associated data descriptor (see 8.7.4.2.2) may be included if any unauthenticated key-associated data is to be associated with logical blocks encrypted with the algorithm and logical block encryption key. The AUTHENTICATED field is reserved. The KEY DESCRIPTOR field shall contain the U-KAD value associated with the encrypted logical block. The device server shall terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER DATA if the UKADF bit is set to one in the logical block encryption algorithm descriptor, the ENCRYPTION MODE field is set to ENCRYPT, and:

- a) the length of the KEY DESCRIPTOR field is not equal to the value in the MAXIMUM UNAUTHENTICATED KEY-ASSOCIATED BYTES field of the logical block encryption algorithm descriptor; or
- b) the parameter data does not contain an unauthenticated key-associated data descriptor.

An authenticated key-associated data descriptor (see 8.7.4.2.3) may be included if any authenticated key-associated data is to be associated with logical blocks encrypted with the algorithm and logical block encryption key. The AUTHENTICATED field is reserved. The KEY DESCRIPTOR field shall contain the A-KAD value associated with the encrypted logical block. The device server shall terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID

FIELD IN PARAMETER DATA if the AKADF bit is set to one in the logical block encryption algorithm descriptor, the ENCRYPTION MODE field is set to ENCRYPT, and:

- a) the length of the KEY DESCRIPTOR field is not equal to the value in the MAXIMUM AUTHENTICATED KEY-ASSOCIATED BYTES field of the logical block encryption algorithm descriptor; or
- b) the parameter data does not contain an authenticated key-associated data descriptor.

If a nonce value descriptor (see 8.7.4.2.4) is included and the algorithm and the device entity supports application client generated nonce values, the value in the KEY DESCRIPTOR field shall be used as the nonce value for the encryption process. If a nonce value descriptor is included and the encryption algorithm or the device entity does not support application client generated nonce values, the device server shall terminate the command with CHECK CONDITION, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST. If the encryption algorithm or the device entity requires an application client generated nonce value and a nonce value descriptor is not included, the device server shall terminate the command with CHECK CONDITION, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INCOMPLETE KEY-ASSOCIATED DATA SET. If a nonce value descriptor is included, the AUTHENTICATED field is reserved. The KEY DESCRIPTOR field shall contain the nonce value associated with the encrypted logical block.

A metadata key-associated data descriptor (see 8.7.4.2.5) may be included if the DECRYPTION MODE field is set to RAW and the encryption algorithm requires any metadata key-associated data to be associated with encrypted logical blocks read if the DECRYPTION MODE field is set to RAW.

A metadata key-associated data descriptor (see 8.7.4.2.5) shall be included if the ENCRYPTION MODE field is set to EXTERNAL and the encryption algorithm requires any metadata key-associated data to be associated with logical blocks written if the ENCRYPTION MODE field is set to EXTERNAL.

The device server shall terminate the command with CHECK CONDITION, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST if an M-KAD is included and:

- a) the ENCRYPTION MODE field is not set to EXTERNAL and the DECRYPTION MODE field is not set to RAW; or
- b) the encryption algorithm specified by the ALGORITHM INDEX field does not support M-KAD.

A wrapped key key-associated data descriptor (see 8.7.4.2.6) may be included if the encryption algorithm indicated by the ALGORITHM INDEX field supports EEDKs (i.e., the MAXIMUM NUMBER OF EEDKS field in the Encryption Capabilities page is set to a non-zero value).

8.7.3.2.2 Plain-text key

If the KEY FORMAT field is set to 00h, the KEY field contains the key in an algorithm-specific format. Table 223 specifies the format of the key in the KEY field if the KEY FORMAT field is set to 00h.

Table 223 — KEY field format with KEY FORMAT field set to 00h

Bit Byte	7	6	5	4	3	2	1	0
0	(MSB)							
...								
n								(LSB)

8.7.3.2.3 Vendor-specific key reference

If the KEY FORMAT field is set to 01h, the KEY field shall contain 8 bytes of T10 vendor identification (see SPC-4) followed by a vendor-specific key reference identifying the logical block encryption key to be used to encrypt or decrypt logical blocks. If the KEY field contains a vendor-specific key reference that is unknown to the device server, the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to VENDOR SPECIFIC KEY REFERENCE NOT FOUND. Table 224 specifies the format of the KEY field if the KEY FORMAT field is set to 01h.

Table 224 — KEY field format with KEY FORMAT field set to 01h

Bit Byte	7	6	5	4	3	2	1	0
0	(MSB)							
...								
7								(LSB)
8	(MSB)							
...								
n								(LSB)

8.7.3.2.4 Key wrapped by device server public key

8.7.3.2.4.1 Key wrapped by device server public key overview

Table 225 specifies the format of the KEY field if the KEY FORMAT field is set to 02h.

Table 225 — KEY field format with KEY FORMAT field set to 02h

Bit Byte	7	6	5	4	3	2	1	0
0	(MSB)							
1								(LSB)
2	(MSB)							
3								(LSB)
4	(MSB)							
...								
n								(LSB)
n+1	(MSB)							
n+2								
n+3	(MSB)							
...								
m								(LSB)
m+1	(MSB)							
m+2								
m+3	(MSB)							
...								
z								(LSB)

The PARAMETER SET field (see table 226) specifies the parameters used in the key wrapping operation.

Table 226 — PARAMETER SET field

Code	Description	Reference
0000h	RSA 2048	8.7.3.2.4.2
0001h to 000Fh	Reserved	
0010h	ECC 521	8.7.3.2.4.3
0011h to BFFFh	Reserved	
C000h to FFFFh	Vendor specific	

The LABEL LENGTH field specifies the length of the LABEL field in bytes.

The LABEL field contains public information associated with the logical block encryption key (e.g., key identification). The LABEL field shall consist of Wrapped Key descriptors (see 8.7.4.3).

The WRAPPED KEY LENGTH field specifies the length of the WRAPPED KEY field in bytes.

The WRAPPED KEY field contains the logical block encryption key encrypted by the parameters specified in the PARAMETER SET field.

The SIGNATURE LENGTH field specifies the length of the SIGNATURE field in bytes.

The SIGNATURE field contains the wrapper's signature as specified by the PARAMETER SET field.

If the device server determines that an incorrect public key was used to wrap the logical block encryption key, the device server shall terminate the command with CHECK CONDITION status, set the sense key to DATA PROTECT, and set the additional sense code to INCORRECT DATA ENCRYPTION KEY.

If a device server that tries to verify the signature determines that it does not have the specified signature verification public key, the device server shall terminate the command with CHECK CONDITION status, set the sense key to DATA PROTECT, and set the additional sense code to UNKNOWN SIGNATURE VERIFICATION KEY.

If the device server encounters an error while unwrapping the logical block encryption key, the device server shall terminate the command with CHECK CONDITION status, set the sense key to DATA PROTECT, and set the additional sense code to UNABLE TO DECRYPT DATA.

If a device server encounters an error while verifying the signature over the wrapped logical block encryption key, the device server shall terminate the command with CHECK CONDITION status, set the sense key to DATA PROTECT, and set the additional sense code to CRYPTOGRAPHIC INTEGRITY VALIDATION FAILED.

8.7.3.2.4.2 Key wrapping with RSA 2048

If the PARAMETER SET field is set to 0000h, the WRAPPED KEY field shall contain the logical block encryption key wrapped using RSAES-OAEP as specified by PKCS #1 V2.1. The RSA modulus length shall be 2048 bits. The hash function used shall be SHA-256. The mask generation function (MGF) used shall be MGF1 with SHA-256 as the hash function.

The WRAPPED KEY shall be the logical block encryption key encrypted with the device server's public key as specified by PKCS #1 V2.1 in 7.1.1 RSAES-OAEP-ENCRYPT ((n, e), M, L), where (n, e) is the public key, M is the raw key, and L is the LABEL field as specified in 8.7.3.2.4.1.

If a signature is included, the SIGNATURE field shall be the RSASSA-PSS signature over the WRAPPED KEY field as specified by PKCS #1 V2.1 in 8.1.1 using the wrapping entity's private key. The RSAES-OAEP-ENCRYPT operation adds an integrity check to both L (i.e., the label) and M (i.e., the key). The signature is for providing proof of the wrapper's identification. The device server may check the signature to verify that the key was wrapped and signed by a valid key management server.

8.7.3.2.4.3 Key wrapping with ECC 521

If the PARAMETER SET field is set to 0010h, the WRAPPED KEY field shall contain the logical block encryption key wrapped using ECIES-HC as specified in ISO/IEC 18033-2. The ECIES-HC requirements and parameters for ECIES-KEM are specified in table 227.

Table 227 — ECIES-HC requirements and parameters for ECIES-KEM

Item	Parameter	Description	Reference
Elliptic curve	Elliptic curve	Curve P-521	Appendix D, FIPS 186-3
KDF		Approved Alternative 1	NIST SP800-56A
	H	SHA-512	
	AlgorithmID	00001h	
	CofactorMode	0	
	OldCofactorMode	0	
	CheckMode	1	
	SingleHashMode	0	
	KeyLen	96	
	fmt	uncompressed	

NOTE 68 - The deviation of the definition of the KDF from ISO/IEC 18033-2 is to allow a conforming implementation to be FIPS 140-2 compliant.

The ECIES-HC requirements and parameters for ECIES-DEM are specified in table 228.

Table 228 — ECIES-HC requirements and parameters for ECIES-DEM

Item	Parameter	Description
DEM		DEM1
	SC	SC1 using AES-256 as the block cipher
	MA	HMAC with SHA-512

If a signature is included, the SIGNATURE field shall be the ECDSA signature over the WRAPPED KEY field as specified in FIPS 186-3 using the wrapping entity's private key. The device server may check the signature to verify that the logical block encryption key was wrapped and signed by a valid key management server.

8.7.3.2.5 Key encrypted using ESP-SCSI

If the KEY FORMAT field is set to 03h, then the KEY field shall contain an ESP-SCSI Data-Out Buffer without length descriptor (see SPC-4) that includes a logical block encryption key that has been encrypted in accordance with an

SA that has been created in the device server (see SPC-4). The SA shall use an encryption algorithm other than ENCR_NULL. The KEY LENGTH field contains the length of the ESP-SCSI Data-Out Buffer without length descriptor.

If the USAGE_TYPE SA parameter in the SA associated with the value in the DS_SAI field in the ESP-SCSI Data-Out Buffer without length descriptor is not set to 0081h (i.e., Tape Data Encryption), then the device server shall terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID SA USAGE.

8.7.3.3 SA Encapsulation page

The SA Encapsulation page shall contain an ESP-SCSI outbound data descriptor (see SPC-4) that has been encrypted in accordance with an SA that has been created in the device server (see SPC-4). The SA shall use an encryption algorithm other than ENCR_NULL.

If the USAGE_TYPE SA parameter in the SA associated with the value in the DS_SAI field in the ESP-SCSI outbound data descriptor is not set to 0081h (i.e., Tape Data Encryption), then the device server shall terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID SA USAGE.

The format of the SA Encapsulation page is specified in table 229.

Table 229 — SA Encapsulation page

Bit Byte	7	6	5	4	3	2	1	0
0	(MSB)				PAGE CODE (0011h)			
1								(LSB)
2								
...					ESP-SCSI outbound data descriptor			
n								

See SPC-4 for a description of the ESP-SCSI outbound data descriptor. The ENCRYPTED OR AUTHENTICATED DATA field in the ESP-SCSI outbound data descriptor contains any Tape Data Encryption security protocol SECURITY PROTOCOL OUT command page except the SA Encapsulation page.

8.7.3.4 Set Encryption Management Attributes page

Table 230 specifies the format of the Set Encryption Management Attributes page.

Table 230 — Set Encryption Management Attributes page

Bit Byte	7	6	5	4	3	2	1	0
0	(MSB)				PAGE CODE (0022h)			
1								(LSB)
2								
...					Reserved			
4								
5					Reserved			CAOD
6	(MSB)				PAGE LENGTH (n-7)			
7								(LSB)
Encryption management attributes descriptor list								
8								
...					Encryption management attributes descriptor [first]			
					⋮			
...					Encryption management attributes descriptor [last]			
n								

See SPC-4 for a description of the PAGE CODE field and the PAGE LENGTH field. The PAGE CODE field shall be set to the value specified in table 230.

A clear attributes on demount (CAOD) bit set to one indicates that the device entity shall set the encryption management attributes to default values upon completion of a volume demount. A CAOD bit set to zero indicates that the demounting of a volume shall not affect the encryption management attributes. If the CAOD bit is set to one and there is no volume mounted, then the device server shall terminate the command with CHECK CONDITION status and set the sense key to ILLEGAL REQUEST and the additional sense code to INVALID FIELD IN PARAMETER LIST. If the CAOD bit is set to zero and there is no volume mounted, then the device server shall process the command normally.

The encryption management attributes descriptor is specified in 8.7.4.4.

8.7.4 SECURITY PROTOCOL IN and SECURITY PROTOCOL OUT descriptors

8.7.4.1 Tape Data Encryption security protocol descriptors overview

Several of the parameter pages used by the SECURITY PROTOCOL IN and SECURITY PROTOCOL OUT commands allow for the inclusion of descriptors to provide additional optional data. This subclause defines the descriptors that are common between multiple pages.

8.7.4.2 Tape Data Encryption descriptors

8.7.4.2.1 Tape Data Encryption descriptors overview

The format of a Tape Data Encryption descriptor is specified in table 231.

Table 231 — Tape Data Encryption descriptor format

Bit Byte	7	6	5	4	3	2	1	0						
0	KEY DESCRIPTOR TYPE													
1	Reserved					AUTHENTICATED								
2	(MSB)	KEY DESCRIPTOR LENGTH (n-3)						(LSB)						
3														
4	(MSB)	KEY DESCRIPTOR												
...														
n														

The KEY DESCRIPTOR TYPE field (see table 232) contains a value that defines the contents of the KEY DESCRIPTOR field.

Table 232 — KEY DESCRIPTOR TYPE field

Code	Description	Reference
00h	Unauthenticated key-associated data (U-KAD)	8.7.4.2.2
01h	Authenticated key-associated data (A-KAD)	8.7.4.2.3
02h	Nonce value	8.7.4.2.4
03h	Metadata key-associated data (M-KAD)	8.7.4.2.5
04h	Wrapped key key-associated data (WK-KAD)	8.7.4.2.6
05h to BFh	Reserved	
C0h to FFh	Vendor specific	

Table 233 specifies the values for the AUTHENTICATED field.

Table 233 — AUTHENTICATED field

Code	Description
0	Reserved
1	The value in the KEY DESCRIPTOR field is not covered by the authentication (i.e., U-KAD).
2	No attempt has been made to authenticate the value in the KEY DESCRIPTOR field.
3	The value in the KEY DESCRIPTOR field has been authenticated.
4	The value in the KEY DESCRIPTOR field has failed authentication.
5 to 7	Reserved

8.7.4.2.2 U-KAD key descriptor

The AUTHENTICATED field in a U-KAD key descriptor shall be set to 1.

The KEY DESCRIPTOR field of a U-KAD key descriptor shall contain any U-KAD assigned to the logical block encryption key.

8.7.4.2.3 A-KAD key descriptor

The AUTHENTICATED field shall be set as defined by the page in which the descriptor is included.

The KEY DESCRIPTOR field of an A-KAD key descriptor shall contain any A-KAD assigned to the logical block encryption key.

8.7.4.2.4 Nonce value key descriptor

The AUTHENTICATED field shall be set as defined by the page in which the key descriptor is included.

The KEY DESCRIPTOR field of a nonce value key descriptor shall contain the nonce value used with the logical block encryption key.

8.7.4.2.5 M-KAD key descriptor

The AUTHENTICATED field in an M-KAD key descriptor shall be set to 2.

The KEY DESCRIPTOR field of an M-KAD key descriptor contains data required by the encryption algorithm for a keyless copy operation (see 4.2.29.5).

8.7.4.2.6 WK-KAD key descriptor

The AUTHENTICATED field in a WK-KAD key descriptor shall be set to 2.

The KEY DESCRIPTOR field of a WK-KAD key descriptor contains one EEDK for use during key wrapping using key manager specific methods (see 4.2.31.4).

8.7.4.3 Wrapped Key descriptors

8.7.4.3.1 Wrapped key descriptors overview

The format of a Wrapped Key descriptor is specified in table 234.

Table 234 — Wrapped Key descriptor format

Bit Byte	7	6	5	4	3	2	1	0
0	WRAPPED KEY DESCRIPTOR TYPE							
1	Reserved							
2	(MSB)	WRAPPED KEY DESCRIPTOR LENGTH (n-3)						(LSB)
3								
4	(MSB)	WRAPPED KEY DESCRIPTOR						
...								
n								(MSB)

The WRAPPED KEY DESCRIPTOR TYPE field (see table 235) contains a value that defines the contents of the WRAPPED KEY DESCRIPTOR field.

Table 235 — WRAPPED KEY DESCRIPTOR TYPE field

Code	Description	Reference	Support
00h	Device server identification descriptor	8.7.4.3.2	M
01h	Key wrapping entity identification descriptor	8.7.4.3.3	M
02h	Wrapped key information descriptor	8.7.4.3.4	O
03h	Wrapped key identification descriptor	8.7.4.3.5	M
04h	Wrapped key length descriptor	8.7.4.3.6	M
05h to BFh	Reserved		
C0h to FFh	Vendor-specific descriptor		

Support key:
M - mandatory for a device server that supports the Device Server Key Wrapping Public Key page (see 8.7.2.11).
O - optional for a device server that supports the Device Server Key Wrapping Public Key page (see 8.7.2.11).

8.7.4.3.2 Device server identification descriptor

The WRAPPED KEY DESCRIPTOR field shall contain the logical unit name (see SAM-5) for the device server. The logical unit name may be used by the device server to validate that the logical block encryption key was wrapped with the correct public key. The logical unit name may also be used by an application client to validate that the logical block encryption key was wrapped with the correct public key, if the logical block encryption key was wrapped by a third-party system (e.g., a key management server).

Editors Note 4 - CCB: The text above is poorly worded. SAM-5 does not define a structure for reporting a logical unit name, it only defines what the logical unit name is. SPC-4 indicates a logical unit name

can be reported in a designator in VPD page 83 so it might be possible to rephrase to use that terminology. The normal method of transferring a name is to use a TransportID, potentially this could be updated to use that structure but that wouldn't be compatible with existing implementations so probably would need a selector.

8.7.4.3.3 Key wrapping entity identification descriptor

The WRAPPED KEY DESCRIPTOR field shall contain identification data that uniquely identifies the key wrapping entity. The identification data may be used by the device server to locate the key wrapper's public key for signature verification.

8.7.4.3.4 Wrapped key information descriptor

The WRAPPED KEY DESCRIPTOR field contains information that may be used to identify the key. This information is not required to be unique (e.g., a label entered by the system operator).

8.7.4.3.5 Wrapped key identification descriptor

The WRAPPED KEY DESCRIPTOR field shall contain the key identifier.

8.7.4.3.6 Wrapped key length descriptor

The WRAPPED KEY DESCRIPTOR field shall contain the length of the wrapped key in bytes.

8.7.4.4 Encryption management attributes descriptor

8.7.4.4.1 Encryption management attributes descriptor overview

The format of an encryption management attributes descriptor is specified in table 236.

Table 236 — Encryption management attributes descriptor format

Bit Byte	7	6	5	4	3	2	1	0
0	(MSB)							
1								(LSB)
2				Reserved				
3	CRIT				Reserved			
4	(MSB)							
5								(LSB)
6								
...								
n								

The device server shall process the encryption management attributes as specified in 4.2.31.5.2.

An application client processes the encryption management attributes as specified in 4.2.31.5.2.

Table 237 specifies the values for the ENCRYPTION MANAGEMENT ATTRIBUTES TYPE field.

Table 237 — ENCRYPTION MANAGEMENT ATTRIBUTES TYPE field

Code	Description	Number of descriptors in Get Encryption Management Attributes	Number of descriptors in Set Encryption Management Attributes	Support	Reference
0000h	Desired key manager operation	0..1	1	M	8.7.4.4.2
0001h	Logical block encryption key selection criteria	0..1	0..1	M	8.7.4.4.3
0002h	Logical block encryption key wrapping attribute	0..n	0..n	O	8.7.4.4.4
0003h to FFFFh	Reserved				
FF00h to FFFFh	Vendor specific			O	
Support key: M - mandatory O - optional					

Encryption management attributes shall be listed in ascending order by encryption management attribute type.

The CRIT bit shall be processed as described in 4.2.31.5.2.

The ENCRYPTION MANAGEMENT ATTRIBUTES LENGTH field indicates the length of the ENCRYPTION MANAGEMENT ATTRIBUTES field.

See the subclause referenced in table 237 for the definition of the ENCRYPTION MANAGEMENT ATTRIBUTES TYPE field.

8.7.4.4.2 Desired key manager operation attribute

If the ENCRYPTION MANAGEMENT ATTRIBUTES TYPE field is set to 0000h (i.e., desired key manager operation), then the ENCRYPTION MANAGEMENT ATTRIBUTES LENGTH field shall be set to 0002h and the ENCRYPTION MANAGEMENT ATTRIBUTES field is specified in table 238.

Table 238 — Desired key manager operation definition

Code	Description	Reference
0000h	Reserved	
0001h	Create	4.2.31.4.3
0002h	Resolve	4.2.31.4.4
0003h to FFEFh	Reserved	
FFF0h to FFFFh	Vendor specific	

There shall be one desired key manager operation attribute as the first attribute in any encryption management attribute descriptor list sent in a Set Encryption Management Attribute page (see 8.7.3.4).

8.7.4.4.3 Logical block encryption key selection criteria attribute

8.7.4.4.3.1 Logical block encryption key selection criteria attribute overview

If the ENCRYPTION MANAGEMENT ATTRIBUTES TYPE field is set to 0001h (i.e., logical block encryption key selection criteria), then the ENCRYPTION MANAGEMENT ATTRIBUTES field format is specified in table 239.

Table 239 — Logical block encryption key selection criteria attribute format

Bit Byte	7	6	5	4	3	2	1	0
Logical block encryption key selection criteria descriptor list								
0								
...								
	Logical block encryption key selection criteria descriptor [first] (see table 240) _____							
	⋮							
...								
n								
Logical block encryption key selection criteria descriptor [last] (see table 240) _____								

There shall be zero or one logical block encryption key selection criteria attribute in any encryption management attribute descriptor list. If there is no logical block encryption key selection criteria attribute, then the key manager should use means outside the scope of this standard to determine how to select the logical block encryption key.

The logical block encryption key selection criteria descriptor format is specified in table 240.

Table 240 — Logical block encryption key selection criteria descriptor format

Bit Byte	7	6	5	4	3	2	1	0
0	CRIT	(MSB)						
1								LOGICAL BLOCK ENCRYPTION KEY SELECTION CRITERIA TYPE (LSB)
2	(MSB)							LOGICAL BLOCK ENCRYPTION KEY SELECTION CRITERIA LENGTH (n-3) (LSB)
3								
4								
...								LOGICAL BLOCK ENCRYPTION KEY SELECTION CRITERIA
n								

The LOGICAL BLOCK ENCRYPTION KEY SELECTION CRITERIA TYPE field is specified in table 241.

Table 241 — LOGICAL BLOCK ENCRYPTION KEY SELECTION CRITERIA TYPE field

Code	Description	Reference
0000h	Reserved	
0001h	Logical block encryption key algorithm	8.7.4.4.3.2
0002h	Logical block encryption key identifier	8.7.4.4.3.3
0003h to 7FEFh	Reserved	
7FF0h to FFFFh	Vendor specific	

8.7.4.4.3.2 Logical block encryption key algorithm

If the LOGICAL BLOCK ENCRYPTION KEY SELECTION CRITERIA TYPE field is set to 0001h (i.e., logical block encryption key algorithm), then the LOGICAL BLOCK ENCRYPTION KEY SELECTION CRITERIA field indicates for which of the encryption algorithms reported by the SECURITY PROTOCOL IN command Logical Block Encryption Capabilities pages the logical block encryption key shall be used.

8.7.4.4.3.3 Logical block encryption key identifier

If the LOGICAL BLOCK ENCRYPTION KEY SELECTION CRITERIA TYPE field is set to 0002h (i.e., logical block encryption key identifier), then the LOGICAL BLOCK ENCRYPTION KEY SELECTION CRITERIA field contains a key identifier for the logical block encryption key.

8.7.4.4.4 Logical block encryption key wrapping attribute

8.7.4.4.4.1 Logical block encryption key wrapping attribute overview

If the ENCRYPTION MANAGEMENT ATTRIBUTES TYPE field is set to 0002h (i.e., logical block encryption key wrapping attribute), then the ENCRYPTION MANAGEMENT ATTRIBUTES field format is specified in table 242.

Table 242 — Logical block encryption key wrapping attribute format

Bit Byte	7	6	5	4	3	2	1	0
Logical block encryption key wrapping attribute descriptor list								
0								
...								
Logical block encryption key wrapping attribute descriptor [first] (see table 243)								
⋮								
...								
n								
Logical block encryption key wrapping attribute descriptor [last] (see table 243)								

There may be zero or more logical block encryption key wrapping attribute descriptors in any encryption management attribute descriptor list.

The logical block encryption key wrapping attribute descriptor format is specified in table 243.

Table 243 — Logical block encryption key wrapping attribute descriptor format

Bit Byte	7	6	5	4	3	2	1	0
0	CRIT	(MSB)						
1								(LSB)
2	(MSB)							
3								(LSB)
4								
...								
n								

The LOGICAL BLOCK ENCRYPTION KEY WRAPPING ATTRIBUTE TYPE field is specified in table 244.

Table 244 — LOGICAL BLOCK ENCRYPTION KEY WRAPPING ATTRIBUTE TYPE field

Code	Description	Reference
0000h	Reserved	
0001h	KEKS	8.7.4.4.4.2
0002h to 7FEFh	Reserved	
7FF0h to FFFFh	Vendor specific	

8.7.4.4.4.2 KEKS

If the LOGICAL BLOCK ENCRYPTION KEY WRAPPING ATTRIBUTE TYPE field is set to 0001h (i.e., KEKS), then the LOGICAL BLOCK ENCRYPTION KEY WRAPPING ATTRIBUTE field is the key encrypting key selector (see 3.1.40).

Annex A (Informative)

Security environment

A.1 Security environment overview

Figure A.1 shows a simplified depiction of a typical security deployment, in which a large number of device servers are being accessed by a set of logical block management servers, each set controlled by a master logical block management server. The addition of encryption in the device servers introduces a centralized key manager that may be part of the master logical block management server. Centralized key managers are used for the same reason that master logical block management servers are used, so that relatively few key managers are installed in an environment with a relatively large number of device servers and logical block management servers.

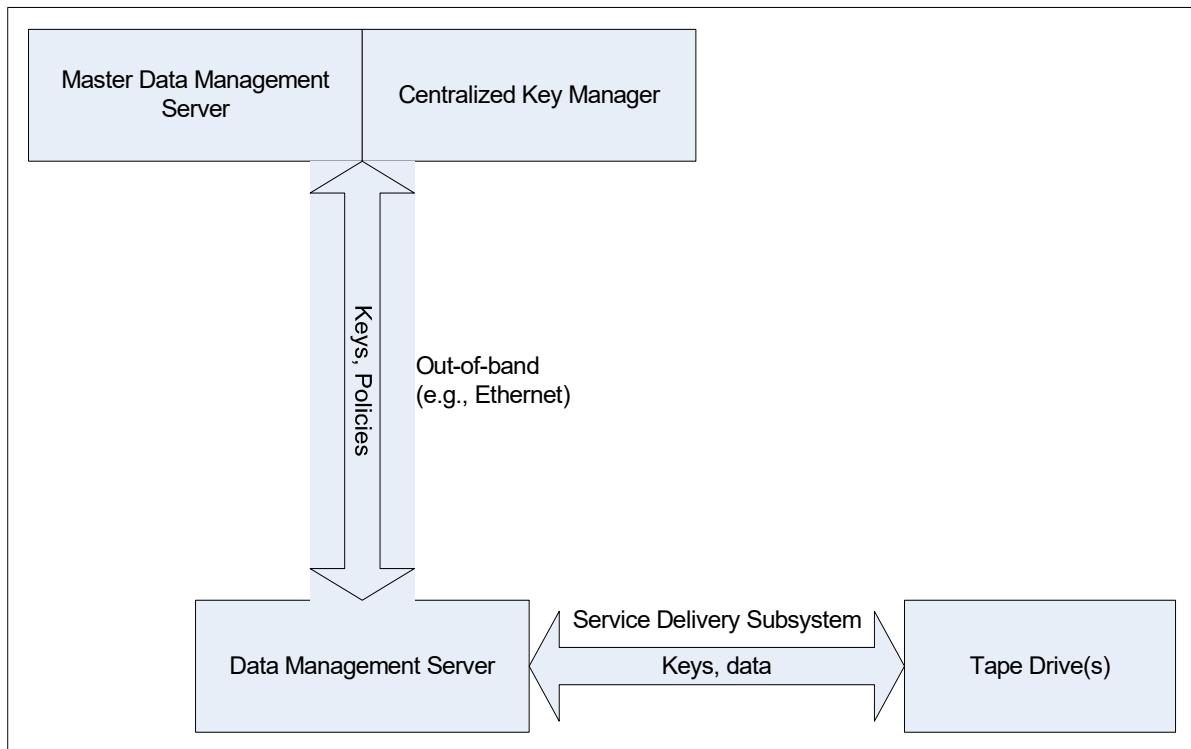


Figure A.1 — Simple security deployment environment

Communication between the centralized key manager, master logical block management server, and the logical block management server is beyond the scope of this standard, while communication between the logical block management server and device server is over a SCSI service delivery subsystem. The following additional assumptions are in place:

- a) the communications path between the centralized key manager and the logical block management server is easily made sufficiently secure (e.g., via SSL with a 128 bit strong cipher suite is probably sufficient for most link based attacks);
- b) the centralized key manager is able to reliably associate keys with logical block management servers;
- c) only the logical block management server is able to associate keys with logical block attributes;
- d) the device server has a minimal cryptographic module, able to encrypt and decrypt logical block, and to perform simple key load/key agreement operations;

- e) attacks against key disclosure or substitution for keys already stored within the device server, or centralized key manager are out of scope; and
- f) the logical block management server, in most cases, has only a simple SSL module implemented in the underlying OS.

As a result of the above, the assumption is threats are against either the logical block management server or the service delivery subsystem.

A.2 Security environment threats

Table A.1 specifies a list of security environment threats and methods to mitigate the threats.

Table A.1 — Security environment threats

Threat	Against logical block management server	Against service delivery subsystem
Subverting key load protocol via CDB modification (e.g., turning off logical block encryption).	Harden the OS of the logical block management server.	Secure the channel from the logical block management server to the device server.
Key disclosure (passive).	<ul style="list-style-type: none"> a) encrypt keys prior to reaching the logical block management server; or b) prevent information leakage by hardening the OS of the logical block management server to prevent caching or core dumps, etc. 	<ul style="list-style-type: none"> a) only send encrypted keys along service delivery subsystem network; or b) secure the channel from the logical block management server to the device server.
Injecting an attacker's key.	<ul style="list-style-type: none"> a) encrypt and sign keys at the centralized key manager prior to shipment to the logical block management server; or b) harden the OS of the logical block management server. 	<ul style="list-style-type: none"> a) only send signed keys along the T10 network; or b) secure the channel from the logical block management server to the device server.
Breaking association of keys to logical block via key replay.	Harden the OS of the logical block management server.	<ul style="list-style-type: none"> a) rotate signing keys frequently; or b) have a secure channel from the logical block management server to the device server.

Annex B
(Informative)

Example keyless copy operation flowchart

B.1 Example keyless copy operation flowchart overview

Figure B.1 and figure B.2 show a simplified flowchart of an example keyless copy operation. The key to the abbreviations in the figures is shown in table B.1.

Table B.1 — Example keyless copy figure codes

Code	Description
AC	Application Client
DM	DECRYPTION MODE field
EM	ENCRYPTION MODE field
ERP	Error recovery procedure
SP	SECURITY PROTOCOL field
SPIN	SECURITY PROTOCOL IN command
SPOUT	SECURITY PROTOCOL OUT command
SPS	SECURITY PROTOCOL SPECIFIC field
CC	CHECK CONDITION
EOP/MD	END-OF-PARTITION/MEDIUM DETECTED
RE	RECOVERED ERROR
NS	NO SENSE
SK	Sense Key
ASC/Q	Additional sense code
FM	Filemark
FMD	FILEMARK DETECTED
DP	DATA PROTECT
NASI	NO ADDITIONAL SENSE INFORMATION

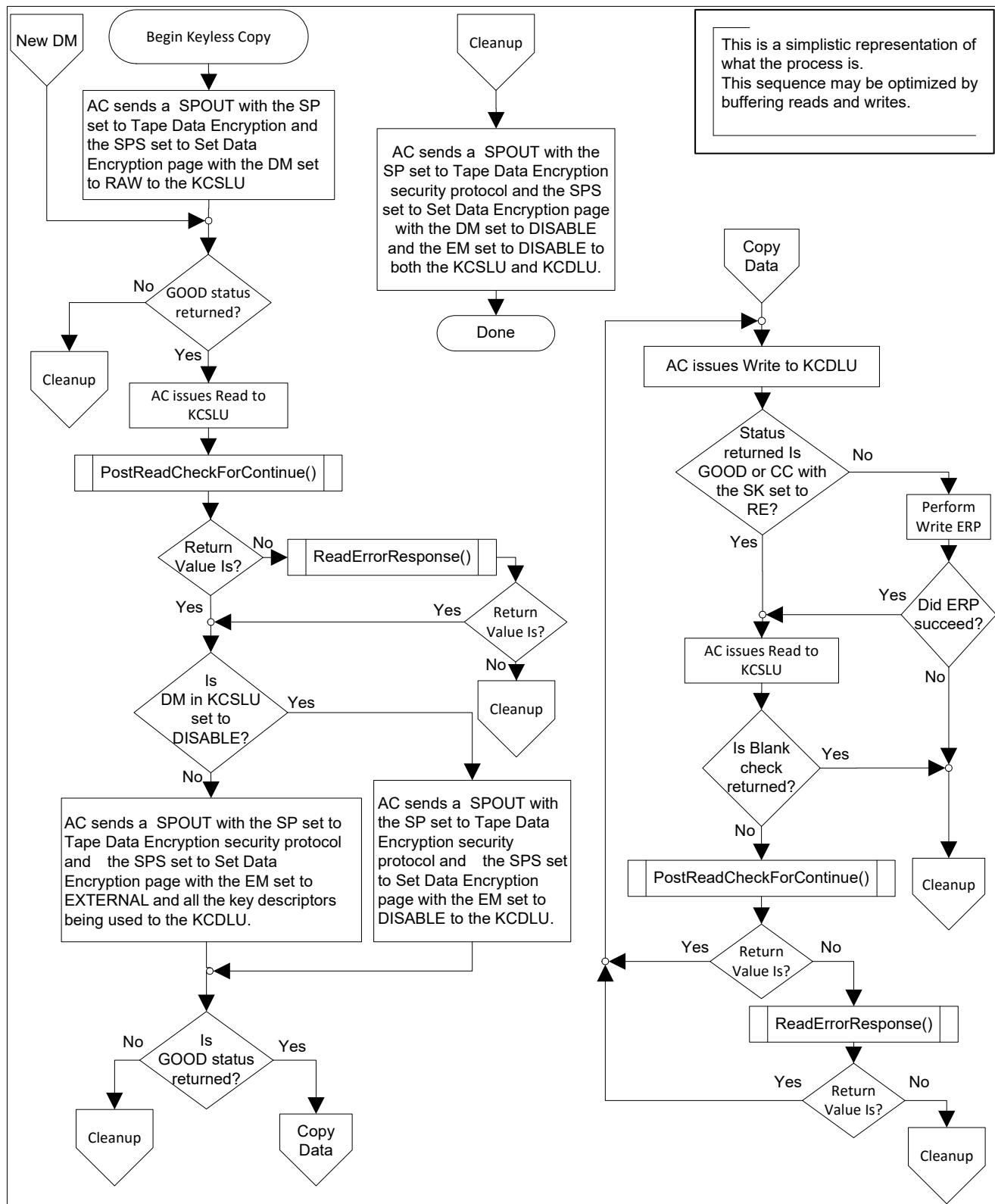


Figure B.1 — Example keyless copy operation flowchart part 1

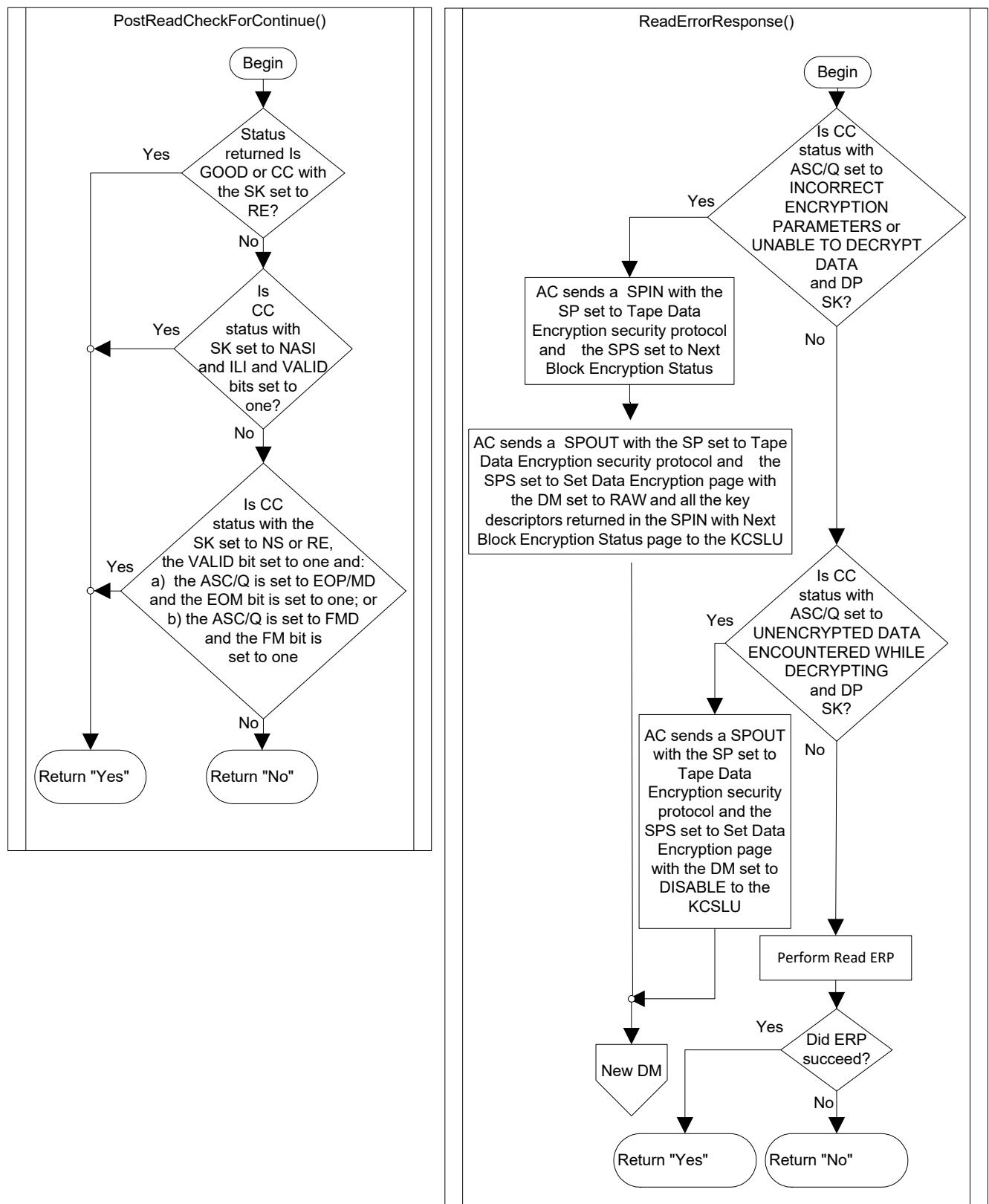


Figure B.2 — Example keyless copy operation flowchart part 2

Annex C

(Informative)

Example application client use of volume coherency

An application client may need to be able to determine if all logical objects on a recording volume are coherent with the last time an application client wrote to this recording volume. The VOLUME COHERENCY INFORMATION attribute (see 4.2.25) of MAM is provided for an application client to collect and save information necessary for this determination.

The VOLUME COHERENCY INFORMATION attribute for each partition is written to MAM by the application client after it has completed a sequence of volume accesses that included writing to the volume (e.g., a job). The VOLUME COHERENCY INFORMATION attribute contains references to a volume coherency set that the application client has written to logical objects on a partition. An application client should not create a VOLUME COHERENCY INFORMATION attribute unless it has written a volume coherency set to that partition. The volume coherency set shall include a volume coherency count. The application client shall maintain one volume coherency count for an entire recording volume and shall monotonically increase the volume coherency count if the state of the volume coherency set changes (e.g., writing identical volume coherency sets on each partition does not force a change of volume coherency count). If the application client writes the VOLUME COHERENCY INFORMATION attribute to MAM for a specific partition, then the VOLUME CHANGE REFERENCE VALUE field of the VOLUME COHERENCY INFORMATION attribute for a partition shall contain the value returned in the ATTRIBUTE VALUE field of the VOLUME CHANGE REFERENCE attribute after the last volume coherency set was written to the recording volume. The VOLUME COHERENCY COUNT field of the VOLUME COHERENCY INFORMATION attribute shall contain the volume coherency count that was written to the last volume coherency set written to that partition. The VOLUME COHERENCY SET IDENTIFIER field of the VOLUME COHERENCY INFORMATION attribute for a partition contains the logical object identifier of the first byte of the last volume coherency set written to that partition. The APPLICATION CLIENT SPECIFIC INFORMATION field of the VOLUME COHERENCY INFORMATION attribute for a partition contains information the application client binds with the coherency set referenced by the VOLUME COHERENCY SET IDENTIFIER field.

NOTE 69 - The application client should guarantee that no other application client updates the logical objects on the recording volume between the time it completes writing and the time it updates the MAM parameter (e.g., use reservations).

If an application client reads the VOLUME COHERENCY INFORMATION attribute for a partition (e.g., during mounting of a recording volume), then the application client may verify that the volume coherency set written in a partition has not changed since the VOLUME COHERENCY INFORMATION attribute was written by comparing the value in the VOLUME CHANGE REFERENCE VALUE field with the value returned in the ATTRIBUTE VALUE field of the VOLUME CHANGE REFERENCE attribute. If the values match, then the volume coherency set written in that partition is unchanged.

To find the most recently written volume coherency set, the application client searches the VOLUME COHERENCY INFORMATION attribute for each partition for which the volume coherency set is unchanged and finds the largest value in the VOLUME COHERENCY COUNT field. The application client then verifies the largest value in the VOLUME COHERENCY COUNT field with the volume coherency count stored in the volume coherency set beginning at the logical object specified by the VOLUME COHERENCY SET IDENTIFIER field. If this matches, then this is the volume coherency set that was most recently written.

The APPLICATION CLIENT SPECIFIC INFORMATION field may also be used by the application client as part of this coherency check. If the information verifies for a partition, then the recording volume is coherent with the last access by this application. If the information does not verify for a partition, then the recording volume is not coherent with the last access by this application.

Annex D (Informative)

Sense data information for error conditions

D.1 Sense data error indications for read and write operations

Table D.1 specifies the sense data error indications for read and write operations.

Table D.1 — Sense data error indications for read and write operations (part 1 of 2)

Condition	Command	Sense key	Additional sense code	EOM	FM
Filemark encountered. ^a	READ(16), READ REVERSE(16), READ(6), READ REVERSE(6), SPACE(6) ^a , SPACE(16) ^a	NO SENSE or RECOVERED ERROR	FILEMARK DETECTED	0	1
End-of-data encountered before EW. ^d	READ(16), READ(6), SPACE(6), SPACE(16)	BLANK CHECK	END-OF-DATA DETECTED	0	X
End-of-data encountered between EW and end-of-partition/medium. ^d	READ(16), READ(6), SPACE(6), SPACE(16)	BLANK CHECK	END-OF-DATA DETECTED	REW ^b	X
EW encountered and REW ^b is set to one in the Device Configuration mode page (see 8.5.3).	READ(16), READ REVERSE(16), READ(6), READ REVERSE(6), SPACE(6), SPACE(16)	NO SENSE or RECOVERED ERROR	END-OF- PARTITION/MEDIUM DETECTED	1	X

Key:

X = not specified.

- a) The application client is only notified of filemarks on SPACE commands if processing a SPACE(6) or SPACE(16) command with the CODE field set to logical blocks (i.e., 0000b).
- b) These values represent the current settings of the corresponding fields in the Device Configuration mode page (see 8.5.3).
- c) The information field is only valid if the VALID bit is set to one in the sense data.
- d) End-of-data is not returned after end-of-partition.
- e) At least one logical block was unable to be written. A READ POSITION command may be used to determine which logical blocks were successfully written. A RECOVER BUFFERED DATA command, if supported by the device server, may be used to read the unwritten logical block from the logical object buffer.
- f) The READ POSITION command should be used to determine the amount of data transferred.
- g) It is only possible to encounter BOP on a SPACE command if the COUNT field is a negative value.
- h) This condition occurs one or more times.
- i) This row is only valid if the SEW bit is set to one in the Device Configuration mode page (see 8.5.3).

Table D.1 — Sense data error indications for read and write operations (part 2 of 2)

Condition	Command	Sense key	Additional sense code	EOM	FM
EW encountered and REW ^b is set to zero in the Device Configuration mode page (see 8.5.3).	READ(16), READ REVERSE(16), READ(6), READ REVERSE(6), SPACE(6), SPACE(16)	N/A	N/A	X	X
End-of-partition/medium encountered before end-of-data.	READ(16), READ(6), SPACE(6), SPACE(16)	MEDIA ERROR	Dependent on the operation	1	X
Beginning-of-partition/medium encountered.	READ REVERSE(16) READ REVERSE(6) SPACE(6) ^g , SPACE(16) ^g	NO SENSE or RECOVERED ERROR	BEGINNING-OF- PARTITION/MEDIUM DETECTED	1	X
EW encountered and data successfully written. ^h	WRITE(6), WRITE(16), WRITE FILEMARKS(6), WRITE FILEMARKS(16)	NO SENSE or RECOVERED ERROR	END-OF- PARTITION/MEDIUM DETECTED	1	X
End-of-partition/medium encountered without successfully writing data. ^e	WRITE(6), WRITE(16), WRITE FILEMARKS(6), WRITE FILEMARKS(16)	VOLUME OVERFLOW	END-OF- PARTITION/MEDIUM DETECTED	1	X

Key:
X = not specified.

- a) The application client is only notified of filemarks on SPACE commands if processing a SPACE(6) or SPACE(16) command with the CODE field set to logical blocks (i.e., 0000b).
- b) These values represent the current settings of the corresponding fields in the Device Configuration mode page (see 8.5.3).
- c) The information field is only valid if the VALID bit is set to one in the sense data.
- d) End-of-data is not returned after end-of-partition.
- e) At least one logical block was unable to be written. A READ POSITION command may be used to determine which logical blocks were successfully written. A RECOVER BUFFERED DATA command, if supported by the device server, may be used to read the unwritten logical block from the logical object buffer.
- f) The READ POSITION command should be used to determine the amount of data transferred.
- g) It is only possible to encounter BOP on a SPACE command if the COUNT field is a negative value.
- h) This condition occurs one or more times.
- i) This row is only valid if the SEW bit is set to one in the Device Configuration mode page (see 8.5.3).

D.2 INFORMATION field and position for read and write operations

Table D.2 specifies the INFORMATION field and position for read and write operations.

Table D.2 — INFORMATION field and position for read and write operations (part 1 of 3)

Condition	Operation	VALID	INFORMATION ^c		Position
			FIXED = 1 ^f	FIXED = 0	
Filemark encountered. ^a	READ(16), READ(6)	1	Transfer length minus length actually read	Transfer length	On EOP side of filemark.
	READ REVERSE(16), READ REVERSE(6)	1	Transfer length minus length actually read	Transfer length	On BOP side of filemark.
	SPACE(6) ^a , SPACE(16) ^a	1	Count of logical blocks traversed		After filemark.
End-of-data encountered before EW. ^d	READ(16), READ(6), SPACE(6), SPACE(16)	1	Transfer length minus length actually read	Transfer length	On EOP side of last logical block (at EOD).
End-of-data encountered between EW and end-of-partition/medium. ^d	READ(16), READ(6), SPACE(6), SPACE(16)	1	Transfer length minus length actually read	Transfer length	On EOP side of last logical block (at EOD).

Key:

X = not specified.

- a) The application client is only notified of filemarks on SPACE commands if processing a SPACE(6) or SPACE(16) command with the CODE field set to logical blocks (i.e., 0000b).
- b) These values represent the current settings of the corresponding fields in the Device Configuration mode page (see 8.5.3).
- c) The information field is only valid if the VALID bit is set to one in the sense data.
- d) End-of-data is not returned after end-of-partition.
- e) At least one logical block was unable to be written. A READ POSITION command may be used to determine which logical blocks were successfully written. A RECOVER BUFFERED DATA command, if supported by the device server, may be used to read the unwritten logical block from the logical object buffer.
- f) The READ POSITION command should be used to determine the amount of data transferred.
- g) It is only possible to encounter BOP on a SPACE command if the COUNT field is a negative value.
- h) This condition occurs one or more times.
- i) This row is only valid if the SEW bit is set to one in the Device Configuration mode page (see 8.5.3).

Table D.2 — INFORMATION field and position for read and write operations (part 2 of 3)

Condition	Operation	VALID	INFORMATION ^c		Position
			FIXED = 1 ^f	FIXED = 0	
EW encountered and REW ^b is set to one in the Device Configuration mode page (see 8.5.3).	READ(16), READ REVERSE(16), READ(6), READ REVERSE(6), SPACE(6), SPACE(16)	1	Transfer length minus length actually transferred	Transfer length minus actual block length	After the last logical object transferred.
EW encountered and REW ^b is set to zero in the Device Configuration mode page (see 8.5.3).	READ(16), READ REVERSE(16), READ(6), READ REVERSE(6), SPACE(6), SPACE(16)	N/A	N/A	N/A	After last requested logical object.
End-of-partition/medium encountered before end-of-data.	READ(16), READ(6), SPACE(6), SPACE(16)	1	Transfer length minus length actually transferred	Transfer length	The medium position following this condition is not defined.

Key:
X = not specified.

- a) The application client is only notified of filemarks on SPACE commands if processing a SPACE(6) or SPACE(16) command with the CODE field set to logical blocks (i.e., 0000b).
- b) These values represent the current settings of the corresponding fields in the Device Configuration mode page (see 8.5.3).
- c) The information field is only valid if the VALID bit is set to one in the sense data.
- d) End-of-data is not returned after end-of-partition.
- e) At least one logical block was unable to be written. A READ POSITION command may be used to determine which logical blocks were successfully written. A RECOVER BUFFERED DATA command, if supported by the device server, may be used to read the unwritten logical block from the logical object buffer.
- f) The READ POSITION command should be used to determine the amount of data transferred.
- g) It is only possible to encounter BOP on a SPACE command if the COUNT field is a negative value.
- h) This condition occurs one or more times.
- i) This row is only valid if the SEW bit is set to one in the Device Configuration mode page (see 8.5.3).

Table D.2 — INFORMATION field and position for read and write operations (part 3 of 3)

Condition	Operation	VALID	INFORMATION ^c		Position
			FIXED = 1 ^f	FIXED = 0	
Beginning-of-partition/medium encountered.	READ REVERSE(16) READ REVERSE(6) SPACE(6) ^g , SPACE(16) ^g	1	Transfer length minus length actually transferred	Transfer length	The medium position following this condition is not defined.
EW encountered and data successfully written. ^{h,i}	WRITE(6), WRITE(16), WRITE FILEMARKS(6), WRITE FILEMARKS(16)	1	Transfer length minus length actually transferred to medium	Transfer length	After logical block(s) written.
End-of-partition/medium encountered without successfully writing data. ^{e,i}	WRITE(6), WRITE(16), WRITE FILEMARKS(6), WRITE FILEMARKS(16)	1	Transfer length minus length actually written	Transfer length	At end-of-partition/ medium.

Key:

X = not specified.

- a) The application client is only notified of filemarks on SPACE commands if processing a SPACE(6) or SPACE(16) command with the CODE field set to logical blocks (i.e., 0000b).
- b) These values represent the current settings of the corresponding fields in the Device Configuration mode page (see 8.5.3).
- c) The information field is only valid if the VALID bit is set to one in the sense data.
- d) End-of-data is not returned after end-of-partition.
- e) At least one logical block was unable to be written. A READ POSITION command may be used to determine which logical blocks were successfully written. A RECOVER BUFFERED DATA command, if supported by the device server, may be used to read the unwritten logical block from the logical object buffer.
- f) The READ POSITION command should be used to determine the amount of data transferred.
- g) It is only possible to encounter BOP on a SPACE command if the COUNT field is a negative value.
- h) This condition occurs one or more times.
- i) This row is only valid if the SEW bit is set to one in the Device Configuration mode page (see 8.5.3).

D.3 Summary of length error conditions on read type commands

Table D.3 provides a summary of length error conditions on read type commands.

Table D.3 — Summary of length error conditions on read type commands

Error condition	Fixed	SILI	BLOCK LENGTH field in mode parameter header	Sense error	ILI	Information	Position ^a
Underlength	0	0	X	NO SENSE / NO ADDITIONAL SENSE	1	Requested length minus block size	After logical block
	0	1	X	None	0	N/A	After logical block
	1	0	Non- zero	NO SENSE / NO ADDITIONAL SENSE	1	Requested length minus blocks read not including incorrect block	After incorrect logical block
Overlength	0	0	X	NO SENSE / NO ADDITIONAL SENSE	1	Requested length minus block size	After logical block
	0	1	0	None	0	N/A	After logical block
	0	1	Non- zero	NO SENSE / NO ADDITIONAL SENSE	-	Requested length minus block size	After logical block
	1	0	Non- zero	NO SENSE / NO ADDITIONAL SENSE	1	Requested length minus blocks read not including incorrect block	After incorrect logical block
Key: X = don't care a) After logical block means on the EOP side of the logical block for a READ(6) or READ(16) command and on the BOP side of the logical block for a READ REVERSE(6) or READ REVERSE(16) command.							

Annex E (Informative)

Logical block protection CRC information

E.1 Reed-Solomon CRC

E.1.1 Reed-Solomon CRC Polynomial

An SSC device may support the Reed Solomon CRC algorithm defined in ECMA-319:

- a) as a format specific symbol written to the medium with each logical block; or
- b) as protection information associated with each logical block transferred between the SCSI device and an application client.

The CRC bytes are Reed-Solomon (N, N-4) codes over GF (256).

A calculation in GF (256) is defined by $P(x) = x^8 + x^4 + x^3 + x^2 + 1$

$$\alpha = (0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 1 \ 0)$$

$$G(x) = x^4 + \alpha^{201}x^3 + \alpha^{246}x^2 + \alpha^{201}x + 1$$

E.1.2 Sample C source code for generating the Reed-Solomon CRC

```
/*
** INPUTS: crc - initial crc (0 for fresh) (i.e., seed)
**          cnt - the number of data bytes to compute CRC for
**          start - the starting address of the data bytes (e.g., data buffer)
** OUTPUTS: UINT32 - crc in big endian (MSB is first byte)
*/
UINT32 GenerateRSCRC(UINT32 crc, UINT32 cnt, const void *start)
{
    static const UINT32 crcTable[256]=
    {
        0x00000000, 0x38CF3801, 0x70837002, 0x484C4803, 0xE01BE004, 0xD8D4D805,
        0x90989006, 0xA857A807, 0xDD36DD08, 0xE5F9E509, 0xADB5AD0A, 0x957A950B,
        0x3D2D3D0C, 0x05E2050D, 0x4DAE4D0E, 0x7561750F, 0xA76CA710, 0x9FA39F11,
        0xD7EF712, 0xEF20EF13, 0x47774714, 0x7FB87F15, 0x37F43716, 0x0F3B0F17,
        0x7A5A7A18, 0x42954219, 0x0AD90A1A, 0x3216321B, 0x9A419A1C, 0xA28EA21D,
        0xEAC2EA1E, 0xD20DD21F, 0x53D85320, 0x6B176B21, 0x235B2322, 0x1B941B23,
        0xB3C3B324, 0x8B0C8B25, 0xC340C326, 0xFB8FFB27, 0x8EE8E28, 0xB621B629,
        0xFE6DFE2A, 0xC6A2C62B, 0x6EF56E2C, 0x563A562D, 0x1E761E2E, 0x26B9262F,
        0xF4B4F430, 0xCC7BCC31, 0x84378432, 0xBCF8BC33, 0x14AF1434, 0x2C602C35,
        0x642C6436, 0x5CE35C37, 0x29822938, 0x114D1139, 0x5901593A, 0x61CE613B,
        0xC999C93C, 0xF156F13D, 0xB91AB93E, 0x81D5813F, 0xA6ADA640, 0x9E629E41,
        0xD62ED642, 0xEEE1EE43, 0x46B64644, 0x7E797E45, 0x36353646, 0x0EFA0E47,
        0x7B9B7B48, 0x43544349, 0x0B180B4A, 0x33D7334B, 0x9B809B4C, 0xA34FA34D,
        0xEB03EB4E, 0xD3CCD34F, 0x01C10150, 0x390E3951, 0x71427152, 0x498D4953,
        0xE1DAE154, 0xD915D955, 0x91599156, 0xA996A957, 0xDC7DC58, 0xE438E459,
        0xAC74AC5A, 0x94BB945B, 0x3CEC3C5C, 0x0423045D, 0x4C6F4C5E, 0x74A0745F,
        0xF575F560, 0xCDBACD61, 0x85F68562, 0xBD39BD63, 0x156E1564, 0x2DA12D65,
```

```

0x65ED6566, 0x5D225D67, 0x28432868, 0x108C1069, 0x58C0586A, 0x600F606B,
0xC858C86C, 0xF097F06D, 0xB8DBB86E, 0x8014806F, 0x52195270, 0x6AD66A71,
0x229A2272, 0x1A551A73, 0xB202B274, 0x8ACD8A75, 0xC281C276, 0xFA4EFA77,
0x8F2F8F78, 0xB7E0B779, 0xFFACFF7A, 0xC763C77B, 0x6F346F7C, 0x57FB577D,
0x1FB71F7E, 0x2778277F, 0x51475180, 0x69886981, 0x21C42182, 0x190B1983,
0xB15CB184, 0x89938985, 0xC1DFC186, 0xF910F987, 0x8C718C88, 0xB4BEB489,
0xFCFC2FC8A, 0xC43DC48B, 0x6C6A6C8C, 0x54A5548D, 0x1CE91C8E, 0x2426248F,
0xF62BF690, 0xCEE4CE91, 0x86A88692, 0xBE67BE93, 0x16301694, 0x2EFF2E95,
0x66B36696, 0x5E7C5E97, 0x2B1D2B98, 0x13D21399, 0x5B9E5B9A, 0x6351639B,
0xCB06CB9C, 0xF3C9F39D, 0xBB85BB9E, 0x834A839F, 0x029F02A0, 0x3A503AA1,
0x721C72A2, 0x4AD34AA3, 0xE284E2A4, 0xDA44BDAA5, 0x920792A6, 0xAAC8AAA7,
0xDFA9DFA8, 0xE766E7A9, 0xAF2AAFAA, 0x97E597AB, 0x3FB23FAC, 0x077D07AD,
0x4F314FAE, 0x77FE77AF, 0xA5F3A5B0, 0x9D3C9DB1, 0xD570D5B2, 0xEDBFEDB3,
0x45E845B4, 0x7D277DB5, 0x356B35B6, 0x0DA40DB7, 0x78C578B8, 0x400A40B9,
0x084608BA, 0x308930BB, 0x98DE98BC, 0xA011A0BD, 0xE85DE8BE, 0xD092D0BF,
0xF7EAF7C0, 0xCF25CFC1, 0x876987C2, 0xBFA6BFC3, 0x17F117C4, 0x2F3E2FC5,
0x677267C6, 0x5FBD5FC7, 0x2ADC2AC8, 0x121312C9, 0x5A5F5ACA, 0x629062CB,
0xCAC7CACC, 0xF208F2CD, 0xBA44BACE, 0x828B82CF, 0x508650D0, 0x684968D1,
0x200520D2, 0x18CA18D3, 0xB09DB0D4, 0x885288D5, 0xC01EC0D6, 0xF8D1F8D7,
0x8DB08DD8, 0xB57FB5D9, 0xFD33FDDA, 0xC5FCC5DB, 0x6DAB6DDC, 0x556455DD,
0x1D281DDE, 0x25E725DF, 0xA432A4E0, 0x9CFD9CE1, 0xD4B1D4E2, 0xEC7EECE3,
0x442944E4, 0x7CE67CE5, 0x34AA34E6, 0x0C650CE7, 0x790479E8, 0x41CB41E9,
0x098709EA, 0x314831EB, 0x991F99EC, 0xA1D0A1ED, 0xE99CE9EE, 0xD153D1EF,
0x035E03F0, 0x3B913BF1, 0x73DD73F2, 0x4B124BF3, 0xE345E3F4, 0xDB8ADBF5,
0x93C693F6, 0xAB09ABF7, 0xDE68DEF8, 0xE6A7E6F9, 0xAEEBAEFA, 0x962496FB,
0x3E733EFC, 0x06BC06FD, 0x4EF04EFE, 0x763F76FF
};

UINT32 i;
const UINT8* d = start;
for ( i=0; i<cnt; i++ )
{
    crc = (crc << 8) ^ crcTable[*d ^ (crc >> 24)];
    d++;
}
return crc;
}

```

E.1.3 Sample C source code for generating and appending the Reed-Solomon CRC

```

/*
** ABSTRACT: function to compute and append LBP CRC to a data block
** INPUTS:   blkbuf - starting address of the data block to protect
**           blklen - length of block to protect (NOT including CRC)
** OUTPUTS:  UINT32 - length of protected block (to write) including LBP CRC
*/
UINT32 BlockProtectRSCRC(UINT8 *blkbuf, UINT32 blklen)
{
    UINT32 crc = GenerateRSCRC(0x00000000, blklen, blkbuf);

    if (blklen == 0) {
        return 0; //no such thing as a zero length block in SSC (write NOP)
    }
}

```

```

    }

//append CRC in proper byte order (regardless of system endian-ness)
blkbuf[blklen+0] = (crc >> 24) & 0xFF;
blkbuf[blklen+1] = (crc >> 16) & 0xFF;
blkbuf[blklen+2] = (crc >> 8) & 0xFF;
blkbuf[blklen+3] = (crc >> 0) & 0xFF;

return (blklen+4); //size of block to be written includes CRC
}

```

E.1.4 Sample C source code for validating a logical block with a Reed-Solomon CRC

```

/*
** ABSTRACT: function to verify block with LBP CRC
** INPUTS:   blkbuf - starting address of the data block to protect
**           blklen - length of block to verify (INCLUDING CRC)
** OUTPUTS:  UINT32 - length of block w/o CRC (0 if verify failed)
*/

```

```

UINT32 BlockVerifyRSCRC(const UINT8 *blkbuf, UINT32 blklen)
{
    if (blklen <= 4) {
        return 0; //block is too small to be valid, cannot check CRC
    }
    blklen -= 4; //user data portion does not include CRC

#if 1 //method 1: calculate CRC on data only and compare against CRC from block
{
    UINT32 crccmp = GenerateRSCRC(0x00000000, blklen, blkbuf);
    UINT32 crcblk;

    //this matches the append method in the function above
    crcblk = (blkbuf[blklen+0] << 24) |
              (blkbuf[blklen+1] << 16) |
              (blkbuf[blklen+2] << 8) |
              (blkbuf[blklen+3] << 0);

    if (crccmp != crcblk)
        return 0; //block CRC is incorrect
    return(blklen);
}
#endif

#if 1 //method 2: calculate including CRC and check magic constant
{
    if (GenerateRSCRC(0x00000000, blklen+4, blkbuf) != 0x00000000) {
        return 0; //block CRC is incorrect (CRC did not neutralize)
    }
    return(blklen);
}
#endif
}

```

E.2 CRC32C (Castagnoli)

E.2.1 CRC32C Polynomial

An SSC device may support using the CRC32C CRC polynomial:

- a) as a format specific symbol written to the medium with each logical block; or
- b) as protection information associated with each logical block transferred between the SCSI device and an application client.

The polynomial is defined by the equation $x^{32}+x^{28}+x^{27}+x^{26}+x^{25}+x^{23}+x^{22}+x^{20}+x^{19}+x^{18}+x^{14}+x^{13}+x^{11}+x^{10}+x^9+x^8+x^6+1$ which is sometimes expressed as 0x1EDC6F41.

NOTE 70 The sample functions for CRC32C in this section use a bit swapped form, which is easier/faster for software implementation as it avoids the need to bit swap bytes of input and output.

NOTE 71 The CRC32C CRC is computed on the actual size of the logical block and appended immediately to the end of the logical block (e.g., the logical block is not padded to a 4-byte boundary).

E.2.2 Sample C source code for generating the CRC32C (Castagnoli) CRC

```
/*
** ABSTRACT: function to compute interim LBP CRC (bit-swapped method)
** INPUTS:    crc      - initial crc (0xFFFFFFFF for fresh)
**            cnt      - the number of data bytes to compute CRC for
**            start   - the starting address of the data bytes
** OUTPUTS:   UINT32  - [inverted] crc in big endian (LSB is first byte)
*/
UINT32 GenerateCRC32C(UINT32 crc, UINT32 cnt, const void *start)
{
    static const UINT32 crcTable[256]=
    { 0x00000000, 0xF26B8303, 0xE13B70F7, 0x1350F3F4, 0xC79A971F, 0x35F1141C,
      0x26A1E7E8, 0xD4CA64EB, 0x8AD958CF, 0x78B2DBCC, 0x6BE22838, 0x9989AB3B,
      0x4D43CFD0, 0xBF284CD3, 0xAC78BF27, 0x5E133C24, 0x105EC76F, 0xE235446C,
      0xF165B798, 0x30E349B, 0xD7C45070, 0x25AFD373, 0x36FF2087, 0xC494A384,
      0x9A879FA0, 0x68EC1CA3, 0x7BCCEF57, 0x89D76C54, 0x5D1D08BF, 0xAF768BBC,
      0xBC267848, 0x4E4DFB4B, 0x20BD8EDE, 0xD2D60DDD, 0xC186FE29, 0x33ED7D2A,
      0xE72719C1, 0x154C9AC2, 0x061C6936, 0xF477EA35, 0xAA64D611, 0x580F5512,
      0x4B5FA6E6, 0xB93425E5, 0x6DFE410E, 0x9F95C20D, 0x8CC531F9, 0x7EAEB2FA,
      0x30E349B1, 0xC288CAB2, 0xD1D83946, 0x23B3BA45, 0xF779DEAE, 0x05125DAD,
      0x1642AE59, 0xE4292D5A, 0xBA3A117E, 0x4851927D, 0x5B016189, 0xA96AE28A,
      0x7DA08661, 0x8FCB0562, 0x9C9BF696, 0x6EF07595, 0x417B1DBC, 0xB3109EBF,
      0xA0406D4B, 0x522BEE48, 0x86E18AA3, 0x748A09A0, 0x67DAFA54, 0x95B17957,
      0xCBA24573, 0x39C9C670, 0x2A993584, 0xD8F2B687, 0x0C38D26C, 0xFE53516F,
      0xED03A29B, 0x1F682198, 0x5125DAD3, 0xA34E59D0, 0xB01EAA24, 0x42752927,
      0x96BF4DCC, 0x64D4CECF, 0x77843D3B, 0x85EFBE38, 0xDBFC821C, 0x2997011F,
      0x3AC7F2EB, 0xC8AC71E8, 0x1C661503, 0xEE0D9600, 0xFD5D65F4, 0x0F36E6F7,
      0x61C69362, 0x93AD1061, 0x80FDE395, 0x72966096, 0xA65C047D, 0x5437877E,
      0x4767748A, 0xB50CF789, 0xEB1FCBAD, 0x197448AE, 0xA24BB5A, 0xF84F3859,
      0x2C855CB2, 0xDEEEDFB1, 0xCDDE2C45, 0x3FD5AF46, 0x7198540D, 0x83F3D70E,
      0x90A324FA, 0x62C8A7F9, 0xB602C312, 0x44694011, 0x5739B3E5, 0xA55230E6,
```

```

0xFB410CC2, 0x092A8FC1, 0x1A7A7C35, 0xE811FF36, 0x3CDB9BDD, 0xCEB018DE,
0xDDE0EB2A, 0x2F8B6829, 0x82F63B78, 0x709DB87B, 0x63CD4B8F, 0x91A6C88C,
0x456CAC67, 0xB7072F64, 0xA457DC90, 0x563C5F93, 0x082F63B7, 0xFA44E0B4,
0xE9141340, 0x1B7F9043, 0xCFB5F4A8, 0x3DDE77AB, 0x2E8E845F, 0xDCE5075C,
0x92A8FC17, 0x60C37F14, 0x73938CE0, 0x81F80FE3, 0x55326B08, 0xA759E80B,
0xB4091BFF, 0x466298FC, 0x1871A4D8, 0xEA1A27DB, 0xF94AD42F, 0x0B21572C,
0xDFEB33C7, 0x2D80B0C4, 0x3ED04330, 0xCCBBC033, 0xA24BB5A6, 0x502036A5,
0x4370C551, 0xB11B4652, 0x65D122B9, 0x97BAA1BA, 0x84EA524E, 0x7681D14D,
0x2892ED69, 0xDAF96E6A, 0xC9A99D9E, 0x3BC21E9D, 0xEF087A76, 0x1D63F975,
0x0E330A81, 0xFC588982, 0xB21572C9, 0x407EF1CA, 0x532E023E, 0xA145813D,
0x758FE5D6, 0x87E466D5, 0x94B49521, 0x66DF1622, 0x38CC2A06, 0xCAA7A905,
0xD9F75AF1, 0x2B9CD9F2, 0xFF56BD19, 0x0D3D3E1A, 0x1E6DCDEE, 0xEC064EED,
0xC38D26C4, 0x31E6A5C7, 0x22B65633, 0xD0DDD530, 0x0417B1DB, 0xF67C32D8,
0xE52CC12C, 0x1747422F, 0x49547E0B, 0xBB3FFD08, 0xA86F0EFC, 0x5A048DFF,
0x8ECEE914, 0x7CA56A17, 0x6FF599E3, 0x9D9E1AE0, 0xD3D3E1AB, 0x21B862A8,
0x32E8915C, 0xC083125F, 0x144976B4, 0xE622F5B7, 0xF5720643, 0x07198540,
0x590AB964, 0xAB613A67, 0xB831C993, 0x4A5A4A90, 0x9E902E7B, 0x6CFBAD78,
0x7FAB5E8C, 0x8DC0DD8F, 0xE330A81A, 0x115B2B19, 0x020BD8ED, 0xF0605BEE,
0x24AA3F05, 0xD6C1BC06, 0xC5914FF2, 0x37FACCF1, 0x69E9F0D5, 0x9B8273D6,
0x88D28022, 0x7AB90321, 0xAE7367CA, 0x5C18E4C9, 0x4F48173D, 0xBD23943E,
0xF36E6F75, 0x0105EC76, 0x12551F82, 0xE03E9C81, 0x34F4F86A, 0xC69F7B69,
0xD5CF889D, 0x27A40B9E, 0x79B737BA, 0x8BDCB4B9, 0x988C474D, 0x6AE7C44E,
0xBE2DA0A5, 0x4C4623A6, 0x5F16D052, 0xAD7D5351 } ;

    UINT32    i;
const UINT8*   d = start;

for ( i=0; i<cnt; i++ )
{
    crc = (crc >> 8) ^ crcTable[*d ^ (crc & 0xFF)];
    d++;
}
return crc;
}

```

E.2.3 Sample C source code for generating and appending the CRC32C CRC

```

/*
** ABSTRACT: function to compute and append LBP CRC to a data block
** INPUTS:    blkbuf - starting address of the data block to protect
**            blklen - length of block to protect (NOT including CRC)
** OUTPUTS:   UINT32 - length of protected block (to write) including LBP CRC
*/

```

```

UINT32 BlockProtectCRC32C(UINT8 *blkbuf, UINT32 blklen)
{
    UINT32 crc = ~GenerateCRC32C(0xFFFFFFFF, blklen, blkbuf); //note bit inversion

    if (blklen == 0) {
        return 0; //no such thing as a zero length block in SSC (write NOP)
    }

    //append CRC in proper byte order (regardless of system endian-ness)

```

```

blkbuf[blklen+0] = (crc >> 0) & 0xFF;
blkbuf[blklen+1] = (crc >> 8) & 0xFF;
blkbuf[blklen+2] = (crc >> 16) & 0xFF;
blkbuf[blklen+3] = (crc >> 24) & 0xFF;

return (blklen+4); //size of block to be written includes CRC
}

```

E.2.4 Sample C source code for validating a logical block with a CRC32C CRC

```

/*
** ABSTRACT: function to verify block with LBP CRC
** INPUTS:   blkbuf - starting address of the data block to protect
**           blklen - length of block to verify (INCLUDING CRC)
** OUTPUTS:  UINT32 - length of block w/o CRC (0 if verify failed)
*/

```

```

UINT32 BlockVerifyCRC32C(const UINT8 *blkbuf, UINT32 blklen)
{
    if (blklen <= 4)
        return 0; //block is too small to be valid, cannot check CRC

    blklen -= 4; //user data portion does not include CRC

#if 1 //method 1: calculate CRC on data only and compare against CRC from block
{
    UINT32 crccmp = ~GenerateRSCRC(0xFFFFFFFF, blklen, blkbuf); //note bit inversion
    UINT32 crcblk;

    //this matches the append method in the function above
    crcblk = (blkbuf[blklen+0] << 0) |
              (blkbuf[blklen+1] << 8) |
              (blkbuf[blklen+2] << 16) |
              (blkbuf[blklen+3] << 24);

    if (crccmp != crcblk)
        return 0; //block CRC is incorrect
    return(blklen);
}
#endif

#if 1 //method 2: calculate including CRC and check magic constant
{
    //NOTE: bit swapped magic constant is also bit+byte swapped
    //      0x1C2D19ED //"nominal" result including [inverted] CRC
    //      0xB798B438 //"swapped" result including [inverted] CRC
    //NOTE: magic constant check below does NOT need bit inversion
    if (GenerateCRC32C(0xFFFFFFFF, blklen+4, blkbuf) != 0xB798B438)
        return 0; //block CRC is incorrect (CRC did not neutralize)
    return(blklen);
}
#endif
}

```

E.2.5 Test vector for CRC32C CRC

An example of a 255 byte logical block containing incrementing data from 01h to FFh with CRC32C appended resulting in a 259 byte protected logical block:

Table E.1 — CRC32C test vector

Byte	0	1	2	3
0:	01h	02h	03h	04h
...				
(CRC begin) 252:	FDh	FEh	FFh	42h
(CRC end) 256:	B3h	F3h	31h	

Annex F (Informative)

Example recommended access order usage

An example of how an application client may use the recommended access order model (see 4.2.25) is to:

- 1) Read the UDS limits (see 7.10.1.1) to determine the number of supported UDS's (see 4.2.10);
- 2) Using methods outside the scope of this standard, compose a list of UDS's to be accessed;
- 3) Generate an RAO list (see 4.2.25) from the list of UDS's to be accessed using the GENERATE RECOMMENDED ACCESS ORDER command (see 7.3);
- 4) Read a portion of the RAO list using the RECEIVE RECOMMENDED ACCESS ORDER command (see 7.10) with the RAO LIST OFFSET field set to zero and the ALLOCATION LENGTH field set as appropriate for the Data-In Buffer;
- 5) Check the RAO PROCESS field and the STATUS field of the RAO list (see 7.10.1.2) to confirm that the RAO list was generated as expected;
- 6) Locate to the UDS described in the first user data segment descriptor of this portion of the RAO list;
- 7) Read that UDS;
- 8) Locate to the UDS described in the next user data segment descriptor of this portion of the RAO list;
- 9) Read that UDS;
- 10) Repeat steps (8) and (9) for all user data segment descriptors returned in this portion of the RAO list;
- 11) if the value in the RAO list RAO DESCRIPTOR LIST LENGTH field returned in step (4) is larger than the sum of the value in the RAO LIST OFFSET field and the size of the portion of the RAO list returned in response to the RAO command, then read another portion of the RAO list using the RECEIVE RECOMMENDED ACCESS ORDER command with the RAO LIST OFFSET field and the ALLOCATION LENGTH field set as appropriate for the Data-In Buffer;
- 12) Repeat steps (6) through (11) as necessary until all UDS's have been read.

An example of the resulting order after a GENERATE RECOMMENDED ACCESS ORDER command with the RAO PROCESS field set to 010b is shown in figure F.1. This simplified example assumes that the recording technology in use is serpentine recording with four bands (called track groups in 4.2.2.3) and 8 wraps (called tracks in 4.2.2.3) per data band for a total of 32 wraps. Different medium format layouts and hardware may have dynamic performance characteristics that affect the recommended access order in complex ways. Such attributes might include lateral head movement performance (coarse and fine), read speed, locate speed, medium acceleration and deceleration, etc.

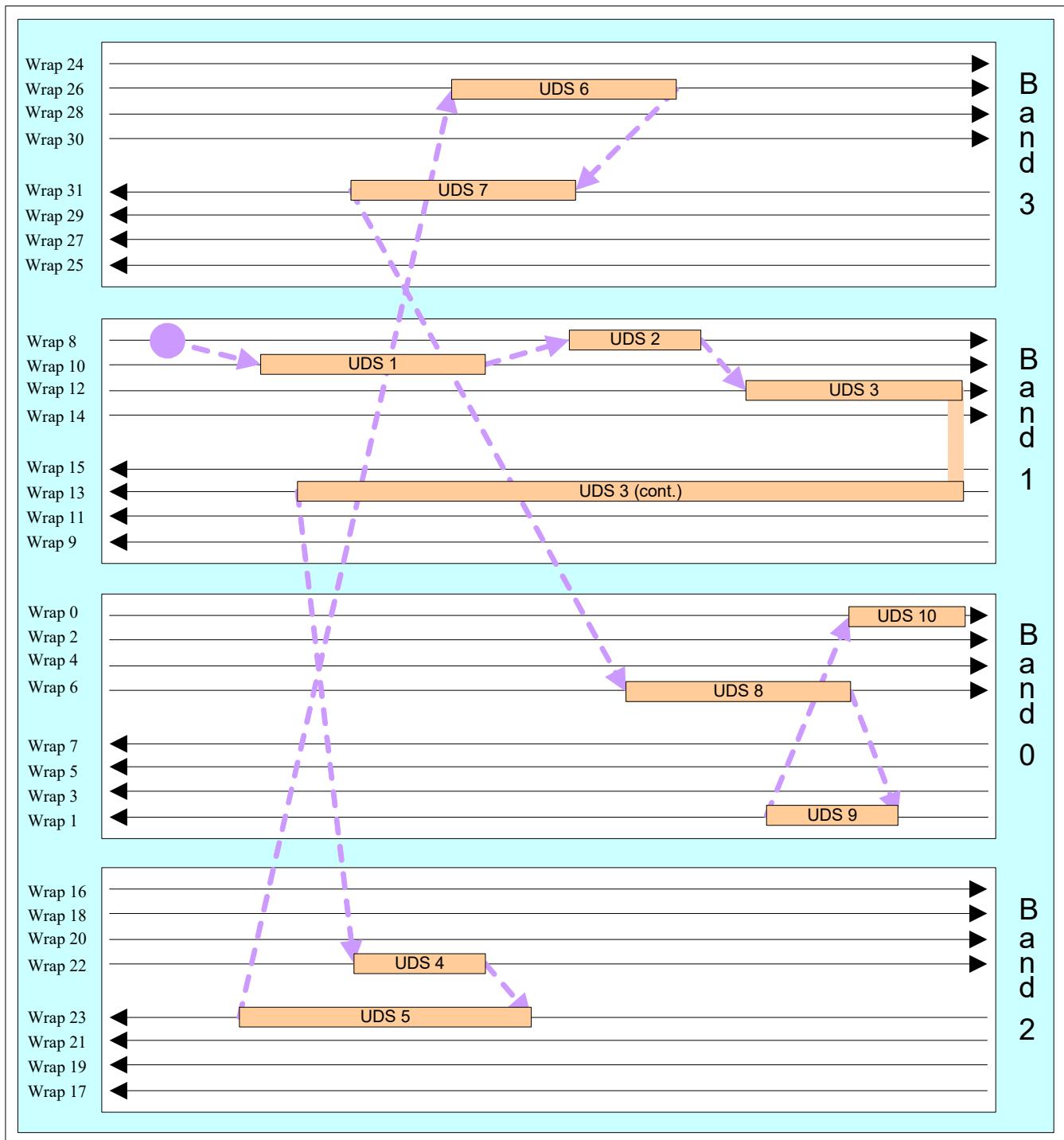


Figure F.1 — Example reordering of UDSs

The UDS numbers represent the sorted output order (i.e. the recommended access order). The sort is based on the current position (i.e., the light purple circle in the figure). The order of the resultant RAO list is independent of the order in the GENERATE RECOMMENDED ACCESS ORDER command parameter list.

Annex G (Informative)

Transfer length examples with and without logical block protection

G.1 Logical block and transfer lengths when logical block protection is supported

If logical block protection is supported, then each logical block stored to medium is recorded with protection information. The protection information is either transferred with the logical block or generated by the device server. In all cases, protection information is recorded with the logical block.

Figure G.1 shows an example of how the protected logical block (i.e., the logical block and its protection information) is used to determine if a read is overlength or underlength. This example is explained after the figure.

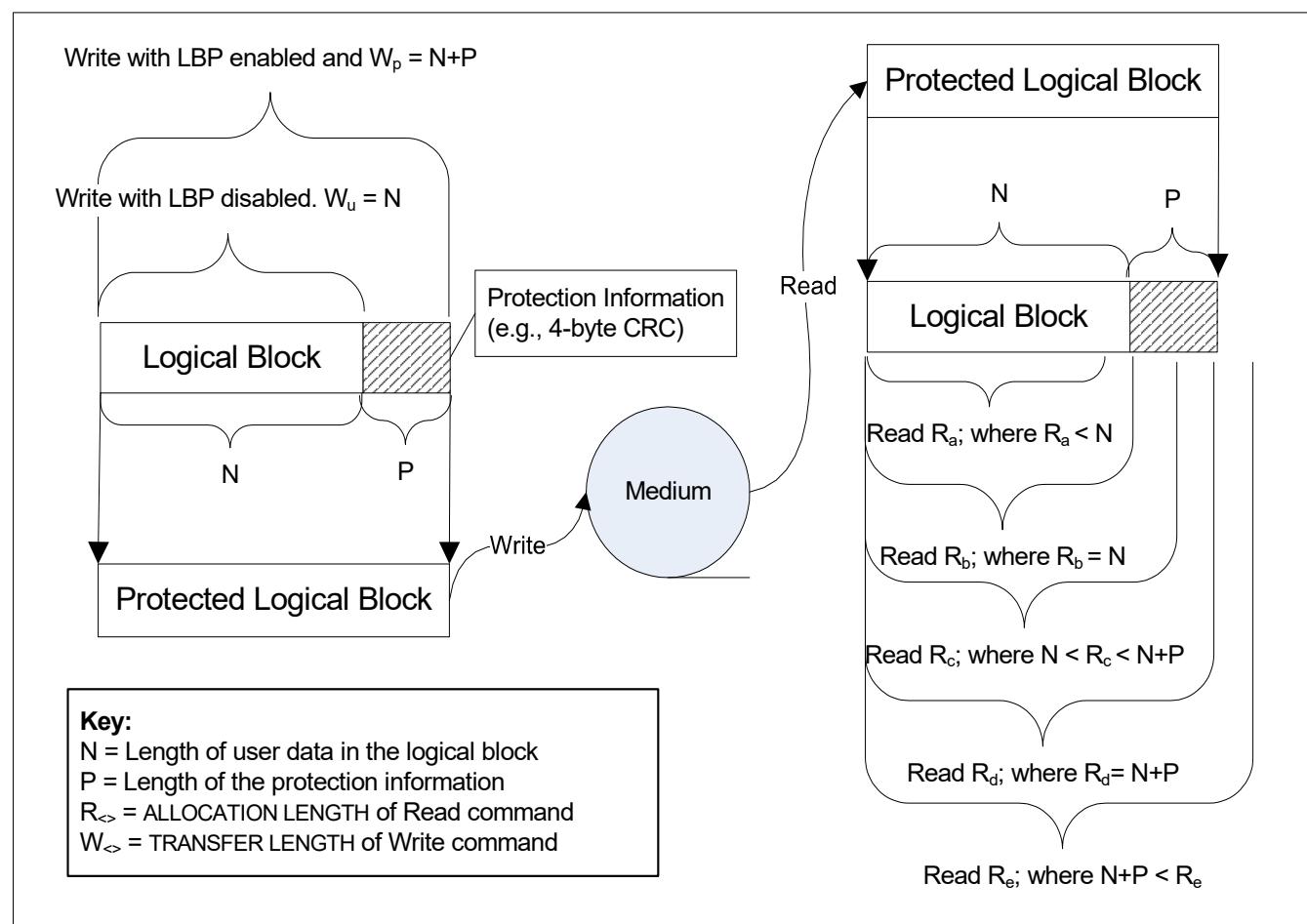


Figure G.1 — Example protected logical block as it relates to write and read commands

This annex assumes that the protection information (e.g., CRC) validates for each example, that when logical block protection is enabled both the LBP_W and LBP_R bits in the Control Data Protection mode page are set to one, and that the same logical block protection method is used for writes and reads, and that the read type command is not a READ REVERSE command.

G.2 General write rules

If logical block protection is enabled, and a write command with the TRANSFER LENGTH field set to W_p is received, then a logical block is written to the medium in a unit called a Protected Logical Block which contains the N bytes of user data and P bytes of protection information. The P bytes of protection information were either transferred in the write command or transformed by the device server.

If logical block protection is disabled, and a write command with the TRANSFER LENGTH field set to W_u is received, then the device server generates P bytes of protection information and a logical block is written to the medium in a unit called a Protected Logical Block which contains the N bytes of user data transferred in the write command and the P bytes of protection information generated by the device server.

In each scenario the user data portion of the protected logical block is N bytes and the protection information is P bytes. If the LOGICAL BLOCK PROTECTION METHOD field of the Control Data Protection mode page (see 8.5.9) is set to 01h (i.e., Use the Reed-Solomon CRC as specified in ECMA-319 as the logical block protection information), then the protection information is a CRC of 4 bytes.

G.3 General read rules

If logical block protection is enabled, then N+P bytes are available for transfer (i.e., the device server validates the protection information and the protection information is included in the data available to be transferred).

If logical block protection is disabled, then N bytes are available for transfer (i.e., the device server validates the protection information and the protection information is not included in the data available to be transferred).

If the read command requests fewer bytes than are available for transfer, then the read is an overlength read. If the read requests more bytes than are available, then the read is an underlength read.

The amount of data returned is the smaller of the bytes available and the allocation length.

Illegal length and CHECK CONDITION status is handled as normal for underlength and overlength conditions:

- a) if a read type command with the SILI bit set to 0b results in an overlength condition, then report CHECK CONDITION for the overlength condition;
- b) if a read type command with the SILI bit set to 1b and the BLOCK LENGTH field in the mode parameter block descriptor set to a non-zero value results in an overlength condition, then report CHECK CONDITION for the overlength condition;
- c) if a read type command with the SILI bit set to 0b results in an underlength condition, then report CHECK CONDITION for the underlength condition; or
- d) all other cases in read type commands do not report an illegal length condition (e.g., report GOOD status).

If a CHECK CONDITION is reported for an illegal length, then the VALID bit in the sense data is set to one and the INFORMATION field in the sense data is set to the value of:

- a) if the request is for a variable-block transfer, ((ALLOCATION LENGTH) - (available bytes)); or
- b) if the request is for a fixed-block transfer, ((ALLOCATION LENGTH) - (number of blocks transferred prior to the illegal length block)).

G.4 Examples from figure G.1 using variable-block transfers and various SILI and BLOCK LENGTH settings

If logical block protection is enabled (i.e., N+P bytes are available for transfer), the command is a read type command, the SILI bit is set to 0b, the FIXED bit is set to 0b, and the ALLOCATION LENGTH is:

- a) Ra where Ra < N, then Ra bytes of data are returned with CHECK CONDITION for overlength condition and the command terminates with the logical position on the EOP side of the incorrect length block;
- b) Rb where Rb = N, then Rb bytes of data are returned with CHECK CONDITION for overlength condition and the command terminates with the logical position on the EOP side of the incorrect length block;
- c) Rc where N < Rc < N+P, then Rc bytes of data are returned with CHECK CONDITION for overlength condition and the command terminates with the logical position on the EOP side of the incorrect length block;
- d) Rd where Rd = N+P, then Rd bytes of data are returned with GOOD status (i.e., the read is a correct length read); or
- e) Re where N+P < Re, then N+P bytes of data are returned with CHECK CONDITION for underlength condition and the command terminates with the logical position on the EOP side of the incorrect length block.

If logical block protection is enabled (i.e., N+P bytes are available for transfer), the command is a read type command, the SILI bit is set to 1b, the FIXED bit is set to 0b, the BLOCK LENGTH in the mode parameter header is non-zero, and the ALLOCATION LENGTH is:

- a) Ra where Ra < N, then Ra bytes of data are returned with CHECK CONDITION for overlength condition and the command terminates with the logical position on the EOP side of the incorrect length block;
- b) Rb where Rb = N, then Rb bytes of data are returned with CHECK CONDITION for overlength condition and the command terminates with the logical position on the EOP side of the incorrect length block;
- c) Rc where N < Rc < N+P, then Rc bytes of data are returned with CHECK CONDITION for overlength condition and the command terminates with the logical position on the EOP side of the incorrect length block;
- d) Rd, then Rd bytes of data are returned with GOOD status (i.e., the read is a correct length read); or
- e) Re, then N+P bytes of data are returned with GOOD status (i.e., the underlength condition is suppressed).

If logical block protection is enabled (i.e., N+P bytes are available for transfer), the command is a read type command, the SILI bit is set to 1b, the FIXED bit is set to 0b, the BLOCK LENGTH in the mode parameter header is zero, and the ALLOCATION LENGTH is:

- a) Ra where Ra < N, then Ra bytes of data are returned with GOOD status (i.e., the overlength condition is suppressed);
- b) Rb where Rb = N, then Rb bytes of data are returned with GOOD status (i.e., the overlength condition is suppressed);
- c) Rc where N < Rc < N+P, then Rc bytes of data are returned with GOOD status (i.e., the overlength condition is suppressed);
- d) Rd where Rd = N+P, then Rd bytes of data are returned with GOOD status (correct length read); or
- e) Re, then N+P bytes of data are returned with GOOD status (i.e., the underlength condition is suppressed).

If logical block protection is disabled (i.e., N bytes are available for transfer), the command is a read type command, the SILI bit is set to 0b, the FIXED bit is set to 0b, and the ALLOCATION LENGTH is:

- a) Ra where Ra < N, then Ra bytes of data are returned with CHECK CONDITION for overlength condition and the command terminates with the logical position on the EOP side of the incorrect length block;
- b) Rb where Rb = N, then Rb bytes of data are returned with GOOD status (i.e., the read is a correct length read); or
- c) Rc, Rd, or Re where R<> > N, then N bytes of data are returned with CHECK CONDITION for underlength condition and the command terminates with the logical position on the EOP side of the incorrect length block.

If logical block protection is disabled (i.e., N bytes are available for transfer), the command is a read type command, the SILI bit is set to 1b, the FIXED bit is set to 0b, the BLOCK LENGTH in the mode parameter block descriptor is non-zero, and the ALLOCATION LENGTH is:

- a) Ra where Ra < N, then Ra bytes of data are returned with CHECK CONDITION for overlength condition and the command terminates with the logical position on the EOP side of the incorrect length block;
- b) Rb where Rb = N, then Rb bytes of data are returned with GOOD status (i.e., the read is a correct length read); or
- c) Rc, Rd, or Re where R<> > N, then N bytes of data are returned with GOOD status (i.e., the underlength condition is suppressed).

If logical block protection is disabled (i.e., N bytes are available for transfer), the command is a read type command, the SILI bit is set to 1b, the FIXED bit is set to 0b, the BLOCK LENGTH in the mode parameter header is zero, and the ALLOCATION LENGTH is:

- a) Ra where Ra < N, then Ra bytes of data are returned with GOOD status (i.e., the overlength condition is suppressed);
- b) Rb where Rb = N, then Rb bytes of data are returned with GOOD status (i.e., the read is a correct length read); or
- c) Rc, Rd, or Re where R<> > N, then N bytes of data are returned with GOOD status (i.e., the underlength condition is suppressed).

G.5 Examples from figure G.1 using fixed-block transfers and various BLOCK LENGTH settings

If logical block protection is enabled (i.e., N+P bytes are available for transfer), the command is a read type command, the SILI bit is set to 0b, the FIXED bit is set to 1b, and the BLOCK LENGTH field in the mode parameter block descriptor is:

- a) Ra where Ra < N, then Ra bytes of data are returned with CHECK CONDITION for overlength condition and the command terminates with the logical position on the EOP side of the incorrect length block;
- b) Rb where Rb = N, then Rb bytes of data are returned with CHECK CONDITION for overlength condition and the command terminates with the logical position on the EOP side of the incorrect length block;
- c) Rc where N < Rc < N+P, then Rc bytes of data are returned with CHECK CONDITION for overlength condition and the command terminates with the logical position on the EOP side of the incorrect length block;
- d) Rd where Rd = N+P, then Rd bytes of data are returned with GOOD status (i.e., the read is a correct length read) and the read continues to the next logical block (i.e., up to ALLOCATION LENGTH number of logical blocks); or
- e) Re where N+P < Re, then N+P bytes of data are returned with CHECK CONDITION for underlength condition and the command terminates with the logical position on the EOP side of the incorrect length block.

If logical block protection is disabled (i.e., N bytes are available for transfer), the command is a read type command, the SILI bit is set to 0b, the FIXED bit is set to 1b, and the BLOCK LENGTH field in the mode parameter block descriptor is:

- a) Ra where Ra < N, then Ra bytes of data are returned with CHECK CONDITION for overlength condition and the command terminates with the logical position on the EOP side of the incorrect length block;
- b) Rb where Rb = N, then Rb bytes of data are returned with GOOD status (i.e., the read is a correct length read) and the read continues to the next logical block (i.e., up to ALLOCATION LENGTH number of logical blocks); or
- c) Rc, Rd, or Re where R<> > N, then N bytes of data are returned with CHECK CONDITION for underlength condition and the command terminates with the logical position on the EOP side of the incorrect length block.