Working Draft

Project T10/1731-D

Revision 31
13 June 2011

Information technology - SCSI Primary Commands - 4 (SPC-4)

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: Ralph O. Weber
ENDL Texas
18484 Preston Road
Suite 102 PMB 178
Dallas, TX 75252
USA

Telephone: 214-912-1373
Facsimile: 972-596-2775
Email: ROWeber at IEEE dot org

Reference number
ISO/IEC 14776-454 : 200x
ANSI INCITS.*:200x**

Printed 5:47 PM Monday 13 June 2011

Points of Contact:

T10 Chair
John B. Lohmeyer
LSI Corp.
4420 Arrows West Drive
Colorado Springs, CO 80907-3444
Tel: (719) 533-7560
Fax: (719) 533-7183
Email: lohmeyer at t10 dot org

T10 Vice-Chair
Mark S. Evans
Western Digital
5863 Rue Ferrari
San Jose, CA 95138
Tel: (408) 363-5257
Fax: (408) 363-5139
Email: Mark dot Evans at wdc dot com

INCITS **Secretariat**
INCITS Secretariat
1101 K Street NW, Suite 610
Washington, DC 20005
Telephone: 202-737-8888
Facsimile: 202-638-4922
Email: incits@itic.org

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

Document Distribution

INCITS Online Store
managed by Techstreet
1327 Jones Drive
Ann Arbor, MI 48105
http://www.techstreet.com/incitsgate.tmpl
Telephone: 1-734-302-7801 or
1-800-699-9277
Facsimile: 1-734-302-7811

or

Global Engineering
15 Inverness Way East
Englewood, CO 80112-5704
http://global.ihs.com/
Telephone: 1-303-792-2181 or
1-800-854-7179
Facsimile: 1-303-792-2192

Revision Information

Changes in the SCSI standards family list, clause 1, are never marked with change bars. Changes in the ASC/ASCQ, Operation Code, Log Page Code, Mode Page Code and Vendor ID tables are usually not marked with change bars.

1 Approved Documents Included

No T10 approved proposals are included in SPC-4 r0. SPC-4 r0 is identical to SPC-3 r23 except that editorial changes made by the ANSI Editor as part of the INCITS Public Review for SPC-3 appear in SPC-4 r0.

The following T10 approved proposals have been incorporated SPC-4 up to and including this revision:

04-198r5	Background Medium Scan
04-262r8	ATA Command Pass-Through
04-335r2	WORM Tamper Read Enable [additional sense code]
04-371r2	Enable Background Operation Error Reporting Bit
04-372r1	I_T NEXUS RESET task management function
04-397r3	SAT ATA control mode page proposal
05-003r3	VPD Page 83 Identifier s/b Designator Changes
05-015r2	Add new ADC-2 command to set Volume Identifier
05-026r2	SMC-3: DTE Prevented Medium Removal [additional sense code]
05-074r1	SPC-4 Command requirements
05-142r4	SAT - LOG SENSE command and SMART [additional sense code]
05-156r7	Application ownership of protection information Reference Tag
05-157r9	Security Commands proposal
05-230r0	Obsolete Per Initiator Port in SPC-4
05-232r2	Log Command Corrections
05-242r2	Combinations of bits and fields in the LOG SELECT CDB and log parameters
05-243r4	SMC-3 REQUEST DATA TRANSFER ELEMENT INQUIRY command {new ASC only}
05-248r4	Statistics and Performance Log Pages
05-252r2	Create well known LUN for trusted commands
05-262r3	SSC-3: ASC/ASCQ for Medium Thread Failure [additional sense code]
05-284r4	Self Describing Command Timeouts
05-296r0	Persistent Reservations PREEMPT AND ABORT Clarification
05-299r1	Correct Log Page Format Tables in SPC-4, SBC-3, & SAS-2
05-317r3	Remove Attached Media Changer model
05-331r1	Allow Mode Sense Through Read Only Persistent Reservation
05-337	(Sep. 2005 CAP minutes) Change IEEE 1212-1994 reference to IEEE 1212-2001
05-340r3	Background scan additions
05-360r2	Extended self test completion time value update
05-368r2	Allow more commands through Write Exclusive reservations
05-370r2	NOTIFY (POWER LOSS EXPECTED) Fixes [additional sense code]
05-374r2	Disabling Reassign on Write Long Logical Blocks
05-375r1	EBACKERR bit Error Reporting Clarification
05-379r0	Unit attention condition trivial wording changes
05-380r0	Mode Page Policy MLUS bit correction
05-383r4	Deferred Microcode Downloads
05-406r1	New ASC/ASCQs for Device Port Address
06-018r0	Update SPC-4 version field
06-079r1	More Log Page Cleanup
06-108r3	Add security protocol and additional sense for tape encryption
06-113r2	Log Parameter Subpages

06-127 Assign Security Protocol code value EFh to the ATA Device Server Password Security protocol (minutes of the March Plenary)
06-147r0 ALL_TG_PT clarification
06-175r2 Wrapping and saturating counter definitions
06-191r0 Request for the reservation/assignment of Security Protocol value
06-195r1 Persistent Reservation additional sense code changes
06-196r0 Persistent Reservation REGISTER AND MOVE from unregistered I_T nexus
06-198r0 Making the basic task management model obsolete
06-219r1 New ASC/ASCQs for LOGICAL UNIT ACCESS NOT AUTHORIZED
06-221r3 Peripheral device identifying information
06-223r1 Vendor-specific service actions for MAINTENANCE (IN) and MAINTENANCE (OUT)
06-224r1 Change 'medium changer' to 'media changer'
06-248r1 Remove the PREVENT ALLOW MEDIUM REMOVAL command from SPC-4
06-259r1 Making linked commands obsolete
06-264r2 REQUEST SENSE parameter data clarifications
06-274r1 REQUEST SENSE and Stopped power condition
06-278r0 Fix persistence of SET DEVICE IDENTIFIER
06-282r4 SPC-4 WRITE BUFFER clarifications
06-298r0 Persistent Reservations correction
06-320r1 Statistics logging for FUA and FUA_NV
06-321r1 Statistics and Performance log pages command cleanup
06-323r1 Multiple service delivery subsystem editorial tweaks
06-325r0 New ASC/ASCQ for SAT queue cleared condition
06-337r0 Statistics Group Number inconsistencies in SPC-4, SSC-3, and MMC-5
06-350r0 Power conditions state machine clarifications
06-362r7 Error History proposal
06-369r8 Security Association Model for SPC-4
06-388r3 Security Goals and Threat Model
06-389r5 Using Public-Key Cryptography for Key Wrapping {new ASC only}
06-390r2 ALUA/TPGS Disconnected state
06-411r2 Clear REPORTED LUNS DATA HAS CHANGED on any command
06-454r0 Clarifying Identifying Information Types Requirements
06-455r1 Lost SET IDENTIFYING INFORMATION Parameter List Length requirement
06-465r1 Alternative Proposal for Management Transport
06-484r1 Encryption MAM Attribute
07-005r2 SSC-3: Encryption unsupported medium {new ASC only}
07-008r8 Expander Notification of Temporary Shutdown
07-022r1 MANAGEMENT PROTOCOL IN command field name fixup
07-029r3 Extended SCSI Commands
07-056r0 Editorial change: IEEE P1667 to IEEE 1667
07-077r0 Stop fruitless attempts to shackle users via the IALUAE Control Ext. Mode bit
07-080r2 Persistent reservations self-preempting clarification
07-100r0 Digital Signing of Microcode
07-111r1 Duplicate persistent reservation wording removal
07-153r1 Protocol-Specific VPD pages
07-159r1 Add QTSS and QUAS bits to REPORT SUPPORTED TMF
07-169r7 ESP-SCSI for Parameter Data
07-172r1 Target Port Group membership clarification
07-184r0 Request for security protocol
07-196r0 LOG SELECT CDB page/subpage field usage
07-203r0 Block Device Characteristics VPD page and medium rotation rate field
07-215r0 Protocol-Specific log page subpages
07-216r1 NAA Locally Assigned designator format
07-218r3 SSC-3: Configurable Early Warning {new ASC only}

07-222r2 SMC-3 Additional sense code for incompatible destination element {new ASC only}
07-244r3 SMC-3 READ ATTRIBUTE and WRITE ATTRIBUTE command clarification {new ASC only}
07-257r1 Prohibited needed as a keyword in SPC-4
07-262r6 Command Security Model
07-269r1 Update Annex C for ADC-2 changes
07-281r1 "Except INQUIRY, REPORT LUNS, and REQUEST SENSE"
07-295r1 Request for TrustedFlash Security Protocol ID Value
07-315 SECURITY CONFLICT IN TRANSLATED DEVICE additional sense code for block devices {new ASC only} (minutes of September CAP WG)
07-361r6 SSC-3 Out of Band Encryption Key Management {new ASCs only}
07-389r1 Persistent Reservation Clarification
07-390r1 Target Port Groups & Logical Unit Groups
07-430r2 UML model for CbCS
07-437r4 Establishing a Security Association using IKEv2
07-451r1 WRITE LONG COR_DIS and WR_UNCOR interaction
07-459r4 Unit attention condition queuing
08-008r2 Download microcode I_T nexus usage
08-029r2 Automation Encryption Control {new ASCs only}
08-034r2 Error history cleanup
08-037r3 SA Creation corrections and clarifications
08-041r1 Use period as decimal separator in T10 standards
08-053r1 Persistent reservation fixes
08-057r0 Allocation length advisories in Inquiry command
08-061 ATA Power Conditions mode subpage assignment (January CAP Minutes)
08-108r1 Completing the SET IDENTIFYING INFORMATION command
08-113r2 UML Conventions and CbCS UML Updates
08-116r3 Protection Type 3 Reference Tag Clarification
08-138r1 Constraints on SPC-4 SA creation based on Usage Type
08-145r3 Capability-based Command Security (CbCS) [the rewrite]
08-156r2 Non-volatile cache becoming volatile
08-158r1 ESP-SCSI field alignments
08-177r0 Bogus Sense Key in SA Creation Parameter Data Error Handling
08-178r0 IKEv2-SCSI Message ID field size correction
08-180r0 SCSI device name cleanup in SPC-4
08-194r1 Common definition for CODE SET
08-199r1 Even more log page cleanup
08-214r1 Allowing ESP-SCSI descriptors in variable length CDBs
08-231r1 SPEC_I_PT clarification
08-254r0 Protocol Specific diagnostic page
08-261r1 Parsing variable-length parameter lists and data
08-262r0 SAI, SQN, IV, & Data must be contiguous
08-263r3 Too much to do … Not enough Progress Counters
08-281r5 UAS Clause 4 (Model) {new ASC only}
08-287r1 ATA Information VPD page nits
08-327r2 Restrict SPC-4 and SAM-5 codes to FC-SB-4 use
08-346r2 Correction to IKEv2-SCSI Certificate Request Payload
08-356r5 Thin Provisioning Commands
08-386r5 Cache hits and power on statistics
08-396r3 Reporting support for all DIF types
08-415r4 Adding a Protection Information Interval
08-423r4 IKEv2-SCSI Authentication
08-450r1 Add Restricted keyword to the Style Guide
09-002r0 Making the PPC bit in the LOG SENSE command obsolete
09-011r8 Thin Provisioning Threshold Notification

09-018r2	Implicit transition (RTPG) timeout notification enhancements
09-019r0	Reclaim some Obsolete bits for future use
09-031	Mark Service Action In/Out (16) service action value 0Fh Restricted for SAT-3
09-033r4	Device constituent VPD page
09-034r1	EMail comment received from Mike Berhan busTRACE
09-054r1	Adding more idle power options
09-072r0	Substantive corrections to 09-054r1 -- Adding more idle power options
09-087r2	Rectifying conflicts for DEVOFFL and UNITOFFL
09-100r5	SBC-3 - Atomic COMPARE AND WRITE command [operation code only]
09-118r0	Mode Page 18h, Change 'Reserved' field to 'Protocol specific' field
09-146r4	Tape end-to-end logical block protection Inquiry … Data
09-162r2	Revise LOG SELECT w/ PCR=1 wording
09-167r2	PL_OC2 I_T Nexus loss fixes [additional sense code]
09-178r2	Add IPv6 target descriptor to XCOPY
09-199r1	Add IP Copy Engine service type to page 85h
09-200r0	Update Access Controls for READ CAPACITY(16)
09-208r1	SPC-3 vs SPC-4 Descriptor format REQUEST SENSE
09-211r1	How many sense data descriptors are too many?
09-219r1	Fix errors in description of log pages 0Eh and 1Ah
09-233r1	AC/DC MSB/LSB
09-237r0	Add clarification to Supported Log Pages log pages
09-241r6	SCSI Referrals [Penokie]
09-250r4	Change power condition timer behavior when commands are received
09-262r8	Bitmaps on Disk - Detailed Changes {new ASC only}
09-281r0	Request ASC/ASCQ for insufficient power condition [ASC only]
09-294r2	Adding a new VPD bit indicating download microcode behavior
09-303r1	SSC-4 Device and volume log page enhancements [log page code only]
09-322r2	SBC-3 Add log page for solid state drive (SSD) [log page code only]
09-342r0	Preempt And Abort has a cross-nexus effect
09-355r2	MAM Write-through cache bit
09-356r5	SSC-4 Append-only mode [operation code and ASC only]
09-357r4	Relationship between power conditions and background tasks
09-359r2	Conflicted List Format Log Parameter Rules
09-360r9	Log Page clean-up
09-363r4	XCOPY with no LIST IDENTIFIER
09-367r0	Volume Coherency MAM parameter
09-369r1	No UA on RELEASE
09-370r1	IKEv2-SCSI Key Exchange Clarification
09-379r1	Another Encryption MAM Attribute
09-381r2	Adding a Copy Manager class into SAM-5
09-401r1	Volume Coherency MAM parameter [MAM Attributes only]
09-408r2	Task to Command conversion
10-005r1	Add Disable PI Check on xxPROTECT=0 feature
10-016r2	Persistent Reservations Clarification
10-032r4	TMF timeouts
10-043r1	Correct usage of term volume in MAM attributes
10-065r1	XCOPY IP Copy Service target descriptor
10-073r0	Extended Logical Unit Address Method added to Report LUNs Command
10-108r2	READ BUFFER from echo buffer mode error
10-115r2	Power conditions transition modifications
10-120r0	Fix to REPORT SUPPORTED TASK MANAGEMENT FUNCTIONS command
10-133r4	Pollable Device State model
10-135r0	Obsolete VPD pages in SPC-4 Annex D
10-138r1	New Supported Attributes service action for READ ATTRIBUTE

10-144r3 Self-test and SEND DIAGNOSTIC command clean-up
10-148r1 More about power conditions and background tasks
10-156r0 Allow RECEIVE COPY RESULTS through reservations
10-161r1 VPD page generic format
10-162r2 Application Tag Mode Page
10-177r2 Page_0 Mode Page Format & Sub-page 00h
10-179r1 Remove implicit head of queue rule that is in SAM-5
10-191r2 EXTENDED COPY Descriptor format Sense Data
10-201r1 SET IDENTIFYING INFORMATION clarifications
10-208r1 AER Obsolete bits in SPC-4
10-210r3 Tweaks to power state machines found during SPL LB Resolution
10-220r4 SSC-4: Verify Files and Logical Block Protection Method {new ASC only}
10-242r2 Too many words about pollable sense data in SPC-4
10-270r1 Possible bugs in Self-test and SEND DIAGNOSTIC command clean-up
10-271r2 Fixing the TMF Timeout REPD bit
10-280r1 Reject Write Without Protection information
10-282r1 Expand definition of self-test completion time field
10-284r7 Coordination of SAS Device Power Consumption {new ASC only}
10-289r1 Power Management Tweak / Clarification
10-300r1 Security protocol ID for JEDEC UFS RPMB Protocol
10-350r0 Clarify TUR - GOOD status can now return sense data
10-390r1 Forwarding GOOD sense data
11-004r0 Remove REPORT LUNS note about SPC
11-010r0 Obsolete READ CAPACITY command PMI bit
11-011r4 Add sanitize command
11-019r1 Logical Block Provisioning corrections [Codes Annex only]
11-021r2 RCR OPERATING PARAMETERS - SNLID bit cleanup
11-026r1 Informational Exceptions Mode Page Modernization in SPC-4, and …
11-060r0 Remove SAS-2 references in power state machines
11-087r1 Standard INQUIRY data target device vs logical unit clarifications
11-114r2 CFA Profile Support VPD page
11-185r1 Returning other sense data for a REQUEST SENSE command
11-186r1 PARAMETER POINTER field values in the LOG SENSE command
11-187r2 Modes using the BUFFER CAPACITY field from the READ BUFFER command
11-204r0 Incorporate Additional Sense Code 23h/00h (only) from this doc {new ASC only}
11-216r1 Error conditions during resetting of log parameters
11-222r0 Minutes of the ADI Working Group (action item 11-008)
11-235r0 One more Log Page correction

T10 approved 07-027r2 (Enabling and disabling Transport Layer Retries), r1 as revised, for inclusion in SPC-4 but the author decided not to include SPC-4 changes in the final revision. Instead, a new proposal 07-153 was started to cover the same information.

T10 approved 07-454r5 (Capability based Command Security), but the proposal contained a large number of problems that required substantive corrections. Later, 08-145r3 (Capability-based Command Security (CbCS) [the rewrite]) was approved as the replacement for 07-454r5 and duly incorporated as noted above.

T10 approved 10-206r4 (Informational Exceptions Control mode page clean-up), but also approved 11-026r1 (Informational Exceptions Mode Page Modernization in SPC-4, SBC-3, SSC-4, and SES-3) at the same meeting. Since 11-026r1 moves the Informational Exceptions Control mode page out of SPC-4, the changes which 10-206r4 designated for inclusion in SPC-4 were made in SBC-3 and SES-3 instead.

To the best of the technical editor's knowledge, all T10 approved proposals have been included in this revision.

2 Revision History

2.1 Revision 0 (1 August 2005)

Revision 0 of SPC-4 is substantially equal to revision 23 of SPC-3. The only differences arise from changes made in SPC-3 by the ANSI Editor during the INCITS Public Review process.

2.2 Revision 1 (3 September 2005)

The following T10 approved proposals were incorporated in SPC-4 revision 1:

04-198r5	Background Medium Scan
04-262r8	ATA Command Pass-Through
04-371r2	Enable Background Operation Error Reporting Bit
04-372r1	I_T NEXUS RESET task management function
04-397r3	SAT ATA control mode page proposal
05-003r3	VPD Page 83 Identifier s/b Designator Changes
05-015r2	Add new ADC-2 command to set Volume Identifier
05-074r1	SPC-4 Command requirements
05-230r0	Obsolete Per Initiator Port in SPC-4

The sentence introducing the 'Parameter control bits for Informational exceptions log parameter' table was changed from 'self test results log parameters' to 'informational exceptions log parameter'.

Media Changer usage was added for the LOGICAL UNIT NOT READY, OFFLINE additional sense code.

A missing column separator line was added in Table C.5 (LOG SELECT save options).

Several purely editorial changes made by the ANSI Editor in the published version of SPC-3 were included in SPC-4 r01.

2.3 Revision 1a (5 September 2005)

The ASCQ assignment for ATA DEVICE FAILED SET FEATURES was changed from F1h to 71h so that it did not fall in the vendor specific ASCQ range.

2.4 Revision 2 (15 September 2005)

The following T10 approved proposals were incorporated in this SPC-4 revision:

05-296r0	Persistent Reservations PREEMPT AND ABORT Clarification
05-299r1	Correct Log Page Format Tables in SPC-4, SBC-3, & SAS-2
05-331r1	Allow Mode Sense Through Read Only Persistent Reservation
05-337	(Sep. 2005 CAP minutes) Change IEEE 1212-1994 reference to IEEE 1212-2001

For consistency with other proposals, one instance of ‘background medium read scan’ was changed to ‘background medium scan’.

The page codes for the Background Control mode page and Background Scan Results log page were added to the Numeric Order Codes annex.

2.5 Revision 3 (19 January 2006)

The following T10 approved proposals were incorporated in this SPC-4 revision:

04-335r2	WORM Tamper Read Enable [additional sense code]
05-026r2	SMC-3: DTE Prevented Medium Removal [additional sense code]
05-142r4	SAT - LOG SENSE command and SMART [additional sense code]
05-156r7	Application ownership of protection information Reference Tag
05-157r9	Security Commands proposal
05-242r2	Combinations of bits and fields in the LOG SELECT CDB and log parameters
05-252r2	Create well known LUN for trusted commands
05-262r3	SSC-3: ASC/ASCQ for Medium Thread Failure [additional sense code]
05-317r3	Remove Attached Media Changer model
05-340r3	Background scan additions
05-360r2	Extended self test completion time value update
05-368r2	Allow more commands through Write Exclusive reservations
05-370r2	NOTIFY (POWER LOSS EXPECTED) Fixes [additional sense code]
05-374r2	Disabling Reassign on Write Long Logical Blocks
05-375r1	EBACKERR bit Error Reporting Clarification
05-379r0	Unit attention condition trivial wording changes
05-380r0	Mode Page Policy MLUS bit correction
05-383r4	Deferred Microcode Downloads
05-406r1	New ASC/ASCQs for Device Port Address
06-018r0	Update SPC-4 version field

In the MAM device type attributes table, code value 020Ah was removed from the reserved list because the same table show it as allocated to DEVICE MAKE/SERIAL NUMBER AT LAST LOAD.

In the mode page codes and subpage codes table, the subpage code value for page code 00h was changed from 'not applicable' to 'vendor specific'. Since mode page code 00h is vendor specific, the applicability of subpage codes must also be vendor specific.

The mode page code information in the numeric order codes informative annex was changed to match the normative code assignments in Mode Parameters subclause. Every code described in the annex as vendor specific was changed to device-type specific, with the exception of page code 00h, which the normative text defines as vendor specific.

Several additional sense code definitions were added as requested by the MMC-5 working group.

Reference columns were added to the mode tables for READ BUFFER and WRITE BUFFER. Several instances of 'accept' were changed to 'process' as in 'If the device server is unable to ~~accept~~ process …'

Some (but not all) of the editorial changes in 05-340r2 that were removed by the CAP working group in r3 were made anyway.

2.6 Revision 4 (9 March 2006)

The following T10 approved proposals were incorporated in this SPC-4 revision:

06-079r1	More Log Page Cleanup
06-127	Assign Security Protocol code value EFh to the ATA Device Server Password Security protocol (minutes of the March Plenary)
06-147r0	ALL_TG_PT clarification

The descriptions of three additional sense codes 10h/01h, 10h/02h, and 10h/03h were modified to match the nomenclature in SBC-2 and SBC-3 as follows:

a) ~~DATA~~ LOGICAL BLOCK GUARD CHECK FAILED;
b) ~~DATA~~ LOGICAL BLOCK APPLICATION TAG CHECK FAILED; and
c) ~~DATA~~ LOGICAL BLOCK REFERENCE TAG CHECK FAILED

The Operation Codes table in the numerical codes annex was updated to:

a) make the SECURITY PROTOCOL IN and SECURITY PROTOCOL OUT commands optional for MMC devices; and
b) add the OPEN/CLOSE IMPORT/EXPORT ELEMENT command for SMC devices.

The ADC usage of Service Buffers Information log page was added to the numerical codes annex.

The supported protection type (SPT) field in the Extended INQUIRY Data VPD page was updated to match the SBC-3 usage.

The definition of additional sense code was update to match existing field reference nomenclature.

2.7 Revision 5 (10 June 2006)

The following T10 approved proposals were incorporated in this SPC-4 revision:

05-248r4	Statistics and Performance Log Pages
06-108r3	Add security protocol and additional sense for tape encryption
06-113r2	Log Parameter Subpages
06-191r0	Request for the reservation/assignment of Security Protocol value
06-195r1	Persistent Reservation additional sense code changes
06-196r0	Persistent Reservation REGISTER AND MOVE from unregistered I_T nexus
06-198r0	Making the basic task management model obsolete
06-219r1	New ASC/ASCQs for LOGICAL UNIT ACCESS NOT AUTHORIZED
06-223r1	Vendor-specific service actions for MAINTENANCE (IN) and MAINTENANCE (OUT)
06-224r1	Change 'medium changer' to 'media changer'
06-248r1	Remove the PREVENT ALLOW MEDIUM REMOVAL command from SPC-4

The operation code list in the informative annex was updated to match MMC-5 usage. The log page code list was updated to match SSC-3.

A reference to the SSCS bit was corrected to the SCCS bit.

The References Under Development subclause was updated.

The MAM device attributes summary table was updated to match the text.

The annex describing Persistent Reservations functionality needed for an implementation equivalent to the model for Reserve/Release was corrected to note that the PERSISTENT RESERVE IN command has no equivalent in the Reserve/Release model.

Numerous editorial changes were made as suggested in 06-271r0. Other minor editorial corrections were made as per requests from T10 members received privately and on the T10 reflector.

2.8 Revision 5a (14 June 2006)

Fix errors, both significant and trivial, in r05. Change bars are cumulative for r05 and r05a.

The revision number in 2.7 was corrected.

Log page parameter code nomenclature was made consistent in the various clause 7 format tables.

The Time Interval log parameter was corrected from 0002h to 0003h.Clarifying changes were made to adjust for moving the *command weight*, *weighted command time*, and *idle time* definitions from the Statistics and Performance log pages overview to the Statistics and Performance log parameter definition. The handling of task priority values of zero was returned to what was defined in 05-248r4. The removal of the TIME INTERVAL DESCRIPTOR field was reflected throughout. The READ/WRITE COMMAND PROCESSING INTERVAL field references were corrected to READ/WRITE COMMAND PROCESSING TIME field.

The coded value tables headers were updated to reference SBC-3, SSC-3, MMC-6, and SMC-3. Hopefully, these will be the active standards at the time SPC-4 is published.

2.9 Revision 6 (18 July 2006)

The following T10 approved proposals were incorporated in this SPC-4 revision:

06-175r2 Wrapping and saturating counter definitions
06-259r1 Making linked commands obsolete
06-264r2 REQUEST SENSE parameter data clarifications
06-274r1 REQUEST SENSE and Stopped power condition
06-278r0 Fix persistence of SET DEVICE IDENTIFIER
06-298r0 Persistent Reservations correction
06-320r1 Statistics logging for FUA and FUA_NV
06-323r1 Multiple service delivery subsystem editorial tweaks
06-325r0 New ASC/ASCQ for SAT queue cleared condition
06-337r0 Statistics Group Number inconsistencies in SPC-4, SSC-3, and MMC-5

In the table showing the format of the Statistics and Performance log parameter, the field names were corrected to match the field names used in the field definitions that follow the table.

Following a suggestion indicated by 06-323r1, all practical instances of SAM-3 were changed to SAM-4.

2.10 Revision 7 (6 October 2006)

The following T10 approved proposals were incorporated in this SPC-4 revision:

06-221r3 Peripheral device identifying information
06-350r0 Power conditions state machine clarifications

The SECURITY PROTOCOL SPECIFIC field for certificates in the SECURITY PROTOCOL IN command was corrected to 0001h. The erroneous value in previous revisions was 0000h. The Certificate data table title and the cross reference to the table were corrected in the same subclause.

In the SECURITY PROTOCOL OUT CDB format table, ALLOCATION LENGTH was changed to TRANSFER LENGTH to match the field definition text following the table.

2.11 Revision 7a (7 October 2006)

Revision 7a is contains the same content as revision 7. The source files have been converted to FrameMaker 7.2 and tables like the additional sense code table have been restructured to use a single column for the DTL… information. The latter change reduces the complexity (i.e., size) of the FrameMaker source files substantially and (hopefully) eliminates a bazillion error messages from the Acrobat PDF Distiller.

2.12 Revision 8 (6 January 2007)

The following T10 approved proposals were incorporated in this SPC-4 revision:

05-232r2 Log Command Corrections
05-284r4 Self Describing Command Timeouts
06-282r4 SPC-4 WRITE BUFFER clarifications
06-321r1 Statistics and Performance log pages command cleanup
06-388r3 Security Goals and Threat Model
06-411r2 Clear REPORTED LUNS DATA HAS CHANGED on any command
06-454r0 Clarifying Identifying Information Types Requirements
06-455r1 Lost SET IDENTIFYING INFORMATION Parameter List Length requirement
06-465r1 Alternative Proposal for Management Transport
06-484r1 Encryption MAM Attribute

The CDB formats for the LOG SENSE and LOG SELECT commands were corrected to be 10 bytes in length.

The cross reference for the IEEE Extended description was corrected.

The information about Fibre Channel standards used by SCSI devices was substantially updated.

The codes annex description for log page 11h was updated.

The contact information for ordering ANSI standards was updated as per a request received from the ANSI editor.

A few less significant wording errors were corrected.

2.13 Revision 9 (17 February 2007)

The following T10 approved proposals were incorporated in this SPC-4 revision:

06-390r2 ALUA/TPGS Disconnected state
07-022r1 MANAGEMENT PROTOCOL IN command field name fixup

The editing pencil was applied liberally to 06-390r2 during its incorporation. The most heavy-handed changes were in the glossary, where 'primary' and 'secondary' versions of asymmetric access terms were added beyond those shown in the proposal. Where necessary, the glossary changes were reflected throughout the working draft. The 'Introduction to asymmetric logical unit access' subclause was heavily restructured to clarify the secondary target port group rules agreed upon by the CAP working group.

An error made during the incorporation of 05-242r2 in r03 which affected two rows in the table titled 'PCR bit, SP bit, and PC field meanings when parameter list length is zero' in the LOG SELECT command definition was corrected.

In the REPORT SUPPORTED OPERATION CODES command, the example CDB usage bit map was updated from 'A3h, 0Ch, 03h, FFh, FFh, FFh, FFh, FFh, FFh, FFh, 00h, 07h' to 'A3h, 0Ch, 87h, FFh, FFh, FFh, FFh, FFh,

FFh, FFh, 00h, 07h' in order to reflect recent changes in the format of the REPORT SUPPORTED OPERATION CODES CDB.

The ATA Information VPD page was added to the appropriate tables with references to SAT-2 where needed.

The Normative References and References Under Development were updated to reflect the publication of some projects and the initiation of new ones.

2.14 Revision 10 (21 April 2007)

The following T10 approved proposals were incorporated in this SPC-4 revision:

06-369r8 Security Association Model for SPC-4
06-389r5 Using Public-Key Cryptography for Key Wrapping {new ASC only}
07-005r2 SSC-3: Encryption unsupported medium {new ASC only}
07-008r8 Expander Notification of Temporary Shutdown
07-056r0 Editorial change: IEEE P1667 to IEEE 1667
07-077r0 Stop fruitless attempts to shackle users via the IALUAE Control Ext. Mode bit
07-080r2 Persistent reservations self-preempting clarification
07-100r0 Digital Signing of Microcode
07-111r1 Duplicate persistent reservation wording removal

In the LOG SELECT command, a brief explanation was added to clarify that parameter data contains log page codes and subpage codes and this makes the use of the CDB fields illegal when parameter data is present.

The larger page length in the General Statistics and Performance log page table was corrected to match the length shown at the bottom of the table.

Several incorrect uses of 'target port group asymmetric access state' were corrected to 'target port asymmetric access state'.

In the numeric codes informative annex, T (i.e., SSC-x tapes support) was added Device Statistics log page.

In tables listing mode pages, the Protocol Specific LUN name was corrected to Protocol Specific Logical Unit.

In the Conventions subclause, the unclear word 'comprised' was globally replaced with 'consisting'.

Several problems with small-caps field names (e.g., field names not followed by the word field) were corrected. MSB and LSB were added to the BUFFER CAPACITY field in the READ BUFFER descriptor. Several other editorial mistakes were corrected.

2.15 Revision 11 (14 May 2007)

The following T10 approved proposals were incorporated in this SPC-4 revision:

07-159r1 Add QTSS and QUAS bits to REPORT SUPPORTED TMF
07-172r1 Target Port Group membership clarification
07-184r0 Request for security protocol
07-196r0 LOG SELECT CDB page/subpage field usage
07-203r0 Block Device Characteristics VPD page and medium rotation rate field
07-216r1 NAA Locally Assigned designator format

In the introductory sentence for the Mandatory/Optional access controls resources tables, 'persistent' was corrected to ‘persist’.

2.16 Revision 12 (11 January 2008)

The following T10 approved proposals were incorporated in this SPC-4 revision:

06-362r7 Error History proposal
07-029r3 Extended SCSI Commands
07-153r1 Protocol-Specific VPD pages
07-215r0 Protocol-Specific log page subpages
07-257r1 Prohibited needed as a keyword in SPC-4
07-262r6 Command Security Model [Weber/Penokie]
07-269r1 Update Annex C for ADC-2 changes
07-281r1 "Except INQUIRY, REPORT LUNS, and REQUEST SENSE"
07-389r1 Persistent Reservation Clarification
07-390r1 Target Port Groups & Logical Unit Groups
07-430r2 UML model for CbCS
07-437r4 Establishing a Security Association using IKEv2

Additional sense codes defined in following T10 approved proposals were incorporated in this SPC-4 revision:

07-218r3 SSC-3: Configurable Early Warning
07-222r2 SMC-3 Additional sense code for incompatible destination element
07-244r3 SMC-3 READ ATTRIBUTE and WRITE ATTRIBUTE command clarification
07-315 SECURITY CONFLICT IN TRANSLATED DEVICE additional sense code for block devices (minutes of September CAP WG)

The following proposal was approved by T10 but not included in this revision because the referenced web site contained no information about standards development at the time this revision was produced:

07-295r1 Request for TrustedFlash Security Protocol ID Value

The correct year was added to the SBP-3 normative reference.

The UCS normative reference was updated to ISO/IEC 10646:2003/Amd.2:2006.

The codes annex was modified to indicate VPD page support by RBC devices and to add several device-type specific VPD pages.

In the PREEMPT service action flowchart figure, three instances of 'registration' (singular) were changed to 'registrations' (plural). These changes are not marked with change bars.

The first paragraph of the Informational Exceptions Control mode page was updated to use 'e.g.' language that modernizes its discussion of the ADDITIONAL SENSE CODE field.

In the security model, several instances of DC_xxx SA parameters were changed to the correct DS_xxx (device server).

The Extended INQUIRY Data VPD page definition was edited heavily to make the nomenclature consistent for all field definitions.

A consistent nomenclature of 'blah blah descriptor [first]' was applied to all table formats that contain first/last descriptors.

Several of 'generate unit attention …' were changed to use 'establish'.

Several instances of 'specifies' were changed to 'indicates' in the VPD page definitions. VPD pages cannot specify anything because they are read only.

The mode page codes table nomenclature for other SCSI standards was updated to match the glossary.

A table footer was added to reference the numeric codes annex in the diagnostic page codes table.

The codes annex diagnostic pages table was updated to match SES-2.

The codes annex log page table was modified to match the normative text.

Several typographical errors in the READ BUFFER command were corrected.

Other obvious typos were corrected and not marked with change bars.

New codes annex tables were added for transport protocol specific log, mode, and VPD pages and VPD pages.

2.17 Revision 13 (11 March 2008)

The following T10 approved proposals were incorporated in this SPC-4 revision:

05-243r4	SMC-3 REQUEST DATA TRANSFER ELEMENT INQUIRY command {new ASC only}
07-169r7	ESP-SCSI for Parameter Data
07-295r1	Request for TrustedFlash Security Protocol ID Value
07-451r1	WRITE LONG COR_DIS and WR_UNCOR interaction
07-459r4	Unit attention condition queuing
08-034r2	Error history cleanup
08-037r3	SA Creation corrections and clarifications
08-041r1	Use period as decimal separator in T10 standards
08-053r1	Persistent reservation fixes
08-057r0	Allocation length advisories in Inquiry command
08-061	ATA Power Conditions mode subpage assignment (January CAP Minutes)

Additional sense codes defined in following T10 approved proposals were incorporated in this SPC-4 revision:

07-361r6	SSC-3 Out of Band Encryption Key Management
08-029r2	Automation Encryption Control

Incorporation of 07-295r1 was accomplished by changing the reference to the Secure Digital Card Association and their web site where the TrustedFlash Security Protocol specification is published.

To maintain consistent terminology with SAM-4, several instances of 'task priority' were changed to 'command priority'.

In the Extended INQUIRY Data VPD page definition, a cross reference was added to define the location of the PROTECT bit, which happens to be located in the Standard INQUIRY Data.

In the fixed format sense data definition, the descriptions for the ILI bit, FILEMARK bit, and EOM bit were modified to align them with the description of the INFORMATION field. The concepts found in the r12 descriptions of these bits were turned into 'e.g.' examples.

Two confusing instances of 'single algorithm' were changed to 'combined integrity check and encryption algorithm' in the IKEv2-SCSI Authentication step model subclause. Wording problems related to combined modes were fixed in protocol details informative annex.

The reference for the ATA Device Server Password Security security protocol was changed to SAT-2.

Since SAT-2 does not support tape devices, the T was removed from the codes annex rows for the Parallel ATA Control and Serial ATA Control mode pages.

A UML reference was added to clause 2. UML notation conventions based on the SAM-4 conventions were added to clause 3.

The reference on the restricted bytes in the REPORT DEVICE IDENTIFYING INFORMATION command table was corrected to read SCC-2.

A normative reference to SSC-2 was added because of the reference to the RSMK bit, which is obsolete in SSC-3.

Other editorial corrections to minor to identify were also made.

2.18 Revision 14 (17 March 2008)

The following T10 approved proposals were incorporated in this SPC-4 revision:

08-008r2	Download microcode I_T nexus usage
08-108r1	Completing the SET IDENTIFYING INFORMATION command
08-113r2	UML Conventions and CbCS UML Updates
08-138r1	Constraints on SPC-4 SA creation based on Usage Type
08-158r1	ESP-SCSI field alignments

Problems with SMC-2 related additional sense code definitions were corrected.

In the IKEv2-SCSI SA Cryptographic Algorithms payload format, the required value in the NUMBER OF ALGORITHM DESCRIPTORS field was corrected from five to six so that it matches the number of entries in the ordered list of algorithm descriptors.

08-008r2 failed to cover the case of non-zero buffer offsets received on the established I_T nexus. The obvious intent was that such commands be processed as described elsewhere in the subclause. This was added as an editorial change.

Based on input from the SAM-4 letter ballot review, all instance of specified status return values were evaluated for the usage of 'terminated' or 'completed' in the description of the command. With the exception of CHECK CONDITION status, commands were described as being 'completed'. For CHECK CONDITION status, the command was described as being 'terminated'. Where such wording could not be applied the terms 'return' and 'response' were used. A few instance of 'task' were replaced with 'command'.

'Principals of SAs' was changed to 'Principles of SAs'.

2.19 Revision 15 (20 June 2008)

The following T10 approved proposals were incorporated in this SPC-4 revision:

08-145r3 Capability-based Command Security (CbCS) [the rewrite]
08-156r2 Non-volatile cache becoming volatile
08-177r0 Bogus Sense Key in SA Creation Parameter Data Error Handling
08-178r0 IKEv2-SCSI Message ID field size correction
08-180r0 SCSI device name cleanup in SPC-4
08-194r1 Common definition for CODE SET
08-199r1 Even more log page cleanup
08-214r1 Allowing ESP-SCSI descriptors in variable length CDBs
08-231r1 SPEC_I_PT clarification

The instruction in 08-053r1(Persistent reservation fixes) were not fully implemented in the 'Register behaviors for a REGISTER AND IGNORE EXISTING KEY service action' table during the preparation of r13. Dangling rows were removed in this revision.

During the incorporation of 08-214r1, several instances where SA protection was restricted to command parameter data were broadened to cover selected CDBs too.

To match SAM-4 Letter Ballot changes:

a) QUERY UNIT ATTENTION was changed to QUERY ASYNCHRONOUS EVENT in the REPORT SUPPORTED TASK MANAGEMENT FUNCTIONS command definition; and
b) 'initial priority' was changed to 'initial command priority' in the REPORT PRIORITY command definition, SET PRIORITY command definition, and the Control Extension mode page.

References to the under development ADC-3 were added wherever appropriate to support the Data Encryption Configuration security protocol.

The POWER CONDITION field name was modified to match the spelling in SBC-3 throughout.

Bogus IKEv2-SCSI Timeout Values an Identification – Device Server next payload code values as well as incorrect references to the IKEv2-SCSI Timeout Values payload were corrected in the per/step next payloads table.

Several mode page entries in the numeric codes annex were updated to match the latest ADC-3 revision.

Several grammar errors were corrected and not marked with change bars.

In hopes of getting a jump on ISO SPC-4, all uses of dashes in range expressions (e.g., 5Ah-FFh) were changed to use the word 'to' (e.g., 5Ah to FFh). These changes were not marked with change bars.

2.20 Revision 16 (29 July 2008)

The following T10 approved proposals were incorporated in this SPC-4 revision:

08-116r3 Protection Type 3 Reference Tag Clarification
08-254r0 Protocol Specific diagnostic page
08-261r1 Parsing variable-length parameter lists and data
08-262r0 SAI, SQN, IV, & Data must be contiguous
08-287r1 ATA Information VPD page nits

In the table defining the ID TYPE field for the Identification – Application Client payload and Identification – Device Server payload, the codepoint values for ID_FC_NAME and ID_KEY_ID were swapped so that the match the values in RFC 4306.

The conventions subclause was updated to include the latest descriptions of the conventions for number value ranges and lettered/numbered lists. Were the new number convention allowed, 'through' was replaced with 'to'.

Several incorrect subclause references discovered during the ISO SPC-3 review and an incorrect table reference were corrected but not marked with change bars.

2.21 Revision 17 (13 November 2008)

The following T10 approved proposals were incorporated in this SPC-4 revision:

08-263r3 Too much to do … Not enough Progress Counters
08-327r2 Restrict SPC-4 and SAM-5 codes to FC-SB-4 use
08-346r2 Correction to IKEv2-SCSI Certificate Request Payload
08-450r1 Add Restricted keyword to the Style Guide

The last paragraph in the IKEv2-SCSI subclause titled 'Errors with high denial of service attack potential' was modified to clarify when the requirements on the SENSE KEY SPECIFIC field apply.

In the informative code values annex, the Parallel ATA Control mode page was constrained to SBC disk devices and the Serial ATA Control mode page was removed.

One instance of NOT READY that 08-177r0 missed was changed to ILLEGAL REQUEST.

The expected and ignored keyword definitions were removed in accordance with decisions reached at the September and November CAP meetings, respectively.

A couple of instances of SET TARGET GROUPS command were corrected to SET TARGET PORT GROUPS command.

In several security algorithms codes tables in the key derivation model subclauses, code values of the form 000x xxxxh were changed to 800x xxxxh to conform to the decision to have IETF-based algorithms numbered 800x xxxxh.

Several minor typos and bogus cross references were corrected and not marked with change bars.

2.22 Revision 18 (18 February 2009)

The following T10 approved proposals were incorporated in this SPC-4 revision:

08-281r5 UAS Clause 4 (Model) {new ASC only}
08-356r5 Thin Provisioning Commands
08-396r3 Reporting support for all DIF types
08-415r4 Adding a Protection Information Interval
08-423r4 IKEv2-SCSI Authentication
09-002r0 Making the PPC bit in the LOG SENSE command obsolete
09-019r0 Reclaim some Obsolete bits for future use
09-031 Mark Service Action In/Out (16) service action value 0Fh Restricted for SAT-3
09-034r1 EMail comment received from Mike Berhan busTRACE
09-054r1 Adding more idle power options

For consistency with the T10 Style Guide, the notation for byte encoded character strings was changed from enclosing such strings in double quotes to enclosing them in single quotes. This change was made throughout the working draft and noted with change bars.

Several minor typos were corrected and not marked with change bars.

2.23 Revision 19 (2 May 2009)

The following T10 approved proposals were incorporated in this SPC-4 revision:

08-386r5 Cache hits and power on statistics
09-018r2 Implicit transition (RTPG) timeout notification enhancements
09-072r0 Substantive corrections to 09-054r1 -- Adding more idle power options
09-118r0 Mode Page 18h, Change 'Reserved' field to 'Protocol specific' field

For consistency with the other LOGICAL UNIT NOT READY additional sense codes, the name for ASC/ASCQ 04h/14h was modified as follows: LOGICAL UNIT NOT READY, SPACE ALLOCATION IN ~~PROCESS~~ PROGRESS.

Several typos were corrected in the paragraph describing parameter codes 0004h and 0006h in the Start-Stop Cycle Counter log page.

Two minor typos were corrected and not marked with change bars.

2.24 Revision 20 (22 May 2009)

The following T10 approved proposals were incorporated in this SPC-4 revision:

09-033r4 Device constituent VPD page
09-087r2 Rectifying conflicts for DEVOFFL and UNITOFFL

To better reflect the information available on the referenced web site, the description for security protocol code EDh was changed to 'SDcard TrustedFlash Security Systems Specification 1.1.3'. This topic was discussed by the CAP working group (minutes in 09-175).

To match the request-response model parameter names in 4.2, the capitalization of all instances of data-in buffer and data-out buffer were corrected to Data-In Buffer and Data-Out Buffer, respectively. These changes were not marked with change bars.

2.25 Revision 20a (25 May 2009)

All lists were updated to use the current ISO/T10 capitalization rules. In a nutshell, the first word of a list entry is capitalized only if that same word would be capitalized if it appeared in the middle of a normal sentence.

The base font was changed from Helvetica to Arial.

None of these changes were marked with change bars, except the change bars shown here. The change bars from revision 20 were left unchanged.

2.26 Revision 21 (25 August 2009)

The following T10 approved proposals were incorporated in this SPC-4 revision:

09-011r8 Thin Provisioning Threshold Notification
09-162r2 Revise LOG SELECT w/ PCR=1 wording
09-167r2 PL_OC2 I_T Nexus loss fixes [additional sense code]
09-178r2 Add IPv6 target descriptor to XCOPY
09-199r1 Add IP Copy Engine service type to page 85h
09-200r0 Update Access Controls for READ CAPACITY(16)
09-208r1 SPC-3 vs SPC-4 Descriptor format REQUEST SENSE
09-211r1 How many sense data descriptors are too many?
09-219r1 Fix errors in description of log pages 0Eh and 1Ah
09-233r1 AC/DC MSB/LSB
09-237r0 Add clarification to Supported Log Pages log pages

As part of incorporating 09-233r1, MSB/LSB markings were also added to the PARAMETER VALUE field in the log parameter definition table. Like the fields cited in 09-233r1, log parameter values can be defined as either single valued or multi-valued.

As part of incorporating 09-237r1, the page structure format tables in all cited subclauses were modernized and the descriptions of values to which fields were required to be set was changed to the 'set as show in table x' form.

The power conditions model overview was modified to reflect the fact that a low power condition (i.e., stopped) is defined in SBC-3.

At the request of the SAT-2 editor, the ATA-PASSTHROUGH Results log page was added to the numeric codes annex.

At the request of the SSC-3 editor:

a) the READ(16) and WRITE(16) commands were made optional for tapes in the numeric codes annex (this change was made in the database and thus not identified with a change bar);
b) the tape usage of the Format Status log page was removed from the numeric codes annex;
c) the Current Service Information log page (2Dh/00h) was added to the numeric codes annex and marked for tapes usage;
d) the Medium Configuration mode page (1Dh/00h) was added to the numeric codes annex and marked for tapes usage;
e) the Automation Device Serial Number VPD page (B3h) was added to the numeric codes annex and marked for tapes usage; and
f) the MAM glossary entry was updated to mirror the one present in SSC-3.

Per an agreement reached in the CAP working group, all page code definition tables were moved to the beginning of the subclause that defines that page format and usage. The page names or descriptions were moved to the left-most column and sorted in alphabetical order. A single row indicating all other codes are reserved was added (or modified) to appear as the last row of the table body. References to the numeric codes annex were modified to name the correct annex subclause.

The SCSI standards family subclause was copied from SAM-5 r02 into this working draft. The figure was altered slightly to reflect its presence in a different standard. The SCSI standards family was removed from clause 1 and placed in the new subclause, if it was not already present there (this movement was not marked with change bars).

The table headings in the normative additional sense codes table and in all tables in numeric codes annex were updated to reference more current versions of SES, ADC, and OSD.

Several editorial corrections (e.g., “a a non-SCSI”) were made but not marked with change bars.

2.27 Revision 22 (28 September 2009)

The following T10 approved proposals were incorporated in this SPC-4 revision:

09-281r0 Request ASC/ASCQ for insufficient power condition [ASC only]
09-342r0 Preempt And Abort has a cross-nexus effect

To emphasize the fact that the list which follows is incomplete, the text that introduces the list of preparatory actions for an EXTENDED COPY command was changed from “Such preparatory actions include:” to “Such preparatory actions include but are not limited to:”.

The Numeric Codes annex list of log page codes reserved to the MS59 Std. (contact AIIM C21 comm.) was corrected to not include the all subpages log page subcode.

The note after the supported subpage descriptor table in the supported subpages log page was corrected to match the table it describes.

Several editorial corrections (e.g., “each of specific command” was changed to “of each specific”) were made but not marked with change bars.

2.28 Revision 23 (9 February 2010)

The following T10 approved proposals were incorporated in this SPC-4 revision:

09-100r5 SBC-3 - Atomic COMPARE AND WRITE command [operation code only]
09-146r4 Tape end-to-end logical block protection Inquiry … Data
09-241r6 SCSI Referrals [Penokie]
09-250r4 Change power condition timer behavior when commands are received
09-294r2 Adding a new VPD bit indicating download microcode behavior
09-303r1 SSC-4 Device and volume log page enhancements [log page code only]
09-322r2 SBC-3 Add log page for solid state drive (SSD) [log page code only]
09-355r2 MAM Write-through cache bit
09-356r5 SSC-4 Append-only mode [operation code and ASC only]
09-359r2 Conflicted List Format Log Parameter Rules
09-363r4 XCOPY with no LIST IDENTIFIER
09-367r0 Volume Coherency MAM parameter
09-369r1 No UA on RELEASE
09-370r1 IKEv2-SCSI Key Exchange Clarification
09-379r1 Another Encryption MAM Attribute
09-381r2 Adding a Copy Manager class into SAM-5
09-401r1 Volume Coherency MAM parameter [MAM Attributes only]
09-408r2 Task to Command conversion
10-005r1 Add Disable PI Check on xxPROTECT=0 feature
10-016r2 Persistent Reservations Clarification

During the incorporation of 09-146r4, discussion of the SPT field being “ignored” was removed and replaced with text that makes currently undefined uses of the field reserved.

During incorporation of 09-294r2, the ACTIVATE MICROCODE field definition was moved out of the model clause into the Extended INQUIRY Data VPD page subclause. This eliminated the only one-field definition table in the model clause.

The definition of the enable warning (EWASC) bit in the Informational Exceptions mode page was updated to clarify the meaning of “warning”.

Because the applicable SAT-3 proposal was withdrawn, two Restricted service actions were changed back to Reserved, one each in the SERVICE ACTION IN(16) and SERVICE ACTION OUT(16) commands, respectively.

The description of the zero case in the QUEUE ALGORITHM MODIFIER field was modified to remove the word 'would'.

For consistency with FC-FS-x, the N_PORT_NAME field in the TransportID for initiator ports using SCSI over Fibre Channel was changed from “Physical Log In (PLOGI)” to “N_Port login (PLOGI)”.

A pair of bogus descriptor-header rows in the Supported Log Pages format table were corrected.

The ADI reference for protocol identifier values was changed from ADT to ADT-2.

Several instanced of “insure” were changed to “ensure”.

In the description of how the EXTENDED COPY command is affected by task management functions, the processor of the task management function was changed from the “device server” to the “task manager”.

The URL for universal time was updated in the glossary.

Several other, minor changes were made but not marked with change bars.

2.29 Revision 24 (14 April 2010)

The following T10 approved proposals were incorporated in this SPC-4 revision:

09-357r4	Relationship between power conditions and background tasks
10-043r1	Correct usage of term volume in MAM attributes
10-065r1	XCOPY IP Copy Service target descriptor
10-073r0	Extended Logical Unit Address Method added to Report LUNs Command
10-120r0	Fix to REPORT SUPPORTED TASK MANAGEMENT FUNCTIONS command

In honor of the SPL letter ballot, several instances of SAS-2 were updated to mention SPL instead.

Since both the ANSI and ISO/IEC versions of SAM-3 have been withdrawn, all references to it were removed. References for several previous in-development standards were moved to published and new references were added to the follow-on in-development working drafts.

A MAINTENANCE (OUT) service action and several additional sense codes defined by the SMC-3 group were added, and the Numeric Codes annex table for MAINTENANCE (OUT/IN) service actions was enhanced slightly.

A new Overview subclause was added to the parameters for all device types clause in which hot links perform the function of bookmarks.

The reference for user data segment referral sense data descriptor was corrected from SSC-3 to SBC-3.

The normative reference for IEC 60027-2 was updated, and 'gebi' was changed to 'gibi' in the binary multipliers table.

The INCITS online store URLs were updated, and 'Information Technology Institute' was modernized to 'Information Technology Industry Council'.

Other trivial editorial changes were made and not marked with change bars.

2.30 Revision 25 (23 June 2010)

The following T10 approved proposals were incorporated in this SPC-4 revision:

10-108r2 READ BUFFER from echo buffer mode error
10-115r2 Power conditions transition modifications
10-133r4 Pollable Device State model
10-135r0 Obsolete VPD pages in SPC-4 Annex D
10-148r1 More about power conditions and background tasks
10-156r0 Allow RECEIVE COPY RESULTS through reservations
10-161r1 VPD page generic format
10-179r1 Remove implicit head of queue rule that is in SAM-5

In the Nonvolatile memory considerations for preserving persistent reservations and registrations model subclause, a cross reference to SBC-3 was added to the START STOP UNIT command usage.

In the description of the RECEIVE COPY RESULTS command with FAILED SEGMENT DETAILS service action, the name of the command to be terminated with CHECK CONDITION was corrected.

Introductory text (that ends with a colon) was added to the a,b lists in the right-most column of the IKEv2-SCSI header checking of SAIs table.

The command ATA COMMAND PASS THROUGH was changed to ATA PASS THROUGH in the opcodes annex. This change was not marked with change bars because it was propagated from a database.

Other minor changes were made without use of change bars.

2.31 Revision 26 (16 August 2010)

The following T10 approved proposals were incorporated in this SPC-4 revision:

10-032r4 TMF timeouts
10-138r1 New Supported Attributes service action for READ ATTRIBUTE
10-144r3 Self-test and SEND DIAGNOSTIC command clean-up
10-177r2 Page_0 Mode Page Format & Sub-page 00h
10-201r1 SET IDENTIFYING INFORMATION clarifications
10-208r1 AER Obsolete bits in SPC-4
10-210r3 Tweaks to power state machines found during SPL LB Resolution

In keeping with a change in 10-144r3, definition paragraphs were added for all instances of the OPERATION CODE field and CONTROL field.

In the Sense key specific sense data descriptor definition, the description of the DESCRIPTOR TYPE field was corrected to contain the proper numeric value.

References to FCP-2 were updated to FCP-4.

Other minor changes were made without use of change bars.

2.32 Revision 27 (11 October 2010)

The following T10 approved proposals were incorporated in this SPC-4 revision:

09-146r4 SSC-4: End-to-end Logical Block Protection {new ASC only, this time, other changes in r23}
09-262r8 Bitmaps on Disk - Detailed Changes {new ASC only}
09-360r9 Log Page clean-up
10-162r2 Application Tag Mode Page
10-242r2 Too many words about pollable sense data in SPC-4
10-270r1 Possible bugs in Self-test and SEND DIAGNOSTIC command clean-up
10-271r2 Fixing the TMF Timeout REPD bit
10-280r1 Reject Write Without Protection information
10-282r1 Expand definition of self-test completion time field
10-289r1 Power Management Tweak / Clarification
10-300r1 Security protocol ID for JEDEC UFS RPMB Protocol

Mistakes made during the r26 incorporation of 10-138r1 (New Supported Attributes service action for READ ATTRIBUTE) were corrected.

Instances where text refers to a table to define the value in a field where updated to the rules agreed upon by the CAP working group.

Other minor changes were made without use of change bars.

2.33 Revision 28 (16 November 2010)

The following T10 approved proposals were incorporated in this SPC-4 revision:

10-191r2 EXTENDED COPY Descriptor format Sense Data
10-220r4 SSC-4: Verify Files and Logical Block Protection Method {new ASC only}
10-350r0 Clarify TUR - GOOD status can now return sense data

During the incorporation of 10-191r2, a terminology conflict was detected between the wording in the EXTENDED COPY command (where 'sense data' was used) and the new sense data descriptor (where ‘additional sense data’ was used). A search of SPC-4 r27 revealed that 'additional sense data' is already used to describe a portion of the sense data. Since the purpose of the new sense data descriptor is to forward all the sense data returned by a third party, the names in the sense data descriptor were revised accordingly.

The definition of the reject write without protection (RWWP) bit in the Control mode page was modified as requested by the SBC-3 editor. The changes were not substantive.

The description of the PARAMETER CODE field in the General Usage Application Client log parameter was corrected to match the descriptions in all other log parameters. The cross reference for all PARAMETER CODE field was modified to point to the correct subclause.

One minor change (i.e., the addition of an 'in') was made without use of change bars.

2.34 Revision 29 (17 January 2011)

The following T10 approved proposals were incorporated in this SPC-4 revision:

11-010r0	Obsolete READ CAPACITY command PMI bit
11-019r1	Logical Block Provisioning corrections [Codes Annex only]
11-021r2	RCR OPERATING PARAMETERS - SNLID bit cleanup
11-026r1	Informational Exceptions Mode Page Modernization in SPC-4, and …
11-060r0	Remove SAS-2 references in power state machines

Four broken cross references caused by the removal of the Informational Exceptions Control mode page subclause as specified in 11-026r1 were fixed by replacing "(see 7.5.12)" with "(see applicable command standard)".

The model description of the extended self-test was corrected to reference both the EXTENDED SELF-TEST COMPLETION MINUTES field in the Extended INQUIRY Data VPD page and the EXTENDED SELF-TEST COMPLETION TIME field in the Control mode page. In this same subclause, a sentence fragment introduced during the incorporation of 10-144r3 in r25 was removed.

In the Power Management model, a reference to MMC-6 was added as an example of the definition of a command which modifies the model. This makes MMC-6 more like SBC-3 in regards to its relationship to the Power Management model.

In the peripheral device type table clause 8 was added as the reference for well known logical units.

A reference to ATA/ATAPI-7 was changed to ACS-2.

The cross reference to log page definitions in the definition of the RLEC bit was changed to cover the entire log pages subclause because sites sprinkled throughout the subclause define RLEC bit functions.

The table which describes the version field in the standard Inquiry data was restructured as requested by the January SPC-4 editing group.

The table headers for all DTL… tables (e.g., the additional sense codes tables) were updated to identify the latest versions of the command standards. This change was **not** applied to areas other than the cited tables. ADC-3 devices were added as supporting the Tape Diagnostic Data log page. Tapes were added as supporting the Data Compression log page and Control Data Protection mode page.

Two typos were corrected in the definition of the Forwarded Sense Data descriptor.

A confusing sentence in the EXTENDED COPY subclause in which multiple ands cannot easily be distinguished was restructured to use unordered lists to reduce the chances for misreading them.

Six minor changes (e.g., changing an instance of 'specifies' to 'indicates') were made without use of change bars.

Neither the additional sense code tables nor the operation codes table were updated in r29 because no requests for changes had been detected.

2.35 Revision 30 (31 March 2011)

The following T10 approved proposals were incorporated in this SPC-4 revision:

10-390r1 Forwarding GOOD sense data
11-004r0 Remove REPORT LUNS note about SPC
11-087r1 Standard INQUIRY data target device vs logical unit clarifications

The paragraphs that describe the OPERATION CODE field and SERVICE ACTION field for each command were updated to the agreed wording. In many cases a paragraph was added for the SERVICE ACTION field.

The definition of the PROTECT bit in the Standard INQUIRY Data was modified so that the reference to the SPT field in the Extended Inquiry Data VPD page does not look like the description of another Standard INQUIRY Data field.

In the Set Master Key – Seed Exchange CbCS page definition, a clearly bogus 'shall be' was changed to 'is' as follows:

> If the PAGE LENGTH field ~~shall be~~ is set to a value that is inconsistent with the Diffie-Hellman algorithm specified for the CbCS master key update CCS, 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 PARAMETER LIST.

The lists in the *Selecting pollable sense data to return* subclause were modified to make them more readable.

Unusual wording of the command introduction was normalized for the SEND DIAGNOSTIC command, SET TARGET PORT GROUPS command, and SET TIMESTAMP command.

The description of the SET TIMESTAMP command was corrected to discuss a 'parameter list' (output data for the command) instead of 'parameter data' (input data returned by the command).

Since the timestamp field has no internal structure, (MSB)/(LSB) markings were added to all tables that include it.

The nomenclature in the table that defines the FORMAT AND LINKING field was changed to make it consistent with the remainder of the log page parameter data clause.

The nomenclature in the table of TransportID formats for specific SCSI transport protocols was tweaked slightly to improve consistency.

Sundry minor editorial changes (e.g., 'power-on' changed to 'power on') without benefit of change bars.

2.36 Revision 31 (13 June 2011)

The following T10 approved proposals were incorporated in this SPC-4 revision:

11-011r4 Add sanitize command
11-114r2 CFA Profile Support VPD page
11-185r1 Returning other sense data for a REQUEST SENSE command
11-186r1 PARAMETER POINTER field values in the LOG SENSE command
11-187r2 Modes using the BUFFER CAPACITY field from the READ BUFFER command
11-216r1 Error conditions during resetting of log parameters
11-222r0 Minutes of the ADI Working Group (action item 11-008)
11-235r0 One more Log Page correction

The code assignments (e.g., additional sense codes) in the following T10 approved proposals were incorporated in this SPC-4 revision:

10-284r7 Coordination of SAS Device Power Consumption
11-204r0 Incorporate Additional Sense Code 23h/00h (only) from this doc

The following changes from the T10 Style Guide 2nd edition (10-316r6) were incorporated in this SPC-4 revision:

Lists convention designed to satisfy ISO/IEC
Copied Numeric and character conventions
Data structure format tables were updated

A few bugs (e.g., a table with two byte n rows) were found and fixed during the data structure format updates. None of the updated tables were marked with change bars.

The table title paragraph and variable styles described in 11-189r0 were incorporated as an editorial change to simplify the transition to ISO source formats.

In the table that defines the LOG SELECT command PC field, references to the table footnote were added to the first two table rows (i.e., threshold values and cumulative values).

In the LOG SELECT command table that defines the PAGE CODE and PAGE SUBCODE fields, the reference to a “previous version of this standard” was changed to “SPC-3”.

The VPD page descriptions were updated to use the “shall be set as shown in table …” nomenclature.

Several “Subclause” column headings were changed to “Reference”.

The acronym definition for ISO was modernized.

Other minor editorial changes were made without benefit of change bars.

ANSI (r)
INCITS.*:200x**

American National Standards for Information Systems -

SCSI Primary Commands - 4 (SPC-4)

Secretariat
InterNational Committee for Information Technology Standards

Approved mm dd yy

American National Standards Institute, Inc.

Abstract

This standard defines the device model for all SCSI devices. This standard defines the SCSI commands that are basic to every device model and the SCSI commands that may apply to any device model.

American National Standard

Approval of an American National Standard requires verification 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 judgment 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 effort be made toward 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 confirming to the standards.

The American National Standards Institute does not develop standards and will in no circumstances give interpretation on any 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 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 licenses are not required to avoid infringement in the use of this standard.

Published by
American National Standards Institute
11 West 42nd Street, New York, NY 10036

Copyright 6/13/11 by American National Standards Institute
All rights reserved.

Printed in the United States of America

Contents

Page

Foreword lxi

Introduction lxiii

SCSI standards family lxiii

1 Scope 1

2 Normative references 1
2.1 General 1
2.2 Approved references 1
2.3 References under development 4
2.4 NIST References 5
2.5 IETF References 5
2.6 OMG references 6
2.7 CompactFlash association references 7
2.8 JEDEC references 7

3 Definitions, symbols, abbreviations, and conventions 8
3.1 Definitions 8
3.2 Symbols and acronyms 20
3.3 Keywords 22
3.4 Conventions 23
3.5 Numeric and character conventions 24
3.5.1 Numeric conventions 24
3.5.2 Units of measure 25
3.5.3 Byte encoded character strings conventions 25
3.6 Bit and byte ordering 26
3.7 Notation conventions 27
3.7.1 Notation for procedure calls 27
3.7.2 Notation for state diagrams 28
3.7.3 Notation for UML figures 28
3.7.3.1 Introduction 28
3.7.3.2 Class notation 29
3.7.3.3 Class association relationships notation 30
3.7.3.4 Class aggregation relationships notation 31
3.7.3.5 Class generalization relationships notation 33
3.7.3.6 Class dependency relationships notation 34
3.7.3.7 Object notation 34

4 General Concepts 35
4.1 Introduction 35
4.2 The request-response model 35
4.3 The Command Descriptor Block (CDB) 35
4.3.1 CDB usage and structure 35
4.3.2 The fixed length CDB formats 36
4.3.3 The variable length CDB formats 40
4.3.4 Extended CDBs 41
4.3.4.1 XCDB model 41
4.3.4.2 The XCDB format 42
4.3.5 Common CDB fields 43

4.3.5.1 Operation code ... 43
4.3.5.2 Service action ... 44
4.3.5.3 Logical block address ... 44
4.3.5.4 Transfer length ... 44
4.3.5.5 Parameter list length ... 45
4.3.5.6 Allocation length ... 45
4.3.5.7 Control ... 45
4.4 Data field requirements ... 45
4.4.1 ASCII data field requirements ... 45
4.4.2 Null data field termination and zero padding requirements ... 46
4.4.3 Variable type data field requirements ... 46
4.5 Sense data ... 46
4.5.1 Sense data introduction ... 46
4.5.2 Descriptor format sense data ... 48
4.5.2.1 Descriptor format sense data overview ... 48
4.5.2.2 Information sense data descriptor ... 50
4.5.2.3 Command-specific information sense data descriptor ... 50
4.5.2.4 Sense key specific sense data descriptor ... 51
4.5.2.4.1 Sense key specific sense data descriptor introduction ... 51
4.5.2.4.2 Field pointer sense key specific data ... 52
4.5.2.4.3 Actual retry count sense key specific data ... 53
4.5.2.4.4 Progress indication sense key specific data ... 53
4.5.2.4.5 Segment pointer sense key specific data ... 54
4.5.2.4.6 Unit attention condition queue overflow sense key specific data ... 54
4.5.2.5 Field replaceable unit sense data descriptor ... 55
4.5.2.6 Progress indication sense data descriptor ... 55
4.5.2.7 Forwarded sense data ... 56
4.5.2.8 Vendor specific sense data descriptors ... 57
4.5.3 Fixed format sense data ... 58
4.5.4 Current information ... 59
4.5.5 Deferred errors ... 59
4.5.6 Sense key and additional sense code definitions ... 61
4.6 Secure random numbers ... 79

5 Model common to all device types ... 80
5.1 Introduction to the model common to all device types ... 80
5.2 Important commands for all SCSI device servers ... 80
5.2.1 Commands implemented by all SCSI device servers ... 80
5.2.2 Commands recommended for all SCSI device servers ... 80
5.2.3 Using the INQUIRY command ... 80
5.2.4 Using the REPORT LUNS command ... 80
5.2.5 Using the TEST UNIT READY command ... 81
5.2.6 Using the REQUEST SENSE command ... 81
5.3 Implicit head of queue ... 81
5.4 Parameter rounding ... 81
5.5 Parsing variable-length parameter lists and parameter data ... 81
5.6 Pollable condition information ... 82
5.6.1 Overview ... 82
5.6.2 REQUEST SENSE pollable sense data ... 82
5.6.2.1 Overview ... 82
5.6.2.2 Selecting pollable sense data to return ... 83
5.6.2.3 Returning one or more progress indications ... 83
5.6.3 Log parameter pollable device condition information ... 84
5.7 Device specific background functions ... 84

5.7.1 Overview ... 84
5.7.2 Suspending and resuming device-specific background functions ... 85
5.8 Self-test operations ... 85
5.8.1 Overview ... 85
5.8.2 Default self-test ... 85
5.8.3 The short self-test and extended self-test ... 86
5.8.4 Self-test modes ... 86
5.8.4.1 Self-test modes overview ... 86
5.8.4.2 Foreground mode ... 86
5.8.4.3 Background mode ... 87
5.8.4.4 Features common to foreground and background self-test modes ... 89
5.9 Reservations ... 90
5.9.1 Persistent Reservations overview ... 90
5.9.2 Third party persistent reservations ... 95
5.9.3 Exceptions to SPC-2 RESERVE and RELEASE behavior ... 95
5.9.4 Persistent reservations interactions with IKEv2-SCSI SA creation ... 96
5.9.5 Preserving persistent reservations and registrations ... 96
5.9.5.1 Preserving persistent reservations and registrations through power loss ... 96
5.9.5.2 Nonvolatile memory considerations for preserving persistent reservations and registrations ... 97
5.9.6 Finding persistent reservations and reservation keys ... 97
5.9.6.1 Summary of commands for finding persistent reservations and reservation keys ... 97
5.9.6.2 Reporting reservation keys ... 97
5.9.6.3 Reporting the persistent reservation ... 97
5.9.6.4 Reporting full status ... 98
5.9.7 Registering ... 98
5.9.8 Registering and moving the reservation ... 102
5.9.9 Reserving ... 103
5.9.10 Persistent reservation holder ... 104
5.9.11 Releasing persistent reservations and removing registrations ... 104
5.9.11.1 Summary of service actions that release persistent reservations and remove registrations ... 104
5.9.11.2 Releasing ... 106
5.9.11.3 Unregistering ... 106
5.9.11.4 Preempting ... 107
5.9.11.4.1 Overview ... 107
5.9.11.4.2 Failed persistent reservation preempt ... 109
5.9.11.4.3 Preempting persistent reservations and registration handling ... 109
5.9.11.4.4 Removing registrations ... 110
5.9.11.5 Preempting and aborting ... 110
5.9.11.6 Clearing ... 111
5.10 Multiple target port and initiator port behavior ... 112
5.11 Target port group access states ... 112
5.11.1 Target port group access overview ... 112
5.11.2 Asymmetric logical unit access ... 112
5.11.2.1 Introduction to asymmetric logical unit access ... 112
5.11.2.2 Explicit and implicit asymmetric logical unit access ... 114
5.11.2.3 Discovery of asymmetric logical unit access behavior ... 114
5.11.2.4 Target port asymmetric access states ... 114
5.11.2.4.1 Target port asymmetric access states overview ... 114
5.11.2.4.2 Active/optimized state ... 115
5.11.2.4.3 Active/non-optimized state ... 115
5.11.2.4.4 Standby state ... 115
5.11.2.4.5 Unavailable state ... 116
5.11.2.4.6 Offline state ... 116
5.11.2.4.7 Logical block dependent state ... 117

5.11.2.5 Transitions between target port asymmetric access states ... 117
5.11.2.6 Preference Indicator ... 118
5.11.2.7 Implicit asymmetric logical units access management ... 118
5.11.2.8 Explicit asymmetric logical units access management ... 119
5.11.2.9 Behavior after power on, hard reset, logical unit reset, and I_T nexus loss ... 119
5.11.2.10 Behavior of target ports that are not accessible from the service delivery subsystem ... 119
5.11.3 Symmetric logical unit access ... 119
5.12 Power management ... 119
5.12.1 Power management overview ... 119
5.12.2 Active power condition ... 121
5.12.3 Idle power conditions ... 121
5.12.4 Standby power conditions ... 121
5.12.5 Power management pollable sense data ... 122
5.12.6 Power condition state machine ... 123
5.12.6.1 Power condition state machine overview ... 123
5.12.6.2 PC0:Powered_On state ... 124
5.12.6.2.1 PC0:Powered_On state description ... 124
5.12.6.2.2 Transition PC0:Powered_On to PC4:Active_Wait ... 124
5.12.6.3 PC1:Active state ... 124
5.12.6.3.1 PC1:Active state description ... 124
5.12.6.3.2 Transition PC1:Active to PC5:Wait_Idle ... 125
5.12.6.3.3 Transition PC1:Active to PC6:Wait_Standby ... 125
5.12.6.4 PC2:Idle state ... 125
5.12.6.4.1 PC2:Idle state description ... 125
5.12.6.4.2 Transition PC2:Idle to PC4:Active_Wait ... 125
5.12.6.4.3 Transition PC2:Idle to PC5:Wait_Idle ... 126
5.12.6.4.4 Transition PC2:Idle to PC6:Wait_Standby ... 126
5.12.6.5 PC3:Standby state ... 126
5.12.6.5.1 PC3:Standby state description ... 126
5.12.6.5.2 Transition PC3:Standby to PC4:Active_Wait ... 126
5.12.6.5.3 Transition PC3:Standby to PC6:Wait_Standby ... 127
5.12.6.6 PC4:Active_Wait state ... 127
5.12.6.6.1 PC4:Active_Wait state description ... 127
5.12.6.6.2 Transition PC4:Active_Wait to PC1:Active ... 128
5.12.6.7 PC5:Wait_Idle state ... 128
5.12.6.7.1 PC5:Wait_Idle state description ... 128
5.12.6.7.2 Transition PC5:Wait_Idle to PC2:Idle ... 128
5.12.6.8 PC6:Wait_Standby state ... 129
5.12.6.8.1 PC6:Wait_Standby state description ... 129
5.12.6.8.2 Transition PC6:Wait_Standby to PC3:Standby ... 129
5.13 Medium auxiliary memory ... 129
5.14 Error history ... 130
5.14.1 Error history overview ... 130
5.14.2 Retrieving error history with the READ BUFFER command ... 131
5.14.3 Adding application client error history with the WRITE BUFFER command ... 133
5.14.4 Clearing error history with the WRITE BUFFER command ... 133
5.15 Device clocks ... 133
5.16 Security Features ... 134
5.16.1 Security goals and threat model ... 134
5.16.1.1 Overview ... 134
5.16.1.2 Security goals ... 134
5.16.1.3 Threat model ... 135
5.16.1.4 Types of attacks ... 136
5.16.1.5 SCSI security considerations ... 136

5.16.2 Security associations ... 137
5.16.2.1 Principles of SAs ... 137
5.16.2.2 SA parameters ... 138
5.16.2.3 Creating an SA ... 141
5.16.3 Key derivation functions ... 141
5.16.3.1 Overview ... 141
5.16.3.2 IKEv2-based iterative KDF ... 142
5.16.3.3 HMAC-based KDFs ... 142
5.16.3.4 AES-XCBC-PRF-128 IKEv2-based iterative KDF ... 143
5.16.4 Using IKEv2-SCSI to create a security association ... 144
5.16.4.1 Overview ... 144
5.16.4.2 IKEv2-SCSI Protocol summary ... 147
5.16.4.3 IKEv2-SCSI Authentication ... 150
5.16.4.3.1 Overview ... 150
5.16.4.3.2 Pre-shared key authentication ... 151
5.16.4.3.3 Digital signature authentication ... 152
5.16.4.3.3.1 Overview ... 152
5.16.4.3.3.2 Certificates and digital signature authentication ... 152
5.16.4.3.3.3 Example of certificate use for digital signature authentication ... 153
5.16.4.3.3.4 Handling of the Certificate Request payload and the Certificate payload ... 153
5.16.4.3.4 Constraints on skipping the Authentication step ... 153
5.16.4.4 Summary of IKEv2-SCSI shared keys nomenclature and shared key sizes ... 155
5.16.4.5 Device Server Capabilities step ... 156
5.16.4.6 IKEv2-SCSI Key Exchange step ... 158
5.16.4.6.1 Overview ... 158
5.16.4.6.2 Key Exchange step SECURITY PROTOCOL OUT command ... 158
5.16.4.6.3 Key Exchange step SECURITY PROTOCOL IN command ... 159
5.16.4.6.4 Key Exchange step completion ... 160
5.16.4.6.5 After the Key Exchange step ... 160
5.16.4.7 IKEv2-SCSI Authentication step ... 160
5.16.4.7.1 Overview ... 160
5.16.4.7.2 Authentication step SECURITY PROTOCOL OUT command ... 161
5.16.4.7.3 Authentication step SECURITY PROTOCOL IN command ... 162
5.16.4.8 Generating shared keys ... 163
5.16.4.8.1 Overview ... 163
5.16.4.8.2 Generating shared keys when the Authentication step is skipped ... 164
5.16.4.8.3 Generating shared keys when the Authentication step is processed ... 164
5.16.4.8.4 Initializing shared key generation ... 164
5.16.4.8.4.1 Initializing for SA creation shared key generation ... 164
5.16.4.8.4.2 Initializing for generation of shared keys used by the created SA ... 165
5.16.4.8.5 Generating shared keys used for SA management ... 165
5.16.4.8.6 Generating shared keys for use by the created SA ... 166
5.16.4.9 IKEv2-SCSI SA generation ... 167
5.16.4.10 Abandoning an IKEv2-SCSI CCS ... 169
5.16.4.11 Deleting an IKEv2-SCSI SA ... 170
5.16.5 Security progress indication ... 170
5.16.6 Command security ... 171
5.16.6.1 Overview ... 171
5.16.6.2 Secure CDB Originator class ... 171
5.16.6.3 Secure CDB Processor class ... 171
5.16.6.4 Enforcement Manager class ... 172
5.16.6.5 Security Manager class ... 172
5.16.6.6 The relationship between SAs and command security ... 173
5.16.6.7 Command security techniques ... 173

5.16.6.8 Capability-based command security technique ... 173
5.16.6.8.1 Overview ... 173
5.16.6.8.2 Security Manager class ... 176
5.16.6.8.3 CbCS Management Device Server class ... 177
5.16.6.8.3.1 CbCS Management Device Server class overview ... 177
5.16.6.8.3.2 Decision Database attribute ... 177
5.16.6.8.4 CbCS Management Application Client class ... 177
5.16.6.8.5 Secure CDB Originator class ... 177
5.16.6.8.6 Secure CDB Processor class ... 178
5.16.6.8.7 Enforcement Manager class ... 178
5.16.6.8.8 CbCS methods ... 179
5.16.6.8.8.1 Overview ... 179
5.16.6.8.8.2 The BASIC CbCS method ... 179
5.16.6.8.8.3 The CAPKEY CbCS method ... 180
5.16.6.8.9 CbCS trust assumptions ... 180
5.16.6.8.10 CbCS security tokens ... 181
5.16.6.8.11 CbCS shared keys ... 182
5.16.6.8.11.1 Overview ... 182
5.16.6.8.11.2 CbCS shared key identifiers ... 183
5.16.6.8.11.3 Specifying which CbCS shared key to change ... 184
5.16.6.8.11.4 Updating a CbCS master key ... 184
5.16.6.8.11.5 Changing a CbCS working keys ... 184
5.16.6.8.12 CbCS credentials ... 185
5.16.6.8.12.1 Overview ... 185
5.16.6.8.12.2 CbCS capability key computations for the secure CDB originator ... 185
5.16.6.8.12.3 CbCS capability key computations for general use ... 185
5.16.6.8.13 CbCS capability descriptors ... 186
5.16.6.8.13.1 Overview ... 186
5.16.6.8.13.2 CbCS extension descriptor validation ... 186
5.16.6.8.13.3 CAPKEY CbCS method capability integrity validation ... 187
5.16.6.8.14 Association between commands and permission bits ... 188
5.16.6.8.15 CbCS parameters ... 190
5.16.6.8.15.1 Overview ... 190
5.16.6.8.16 CbCS extension descriptor format ... 192
5.16.7 ESP-SCSI for parameter data ... 193
5.16.7.1 Overview ... 193
5.16.7.2 ESP-SCSI required inputs ... 193
5.16.7.3 ESP-SCSI data format before encryption and after decryption ... 194
5.16.7.4 ESP-SCSI outbound data descriptors ... 195
5.16.7.4.1 Overview ... 195
5.16.7.4.2 ESP-SCSI CDBs or Data-Out Buffer parameter lists including a descriptor length ... 196
5.16.7.4.3 ESP-SCSI Data-Out Buffer parameter lists for externally specified descriptor length ... 199
5.16.7.5 ESP-SCSI Data-In Buffer parameter data descriptors ... 200
5.16.7.5.1 Overview ... 200
5.16.7.5.2 ESP-SCSI Data-In Buffer parameter data including a descriptor length ... 201
5.16.7.5.3 ESP-SCSI Data-In Buffer parameter data for externally specified descriptor length ... 204
5.16.8 Security algorithm codes ... 206
5.17 Identifying information ... 208
5.18 Downloading and activating microcode ... 208

6 Commands for all device types ... 213
6.1 Summary of commands for all device types ... 213
6.2 CHANGE ALIASES command ... 215
6.2.1 CHANGE ALIASES command introduction ... 215

6.2.2 Alias entry format ... 217
6.2.3 Alias designation validation ... 218
6.2.4 Alias entry protocol independent designations ... 218
6.2.4.1 Alias entry protocol independent designations overview ... 218
6.2.4.2 NULL DESIGNATION alias format ... 219
6.3 EXTENDED COPY command ... 220
6.3.1 EXTENDED COPY command introduction ... 220
6.3.2 Errors detected before starting processing of the segment descriptors ... 224
6.3.3 Errors detected during processing of segment descriptors ... 224
6.3.4 Abort task management functions ... 226
6.3.5 Descriptor type codes ... 227
6.3.6 Target descriptors ... 227
6.3.6.1 Target descriptors introduction ... 227
6.3.6.2 The target descriptor extension ... 230
6.3.6.3 Identification descriptor target descriptor format ... 231
6.3.6.4 Alias target descriptor format ... 232
6.3.6.5 IP Copy Service target descriptor ... 233
6.3.6.6 Device type specific target descriptor parameters for block device types ... 234
6.3.6.7 Device type specific target descriptor parameters for sequential-access device types ... 235
6.3.6.8 Device type specific target descriptor parameters for processor device types ... 236
6.3.7 Segment descriptors ... 236
6.3.7.1 Segment descriptors introduction ... 236
6.3.7.2 Segment descriptor processing ... 238
6.3.7.3 Block device to stream device operations ... 242
6.3.7.4 Stream device to block device operations ... 243
6.3.7.5 Block device to block device operations ... 244
6.3.7.6 Stream device to stream device operations ... 246
6.3.7.7 Inline data to stream device operation ... 248
6.3.7.8 Embedded data to stream device operation ... 249
6.3.7.9 Stream device to discard operation ... 250
6.3.7.10 Verify device operation ... 251
6.3.7.11 Block device with offset to stream device operation ... 252
6.3.7.12 Stream device to block device with offset operation ... 253
6.3.7.13 Block device with offset to block device with offset operation ... 254
6.3.7.14 Write filemarks operation ... 255
6.3.7.15 Space operation ... 256
6.3.7.16 Locate operation ... 257
6.3.7.17 Tape device image copy operation ... 258
6.3.7.18 Register persistent reservation key operation ... 259
6.3.7.19 Third party persistent reservations source I_T nexus ... 260
6.4 INQUIRY command ... 263
6.4.1 INQUIRY command introduction ... 263
6.4.2 Standard INQUIRY data ... 264
6.4.3 SCSI Parallel Interface specific INQUIRY data ... 279
6.5 LOG SELECT command ... 281
6.5.1 Introduction ... 281
6.5.2 Processing LOG SELECT when the parameter list length is zero ... 283
6.6 LOG SENSE command ... 286
6.7 MANAGEMENT PROTOCOL IN command ... 288
6.7.1 MANAGEMENT PROTOCOL IN command description ... 288
6.7.2 Management protocol information description ... 289
6.7.2.1 Overview ... 289
6.7.2.2 CDB description ... 289
6.7.2.3 Supported management protocols list description ... 290

6.8 MANAGEMENT PROTOCOL OUT command ... 291
6.9 MODE SELECT(6) command ... 292
6.10 MODE SELECT(10) command ... 294
6.11 MODE SENSE(6) command ... 295
6.11.1 MODE SENSE(6) command introduction ... 295
6.11.2 Current values ... 297
6.11.3 Changeable values ... 297
6.11.4 Default values ... 297
6.11.5 Saved values ... 297
6.11.6 Initial responses ... 298
6.12 MODE SENSE(10) command ... 298
6.13 PERSISTENT RESERVE IN command ... 300
6.13.1 PERSISTENT RESERVE IN command introduction ... 300
6.13.2 READ KEYS service action ... 301
6.13.3 READ RESERVATION service action ... 302
6.13.3.1 READ RESERVATION service action introduction ... 302
6.13.3.2 Format of PERSISTENT RESERVE IN parameter data for READ RESERVATION ... 302
6.13.3.3 Persistent reservations scope ... 304
6.13.3.4 Persistent reservations type ... 304
6.13.4 REPORT CAPABILITIES service action ... 305
6.13.5 READ FULL STATUS service action ... 309
6.14 PERSISTENT RESERVE OUT command ... 311
6.14.1 PERSISTENT RESERVE OUT command introduction ... 311
6.14.2 PERSISTENT RESERVE OUT service actions ... 313
6.14.3 Basic PERSISTENT RESERVE OUT parameter list ... 314
6.14.4 PERSISTENT RESERVE OUT command with REGISTER AND MOVE service action parameters . 319
6.15 READ ATTRIBUTE command ... 321
6.15.1 READ ATTRIBUTE command introduction ... 321
6.15.2 ATTRIBUTE VALUES service action ... 323
6.15.3 ATTRIBUTE LIST service action ... 324
6.15.4 LOGICAL VOLUME LIST service action ... 324
6.15.5 PARTITION LIST service action ... 325
6.15.6 SUPPORTED ATTRIBUTES service action ... 325
6.16 READ BUFFER command ... 326
6.16.1 READ BUFFER command introduction ... 326
6.16.2 Combined header and data mode (00h) ... 327
6.16.3 Vendor specific mode (01h) ... 328
6.16.4 Data mode (02h) ... 328
6.16.5 Descriptor mode (03h) ... 328
6.16.6 Read data from echo buffer mode (0Ah) ... 329
6.16.7 Echo buffer descriptor mode (0Bh) ... 329
6.16.8 Enable expander communications protocol and Echo buffer (1Ah) ... 330
6.16.9 Error history mode (1Ch) ... 330
6.16.9.1 Error history overview ... 330
6.16.9.2 Error history directory ... 332
6.16.9.3 Error history data buffer ... 334
6.16.9.4 Clear error history I_T nexus ... 334
6.16.9.5 Clear error history I_T nexus and release snapshot ... 334
6.17 READ MEDIA SERIAL NUMBER command ... 335
6.18 RECEIVE COPY RESULTS command ... 336
6.18.1 RECEIVE COPY RESULTS command introduction ... 336
6.18.2 COPY STATUS service action ... 337
6.18.3 RECEIVE DATA service action ... 340
6.18.4 OPERATING PARAMETERS service action ... 341

6.18.5 FAILED SEGMENT DETAILS service action ... 344
6.19 RECEIVE CREDENTIAL command ... 346
6.19.1 RECEIVE CREDENTIAL command description ... 346
6.19.1.1 Overview ... 346
6.19.1.2 CbCS logical unit credential request descriptor ... 348
6.19.1.3 CbCS logical unit and volume credential request descriptor ... 348
6.19.2 RECEIVE CREDENTIAL parameter data ... 349
6.19.2.1 RECEIVE CREDENTIAL parameter data encryption ... 349
6.19.2.2 RECEIVE CREDENTIAL decrypted parameter data ... 349
6.19.2.3 CbCS capability descriptor ... 350
6.19.2.3.1 Overview ... 350
6.19.2.3.2 Logical unit designation descriptor format ... 353
6.19.2.3.3 Volume designation descriptors ... 353
6.20 RECEIVE DIAGNOSTIC RESULTS command ... 354
6.21 REPORT ALIASES command ... 355
6.22 REPORT IDENTIFYING INFORMATION command ... 356
6.22.1 REPORT IDENTIFYING INFORMATION command overview ... 356
6.22.2 IDENTIFYING INFORMATION parameter data ... 358
6.22.3 IDENTIFYING INFORMATION SUPPORTED parameter data ... 359
6.23 REPORT LUNS command ... 360
6.24 REPORT PRIORITY command ... 362
6.25 REPORT SUPPORTED OPERATION CODES command ... 364
6.25.1 REPORT SUPPORTED OPERATION CODES command introduction ... 364
6.25.2 All_commands parameter data format ... 366
6.25.3 One_command parameter data format ... 367
6.25.4 Command timeouts descriptor ... 368
6.25.4.1 Overview ... 368
6.25.4.2 WRITE BUFFER command timeouts descriptor COMMAND SPECIFIC field usage ... 370
6.26 REPORT SUPPORTED TASK MANAGEMENT FUNCTIONS command ... 370
6.27 REPORT TARGET PORT GROUPS command ... 374
6.28 REPORT TIMESTAMP command ... 379
6.29 REQUEST SENSE command ... 380
6.30 SECURITY PROTOCOL IN command ... 383
6.31 SECURITY PROTOCOL OUT command ... 385
6.32 SEND DIAGNOSTIC command ... 387
6.33 SET IDENTIFYING INFORMATION command ... 393
6.34 SET PRIORITY command ... 395
6.35 SET TARGET PORT GROUPS command ... 398
6.36 SET TIMESTAMP command ... 401
6.37 TEST UNIT READY command ... 403
6.38 WRITE ATTRIBUTE command ... 404
6.39 WRITE BUFFER command ... 406
6.39.1 WRITE BUFFER command introduction ... 406
6.39.2 Combined header and data mode (00h) ... 407
6.39.3 Vendor specific mode (01h) ... 408
6.39.4 Data mode (02h) ... 408
6.39.5 Download microcode and activate mode (04h) ... 408
6.39.6 Download microcode, save, and activate mode (05h) ... 408
6.39.7 Download microcode with offsets and activate mode (06h) ... 408
6.39.8 Download microcode with offsets, save, and activate mode (07h) ... 409
6.39.9 Write data to echo buffer mode (0Ah) ... 409
6.39.10 Download microcode with offsets, save, and defer activate mode (0Eh) ... 409
6.39.11 Activate deferred microcode mode (0Fh) ... 410
6.39.12 Enable expander communications protocol and Echo buffer mode (1Ah) ... 410

6.39.13 Disable expander communications protocol mode (1Bh) ... 410
6.39.14 Download application client error history mode (1Ch) ... 410

7 Parameters for all device types ... 414
7.1 Overview ... 414
7.2 Diagnostic parameters ... 414
7.2.1 Summary of diagnostic page codes ... 414
7.2.2 Diagnostic page format and page codes for all device types ... 414
7.2.3 Protocol Specific ... 415
7.2.4 Supported Diagnostic Pages ... 416
7.3 Log parameters ... 417
7.3.1 Summary of log page codes ... 417
7.3.2 Log page structure and log parameter structure for all device types ... 418
7.3.2.1 Log page structure ... 418
7.3.2.2 Log parameter structure ... 420
7.3.2.2.1 Introduction ... 420
7.3.2.2.2 Parameter control byte ... 421
7.3.2.2.2.1 Introduction ... 421
7.3.2.2.2.2 Parameter control byte values for bounded data counter parameters ... 423
7.3.2.2.2.3 Parameter control byte values for unbounded data counter parameters ... 424
7.3.2.2.2.4 Parameter control byte values for ASCII format list log parameters ... 425
7.3.2.2.2.5 Parameter control byte values for binary format list log parameters ... 426
7.3.2.2.2.6 Resetting and setting log parameters ... 427
7.3.3 Application Client log page ... 427
7.3.3.1 Overview ... 427
7.3.3.2 General Usage Application Client log parameter ... 428
7.3.4 Buffer Over-Run/Under-Run log page ... 429
7.3.4.1 Overview ... 429
7.3.4.2 Buffer Over-run/Under-run log parameter ... 431
7.3.5 Cache Memory Statistics log page ... 432
7.3.5.1 Overview ... 432
7.3.5.2 Read Cache Memory Hits log parameter ... 434
7.3.5.3 Reads To Cache Memory Hits log parameter ... 435
7.3.5.4 Write Cache Memory Hits log parameter ... 436
7.3.5.5 Writes From Cache Memory Hits log parameter ... 437
7.3.5.6 Time From Last Hard Reset log parameter ... 438
7.3.5.7 Time Interval log parameter ... 439
7.3.6 General Statistics and Performance log pages ... 440
7.3.6.1 Overview ... 440
7.3.6.2 General Access Statistics and Performance log parameter ... 442
7.3.6.3 Idle Time log parameter ... 444
7.3.6.4 Force Unit Access Statistics and Performance log parameter ... 445
7.3.7 Group Statistics and Performance (n) log pages ... 446
7.3.7.1 Overview ... 446
7.3.7.2 Group n Statistics and Performance log parameter ... 450
7.3.7.3 Group n Force Unit Access Statistics and Performance log parameter ... 452
7.3.8 Informational Exceptions log page ... 453
7.3.8.1 Overview ... 453
7.3.8.2 Informational Exceptions General log parameter ... 454
7.3.9 Last n Deferred Errors or Asynchronous Events log page ... 455
7.3.9.1 Overview ... 455
7.3.9.2 Deferred Error or Asynchronous Event log parameters ... 457
7.3.10 Last n Error Events log page ... 457
7.3.10.1 Overview ... 457

7.3.10.2 Error Event log parameters ... 459
7.3.11 Non-Medium Error log page ... 459
7.3.11.1 Overview ... 459
7.3.11.2 Non-Medium Error Count log parameter ... 460
7.3.12 Power Condition Transitions log page ... 461
7.3.12.1 Overview ... 461
7.3.12.2 Accumulated Transitions log parameter ... 462
7.3.13 Protocol Specific Port log page ... 463
7.3.13.1 Overview ... 463
7.3.13.2 Generic protocol specific port log parameter ... 464
7.3.14 Read Error Counters log page ... 465
7.3.14.1 Overview ... 465
7.3.14.2 Read Error Counter log parameter ... 466
7.3.15 Read Reverse Error Counters log page ... 467
7.3.15.1 Overview ... 467
7.3.15.2 Read Reverse Error Counter log parameter ... 468
7.3.16 Self-Test Results log page ... 469
7.3.16.1 Overview ... 469
7.3.16.2 Self-Test Results log parameters ... 471
7.3.17 Start-Stop Cycle Counter log page ... 473
7.3.17.1 Overview ... 473
7.3.17.2 Date of Manufacture log parameter ... 474
7.3.17.3 Accounting Date log parameter ... 475
7.3.17.4 Specified Cycle Count Over Device Lifetime log parameter ... 476
7.3.17.5 Accumulated Start-Stop Cycles log parameter ... 476
7.3.17.6 Specified Load-Unload Count Over Device Lifetime log parameter ... 477
7.3.17.7 Accumulated Load-Unload Cycles log parameter ... 478
7.3.18 Supported Log Pages log page ... 479
7.3.19 Supported Log Pages and Subpages log page ... 480
7.3.20 Supported Subpages log page ... 481
7.3.21 Temperature log page ... 482
7.3.21.1 Overview ... 482
7.3.21.2 Temperature log parameter ... 483
7.3.21.3 Reference Temperature log parameter ... 484
7.3.22 Verify Error Counters log page ... 485
7.3.22.1 Overview ... 485
7.3.22.2 Verify Error Counter log parameter ... 486
7.3.23 Write Error Counters log page ... 487
7.3.23.1 Overview ... 487
7.3.23.2 Write Error Counter log parameter ... 488
7.4 Medium auxiliary memory attributes ... 490
7.4.1 Attribute format ... 490
7.4.2 Attribute identifier values ... 491
7.4.2.1 Attribute identifier values overview ... 491
7.4.2.2 Device type attributes ... 492
7.4.2.3 Medium type attributes ... 499
7.4.2.4 Host type attributes ... 500
7.5 Mode parameters ... 502
7.5.1 Summary of mode page codes ... 502
7.5.2 Mode parameters overview ... 502
7.5.3 Mode parameter list format ... 503
7.5.4 Mode parameter header formats ... 503
7.5.5 Mode parameter block descriptor formats ... 504
7.5.5.1 General block descriptor format ... 504

7.5.6 Mode page and subpage formats and page codes ... 505
7.5.7 Control mode page ... 507
7.5.8 Control Extension mode page ... 512
7.5.9 Disconnect-Reconnect mode page ... 513
7.5.10 Extended mode page ... 516
7.5.11 Extended Device-Type Specific mode page ... 516
7.5.12 Power Condition mode page ... 516
7.5.13 Protocol Specific Logical Unit mode page ... 520
7.5.14 Protocol Specific Port mode page ... 521
7.6 Protocol specific parameters ... 523
7.6.1 Protocol specific parameters introduction ... 523
7.6.2 Alias entry protocol specific designations ... 523
7.6.2.1 Introduction to alias entry protocol specific designations ... 523
7.6.2.2 Fibre Channel specific alias entry designations ... 524
7.6.2.2.1 Introduction to Fibre Channel specific alias entry designations ... 524
7.6.2.2.2 Fibre Channel world wide port name alias entry designation ... 524
7.6.2.2.3 Fibre Channel world wide port name with N_Port checking alias entry designation ... 525
7.6.2.3 RDMA specific alias entry designations ... 525
7.6.2.3.1 Introduction to RDMA specific alias entry designations ... 525
7.6.2.3.2 RDMA target port identifier alias entry designation ... 526
7.6.2.3.3 InfiniBand global identifier with target port identifier checking alias entry designation ... 526
7.6.2.4 Internet SCSI specific alias entry designations ... 527
7.6.2.4.1 Introduction to Internet SCSI specific alias entry designations ... 527
7.6.2.4.2 iSCSI name alias entry designation ... 527
7.6.2.4.3 iSCSI name with binary IPv4 address alias entry designation ... 528
7.6.2.4.4 iSCSI name with IPname alias entry designation ... 529
7.6.2.4.5 iSCSI name with binary IPv6 address alias entry designation ... 530
7.6.3 EXTENDED COPY protocol specific target descriptors ... 531
7.6.3.1 Introduction to EXTENDED COPY protocol specific target descriptors ... 531
7.6.3.2 Fibre Channel N_Port_Name EXTENDED COPY target descriptor format ... 531
7.6.3.3 Fibre Channel N_Port_ID EXTENDED COPY target descriptor format ... 532
7.6.3.4 Fibre Channel N_Port_ID with N_Port_Name checking EXTENDED COPY target descriptor format 533
7.6.3.5 SCSI Parallel T_L EXTENDED COPY target descriptor format ... 534
7.6.3.6 IEEE 1394 EUI-64 EXTENDED COPY target descriptor format ... 535
7.6.3.7 RDMA EXTENDED COPY target descriptor format ... 536
7.6.3.8 iSCSI binary IPv4 address EXTENDED COPY target descriptor format ... 537
7.6.3.9 iSCSI binary IPv6 address EXTENDED COPY target descriptor format ... 538
7.6.3.10 SAS serial SCSI protocol target descriptor format ... 539
7.6.4 TransportID identifiers ... 539
7.6.4.1 Overview of TransportID identifiers ... 539
7.6.4.2 TransportID for initiator ports using SCSI over Fibre Channel ... 541
7.6.4.3 TransportID for initiator ports using a parallel SCSI bus ... 541
7.6.4.4 TransportID for initiator ports using SCSI over IEEE 1394 ... 542
7.6.4.5 TransportID for initiator ports using SCSI over an RDMA interface ... 542
7.6.4.6 TransportID for initiator ports using SCSI over iSCSI ... 543
7.6.4.7 TransportID for initiator ports using SCSI over SAS Serial SCSI Protocol ... 545
7.7 Security protocol parameters ... 546
7.7.1 Security protocol information description ... 546
7.7.1.1 Overview ... 546
7.7.1.2 CDB description ... 546
7.7.1.3 Supported security protocols list description ... 547
7.7.1.4 Certificate data description ... 548
7.7.1.4.1 Certificate overview ... 548
7.7.1.4.2 Public Key certificate description ... 548

7.7.1.4.3 Attribute certificate description ... 548
7.7.2 SA creation capabilities ... 549
7.7.2.1 Overview ... 549
7.7.2.2 SA creation capabilities CDB description ... 549
7.7.2.3 SA creation capabilities parameter data formats ... 550
7.7.2.3.1 Supported device server capabilities formats parameter data format ... 550
7.7.2.3.2 IKEv2-SCSI device server capabilities parameter data format ... 550
7.7.3 IKEv2-SCSI ... 551
7.7.3.1 Overview ... 551
7.7.3.2 IKEv2-SCSI SECURITY PROTOCOL IN CDB description ... 551
7.7.3.3 IKEv2-SCSI SECURITY PROTOCOL OUT CDB description ... 552
7.7.3.4 IKEv2-SCSI parameter data format ... 553
7.7.3.5 IKEv2-SCSI payloads ... 561
7.7.3.5.1 IKEv2-SCSI payload format ... 561
7.7.3.5.2 No Next payload ... 562
7.7.3.5.3 Key Exchange payload ... 562
7.7.3.5.4 Identification – Application Client payload and Identification – Device Server payload ... 563
7.7.3.5.5 Certificate payload ... 564
7.7.3.5.6 Certificate Request payload ... 565
7.7.3.5.7 Authentication payload ... 566
7.7.3.5.8 Nonce payload ... 569
7.7.3.5.9 Notify payload ... 570
7.7.3.5.10 Delete payload ... 571
7.7.3.5.11 Encrypted payload ... 572
7.7.3.5.11.1 Combined mode encryption ... 572
7.7.3.5.11.2 Encrypted payload introduction ... 573
7.7.3.5.11.3 IKEv2-SCSI AAD ... 575
7.7.3.5.11.4 Processing a received Encrypted payload ... 576
7.7.3.5.12 IKEv2-SCSI SA Creation Capabilities payload ... 578
7.7.3.5.13 IKEv2-SCSI SA Cryptographic Algorithms payload ... 579
7.7.3.5.14 IKEv2-SCSI SAUT Cryptographic Algorithms payload ... 581
7.7.3.5.15 IKEv2-SCSI Timeout Values payload ... 583
7.7.3.6 IKEv2-SCSI cryptographic algorithm descriptors ... 584
7.7.3.6.1 Overview ... 584
7.7.3.6.2 Encryption algorithm (ENCR) IKEv2-SCSI cryptographic algorithm descriptor ... 585
7.7.3.6.3 Pseudo-random function (PRF) IKEv2-SCSI cryptographic algorithm descriptor ... 587
7.7.3.6.4 Integrity algorithm (INTEG) IKEv2-SCSI cryptographic algorithm descriptor ... 589
7.7.3.6.5 Diffie-Hellman group (D-H) IKEv2-SCSI cryptographic algorithm descriptor ... 590
7.7.3.6.6 IKEv2-SCSI authentication algorithm IKEv2-SCSI cryptographic algorithm descriptor ... 592
7.7.3.7 Errors in IKEv2-SCSI security protocol commands ... 595
7.7.3.8 Errors in IKEv2-SCSI security protocol parameter data ... 597
7.7.3.8.1 Overview ... 597
7.7.3.8.2 Errors with high denial of service attack potential ... 597
7.7.3.8.3 Errors with minimal denial of service attack potential ... 598
7.7.3.9 Translating IKEv2 errors ... 599
7.7.4 CbCS security protocol ... 600
7.7.4.1 Overview ... 600
7.7.4.2 CbCS SECURITY PROTOCOL IN CDB description ... 600
7.7.4.3 CbCS SECURITY PROTOCOL IN parameter data ... 601
7.7.4.3.1 Supported CbCS SECURITY PROTOCOL IN Pages CbCS page ... 601
7.7.4.3.2 Supported CbCS SECURITY PROTOCOL OUT pages CbCS page ... 602
7.7.4.3.3 Unchangeable CbCS Parameters CbCS page ... 603
7.7.4.3.4 Security Token CbCS page ... 605
7.7.4.3.5 Current CbCS Parameters CbCS page ... 606

7.7.4.3.6 Set Master Key – Seed Exchange CbCS page ... 608
7.7.4.4 CbCS SECURITY PROTOCOL OUT CDB description ... 610
7.7.4.5 CbCS SECURITY PROTOCOL OUT parameter data ... 611
7.7.4.5.1 Set Policy Access Tag CbCS page ... 611
7.7.4.5.2 Set Minimum CbCS Method CbCS page ... 612
7.7.4.5.3 Invalidate Key CbCS page ... 613
7.7.4.5.4 Set Key CbCS page ... 614
7.7.4.5.5 Set Master Key – Seed Exchange CbCS page ... 615
7.7.4.5.6 Set Master Key – Change Master Key CbCS page ... 616
7.8 Vital product data parameters ... 618
7.8.1 Vital product data parameters overview and page codes ... 618
7.8.2 VPD page format and page codes for all device types ... 619
7.8.3 ASCII Information VPD page ... 620
7.8.4 CFA Profile Information VPD page ... 621
7.8.5 Device Constituents VPD page ... 622
7.8.5.1 Device Constituents VPD page overview ... 622
7.8.5.2 Virtual tape library constituent type specific descriptor ... 624
7.8.5.3 Virtual tape drive constituent type specific descriptor ... 625
7.8.6 Device Identification VPD page ... 625
7.8.6.1 Device Identification VPD page overview ... 625
7.8.6.2 Device designation descriptor requirements ... 628
7.8.6.2.1 Designation descriptors for logical units other than well known logical units ... 628
7.8.6.2.2 Designation descriptors for well known logical units ... 628
7.8.6.2.3 Designation descriptors for SCSI target ports ... 629
7.8.6.2.3.1 Relative target port identifiers ... 629
7.8.6.2.3.2 Target port names or identifiers ... 629
7.8.6.2.4 Designation descriptors for SCSI target devices ... 629
7.8.6.3 Vendor specific designator format ... 630
7.8.6.4 T10 vendor ID based designator format ... 630
7.8.6.5 EUI-64 based designator format ... 631
7.8.6.5.1 EUI-64 based designator format overview ... 631
7.8.6.5.2 EUI-64 designator format ... 631
7.8.6.5.3 EUI-64 based 12-byte designator format ... 632
7.8.6.5.4 EUI-64 based 16-byte designator format ... 632
7.8.6.6 NAA designator format ... 633
7.8.6.6.1 NAA identifier basic format ... 633
7.8.6.6.2 NAA IEEE Extended designator format ... 633
7.8.6.6.3 NAA Locally Assigned designator format ... 634
7.8.6.6.4 NAA IEEE Registered designator format ... 634
7.8.6.6.5 NAA IEEE Registered Extended designator format ... 635
7.8.6.7 Relative target port designator format ... 635
7.8.6.8 Target port group designator format ... 636
7.8.6.9 Logical unit group designator format ... 636
7.8.6.10 MD5 logical unit designator format ... 637
7.8.6.11 SCSI name string designator format ... 638
7.8.7 Extended INQUIRY Data VPD page ... 639
7.8.8 Management Network Addresses VPD page ... 643
7.8.9 Mode Page Policy VPD page ... 644
7.8.10 Power Condition VPD page ... 646
7.8.11 SCSI Ports VPD page ... 648
7.8.12 Protocol Specific Logical Unit Information VPD page ... 651
7.8.13 Protocol Specific Port Information VPD page ... 652
7.8.14 Software Interface Identification VPD page ... 654
7.8.15 Supported VPD Pages VPD page ... 655

7.8.16 Unit Serial Number VPD page ... 655

8 Well known logical units ... 657
8.1 Model for well known logical units ... 657
8.2 REPORT LUNS well known logical unit ... 657
8.3 ACCESS CONTROLS well known logical unit ... 658
8.3.1 Access controls model ... 658
8.3.1.1 Access controls commands ... 658
8.3.1.2 Access controls overview ... 658
8.3.1.3 The access control list (ACL) ... 659
8.3.1.3.1 ACL overview ... 659
8.3.1.3.2 Access identifiers ... 660
8.3.1.3.3 Logical unit access control descriptors ... 661
8.3.1.4 Managing the ACL ... 661
8.3.1.4.1 ACL management overview ... 661
8.3.1.4.2 Authorizing ACL management ... 661
8.3.1.4.3 Identifying logical units during ACL management ... 662
8.3.1.4.4 Tracking changes in logical unit identification ... 662
8.3.1.5 Enrolling AccessIDs ... 663
8.3.1.5.1 Enrollment states ... 663
8.3.1.5.1.1 Summary of enrollment states ... 663
8.3.1.5.1.2 Not-enrolled state ... 663
8.3.1.5.1.3 Enrolled state ... 664
8.3.1.5.1.4 Pending-enrolled state ... 664
8.3.1.5.2 ACL LUN conflict resolution ... 665
8.3.1.6 Granting and revoking access rights ... 665
8.3.1.6.1 Non-proxy access rights ... 665
8.3.1.6.2 Proxy access ... 666
8.3.1.6.2.1 Proxy tokens ... 666
8.3.1.6.2.2 Proxy LUNs ... 666
8.3.1.7 Verifying access rights ... 667
8.3.1.8 The management identifier key ... 668
8.3.1.8.1 Management identifier key usage ... 668
8.3.1.8.2 Overriding the management identifier key ... 668
8.3.1.8.2.1 The OVERRIDE MGMT ID KEY service action ... 668
8.3.1.8.2.2 The override lockout timer ... 669
8.3.1.9 Reporting access control information ... 669
8.3.1.10 Access controls log ... 670
8.3.1.11 Interactions of access controls and other features ... 671
8.3.1.11.1 Task set management and access controls ... 671
8.3.1.11.2 Existing reservations and ACL changes ... 671
8.3.1.12 Access controls information persistence and memory usage requirements ... 671
8.3.1.13 Access identifier formats ... 673
8.3.1.13.1 Access identifier type ... 673
8.3.1.13.2 AccessID access identifiers ... 673
8.3.2 ACCESS CONTROL IN command ... 674
8.3.2.1 ACCESS CONTROL IN introduction ... 674
8.3.2.2 REPORT ACL service action ... 675
8.3.2.2.1 REPORT ACL introduction ... 675
8.3.2.2.2 REPORT ACL parameter data format ... 676
8.3.2.2.2.1 REPORT ACL parameter data introduction ... 676
8.3.2.2.2.2 Granted ACL data page format ... 677
8.3.2.2.2.3 Granted All ACL data page format ... 679
8.3.2.2.2.4 Proxy Tokens ACL data page format ... 680

8.3.2.3 REPORT LU DESCRIPTORS service action ... 681
8.3.2.3.1 REPORT LU DESCRIPTORS introduction ... 681
8.3.2.3.2 REPORT LU DESCRIPTORS parameter data format ... 682
8.3.2.4 REPORT ACCESS CONTROLS LOG service action ... 686
8.3.2.4.1 REPORT ACCESS CONTROLS LOG introduction ... 686
8.3.2.4.2 REPORT ACCESS CONTROLS LOG parameter data format ... 687
8.3.2.4.2.1 REPORT ACCESS CONTROLS LOG parameter data introduction ... 687
8.3.2.4.2.2 Key Overrides access controls log portion page format ... 688
8.3.2.4.2.3 Invalid Keys access controls log portion page format ... 689
8.3.2.4.2.4 ACL LUN Conflicts access controls log portion page format ... 690
8.3.2.5 REPORT OVERRIDE LOCKOUT TIMER service action ... 691
8.3.2.6 REQUEST PROXY TOKEN service action ... 692
8.3.3 ACCESS CONTROL OUT command ... 694
8.3.3.1 ACCESS CONTROL OUT introduction ... 694
8.3.3.2 MANAGE ACL service action ... 695
8.3.3.2.1 MANAGE ACL introduction ... 695
8.3.3.2.2 The Grant/Revoke ACE page ... 698
8.3.3.2.3 The Grant All ACE page ... 701
8.3.3.2.4 The Revoke Proxy Token ACE page ... 702
8.3.3.2.5 The Revoke All Proxy Tokens ACE page ... 702
8.3.3.3 DISABLE ACCESS CONTROLS service action ... 703
8.3.3.4 ACCESS ID ENROLL service action ... 704
8.3.3.5 CANCEL ENROLLMENT service action ... 705
8.3.3.6 CLEAR ACCESS CONTROLS LOG service action ... 705
8.3.3.7 MANAGE OVERRIDE LOCKOUT TIMER service action ... 706
8.3.3.8 OVERRIDE MGMT ID KEY service action ... 707
8.3.3.9 REVOKE PROXY TOKEN service action ... 708
8.3.3.10 REVOKE ALL PROXY TOKENS service action ... 709
8.3.3.11 ASSIGN PROXY LUN service action ... 710
8.3.3.12 RELEASE PROXY LUN service action ... 711
8.4 TARGET LOG PAGES well known logical unit ... 712
8.5 SECURITY PROTOCOL well known logical unit ... 713
8.6 MANAGEMENT PROTOCOL well known logical unit ... 713

9 Security manager command set ... 714

Annex A (informative) Terminology mapping ... 715

Annex B (Informative) PERSISTENT RESERVE IN/OUT functionality for RESERVE/RELEASE replacement ... 716
B.1 Introduction ... 716
B.2 Replacing the reserve/release method with the PERSISTENT RESERVE OUT COMMAND ... 716
B.3 Third party reservations ... 717

Annex C (Informative) Variations between this standard and equivalent security protocols ... 718
C.1 IKEv2 protocol details and variations for IKEv2-SCSI ... 718
C.2 ESP protocol details and variations for ESP-SCSI ... 720

Annex D (informative) Numeric order codes ... 721
D.1 Numeric order codes introduction ... 721
D.2 Additional sense codes ... 721
D.3 Operation codes ... 739
D.3.1 Operation codes ... 739
D.3.2 Additional operation codes for devices with the ENCSERV bit set to one ... 745
D.3.3 MAINTENANCE (IN) and MAINTENANCE (OUT) service actions ... 746

D.3.4 SERVICE ACTION IN and SERVICE ACTION OUT service actions 747
D.3.5 Variable length CDB service action codes 748
D.4 Diagnostic page codes 749
D.5 Log page codes 750
D.6 Mode page codes 753
D.7 VPD page codes 756
D.8 Version descriptor values 758
D.9 T10 IEEE binary identifiers 777

Annex E (informative) T10 vendor identification 778

Table – 1 Numbering conventions examples ... 24
Table – 2 Comparison of decimal prefixes and binary prefixes ... 25
Table – 3 Class diagram constraints and notes notation ... 28
Table – 4 Class diagram multiplicity notation ... 29
Table – 5 Class diagram notation for classes ... 29
Table – 6 Class diagram notation for associations ... 30
Table – 7 Class diagram notation for aggregations ... 31
Table – 8 Class diagram notation for generalizations ... 33
Table – 9 Class diagram notation for dependencies ... 34
Table – 10 Notation for objects ... 34
Table – 11 Typical CDB for 6-byte commands ... 36
Table – 13 Typical CDB for 12-byte commands ... 37
Table – 12 Typical CDB for 10-byte commands ... 37
Table – 14 Typical CDB for 16-byte commands ... 38
Table – 15 Typical CDB for long LBA 16-byte commands ... 39
Table – 16 Typical variable length CDB ... 40
Table – 17 Typical variable length CDB for long LBA 32-byte commands ... 41
Table – 18 XCDB format ... 42
Table – 19 XCDB descriptor format ... 42
Table – 20 EXTENSION TYPE field ... 43
Table – 21 OPERATION CODE byte ... 43
Table – 22 Group Code values ... 44
Table – 23 Code set enumeration ... 46
Table – 24 Sense data response codes ... 47
Table – 25 Descriptor format sense data ... 48
Table – 26 Sense data descriptor format ... 49
Table – 27 DESCRIPTOR TYPE field ... 49
Table – 28 Information sense data descriptor format ... 50
Table – 29 Command-specific information sense data descriptor format ... 50
Table – 30 Sense key specific sense data descriptor format ... 51
Table – 31 Sense key specific sense data descriptor definitions ... 52
Table – 32 Field pointer sense key specific data ... 52
Table – 33 Actual retry count sense key specific data ... 53
Table – 34 Progress indication sense key specific data ... 53
Table – 35 Segment pointer sense key specific data ... 54
Table – 36 Unit attention condition queue overflow sense key specific data ... 54
Table – 37 Field replaceable unit sense data descriptor format ... 55
Table – 38 Progress indication sense data descriptor format ... 55
Table – 39 Forwarded sense data descriptor format ... 56
Table – 40 SENSE DATA SOURCE field ... 57
Table – 41 Vendor specific sense data descriptor format ... 57
Table – 42 Fixed format sense data ... 58
Table – 43 Sense key descriptions ... 61
Table – 44 ASC and ASCQ assignments ... 62
Table – 45 Exception commands for background self-tests ... 88
Table – 46 Self-test mode summary ... 90
Table – 47 SPC-4 commands that are allowed in the presence of various reservations ... 93
Table – 48 PERSISTENT RESERVE OUT service actions allowed in the presence of various reservations ... 95
Table – 49 Register behaviors for a REGISTER service action ... 99
Table – 50 Register behaviors for a REGISTER AND IGNORE EXISTING KEY service action ... 100
Table – 51 I_T Nexuses being registered ... 101
Table – 52 Register behaviors for a REGISTER AND MOVE service action ... 102
Table – 53 Processing for a released or preempted persistent reservation ... 105

Table – 54 Preempting actions ... 107
Table – 55 PC state machine states ... 123
Table – 56 PC state machine timers ... 123
Table – 57 Types of MAM attributes ... 130
Table – 58 MAM attribute states ... 130
Table – 59 TIMESTAMP ORIGIN field ... 133
Table – 60 TIMESTAMP field format ... 134
Table – 61 Minimum SA parameters ... 138
Table – 62 USAGE_TYPE SA parameter values ... 140
Table – 63 Security protocols that create SAs ... 141
Table – 64 Key derivation functions summary ... 141
Table – 65 HMAC-based key derivation functions ... 143
Table – 66 Hash functions used by HMAC based on KDF_ID ... 143
Table – 67 RFC 3566 parameter translations KDF based on AES-XCBC-PRF-128 ... 144
Table – 68 IKEv2-SCSI shared key names and SA shared key names ... 155
Table – 69 Shared key size determination ... 156
Table – 70 Device Server Capabilities step parameter data requirements ... 157
Table – 71 IKEv2-SCSI command terminations that do not abandon the CCS, if any ... 169
Table – 72 Security Manager class relationships ... 172
Table – 73 CbCS methods ... 179
Table – 74 CbCS communications trust requirement ... 181
Table – 75 Summary of CbCS shared keys ... 183
Table – 76 CbCS shared key identifier values ... 183
Table – 77 Associations between commands and permissions ... 188
Table – 78 Associations between security protocol commands and permissions ... 190
Table – 79 Summary of changeable CbCS parameters ... 191
Table – 80 CbCS extension descriptor format ... 192
Table – 81 ESP-SCSI data format before encryption and after decryption ... 194
Table – 82 ESP-SCSI outbound data descriptors ... 195
Table – 83 ESP-SCSI CDBs or Data-Out Buffer parameter list descriptor without initialization vector ... 196
Table – 84 ESP-SCSI CDBs or Data-Out Buffer full parameter list descriptor ... 197
Table – 85 ESP-SCSI Data-Out Buffer parameter list descriptor without length and initialization vector ... 199
Table – 86 ESP-SCSI Data-Out Buffer parameter list descriptor without length ... 200
Table – 87 ESP-SCSI Data-In Buffer parameter data descriptors ... 201
Table – 88 ESP-SCSI Data-In Buffer parameter data descriptor without initialization vector ... 201
Table – 89 ESP-SCSI Data-In Buffer full parameter data descriptor ... 202
Table – 90 ESP-SCSI Data-In Buffer parameter data descriptor without length and initialization vector ... 204
Table – 91 ESP-SCSI Data-In Buffer parameter data descriptor without length ... 205
Table – 92 Security algorithm codes ... 206
Table – 93 Identifying information types ... 208
Table – 94 WRITE BUFFER download microcode modes ... 209
Table – 95 WRITE BUFFER download microcode field processing ... 210
Table – 96 Multiple I_T nexus handling for WRITE BUFFER download microcode modes ... 211
Table – 97 Commands for all device types ... 213
Table – 98 CHANGE ALIASES command ... 215
Table – 99 CHANGE ALIASES parameter list ... 216
Table – 100 Alias entry format ... 217
Table – 101 Alias entry PROTOCOL IDENTIFIER field ... 217
Table – 102 Protocol independent alias entry format codes ... 218
Table – 103 EXTENDED COPY command ... 220
Table – 104 EXTENDED COPY parameter list ... 221
Table – 105 LIST ID USAGE field ... 222
Table – 106 EXTENDED COPY descriptor type codes ... 227
Table – 107 EXTENDED COPY target descriptor type codes ... 227

Table – 108 Target descriptor format ... 228
Table – 109 LU ID TYPE field ... 228
Table – 110 Device type specific parameters in target descriptors ... 229
Table – 111 Target descriptor extension format ... 230
Table – 112 Identification descriptor target descriptor format ... 231
Table – 113 Alias target descriptor format ... 232
Table – 114 IP Copy Service EXTENDED COPY target descriptor format ... 233
Table – 115 Device type specific target descriptor parameters for block device types ... 234
Table – 116 Device type specific target descriptor parameters for sequential-access device types ... 235
Table – 117 Stream device transfer lengths ... 235
Table – 118 Device type specific target descriptor parameters for processor device types ... 236
Table – 119 EXTENDED COPY segment descriptor type codes ... 236
Table – 120 Segment descriptor header ... 238
Table – 121 Descriptor Type Code Dependent Copy Manager Processing ... 239
Table – 122 PAD and CAT bit definitions ... 241
Table – 123 Block device to or from stream device segment descriptor ... 242
Table – 124 Block device to block device segment descriptor ... 244
Table – 125 Stream device to stream device segment descriptor ... 246
Table – 126 Inline data to stream device segment descriptor ... 248
Table – 127 Embedded data to stream device segment descriptor ... 249
Table – 128 Stream device to discard segment descriptor ... 250
Table – 129 Verify device operation segment descriptor ... 251
Table – 130 Block device with offset to or from stream device segment descriptor ... 252
Table – 131 Block device with offset to block device with offset segment descriptor ... 254
Table – 132 Write filemarks operation segment descriptor ... 255
Table – 133 Space operation segment descriptor ... 256
Table – 134 Locate operation segment descriptor ... 257
Table – 135 Tape device image copy segment descriptor ... 258
Table – 136 Register persistent reservation key segment descriptor ... 259
Table – 137 Third party persistent reservations source I_T nexus segment descriptor ... 261
Table – 138 INQUIRY command ... 263
Table – 139 Standard INQUIRY data format ... 264
Table – 140 Peripheral qualifier ... 265
Table – 141 Peripheral device type ... 266
Table – 142 Version ... 267
Table – 143 TPGS field ... 268
Table – 144 Version descriptor values ... 269
Table – 145 SPI-specific standard INQUIRY bits ... 279
Table – 146 Maximum logical device configuration table ... 280
Table – 147 CLOCKING field ... 280
Table – 148 LOG SELECT command ... 281
Table – 149 Page control (PC) field ... 282
Table – 150 PAGE CODE field and SUBPAGE CODE field ... 283
Table – 151 PCR bit, SP bit, and PC field meanings when parameter list length is zero ... 283
Table – 152 LOG SENSE command ... 286
Table – 153 MANAGEMENT PROTOCOL IN command ... 288
Table – 154 MANAGEMENT PROTOCOL field in MANAGEMENT PROTOCOL IN command ... 288
Table – 155 MANAGEMENT PROTOCOL SPECIFIC2 field for MANAGEMENT PROTOCOL IN protocol 00h ... 289
Table – 156 Supported management protocols MANAGEMENT PROTOCOL IN parameter data ... 290
Table – 157 MANAGEMENT PROTOCOL OUT command ... 291
Table – 158 MANAGEMENT PROTOCOL field in MANAGEMENT PROTOCOL OUT command ... 291
Table – 159 MODE SELECT(6) command ... 292
Table – 160 Mode page policies ... 293
Table – 162 MODE SENSE(6) command ... 295

Table – 161 MODE SELECT(10) command ... 295
Table – 163 Page control (PC) field ... 296
Table – 164 Mode page code usage for all devices ... 296
Table – 165 MODE SENSE(10) command ... 298
Table – 166 PERSISTENT RESERVE IN command ... 300
Table – 167 PERSISTENT RESERVE IN service action codes ... 300
Table – 168 PERSISTENT RESERVE IN parameter data for READ KEYS ... 301
Table – 169 PERSISTENT RESERVE IN parameter data for READ RESERVATION with no reservation held . 302
Table – 170 PERSISTENT RESERVE IN parameter data for READ RESERVATION with reservation ... 303
Table – 171 Persistent reservation SCOPE field ... 304
Table – 172 Persistent reservation TYPE field ... 304
Table – 173 PERSISTENT RESERVE IN parameter data for REPORT CAPABILITIES ... 305
Table – 174 ALLOW COMMANDS field ... 307
Table – 175 Persistent Reservation Type Mask format ... 308
Table – 176 PERSISTENT RESERVE IN parameter data for READ FULL STATUS ... 309
Table – 177 PERSISTENT RESERVE IN full status descriptor format ... 310
Table – 178 PERSISTENT RESERVE OUT command ... 311
Table – 179 PERSISTENT RESERVE OUT service action codes ... 313
Table – 180 PERSISTENT RESERVE OUT parameter list ... 314
Table – 181 PERSISTENT RESERVE OUT specify initiator ports additional parameter data ... 315
Table – 182 PERSISTENT RESERVE OUT service actions and valid parameters (part 1 of 2) ... 317
Table – 183 PERSISTENT RESERVE OUT with REGISTER AND MOVE service action parameter list ... 319
Table – 184 READ ATTRIBUTE command ... 321
Table – 185 READ ATTRIBUTE service action codes ... 322
Table – 186 READ ATTRIBUTE with ATTRIBUTE VALUES service action parameter list format ... 323
Table – 187 READ ATTRIBUTE with ATTRIBUTE LIST service action parameter list format ... 324
Table – 188 READ ATTRIBUTE with LOGICAL VOLUME LIST service action parameter list format ... 324
Table – 189 READ ATTRIBUTE with PARTITION LIST service action parameter list format ... 325
Table – 190 READ ATTRIBUTE with SUPPORTED ATTRIBUTES service action parameter list format ... 325
Table – 191 READ BUFFER command ... 326
Table – 192 READ BUFFER MODE field ... 327
Table – 193 READ BUFFER header ... 327
Table – 194 READ BUFFER descriptor ... 328
Table – 195 OFFSET BOUNDARY field ... 329
Table – 196 Echo buffer descriptor ... 330
Table – 197 Error history BUFFER ID field ... 331
Table – 198 Summary of error history directory device server actions ... 332
Table – 199 Error history directory ... 332
Table – 200 EHS_RETRIEVED field ... 333
Table – 201 EHS_SOURCE field ... 333
Table – 202 Error history directory entry ... 334
Table – 203 READ MEDIA SERIAL NUMBER command ... 335
Table – 204 READ MEDIA SERIAL NUMBER parameter data format ... 335
Table – 205 RECEIVE COPY RESULTS command ... 336
Table – 206 RECEIVE COPY RESULTS service action codes ... 337
Table – 207 Parameter data for the COPY STATUS service action ... 338
Table – 208 COPY MANAGER STATUS field ... 338
Table – 209 COPY STATUS TRANSFER COUNT UNITS field ... 339
Table – 210 Parameter data for the RECEIVE DATA service action ... 340
Table – 211 Parameter data for the OPERATING PARAMETERS service action ... 341
Table – 212 Parameter data for the FAILED SEGMENT DETAILS service action ... 344
Table – 213 RECEIVE CREDENTIAL command ... 346
Table – 214 RECEIVE CREDENTIAL command unencrypted bytes format ... 347
Table – 215 CREDENTIAL REQUEST TYPE field ... 347

Table – 216 CbCS logical unit credential request descriptor format ... 348
Table – 217 CbCS logical unit and volume credential request descriptor format ... 348
Table – 218 CbCS credential format ... 349
Table – 219 Credential format values ... 349
Table – 220 CbCS capability descriptor format ... 350
Table – 221 DESIGNATION TYPE field ... 351
Table – 222 CBCS METHOD field ... 351
Table – 223 PERMISSIONS BIT MASK field format ... 352
Table – 224 Logical unit designation descriptor format ... 353
Table – 225 Volume designation descriptor format ... 353
Table – 226 RECEIVE DIAGNOSTIC RESULTS command ... 354
Table – 227 REPORT ALIASES command ... 355
Table – 228 REPORT ALIASES parameter data ... 356
Table – 229 REPORT IDENTIFYING INFORMATION command ... 357
Table – 230 INFORMATION TYPE field ... 358
Table – 231 REPORT IDENTIFYING INFORMATION parameter data ... 358
Table – 232 REPORT IDENTIFYING INFORMATION SUPPORTED parameter data ... 359
Table – 233 Identifying information descriptor ... 359
Table – 234 REPORT LUNS command ... 360
Table – 235 SELECT REPORT field ... 360
Table – 236 REPORT LUNS parameter data format ... 361
Table – 237 REPORT PRIORITY command ... 362
Table – 238 PRIORITY REPORTED field ... 362
Table – 239 REPORT PRIORITY parameter data format ... 363
Table – 240 Priority descriptor format ... 363
Table – 241 REPORT SUPPORTED OPERATION CODES command ... 364
Table – 242 REPORT SUPPORTED OPERATION CODES REPORTING OPTIONS field ... 365
Table – 243 All_commands parameter data ... 366
Table – 244 Command descriptor format ... 366
Table – 245 One_command parameter data ... 367
Table – 246 SUPPORT values ... 368
Table – 247 Command timeouts descriptor format ... 369
Table – 248 Command timeouts descriptor COMMAND SPECIFIC field usage in this standard ... 369
Table – 249 REPORT SUPPORTED TASK MANAGEMENT FUNCTIONS command ... 370
Table – 250 REPORT SUPPORTED TASK MANAGEMENT FUNCTIONS basic parameter data ... 371
Table – 251 REPORT SUPPORTED TASK MANAGEMENT FUNCTIONS extended parameter data ... 372
Table – 252 REPORT TARGET PORT GROUPS command ... 374
Table – 253 PARAMETER DATA FORMAT field ... 375
Table – 254 Length only header parameter data format ... 375
Table – 255 Extended header parameter data format ... 376
Table – 256 Target port group descriptor format ... 377
Table – 257 ASYMMETRIC ACCESS STATE field ... 377
Table – 258 STATUS CODE field ... 378
Table – 259 Target port descriptor format ... 379
Table – 260 REPORT TIMESTAMP command ... 379
Table – 261 REPORT TIMESTAMP parameter data format ... 380
Table – 262 REQUEST SENSE command ... 380
Table – 263 DESC bit ... 381
Table – 264 SECURITY PROTOCOL IN command ... 383
Table – 265 SECURITY PROTOCOL field in SECURITY PROTOCOL IN command ... 384
Table – 266 SECURITY PROTOCOL OUT command ... 385
Table – 267 SECURITY PROTOCOL field in SECURITY PROTOCOL OUT command ... 386
Table – 268 SEND DIAGNOSTIC command ... 387
Table – 269 SELF-TEST CODE field ... 387

Table – 270 SELF-TEST CODE field, the PF bit, the SELFTEST bit, and the PARAMETER LIST LENGTH field meanings 389
Table – 271 SET IDENTIFYING INFORMATION command ... 393
Table – 272 INFORMATION TYPE field ... 394
Table – 273 SET IDENTIFYING INFORMATION parameter list ... 394
Table – 274 SET PRIORITY command ... 395
Table – 275 I_T_L NEXUS TO SET field ... 396
Table – 276 SET PRIORITY parameter list format ... 397
Table – 277 SET TARGET PORT GROUPS command ... 398
Table – 278 SET TARGET PORT GROUPS parameter list format ... 399
Table – 279 Set target port group descriptor parameter list ... 400
Table – 280 ASYMMETRIC ACCESS STATE field ... 400
Table – 281 SET TIMESTAMP command ... 401
Table – 282 SET TIMESTAMP parameter list format ... 402
Table – 283 TEST UNIT READY command ... 403
Table – 284 Preferred TEST UNIT READY responses ... 403
Table – 285 WRITE ATTRIBUTE command ... 404
Table – 286 WRITE ATTRIBUTE parameter list format ... 405
Table – 287 WRITE BUFFER command ... 406
Table – 288 WRITE BUFFER MODE field ... 407
Table – 289 Application client error history parameter list format ... 411
Table – 290 ERROR TYPE field ... 412
Table – 291 ERROR LOCATION FORMAT field ... 413
Table – 292 Summary of diagnostic page codes ... 414
Table – 293 Diagnostic page format ... 415
Table – 294 Protocol Specific diagnostic page ... 415
Table – 295 Supported Diagnostic Pages diagnostic page ... 416
Table – 296 Summary of log page codes ... 417
Table – 297 Log page format ... 418
Table – 298 LOG SELECT PCR bit, SP bit, and DS bit meanings when parameter list length is not zero ... 419
Table – 299 Log parameter ... 420
Table – 300 Threshold met criteria (TMC) field ... 422
Table – 301 FORMAT AND LINKING field ... 422
Table – 302 Parameter control byte values for bounded data counter parameters ... 423
Table – 303 Parameter control byte values for unbounded data counter parameters ... 424
Table – 304 Parameter control byte values for ASCII format list log parameters ... 425
Table – 305 Parameter control byte values for binary format list log parameters ... 426
Table – 306 Keywords for resetting or changing log parameters ... 427
Table – 307 Application Client log page parameter codes ... 427
Table – 308 Application Client log page ... 428
Table – 309 General Usage Application Client log parameter ... 428
Table – 310 Buffer Over-Run/Under-Run log page parameter codes ... 429
Table – 311 Buffer Over-Run/Under-Run log page ... 431
Table – 312 Buffer Over-run/Under-run log parameter ... 431
Table – 313 Cache Memory Statistics log page parameter codes ... 432
Table – 314 Cache Memory Statistics log page commands ... 433
Table – 315 Cache Memory Statistics log page ... 433
Table – 316 Read Cache Memory Hits log parameter ... 434
Table – 317 Reads To Cache Memory log parameter ... 435
Table – 318 Write Cache Memory Hits log parameter ... 436
Table – 319 Writes From Cache Memory log parameter ... 437
Table – 320 Time From Last Hard Reset log parameter ... 438
Table – 321 Time Interval log parameter ... 439
Table – 322 Time interval descriptor ... 439
Table – 323 General Statistics and Performance log page parameter codes ... 440

Table – 324 Statistics and Performance log pages commands ... 441
Table – 325 General Statistics and Performance log page ... 441
Table – 326 General Access Statistics and Performance log parameter ... 442
Table – 327 Idle Time log parameter ... 444
Table – 328 Force Unit Access Statistics and Performance log parameter ... 445
Table – 329 Group Statistics and Performance log page parameter codes ... 446
Table – 330 Group Statistics and Performance (n) log page ... 447
Table – 331 Group Statistics and Performance (n) subpage codes ... 448
Table – 332 Group n Statistics and Performance log parameter ... 450
Table – 333 Group n Force Unit Access Statistics and Performance log parameter ... 452
Table – 334 Informational Exceptions log page parameter codes ... 453
Table – 335 Informational Exceptions log page ... 454
Table – 336 Informational Exceptions General log parameter ... 454
Table – 337 Last n Deferred Errors or Asynchronous Events log page parameter codes ... 455
Table – 338 Last n Deferred Errors or Asynchronous Events log page ... 456
Table – 339 Deferred Error or Asynchronous Event log parameter ... 457
Table – 340 Last n Error Events log page parameter codes ... 457
Table – 341 Last n Error Events log page ... 458
Table – 342 Error Event log parameter ... 459
Table – 343 Non-Medium Error log page parameter codes ... 459
Table – 344 Non-Medium Error log page ... 460
Table – 345 Non-Medium Error Count log parameter ... 460
Table – 346 Power Conditions Transitions log page parameter codes ... 461
Table – 347 Power Condition Transitions log page ... 462
Table – 348 Accumulated Transitions log parameter ... 462
Table – 349 Accumulated Transitions parameter codes and saturating counters ... 463
Table – 350 Protocol Specific Port log page ... 464
Table – 351 Generic protocol specific port log parameter ... 464
Table – 352 Read Error Counters log page parameter codes ... 465
Table – 353 Read Error Counters log page ... 466
Table – 354 Read Error Counter log parameter ... 466
Table – 355 Read Reverse Error Counters log page parameter codes ... 467
Table – 356 Read Reverse Error Counters log page ... 468
Table – 357 Read Reverse Error Counter log parameter ... 468
Table – 358 Self-Test Results log page parameter codes ... 469
Table – 359 Self-Test Results log page ... 469
Table – 360 Unused Self-Test Results log parameter ... 470
Table – 361 Self-Test Results log parameter ... 471
Table – 362 SELF-TEST RESULTS field ... 472
Table – 363 Start-Stop Cycle Counter log page parameter codes ... 473
Table – 364 Start-Stop Cycle Counter log page ... 473
Table – 365 Date of Manufacture log parameter ... 474
Table – 366 Accounting Date log parameter ... 475
Table – 367 Specified Cycle Count Over Device Lifetime log parameter ... 476
Table – 368 Accumulated Start-Stop Cycles log parameter ... 476
Table – 369 Specified Load-Unload Count Over Device Lifetime log parameter ... 477
Table – 370 Accumulated Load-Unload Cycles log parameter ... 478
Table – 371 Supported Log Pages log page ... 479
Table – 372 Supported page descriptor ... 479
Table – 373 Supported Log Pages and Subpages log page ... 480
Table – 374 Supported page/subpage descriptor ... 480
Table – 375 Supported Subpages log page ... 481
Table – 376 Supported subpage descriptor ... 481
Table – 377 Temperature log page parameter codes ... 482

Table – 378 Temperature log page ... 482
Table – 379 Temperature log parameter ... 483
Table – 380 Reference Temperature log parameter ... 484
Table – 381 Verify Error Counters log page parameter codes ... 485
Table – 382 Verify Error Counters log page ... 486
Table – 383 Verify Error Counter log parameter ... 486
Table – 384 Write Error Counters log page parameter codes ... 487
Table – 385 Write Error Counters log page ... 488
Table – 386 Write Error Counter log parameter ... 488
Table – 387 MAM ATTRIBUTE format ... 490
Table – 388 MAM attribute FORMAT field ... 490
Table – 389 MAM attribute identifier range assignments ... 491
Table – 390 Device type attributes ... 492
Table – 391 DEVICE VENDOR/SERIAL NUMBER attribute format ... 493
Table – 392 MEDIUM USAGE HISTORY attribute format ... 494
Table – 393 PARTITION USAGE HISTORY attribute format ... 496
Table – 394 Medium type attributes ... 499
Table – 395 MEDIUM TYPE and MEDIUM TYPE INFORMATION attributes ... 499
Table – 396 Host type attributes ... 500
Table – 397 TEXT LOCALIZATION IDENTIFIER attribute values ... 501
Table – 398 Summary of mode page codes ... 502
Table – 399 Mode parameter list ... 503
Table – 400 Mode parameter header(6) ... 503
Table – 401 Mode parameter header(10) ... 503
Table – 402 General mode parameter block descriptor ... 504
Table – 403 Page_0 mode page format ... 505
Table – 404 Sub_page mode page format ... 506
Table – 405 Control mode page ... 507
Table – 406 Task set type (TST) field ... 507
Table – 407 QUEUE ALGORITHM MODIFIER field ... 508
Table – 408 Queue error management (QERR) field ... 509
Table – 409 Unit attention interlocks control (UA_INTLCK_CTRL) field ... 510
Table – 410 AUTOLOAD MODE field ... 511
Table – 411 Control Extension mode page ... 512
Table – 412 Disconnect-Reconnect mode page ... 513
Table – 413 Data transfer disconnect control (DTDC) field ... 515
Table – 414 Extended mode page ... 516
Table – 415 Extended Device-Type Specific mode page ... 516
Table – 416 Power Condition mode page ... 517
Table – 417 PM_BG_PRECEDENCE field ... 517
Table – 418 FIDCPC field ... 519
Table – 419 FSBCPC field ... 520
Table – 420 FSTCPC field ... 520
Table – 421 Protocol Specific Logical Unit mode page ... 520
Table – 422 Page_0 format Protocol Specific Port mode page ... 521
Table – 423 Sub_page format Protocol Specific Port mode page ... 522
Table – 424 PROTOCOL IDENTIFIER values ... 523
Table – 425 Fibre Channel alias entry format codes ... 524
Table – 426 Fibre Channel world wide port name alias entry designation ... 524
Table – 427 Fibre Channel world wide port name with N_Port checking alias entry designation ... 525
Table – 428 RDMA alias entry format codes ... 525
Table – 429 RDMA target port identifier alias entry designation ... 526
Table – 430 InfiniBand global identifier with target port identifier checking alias entry designation ... 526
Table – 431 iSCSI alias entry format codes ... 527

Table – 432 iSCSI name alias entry designation ... 527
Table – 433 iSCSI name with binary IPv4 address alias entry designation ... 528
Table – 434 iSCSI name with IPname alias entry designation ... 529
Table – 435 iSCSI name with binary IPv6 address alias entry designation ... 530
Table – 436 Fibre Channel N_Port_Name EXTENDED COPY target descriptor format ... 531
Table – 437 Fibre Channel N_Port_ID EXTENDED COPY target descriptor format ... 532
Table – 438 Fibre Channel N_Port_ID with N_Port_Name checking target descriptor format ... 533
Table – 439 SCSI Parallel T_L EXTENDED COPY target descriptor format ... 534
Table – 440 IEEE 1394 EUI-64 EXTENDED COPY target descriptor format ... 535
Table – 441 RDMA EXTENDED COPY target descriptor format ... 536
Table – 442 iSCSI binary IPv4 address EXTENDED COPY target descriptor format ... 537
Table – 443 iSCSI binary IPv6 address EXTENDED COPY target descriptor format ... 538
Table – 444 SAS serial SCSI protocol EXTENDED COPY target descriptor format ... 539
Table – 445 TransportID format ... 540
Table – 446 TransportID formats for specific SCSI transport protocols ... 540
Table – 447 Fibre Channel TransportID format ... 541
Table – 448 Parallel SCSI bus TransportID format ... 541
Table – 449 IEEE 1394 TransportID format ... 542
Table – 450 RDMA TransportID format ... 542
Table – 451 iSCSI TransportID formats ... 543
Table – 452 iSCSI initiator device TransportID format ... 543
Table – 453 iSCSI initiator port TransportID format ... 544
Table – 454 SAS Serial SCSI Protocol TransportID format ... 545
Table – 455 SECURITY PROTOCOL SPECIFIC field for SECURITY PROTOCOL IN protocol 00h ... 546
Table – 456 Supported security protocols SECURITY PROTOCOL IN parameter data ... 547
Table – 457 Certificate data SECURITY PROTOCOL IN parameter data ... 548
Table – 458 SECURITY PROTOCOL SPECIFIC field for the SA creation capabilities ... 549
Table – 459 Supported device server capabilities formats parameter data ... 550
Table – 460 IKEv2-SCSI device server capabilities parameter data ... 550
Table – 461 SECURITY PROTOCOL SPECIFIC field for the IKEv2-SCSI SECURITY PROTOCOL IN command ... 551
Table – 462 SECURITY PROTOCOL SPECIFIC field for the IKEv2-SCSI SECURITY PROTOCOL OUT command ... 552
Table – 463 IKEv2-SCSI SECURITY PROTOCOL OUT and SECURITY PROTOCOL IN parameter data ... 553
Table – 464 IKEv2-SCSI header checking of SAIs ... 555
Table – 465 NEXT PAYLOAD field ... 556
Table – 466 MESSAGE ID field ... 557
Table – 467 Next payload values in SECURITY PROTOCOL OUT/IN parameter data ... 558
Table – 468 IKEv2-SCSI payload format ... 561
Table – 469 Key Exchange payload format ... 562
Table – 470 Identification payload format ... 563
Table – 471 ID TYPE field ... 563
Table – 472 Certificate payload format ... 564
Table – 473 CERTIFICATE ENCODING field ... 565
Table – 474 Certificate Request payload format ... 565
Table – 475 Authentication payload format ... 566
Table – 476 Nonce payload format ... 569
Table – 477 Notify payload format ... 570
Table – 478 Delete payload format ... 571
Table – 479 Encrypted payload format ... 573
Table – 480 Plaintext format for Encrypted payload CIPHERTEXT field ... 575
Table – 481 IKEv2-SCSI SA Creation Capabilities payload format ... 578
Table – 482 IKEv2-SCSI SA Cryptographic Algorithms payload format ... 579
Table – 483 IKEv2-SCSI SAUT Cryptographic Algorithms payload format ... 581
Table – 484 IKEv2-SCSI Timeout Values payload format ... 583
Table – 485 IKEv2-SCSI cryptographic algorithm descriptor format ... 584

Table – 486 ALGORITHM TYPE field ... 584
Table – 487 ENCR IKEv2-SCSI cryptographic algorithm descriptor format ... 585
Table – 488 ENCR ALGORITHM IDENTIFIER field ... 586
Table – 489 PRF IKEv2-SCSI cryptographic algorithm descriptor format ... 587
Table – 490 PRF ALGORITHM IDENTIFIER field ... 588
Table – 491 INTEG IKEv2-SCSI cryptographic algorithm descriptor format ... 589
Table – 492 INTEG ALGORITHM IDENTIFIER field ... 589
Table – 493 D-H IKEv2-SCSI cryptographic algorithm descriptor format ... 590
Table – 494 D-H ALGORITHM IDENTIFIER field ... 591
Table – 495 SA_AUTH_OUT and SA_AUTH_IN IKEv2-SCSI cryptographic algorithm descriptor format ... 592
Table – 496 SA_AUTH_OUT and SA_AUTH_IN ALGORITHM IDENTIFIER field ... 593
Table – 497 IKEv2-SCSI command ordering processing requirements on a single I_T_L nexus ... 595
Table – 498 IKEv2-SCSI parameter error categories ... 597
Table – 499 IKEv2 Notify payload error translations for IKEv2-SCSI ... 599
Table – 500 SECURITY PROTOCOL SPECIFIC field for the CbCS SECURITY PROTOCOL IN command ... 600
Table – 501 Supported CbCS SECURITY PROTOCOL IN Pages CbCS page format ... 601
Table – 502 Supported CbCS SECURITY PROTOCOL OUT Pages CbCS page format ... 602
Table – 503 Unchangeable CbCS Parameters CbCS page format ... 603
Table – 504 KEYS SUPPORT field ... 604
Table – 505 MIN CBCS METHOD SUP field ... 604
Table – 506 Security Token CbCS page format ... 605
Table – 507 Current CbCS Parameters CbCS page format ... 606
Table – 508 Set Master Key – Seed Exchange CbCS page format ... 608
Table – 509 SECURITY PROTOCOL SPECIFIC field for the CbCS SECURITY PROTOCOL OUT command ... 610
Table – 510 Set Policy Access Tag CbCS page format ... 611
Table – 511 Set Minimum CbCS Method CbCS page format ... 612
Table – 512 Minimum CbCS Method CbCS Parameter set ... 612
Table – 513 Invalidate Key CbCS page format ... 613
Table – 514 Set Key CbCS page format ... 614
Table – 515 Set Master Key – Seed Exchange CbCS page format ... 615
Table – 516 Set Master Key – Change Master Key CbCS page format ... 616
Table – 517 Vital product data page codes ... 618
Table – 518 VPD page format ... 619
Table – 519 ASCII Information VPD page ... 620
Table – 520 CFA Profile Information VPD page ... 621
Table – 521 CFA profile descriptor ... 621
Table – 522 Device Constituents VPD page ... 622
Table – 523 Constituent descriptor ... 623
Table – 524 CONSTITUENT TYPE field ... 623
Table – 525 CONSTITUENT DEVICE TYPE field ... 624
Table – 526 Virtual tape library constituent type specific descriptor format ... 624
Table – 527 VIRTUAL TAPE LIBRARY SPECIFIC TYPE field ... 624
Table – 528 Virtual tape drive constituent type specific descriptor format ... 625
Table – 529 VIRTUAL TAPE DRIVE SPECIFIC TYPE field ... 625
Table – 530 Device Identification VPD page ... 626
Table – 531 Designation descriptor ... 626
Table – 532 ASSOCIATION field ... 627
Table – 533 DESIGNATOR TYPE field ... 627
Table – 534 Vendor specific DESIGNATOR field format ... 630
Table – 535 T10 vendor ID based DESIGNATOR field format ... 630
Table – 536 EUI-64 based designator lengths ... 631
Table – 537 EUI-64 DESIGNATOR field format ... 631
Table – 538 EUI-64 based 12-byte DESIGNATOR field format ... 632
Table – 539 EUI-64 based 16-byte DESIGNATOR field format ... 632

Table – 540 NAA DESIGNATOR field format ... 633
Table – 541 Name Address Authority (NAA) field ... 633
Table – 542 NAA IEEE Extended DESIGNATOR field format ... 633
Table – 543 NAA Locally Assigned DESIGNATOR field format ... 634
Table – 544 NAA IEEE Registered DESIGNATOR field format ... 634
Table – 545 NAA IEEE Registered Extended DESIGNATOR field format ... 635
Table – 546 Relative target port DESIGNATOR field format ... 635
Table – 547 RELATIVE TARGET PORT IDENTIFIER field ... 636
Table – 548 Target port group DESIGNATOR field format ... 636
Table – 549 Logical unit group DESIGNATOR field format ... 636
Table – 550 MD5 logical unit DESIGNATOR field format ... 637
Table – 551 MD5 logical unit identifier example available data ... 637
Table – 552 Example MD5 input for computation of a logical unit identifier ... 638
Table – 553 SCSI name string DESIGNATOR field format ... 638
Table – 554 Extended INQUIRY Data VPD page ... 639
Table – 555 ACTIVATE MICROCODE field ... 640
Table – 556 SPT field for peripheral device type 00h ... 640
Table – 557 SPT field for peripheral device type 01h ... 641
Table – 558 Management Network Addresses VPD page ... 643
Table – 559 Network service descriptor format ... 643
Table – 560 SERVICE TYPE field ... 644
Table – 561 Mode Page Policy VPD page ... 644
Table – 562 Mode page policy descriptor ... 645
Table – 563 MODE PAGE POLICY field ... 646
Table – 564 Power Condition VPD page ... 646
Table – 565 SCSI Ports VPD page ... 648
Table – 566 SCSI port designation descriptor ... 649
Table – 567 RELATIVE PORT IDENTIFIER field ... 649
Table – 568 Target port descriptor ... 650
Table – 569 Protocol Specific Logical Unit Information VPD page ... 651
Table – 570 Logical unit information descriptor ... 651
Table – 571 Protocol Specific Port Information VPD page ... 652
Table – 572 Port information descriptor ... 653
Table – 573 Software Interface Identification VPD page ... 654
Table – 574 Software interface identifier format ... 654
Table – 575 Supported VPD Pages VPD page ... 655
Table – 576 Unit Serial Number VPD page ... 655
Table – 577 Well known logical unit numbers ... 657
Table – 578 Commands for the REPORT LUNS well known logical unit ... 657
Table – 579 Commands for the ACCESS CONTROLS well known logical unit ... 658
Table – 580 ACCESS CONTROL OUT management identifier key requirements ... 661
Table – 581 ACCESS CONTROL IN management identifier key requirements ... 662
Table – 582 Mandatory access controls resources ... 672
Table – 583 Optional access controls resources ... 673
Table – 584 ACCESS IDENTIFIER TYPE field ... 673
Table – 585 AccessID access identifier format ... 673
Table – 586 ACCESS CONTROL IN service actions ... 674
Table – 587 ACCESS CONTROL IN command with REPORT ACL service action ... 675
Table – 588 ACCESS CONTROL IN with REPORT ACL parameter data format ... 676
Table – 589 ACL data page codes ... 676
Table – 590 Granted ACL data page format ... 677
Table – 591 Granted ACL data page LUACD descriptor format ... 678
Table – 592 ACCESS MODE field ... 678
Table – 593 Granted All ACL data page format ... 679

Table – 594 Proxy Tokens ACL data page format ... 680
Table – 595 Proxy token descriptor format ... 680
Table – 596 ACCESS CONTROL IN command with REPORT LU DESCRIPTORS service action ... 681
Table – 597 ACCESS CONTROL IN with REPORT LU DESCRIPTORS parameter data format ... 682
Table – 598 SUPPORTED LUN MASK FORMAT field format ... 683
Table – 599 Logical Unit descriptor format ... 684
Table – 600 ACCESS CONTROL IN command with REPORT ACCESS CONTROLS LOG service action ... 686
Table – 601 CDB LOG PORTION field ... 687
Table – 602 ACCESS CONTROL IN with REPORT ACCESS CONTROLS LOG parameter data format ... 687
Table – 603 Parameter data LOG PORTION field ... 688
Table – 604 Key Overrides access controls log portion page format ... 688
Table – 605 Invalid Keys access controls log portion page format ... 689
Table – 606 ACL LUN Conflicts access controls log portion page format ... 690
Table – 607 ACCESS CONTROL IN command with REPORT OVERRIDE LOCKOUT TIMER service action ... 691
Table – 608 ACCESS CONTROL IN with REPORT OVERRIDE LOCKOUT TIMER parameter data ... 692
Table – 609 ACCESS CONTROL IN command with REQUEST PROXY TOKEN service action ... 693
Table – 610 ACCESS CONTROL IN with REQUEST PROXY TOKEN parameter data ... 694
Table – 611 ACCESS CONTROL OUT service actions ... 694
Table – 612 ACCESS CONTROL OUT command format ... 695
Table – 613 ACCESS CONTROL OUT with MANAGE ACL parameter data format ... 696
Table – 614 ACE page codes ... 697
Table – 615 Grant/Revoke ACE page format ... 698
Table – 616 ACE page LUACD descriptor format ... 699
Table – 617 Access Coordinator Grant/Revoke ACE page actions ... 700
Table – 618 Grant All ACE page format ... 701
Table – 619 Revoke Proxy Token ACE page format ... 702
Table – 620 Revoke All Proxy Tokens ACE page format ... 702
Table – 621 ACCESS CONTROL OUT with DISABLE ACCESS CONTROLS parameter data format ... 703
Table – 622 ACCESS CONTROL OUT with ACCESS ID ENROLL parameter data format ... 704
Table – 623 ACCESS CONTROL OUT with CLEAR ACCESS CONTROLS LOG parameter data format ... 706
Table – 624 CLEAR ACCESS CONTROLS LOG LOG PORTION field ... 706
Table – 625 ACCESS CONTROL OUT with MANAGE OVERRIDE LOCKOUT TIMER parameter data format ... 707
Table – 626 ACCESS CONTROL OUT with OVERRIDE MGMT ID KEY parameter data format ... 708
Table – 627 ACCESS CONTROL OUT with REVOKE PROXY TOKEN parameter data format ... 709
Table – 628 ACCESS CONTROL OUT with REVOKE ALL PROXY TOKENS parameter data format ... 709
Table – 629 ACCESS CONTROL OUT with ASSIGN PROXY LUN parameter data format ... 710
Table – 630 ACCESS CONTROL OUT with RELEASE PROXY LUN parameter data format ... 712
Table – 631 Commands for the TARGET LOG PAGES well known logical unit ... 712
Table – 632 Commands for the SECURITY PROTOCOL well known logical unit ... 713
Table – 633 Commands for the MANAGEMENT PROTOCOL well known logical unit ... 713
Table – 634 Commands for a security manager ... 714
Table – A.1 This standard to SPC-2 terminology mapping ... 715
Table – B.1 PERSISTENT RESERVE OUT command features ... 716
Table – C.1 IKE payload names shorthand ... 720
Table – D.1 ASC and ASCQ assignments ... 721
Table – D.2 Operation codes ... 739
Table – D.3 Additional operation codes for devices with the ENCSERV bit set to one ... 745
Table – D.4 MAINTENANCE (IN) and MAINTENANCE (OUT) service actions ... 746
Table – D.5 SERVICE ACTION IN(12) and SERVICE ACTION OUT(12) service actions ... 747
Table – D.6 SERVICE ACTION IN(16) and SERVICE ACTION OUT(16) service actions ... 747
Table – D.7 Variable Length CDB Service Action Code Ranges ... 748
Table – D.8 Variable Length CDB Service Action Codes Used by All Device Types ... 748
Table – D.9 Diagnostic page codes ... 749
Table – D.10 Log page codes ... 750

Table – D.11 Transport protocol specific log page codes 752
Table – D.12 Mode page codes 753
Table – D.13 Transport protocol specific mode page codes 755
Table – D.14 VPD page codes 756
Table – D.15 Transport protocol specific VPD page codes 757
Table – D.16 Version descriptor assignments 758
Table – D.17 Standard code value guidelines 771
Table – D.18 IEEE binary identifiers assigned by T10 777
Table – E.1 T10 vendor identification list 778

Figure – 0 SCSI document structure lxiii
Figure – 1 Example state diagram 28
Figure – 2 Example class association relationships 31
Figure – 3 Example class aggregation relationships 32
Figure – 4 Example class generalization relationships 33
Figure – 5 Example class dependency relationships 34
Figure – 6 Device server interpretation of PREEMPT service action 108
Figure – 7 Primary target port group example 113
Figure – 8 Power condition state machine 124
Figure – 9 SA relationships 137
Figure – 10 IKEv2-SCSI Device Server Capabilities step 148
Figure – 11 IKEv2-SCSI Key Exchange step 148
Figure – 12 IKEv2-SCSI Authentication step 149
Figure – 13 IKEv2-SCSI Delete operation 150
Figure – 14 CbCS overview class diagram 175
Figure – 15 ACL Structure 660

Foreword

This foreword is not part of American National Standard INCITS.***:200x.

The SCSI command set is designed to provide efficient peer-to-peer operation of SCSI devices (e.g., disks, tapes, printers) by an operating system. The SCSI command set assumes an underlying command-response protocol.

The SCSI command set provides multiple operating systems concurrent control over one or more SCSI devices. However, proper coordination of activities between the multiple operating systems is critical to avoid data corruption. Commands that assist with coordination between multiple operating systems are described in this standard. However, details of the coordination are beyond the scope of the SCSI command set.

This standard defines the device model for all SCSI devices. This standard defines the SCSI commands that are basic to every device model and the SCSI commands that may apply to any device model.

With any technical document there may arise questions of interpretation as new products are implemented. INCITS has established procedures to issue technical opinions concerning the standards developed by INCITS. These procedures may result in SCSI Technical Information Bulletins being published by INCITS.

These Bulletins, while reflecting the opinion of the Technical Committee that developed the standard, are intended solely as supplementary information to other users of the standard. This standard, ANSI INCITS.***:200x, as approved through the publication and voting procedures of the American National Standards Institute, is not altered by these bulletins. Any subsequent revision to this standard may or may not reflect the contents of these Technical Information Bulletins.

Current INCITS practice is to make Technical Information Bulletins available through:

INCITS Online Store	http://www.techstreet.com/incitsgate.tmpl
managed by Techstreet	Telephone: 1-734-302-7801 or
1327 Jones Drive	1-800-699-9277
Ann Arbor, MI 48105	Facsimile: 1-734-302-7811

or

Global Engineering	http://global.ihs.com/
15 Inverness Way East	Telephone: 1-303-792-2181 or
Englewood, CO 80112-5704	1-800-854-7179
	Facsimile: 1-303-792-2192

Requests for interpretation, suggestions for improvement and addenda, or defect reports are welcome. They should be sent to the INCITS Secretariat, National Committee for Information Technology Standards, Information Technology Industry Council, 1101 K Street NW, Suite 610, Washington, DC 20005-3922.

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 it approved this standard, INCITS had the following members:

<<Insert INCITS member list>>

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

John B. Lohmeyer, Chair
Mark S. Evans, Vice-Chair
Ralph O. Weber, Secretary

<<Insert T10 member list>>

Introduction

The SCSI Primary Commands - 4 (SPC-4) standard is divided into the following clauses and annexes:

Clause 1 is the scope.
Clause 2 enumerates the normative references that apply to this standard.
Clause 3 describes the definitions, symbols, and abbreviations used in this standard.
Clause 4 describes the conceptual relationship between this document and the SCSI-3 Architecture Model.
Clause 5 describes the command model for all SCSI devices.
Clause 6 defines the commands that may be implemented by any SCSI device.
Clause 7 defines the parameter data formats that may be implemented by any SCSI device.
Clause 8 defines the well known logical units that may be implemented by any SCSI device.
Annex A identifies differences between the terminology used in this standard and previous versions of this standard. (informative)
Annex B describes the PERSISTENT RESERVE OUT command features necessary to replace the reserve/release management method and provides guidance on how to perform a third party reservation using persistent reservations. (informative)
Annex C identifies the differences between security protocols defined by other standards (e.g., IKEv2 (see RFC 4306)) and the equivalent protocols defined by this standard (e.g., the IKEv2-SCSI SA creation protocol). (informative)
Annex D lists code values in numeric order. (informative)
Annex E lists assigned vendor identifiers. (informative)

The annexes provide information to assist with implementation of this standard. The information in the annexes applies to all the SCSI command standards. See 3.1.31 for more information about other SCSI command standards.

SCSI standards family

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

Figure 0 — SCSI document structure

The SCSI document structure in figure 0 is intended to show the general applicability of the documents to one another. Figure 0 is not intended to imply a relationship such as a hierarchy, protocol stack, or system architecture.

SCSI Architecture Model: Defines the SCSI systems model, the functional partitioning of the SCSI standard set and requirements applicable to all SCSI implementations and implementation standards.

Device-Type Specific Command Sets: Implementation standards that define specific device types including a device model for each device type. These standards specify the required commands and behaviors that are specific to a given device type and prescribe the requirements to be followed by a SCSI initiator device when sending commands to a SCSI target device having the specific device type. The commands and behaviors for a specific device type may include by reference commands and behaviors that are shared by all SCSI devices.

Shared Command Set: An implementation standard that defines a model for all SCSI device types. This standard specifies the required commands and behavior that is common to all SCSI devices, regardless of device type, and prescribes the requirements to be followed by a SCSI initiator device when sending commands to any SCSI target device.

SCSI Transport Protocols: Implementation standards that define the requirements for exchanging information so that different SCSI devices are capable of communicating.

Interconnects: Implementation standards that define the communications mechanism employed by the SCSI transport protocols. These standards may describe the electrical and signaling requirements essential for SCSI devices to interoperate over a given interconnect. Interconnect standards may allow the interconnection of devices other than SCSI devices in ways that are outside the scope of this standard.

The term SCSI is used to refer to the family of standards described in this subclause.

At the time this standard was generated, examples of the SCSI general structure included:

Interconnects:

Fibre Channel Arbitrated Loop - 2	FC-AL-2	[ISO/IEC 14165-122] [ANSI INCITS 332-1999] [ANSI INCITS 332/AM1-2003] [ANSI INCITS 332/AM2-2006]
Fibre Channel Physical Interfaces	FC-PI	[ISO/IEC 14165-115] [ANSI INCITS 352-2002]
Fibre Channel Physical Interfaces - 2	FC-PI-2	[ISO/IEC 14165-142] [ANSI INCITS 404-2006]
Fibre Channel Physical Interfaces - 3	FC-PI-3	[ISO/IEC 14165-143] [T11/1625-D]
Fibre Channel Physical Interfaces - 4	FC-PI-4	[ISO/IEC 14165-144] [ANSI INCITS 450-2009]
Fibre Channel Physical Interfaces - 5	FC-PI-5	[ISO/IEC 14165-145] [T11/2112-D]
Fibre Channel Physical Interfaces - 6	FC-PI-6	[ISO/IEC 14165-146] [T11/2221-D]
Fibre Channel - 10 Gigabit	10GFC	[ISO/IEC 14165-116] [ANSI INCITS 364-2003] [ANSI INCITS 364/AM1-2007]
Fibre Channel Framing and Signaling Interface	FC-FS	[ISO/IEC 14165-251] [ANSI INCITS 373-2003]
Fibre Channel Framing and Signaling Interface - 2	FC-FS-2	[ISO/IEC 14165-252] [ANSI INCITS 424-2007] [ANSI INCITS 424/AM1-2007]

Fibre Channel Framing and Signaling Interface - 3	FC-FS-3	[ISO/IEC 14165-253] [T11/1861-D]
Fibre Channel Link Services	FC-LS	[ANSI INCITS 433-2007]
Fibre Channel Link Services - 2	FC-LS-2	[T11/2103-D]
Fibre Channel Security Protocols	FC-SP	[ISO/IEC 14165-431] [ANSI INCITS 426-2007]
Fibre Channel Security Protocols - 2	FC-SP-2	[ISO/IEC 14165-432] [T11/1835-D]
High Performance Serial Bus		[ANSI IEEE 1394-1995]
High Performance Serial Bus (supplement to ANSI/IEEE 1394-1995)		[ANSI IEEE 1394a-2000]
SCSI Parallel Interface - 2	SPI-2	[ISO/IEC 14776-112] [ANSI INCITS 302-1999]
SCSI Parallel Interface - 3	SPI-3	[ISO/IEC 14776-113] [ANSI INCITS 336-2000]
SCSI Parallel Interface - 4	SPI-4	[ISO/IEC 14776-114] [ANSI INCITS 362-2002]
SCSI Parallel Interface - 5	SPI-5	[ISO/IEC 14776-115] [ANSI INCITS 367-2003]
Serial Storage Architecture Physical Layer 1	SSA-PH	[ANSI INCITS 293-1996]
Serial Storage Architecture Physical Layer 2	SSA-PH-2	[ANSI INCITS 307-1998]
Serial Attached SCSI	SAS	[ISO/IEC 14776-150] [ANSI INCITS 376-2003]
Serial Attached SCSI - 1.1	SAS-1.1	[ISO/IEC 14776-151] [ANSI INCITS 417-2006]
Serial Attached SCSI - 2	SAS-2	[ISO/IEC 14776-152] [T10/1760-D]
Serial Attached SCSI - 2.1	SAS-2.1	[ISO/IEC 14776-153] [T10/2125-D]
Serial Attached SCSI - 3	SAS-3	[ISO/IEC 14776-154] [T10/2212-D]

SCSI Transport Protocols:

Automation/Drive Interface - Transport Protocol	ADT	[ANSI INCITS 406-2005]
Automation/Drive Interface - Transport Protocol - 2	ADT-2	[ISO/IEC 14776-192] [T10/1742-D]
Serial Storage Architecture Transport Layer 1	SSA-TL-1	[ANSI INCITS 295-1996]
Serial Storage Architecture Transport Layer 2	SSA-TL-2	[ANSI INCITS 308-1998]
SCSI-3 Fibre Channel Protocol	FCP	[ISO/IEC 14776-221] [ANSI INCITS 269-1996]
Fibre Channel Protocol for SCSI - 2	FCP-2	[ISO/IEC 14776-222] [ANSI INCITS 350-2003]
Fibre Channel Protocol for SCSI - 3	FCP-3	[ISO/IEC 14776-223] [ANSI INCITS 416-2006]
Fibre Channel Protocol for SCSI - 4	FCP-4	[ISO/IEC 14776-223] [T10/1828-D]
Serial Bus Protocol - 2	SBP-2	[ISO/IEC 14776-232] [ANSI INCITS 325-1998]
Serial Bus Protocol - 3	SBP-3	[ANSI INCITS 375-2004]
Serial Storage Architecture SCSI-3 Protocol	SSA-S3P	[ANSI INCITS 309-1998]
SCSI RDMA Protocol	SRP	[ISO/IEC 14776-241] [ANSI INCITS 365-2002]
USB Queuing Transport	UAS	[ISO/IEC 14776-251] [T10/2095-D]

USB Queuing Transport - 2	UAS-2	[ISO/IEC 14776-252] [T10/2229-D]
SAS Protocol Layer	SPL	[ISO/IEC 14776-261] [T10/2124-D]
SAS Protocol Layer - 2	SPL-2	[ISO/IEC 14776-262] [T10/2228-D]
Primary Command Set:		
SCSI-3 Primary Commands	SPC	[ANSI INCITS 301-1997]
SCSI Primary Commands - 2	SPC-2	[ISO/IEC 14776-452] [ANSI INCITS 351-2001]
SCSI Primary Commands - 3	SPC-3	[ISO/IEC 14776-453] [ANSI INCITS 408-2005]
SCSI Primary Commands - 4	SPC-4	[ISO/IEC 14776-454] [T10/1731-D]
Device-Type Specific Command Sets:		
SCSI-3 Block Commands	SBC	[ISO/IEC 14776-321] [ANSI INCITS 306-1998]
SCSI Block Commands - 2	SBC-2	[ISO/IEC 14776-322] [ANSI INCITS 405-2005]
SCSI Block Commands - 3	SBC-3	[ISO/IEC 14776-323] [T10/1799-D]
SCSI-3 Stream Commands	SSC	[ISO/IEC 14776-331] [ANSI INCITS 335-2000]
SCSI Stream Commands - 2	SSC-2	[ISO/IEC 14776-332] [ANSI INCITS 380-2003]
SCSI Stream Commands - 3	SSC-3	[ISO/IEC 14776-333] [T10/1611-D]
SCSI Stream Commands - 4	SSC-5	[ISO/IEC 14776-334] [T10/2123-D]
SCSI-3 Medium Changer Commands	SMC	[ISO/IEC 14776-351] [ANSI INCITS 314-1998]
SCSI Media Changer Commands - 2	SMC-2	[ISO/IEC 14776-352] [ANSI INCITS 382-2004]
SCSI Media Changer Commands - 3	SMC-3	[ISO/IEC 14776-353] [T10/1730-D]
SCSI-3 Multimedia Command Set	MMC	[ANSI INCITS 304-1997]
SCSI Multimedia Command Set - 2	MMC-2	[ISO/IEC 14776-362] [ANSI INCITS 333-2000]
SCSI Multimedia Command Set - 3	MMC-3	[ISO/IEC 14776-363] [ANSI INCITS 360-2002]
SCSI Multimedia Command Set - 4	MMC-4	[ISO/IEC 14776-364] [ANSI INCITS 401-2005]
SCSI Multimedia Command Set - 5	MMC-5	[ISO/IEC 14776-365] [ANSI INCITS 430-2007]
SCSI Multimedia Command Set - 6	MMC-6	[ISO/IEC 14776-366] [T10/1836-D]
SCSI Controller Commands - 2	SCC-2	[ISO/IEC 14776-342] [ANSI INCITS 318-1998]
SCSI Reduced Block Commands	RBC	[ISO/IEC 14776-326] [ANSI INCITS 330-2000] [ANSI INCITS 330-2003/AM1]

SCSI-3 Enclosure Services Commands	SES	[ISO/IEC 14776-371] [ANSI INCITS 305-1998] [ANSI INCITS 305-2000/AM1]
SCSI Enclosure Services Commands - 2	SES-2	[ISO/IEC 14776-372] [ANSI INCITS 448-2008]
SCSI Enclosure Services Commands - 3	SES-3	[ISO/IEC 14776-373] [T10/2149-D]
SCSI Specification for Optical Card Reader/Writer	OCRW	[ISO/IEC 14776-381]
Automation/Drive Interface - Commands	ADC	[ISO/IEC 14776-356] [ANSI INCITS 403-2005]
Automation/Drive Interface - Commands - 2	ADC-2	[ISO/IEC 14776-357] [T10/1741-D]
Automation/Drive Interface - Commands - 3	ADC-3	[ISO/IEC 14776-358] [T10/1895-D]
Object-based Storage Device Commands	OSD	[ANSI INCITS 400-2004]
Object-based Storage Device Commands - 2	OSD-2	[T10/1729-D]
Object-based Storage Device Commands - 3	OSD-3	[T10/2128-D]

Translation Protocols:

SCSI / ATA Translation	SAT	[ISO/IEC 14776-921] [ANSI INCITS 431-2007]
SCSI / ATA Translation - 2	SAT-2	[ISO/IEC 14776-922] [T10/1826-D]
SCSI / ATA Translation - 3	SAT-3	[ISO/IEC 14776-923] [T10/2126-D]

Architecture Model:

SCSI-3 Architecture Model	SAM	[ISO/IEC 14776-411] [ANSI INCITS 270-1996]
SCSI Architecture Model - 2	SAM-2	[ISO/IEC 14776-412] [ANSI INCITS 366-2003]
SCSI Architecture Model - 3	SAM-3	[ISO/IEC 14776-413] [ANSI INCITS 402-2005]
SCSI Architecture Model - 4	SAM-4	[ISO/IEC 14776-414] [ANSI INCITS 447-2008]
SCSI Architecture Model - 5	SAM-5	[ISO/IEC 14776-415] [T10/2104-D]

AMERICAN NATIONAL STANDARD **INCITS.***:200x**

American National Standard for Information Technology -

SCSI Primary Commands - 4 (SPC-4)

1 Scope

The SCSI family of standards provides for many different types of SCSI devices (e.g., disks, tapes, printers, scanners). This standard defines a device model that is applicable to all SCSI devices. Other SCSI command standards (see 3.1.31) expand on the general SCSI device model in ways appropriate to specific types of SCSI devices.

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.

This standard defines the SCSI commands that are mandatory and optional for all SCSI devices. Support for any feature defined in this standard is optional unless otherwise stated. This standard also defines the SCSI commands that may apply to any device model.

The following commands, parameter data, and features defined in previous versions of this standard are made obsolete by this standard:

a) the TARGET RESET supported (TRS) bit and the WAKEUP supported (WAKES) bit in the REPORT SUPPORTED TASK MANAGEMENT FUNCTIONS parameter data;
b) code value 10b (i.e., Per initiator port) in the MODE PAGE POLICY field in the mode page policy descriptor in the Mode Page Policy VPD page;
c) the removable medium devices with an attached medium changer model, MCHNGR bit in the standard INQUIRY data, the MOVE MEDIUM ATTACHED command in disks and tapes, and the READ ELEMENT STATUS ATTACHED command in disks and tapes;
d) linked commands; and
e) the PPC bit in the LOG SENSE command.

2 Normative references

2.1 General

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.

2.2 Approved references

Copies of the following documents may be obtained from ANSI, an ISO member organization:

a) approved ANSI standards;

b) approved international and regional standards (ISO and IEC); and
c) approved foreign standards (including JIS and DIN).

For further information, contact the ANSI Customer Service Department:

Phone +1 212-642-4980
Fax: +1 212-302-1286
Web: http://www.ansi.org
E-mail: ansionline@ansi.org

or the InterNational Committee for Information Technology Standards (INCITS):

Phone +1 202-626-5738
Web: http://www.incits.org
E-mail: incits@itic.org

ISO/IEC 14776-412, *SCSI Architecture Model - 2 (SAM-2) [ANSI INCITS 366-2003]*

ISO/IEC 14776-414, *SCSI Architecture Model - 4 (SAM-4) [ANSI INCITS 447-2008]*

ISO/IEC 14776-452, *SCSI Primary Commands - 2 (SPC-2) [ANSI INCITS 351-2001]*

ISO/IEC 14776-113, *SCSI Parallel Interface - 5 (SPI-5) [ANSI INCITS 367-2003]*

ISO/IEC 14776-223, *Fibre Channel Protocol for SCSI - 3 (FCP-3) [ANSI INCITS 416-2006]*

ISO/IEC 14776-241, *SCSI RDMA Protocol (SRP) [ANSI INCITS 365-2002]*

ISO/IEC 14776-150, *Serial Attached SCSI (SAS) [ANSI INCITS 376-2003]*

ISO/IEC 14776-921, *SCSI / ATA Translation (SAT) [ANSI INCITS 431-2007]*

ISO/IEC 14776-922, *SCSI / ATA Translation - 2 (SAT-2) [ANSI INCITS 465-2010]*

ISO/IEC 14776-322, *SCSI Block Commands - 2 (SBC-2) [ANSI INCITS 405-2005]*

ISO/IEC 14776-332, *SCSI Stream Commands - 2 (SSC-2) [ANSI INCITS 380-2003]*

ISO/IEC 14776-333, *SCSI Stream Commands - 3 (SSC-3) [ANSI INCITS 467-201x]*

ISO/IEC 14776-352, *SCSI Media Changer Commands - 2, (SMC-2) [ANSI INCITS 382:2004]*

ANSI INCITS 309-1998, *Serial Storage Architecture SCSI-3 Protocol (SSA-S3P)*

ANSI INCITS 375-2004, *Serial Bus Protocol - 3 (SBP-3)*

ANSI INCITS 406-2005, *Automation/Drive Interface - Transport Protocol (ADT)*

ISO/IEC 14165-251, *Fibre Channel Framing and Signaling Interface (FC-FS) [ANSI INCITS 373-2003]*

ISO/IEC 14165-431, *Fibre Channel Security Protocols (FC-SP) [ANSI INCITS 426-2007]*

ANSI INCITS 466-201x, *Single-Byte Command Code Sets-4 Mapping Protocol (FC-SB-4)*

ISO/IEC 24739 (all parts), *Information technology – (all parts) AT Attachment with Packet Interface - 7 (AT/ATAPI-7)*

IEC 60027-2 ed3.0, *Letter symbols to be used in electrical technology - Part 2: Telecommunications and electronics*

ANSI/IEEE 1394a-2000, *High Performance Serial Bus (supplement to ANSI/IEEE 1394-1995)*

ANSI/IEEE 1619.1-20xx, *Standard for Authenticated Encryption with Length Expansion for Storage Devices*

ANSI INCITS 4-1986 (R2002), *Information Systems - Coded Character Sets - 7-Bit American National Standard Code for Information Interchange (7-Bit ASCII)*

ISO/IEC 13213, *Information technology - Microprocessor systems - Control and Status Registers Architecture for microcomputer buses [ANSI/IEEE 1212, 2001 Edition]*

ISO/IEC 646:1991, *Information technology - ISO 7-bit coded character set for information interchange (third edition).*

ISO/IEC 8859-1:1998, *Information technology - 8-bit single-byte coded graphic character sets - Part 1: Latin alphabet No. 1*

ISO/IEC 8859-2:1999, *Information technology - 8-bit single-byte coded graphic character sets - Part 2: Latin alphabet No. 2*

ISO/IEC 8859-3:1999, *Information technology - 8-bit single-byte coded graphic character sets - Part 3: Latin alphabet No. 3*

ISO/IEC 8859-4:1998, *Information technology - 8-bit single-byte coded graphic character sets - Part 4: Latin alphabet No. 4*

ISO/IEC 8859-5:1999, *Information technology - 8-bit single-byte coded graphic character sets - Part 5: Latin/Cyrillic alphabet*

ISO/IEC 8859-6:1999, *Information technology - 8-bit single-byte coded graphic character sets - Part 6: Latin/Arabic alphabet*

ISO/IEC 8859-7:1987, *Information processing - 8-bit single-byte coded graphic character sets - Part 7: Latin/Greek alphabet*

ISO/IEC 8859-8:1999, *Information technology - 8-bit single-byte coded graphic character sets - Part 8: Latin/Hebrew alphabet*

ISO/IEC 8859-9:1999, *Information technology - 8-bit single-byte coded graphic character sets - Part 9: Latin alphabet No. 5*

ISO/IEC 8859-10:1998, *Information technology - 8-bit single-byte coded graphic character sets - Part 10: Latin alphabet No. 6*

ISO/IEC 10646:2003, *Information technology - Universal Multiple-Octet Coded Character Set (UCS)*

ISO/IEC 9594-8:2001, *Information technology - Open Systems Interconnection - The Directory: Public-key and attribute certificate frameworks [ITU-T Recommendation X.509:2000 – http://www.itu.int/ITU-T/]*

ISO/IEC 9594-8:2001/Cor 1:2002, *Information technology - Open Systems Interconnection - The Directory: Public-key and attribute certificate frameworks*

ISO/IEC 9594-8:2001/Cor 2:2002, *Information technology - Open Systems Interconnection - The Directory: Public-key and attribute certificate frameworks*

ISO/IEC 9594-8:2001/Cor 3:2005, *Information technology - Open Systems Interconnection - The Directory: Public-key and attribute certificate frameworks*

ISO/IEC 8824-1:2002, *Information technology - Abstract Syntax Notation One (ASN.1): Specification of basic notation*

ISO/IEC 8824-1:2002/Amd 1:2004, *Information technology - Abstract Syntax Notation One (ASN.1): Specification of basic notation*

ISO/IEC 8824-1:2002/Amd 2:2005, *Information technology - Abstract Syntax Notation One (ASN.1): Specification of basic notation*

ISO/IEC 8824-2:2002, *Information technology - Abstract Syntax Notation One (ASN.1): Information object specification*

ISO/IEC 8824-2:2002/Amd 1:2004, *Information technology - Abstract Syntax Notation One (ASN.1): Information object specification*

ISO/IEC 8824-3:2002, *Information technology - Abstract Syntax Notation One (ASN.1): Constraint specification*

ISO/IEC 8824-4:2002, *Information technology - Abstract Syntax Notation One (ASN.1): Parameterization of ASN.1 specifications*

ISO/IEC 9834-1:2005, *Information technology - Open Systems Interconnection - Procedures for the operation of OSI Registration Authorities: General procedures and top arcs of the ASN.1 Object Identifier tree*

ANSI/IEEE 1667, *Standard Protocol for Authentication in Host Attachments of Transient Storage Devices*

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 (SAM-5) [T10/2104-D]*

ISO/IEC 14776-323, *SCSI Block Commands - 3 (SBC-3) [T10/1799-D]*

ISO/IEC 14776-334, *SCSI Stream Commands - 4 (SSC-4) [T10/2123-D]*

ISO/IEC 14776-353, *SCSI Media Changer Commands - 3 (SMC-3) [T10/1730-D]*

ISO/IEC 14776-224, *Fibre Channel Protocol for SCSI - 4 (FCP-4) [T10/1828-D]*

ISO/IEC 14776-358, *Automation/Drive Interface - Commands - 3 (ADC-3) [T10/1895-D]*

ISO/IEC 14776-261, *SAS Protocol Layer (SPL) [T10/2124-D]*

ISO/IEC 14776-192, *Automation/Drive Interface* - Transport Protocol - *2 (ADT-2) [T10/1742-D]*

ISO/IEC 14776-923, *SCSI / ATA Translation - 3 (SAT-3) [T10/2126-D]*

2.4 NIST References

Copies of the following 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) 800-38D, *Recommendation for Block Cipher Modes of Operation: Galois/Counter (GCM) Mode for Confidentiality and Authentication*

FIPS 140-2, *Annex C: Approved Random Number Generators*

FIPS 180-2 with Change Notice 1 dated February 25, 2004, *Secure Hash Standard*

FIPS 198a, *The Keyed-Hash Message Authentication Code (HMAC)*

2.5 IETF References

Copies of the following approved IETF standards may be obtained through the Internet Engineering Task Force (IETF) at www.ietf.org.

RFC 791, *Internet Protocol - DARPA Internet Program - Protocol Specification*

RFC 793, *Transmission Control Protocol - DARPA Internet Program - Protocol Specification*

RFC 1035, *Domain Names - Implementation and Specification*

RFC 1321, *The MD5 Message-Digest Algorithm*

RFC 1591, *Domain Name System Structure and Delegation*

RFC 2104, *HMAC: Keyed-Hashing for Message Authentication*

RFC 2279, *UTF-8, a transformation format of ISO 10646*

RFC 2396, *Uniform Resource Identifiers (URI): Generic Syntax*

RFC 2410, *The NULL Encryption Algorithm and Its Use With IPsec*

RFC 2437, *PKCS #1: RSA Cryptography Specifications Version 2.0*

RFC 2616, *Hypertext Transfer Protocol - HTTP/1.1*

RFC 2753, *A Framework for Policy-based Admission Control*

RFC 3281, *An Internet Attribute Certificate Profile for Authorization*

RFC 3305, *Report from the Joint W3C/IETF URI Planning Interest Group: Uniform Resource Identifiers (URIs), URLs, and Uniform Resource Names (URNs): Clarifications and Recommendations*

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

RFC 3526, *More Modular Exponential (MODP) Diffie-Hellman groups for Internet Key Exchange (IKE)*

RFC 3552, *Guidelines for Writing RFC Text on Security Considerations*

RFC 3566, *The AES-XCBC-MAC-96 Algorithm and Its Use With IPsec*

RFC 3602, *The AES-CBC Cipher Algorithm and Its Use with IPsec*

RFC 3720, *Internet Small Computer Systems Interface (iSCSI)*

RFC 3766, *Determining Strengths For Public Keys Used For Exchanging Symmetric Keys*

RFC 4086, *Randomness Requirements for Security*

RFC 4106, *The Use of Galois/Counter Mode (GCM) in IPsec Encapsulating Security Payload*

RFC 4291, *IP Version 6 Addressing Architecture*

RFC 4303, *IP Encapsulating Security Payload (ESP)*

RFC 4306, *Internet Key Exchange (IKEv2) Protocol*

RFC 4309, *Using Advanced Encryption Standard (AES) CCM Mode with IPsec Encapsulating Security Payload*

RFC 4434, *The AES-XCBC-PRF-128 Algorithm for the Internet Key Exchange Protocol (IKE)*

RFC 4595, *Use of IKEv2 in the Fibre Channel Security Association Management Protocol*

RFC 4718, *IKEv2 Clarifications and Implementation Guidelines*

RFC 4753, *ECP Groups for IKE and IKEv2*

RFC Errata for RFC 4753, *http://rfc-editor.org/errata_search.php?rfc=4753*

RFC 4754, *IKE and IKEv2 Authentication Using the Elliptic Curve Digital Signature Algorithm (ECDSA)*

RFC 4868, *Using HMAC-SHA-256, HMAC-SHA-384, and HMAC-SHA-512 with IPsec*

RFC 5280, *Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*

RFC 5282, *Using Authenticated Encryption Algorithms with the Encrypted Payload of the Internet Key Exchange version 2 (IKEv2) Protocol*

2.6 OMG references

Copies of the following approved OMG specifications may be obtained through the Object Management Group (OMG) at www.omg.org.

OMG Unified Modeling Language (UML) Specification Version 1.5, March 2003

2.7 CompactFlash association references

Copies of the following standards may be obtained from the CompactFlash association at www.compactflash.org.

CF Performance Guarantee Profile 1 Revision 0.7r0 (VPG1), May 3, 2011

2.8 JEDEC references

Copies of the following standards may be obtained from JEDEC at www.jedec.org.

JEDEC JC64.1, *Universal Flash Storage (UFS) standard (in development)*

3 Definitions, symbols, abbreviations, and conventions

3.1 Definitions

3.1.1 access control list (ACL): The data used by a SCSI target device to configure access rights for initiator ports according to the access controls state of the SCSI target device (see 8.3.1.3).

3.1.2 access control list entry (ACE): One entry in the access control list (see 3.1.1).

3.1.3 access controls: An optional SCSI target device feature that restricts initiator port access to specific logical units and modifies the information about logical units in the parameter data of the INQUIRY and REPORT LUNS commands (see 8.3.1).

3.1.4 access controls coordinator: The entity within a SCSI target device that coordinates the management and enforcement of access controls (see 8.3.1) for all logical units within the SCSI target device. The access controls coordinator is always addressable through the ACCESS CONTROLS well known logical unit (see 8.3) and LUN 0.

3.1.5 active power condition: The power condition in which a device server is capable of completing the processing of all its supported commands. See 5.12.2.

3.1.6 additional authenticated data (AAD): Bytes of data input to a combined mode encryption algorithm (e.g., AES-GCM) as part of integrity checking computations (e.g., the IKEv2-SCSI Encrypted payload (see 7.7.3.5.11) uses AAD).

3.1.7 additional sense code: A combination of the ADDITIONAL SENSE CODE field and the ADDITIONAL SENSE CODE QUALIFIER field in sense data (see 4.5).

3.1.8 alias list: A list of alias values (see 3.1.9) and their associated designations (see 3.1.42) maintained by the device server and managed by the CHANGE ALIASES command (see 6.2) and REPORT ALIASES command (see 6.21).

3.1.9 alias value: A numeric value associated to a designation (see 3.1.42) in the alias list (see 3.1.8) and used in command or parameter data to reference a SCSI target device or SCSI target port. See 6.2.2.

3.1.10 application client: An object that is the source of SCSI commands. Further definition of an application client may be found in SAM-4.

3.1.11 ASCII format list log parameter: a log parameter that contains ASCII data in a list format. See 7.3.2.2.2.4.

3.1.12 attribute: A single unit of MAM (see 3.1.91) information.

3.1.13 auto contingent allegiance (ACA): The task set condition established following the return of CHECK CONDITION status when the NACA bit is set to one in the CONTROL byte. A detailed definition of ACA may be found in SAM-4.

3.1.14 background function: either a background scan operation (see SBC-3) or a device specific background function (see 5.7).

3.1.15 binary format list log parameter: a log parameter that contains binary data in a list format. See 7.3.2.2.2.5.

3.1.16 blocked command: A command that is in the blocked state. Commands become blocked when an ACA condition occurs. The blocked state ends when the ACA condition is cleared. A detailed definition of the blocked command state may be found in SAM-4.

3.1.17 bounded data counter log parameter: a log parameter that contains one saturating counter (see 3.1.135) value whose affect on other log parameters in the same log page is determined by the format and linking field in the parameter control byte and for which the establishment of thresholds is allowed. See 7.3.2.2.2.2.

3.1.18 byte: A sequence of eight contiguous bits considered as a unit.

3.1.19 cache memory: A temporary and often volatile data storage area outside the area accessible by application clients that may contain a subset of the data stored in the non-volatile data storage area.

3.1.20 CbCS capability: A descriptor that is a component of a CbCS credential (see 3.1.20) and a CbCS extension descriptor (see 5.16.6.8.16) that specifies access to a logical unit or volume (see SSC-3) for specific commands. See 5.16.6.8.13.

3.1.21 CbCS capability key: A shared key (see 3.1.162) that is cryptographically associated with a specific CbCS capability (see 3.1.20).

3.1.22 CbCS credential: The combination of a CbCS capability (see 3.1.20) and a CbCS capability key (see 3.1.21) that is returned by a RECEIVE CREDENTIAL command (see 6.19). See 5.16.6.8.12.

3.1.23 CbCS Management Application Client class: The CbCS (see 5.16.6.8) class whose objects manage, or an object that manages CbCS shared keys (see 5.16.6.8.11). See 5.16.6.8.4.

3.1.24 CbCS Management Device Server class: The CbCS (see 5.16.6.8) class whose objects prepare, or an object that prepares CbCS credentials (see 5.16.6.8.12) in response to RECEIVE CREDENTIAL command (see 6.19) requests from secure CDB originator (see 5.16.6.8.5). See 5.16.6.8.3.

3.1.25 CbCS master key: A shared key (see 3.1.162) from which CbCS working keys (see 3.1.28) are derived that is composed of an authentication key component and a generation key component. See 5.16.6.8.11.

3.1.26 CbCS security token: A random nonce (see 3.1.122) that is chosen by the enforcement manager (see 5.16.6.8.7) for a given I_T nexus and returned to a secure CDB originator (see 5.16.6.8.5). See 5.16.6.8.10.

3.1.27 CbCS working keys: Shared keys (see 3.1.162) that are used to cryptographically associate a CbCS capability (see 3.1.20) with a CbCS capability key (see 3.1.21). See 5.16.6.8.11.

3.1.28 code set enumeration: A coded value used in one field to indicate the format of data in other fields or descriptors. See 4.4.3.

3.1.29 command: A request describing a unit of work to be performed by a device server. A detailed definition of a command may be found in SAM-4.

3.1.30 command descriptor block (CDB): The structure used to communicate commands from an application client to a device server (see 4.3). A CDB may have a fixed length of up to 16 bytes or a variable length of between 12 and 260 bytes.

3.1.31 command standard: A SCSI standard that defines the model, commands, and parameter data for a device type (e.g., SBC-2, SSC-2, SMC-2, MMC-4, or SES-2). See clause 1.

3.1.32 company_id: Synonym for OUI (see 3.1.103).

3.1.33 Control mode page: A mode page that provides controls over SCSI features (e.g., task set management and error logging) that are applicable to all device types. See 7.5.7.

3.1.34 Control Extension mode page: A mode page that provides controls over SCSI features that are applicable to all device types. See 7.5.8.

3.1.35 copy manager: An object that manages the processing of copy commands (i.e., the EXTENDED COPY command (see 6.3) and the RECEIVE COPY RESULTS command (see 6.18)). See SAM-5.

3.1.36 copy target device: The name given by the EXTENDED COPY command (see 6.3) to a source or destination logical unit (i.e., a copy target device is a logical unit (see 3.1.84), not a SCSI target device (see 3.1.145)).

3.1.37 cryptographic command sequence (CCS): A defined sequence of SECURITY PROTOCOL IN commands (see 6.30) and SECURITY PROTOCOL OUT commands (see 6.31) that realize the cryptographic protocol of a specified security operation (e.g., the SECURITY PROTOCOL IN commands and SECURITY PROTOCOL OUT commands in the Key Exchange step and Authentication step, if any, of an IKEv2-SCSI (see 3.1.69) SA creation transaction (see 3.1.130)).

3.1.38 data counter log parameter: a bounded data counter log parameter (see 7.3.2.2.2.2) or unbounded data counter log parameter (see 7.3.2.2.2.3).

3.1.39 Data-In Buffer: The buffer specified by the application client to receive data from the device server during the processing of a command (see 4.2 and SAM-4).

3.1.40 Data-Out Buffer: The buffer specified by the application client to supply data that is sent from the application client to the device server during the processing of a command (see 4.2 and SAM-4).

3.1.41 deferred error: A CHECK CONDITION status and sense data that is returned as the result of an error or exception condition that occurred during processing of a previous command for which GOOD status or CONDITION MET status has already been returned. See 4.5.5.

3.1.42 designation: When used in reference to access controls, a name and optional identifier information that specifies a SCSI target device or SCSI target port for association with an alias value (see 3.1.9) in the alias list (see 3.1.8). See 6.2.2. Otherwise, a distinguishing name, identifier, or title.

3.1.43 Device Identification VPD page: A VPD page that provides the means to retrieve identification information about the SCSI device, logical unit, and SCSI port. See 7.8.6.

3.1.44 device server: An object within a logical unit that processes SCSI commands according to the rules of command management. A detailed definition of a device server may be found in SAM-4.

3.1.45 device service request: A request, submitted by an application client, conveying a SCSI command to a device server. A detailed definition of a device service request may be found in SAM-4.

3.1.46 device service response: The response returned to an application client by a device server on completion of a SCSI command. A detailed definition of a device service response may be found in SAM-4.

3.1.47 device type: The type of peripheral device (i.e., device model) implemented by the device server and indicated by the contents of the PERIPHERAL DEVICE TYPE field in the standard INQUIRY data (see 6.4.2).

3.1.48 Disconnect-Reconnect mode page: A mode page that provides the application client the means to tune the performance of a service delivery subsystem. See 7.5.9.

3.1.49 element: An addressable physical component of a medium changer SCSI device that may serve as the location of a removable unit of data storage medium. A detailed definition of an element may be found in SMC-2.

3.1.50 enabled command state: The only command state in which a command may make progress towards completion. A detailed definition of the enabled command state may be found in SAM-4.

3.1.51 Encapsulating Security Payload for SCSI (ESP-SCSI): A method for transferring encrypted and/or integrity checked parameter data in Data-In Buffers and/or Data-Out Buffers based on Encapsulating Security Payload (see RFC 4303). See 5.16.7.

3.1.52 Enforcement Manager class: The command security (see 5.16.6) class whose objects are, or an object that is consulted by the Secure CDB Processor class (see 3.1.151) or Secure CDB Processor class objects to determine if the processing of a command is permitted or prohibited. Equivalent to the Policy Enforcement Point in the policy model defined by RFC 2753 and similar policy-based authorization standards. See 5.16.6.4.

3.1.53 Error history I_T nexus: An I_T nexus for which the device server has reserved access to the error history snapshot (see 3.1.54).

3.1.54 Error history snapshot: The contents of the error history at a specific point in time. See 5.14.2.

3.1.55 Extended CDB (XCDB): A CDB that contains another CDB and additional information to support extra processing (e.g., security features). See 4.3.4.

3.1.56 Extended Unique Identifier, a 48-bit globally unique identifier (EUI-48): The IEEE maintains a tutorial describing EUI-48 at http://standards.ieee.org/regauth/oui/tutorials/EUI48.html.

3.1.57 Extended Unique Identifier, a 64-bit globally unique identifier (EUI-64): The IEEE maintains a tutorial describing EUI-64 at http://standards.ieee.org/regauth/oui/tutorials/EUI64.html.

3.1.58 faulted I_T nexus: The I_T nexus on which a CHECK CONDITION status was returned that resulted in the establishment of an ACA. The faulted I_T nexus condition is cleared when the ACA condition is cleared. See SAM-4.

3.1.59 field: A group of one or more contiguous bits, a part of a larger structure such as a CDB (see 3.1.30) or sense data (see 3.1.158).

3.1.60 hard reset: A condition resulting from the events defined by SAM-4 in which the SCSI device performs the hard reset operations described in SAM-4, this standard, and the applicable command standards.

3.1.61 hashed message authentication code (HMAC): A type of message authentication code that is calculated using the cryptographic hash function as defined in FIPS 198a (see 2.4) in combination with a secret key.

3.1.62 host: A SCSI device with the characteristics of a primary computing device, typically a personal computer, workstation, server, minicomputer, mainframe computer, or auxiliary computing device. A host includes one or more SCSI initiator devices.

3.1.63 IEEE company_id: Synonym for OUI (see 3.1.103).

3.1.64 I_T nexus: A nexus between a SCSI initiator port and a SCSI target port. See SAM-4.

3.1.65 I_T nexus loss: A condition resulting from the events defined by SAM-4 in which the SCSI device performs the I_T nexus loss operations described in SAM-4, this standard, and the applicable command standards.

3.1.66 I_T_L nexus: A nexus between a SCSI initiator port, a SCSI target port, and a logical unit. See SAM-4.

3.1.67 I_T_L_Q nexus transaction: The information transferred between SCSI ports in a single data structure with defined boundaries (e.g., an information unit).

3.1.68 idle power condition: The power condition in which a device server is capable of completing the processing of all its supported commands, except those that require the logical unit be in the active power condition (see 3.1.5) to be capable of completing the command with GOOD status. See 5.12.3.

3.1.69 IKEv2-SCSI: Internet Key Exchange protocol version 2 for SCSI. See 5.16.4.

3.1.70 IKEv2-SCSI keys: Shared keys (see 3.1.162) used to provide security for IKEv2-SCSI operations (e.g., creation and deletion of the SA). IKEv2-SCSI keys that are used after SA creation is complete are maintained in the MGMT_DATA SA parameter (see 3.1.133). The shared keys names used by IKEv2-SCSI are listed in 5.16.4.4.

3.1.71 IKEv2-SCSI CCS: The CCS (see 3.1.37) that is the Key Exchange step and Authentication step, if any, of an IKEv2-SCSI (see 3.1.69) SA creation transaction (see 3.1.130).

3.1.72 implicit head of queue: An optional processing model for specified commands wherein the specified commands may be treated as if they had been received with a HEAD OF QUEUE task attribute (see SAM-4 and 5.3).

3.1.73 information unit (IU): A delimited and sequenced set of information in a format appropriate for transport by the service delivery subsystem (e.g., a CDB for a specific SCSI command).

3.1.74 initiator port: Synonymous with SCSI initiator port (see 3.1.140).

3.1.75 initiator port identifier: A value by which a SCSI initiator port (see 3.1.140) is referenced within a SCSI domain (see SAM-4).

3.1.76 initiator port name: A SCSI port name (see 3.1.143) of a SCSI initiator port (see 3.1.140) or of a SCSI target/initiator port when operating as a SCSI initiator port (see SAM-4).

3.1.77 integrity check value: A value used to cryptographically validate the integrity of a specified set of bytes that contain specified data based on a shared key (see 3.1.162).

3.1.78 internet assigned numbers authority (IANA): A service that assigns and manages registries of code values (i.e., code points) for the IETF (see http://www.ietf.org) and other users of the public internet. See http://www.iana.org. IANA maintains code values used by Internet Key Exchange version 2 (IKEv2) at http://www.iana.org/assignments/ikev2-parameters.

3.1.79 Internet protocol domain name: The name of a computer or hierarchy of computers within the domain name system defined by the IETF (see RFC 1035 and RFC 1591). The Internet Assigned Numbers Authority maintains a list of domain name assignments at http://www.iana.org/assignments/domain-names.

3.1.80 Internet protocol number: A coded value assigned to identify protocols that layer on the Internet protocol (see RFC 791). The Internet protocol number assigned to the transmission control protocol (TCP, see RFC 793) is six. The Internet Assigned Numbers Authority maintains a list of Internet protocol number assignments at http://www.iana.org/assignments/protocol-numbers.

3.1.81 key derivation function (KDF): An algorithm that is used to derive cryptographic keying material from a shared secret and other information.

3.1.82 least significant bit (LSB): In a binary code, the bit or bit position with the smallest numerical weighting in a group of bits that, when taken as a whole, represent a numerical value (e.g., in the number 0001b, the bit that is set to one).

3.1.83 left-aligned: A type of field containing ASCII data in which unused bytes are placed at the end of the field (highest offset) and are filled with ASCII space (20h) characters. See 4.4.1.

3.1.84 logical unit: An externally addressable entity within a SCSI target device that implements a SCSI device model and contains a device server. A detailed definition of a logical unit may be found in SAM-4.

3.1.85 logical unit access control descriptor (LUACD): The structure within an ACE (see 3.1.2) that identifies a logical unit to which access is allowed and specifies the LUN by which the logical unit is to be accessed (see 8.3.1.3.3).

3.1.86 logical unit inventory: The list of the logical unit numbers reported by a REPORT LUNS command (see 6.23).

3.1.87 logical unit number (LUN): An encoded 64-bit identifier for a logical unit. A detailed definition of a logical unit number may be found in SAM-4.

3.1.88 logical unit reset: A condition resulting from the events defined by SAM-4 in which the logical unit performs the logical unit reset operations described in SAM-4, this standard, and the applicable command standards.

3.1.89 low power condition: Any power condition other than the active power condition (see 5.12.2).

3.1.90 medium: A physical entity that stores data in a nonvolatile manner (i.e., retained through a power cycle) in accordance with commands processed by the device server.

3.1.91 medium auxiliary memory (MAM): An auxiliary memory residing in a volume (e.g., a tape cartridge) that is accessible to the device server. See 7.4.

3.1.92 media changer: A device that mechanizes the movement of media to and from the SCSI device that records on or reads from the media. A detailed definition of a medium changer may be found in SMC-2.

3.1.93 most significant bit (MSB): In a binary code, the bit or bit position with the largest numerical weighting in a group of bits that, when taken as a whole, represent a numerical value (e.g., in the number 1000b, the bit that is set to one).

3.1.94 name: A label of an object that is unique within a specified context and should never change (e.g., the term name and worldwide identifier (WWID) may be interchangeable).

3.1.95 network address authority (NAA): A field within a name (see 3.1.94) that specifies the format and length of that name. See 7.8.6.6 and FC-FS.

3.1.96 nexus: A relationship between two SCSI devices, and the SCSI initiator port and SCSI target port objects within those SCSI devices. See SAM-4.

3.1.97 non-volatile cache memory: Cache memory (see 3.1.19) that retains data through power cycles.

3.1.98 nonce: A value that is used one and only one time to provide uniqueness to a value (e.g., a secure cryptographic key) in whose derivation it participates or to uniquely identify a single instance of something (e.g., a timestamp) exchanged between an application client and a device server.

3.1.99 null-padded: A type of field in which unused bytes are placed at the end of the field (i.e., highest offset) and are filled with ASCII null (00h) characters. See 4.4.2.

3.1.100 null-terminated: A type of field in which the last used byte (i.e., highest offset) is required to contain an ASCII null (00h) character. See 4.4.2.

3.1.101 Object Identifier (OID): An ASN.1 format (see 2.2) identifier for an object in or related to a certificate. ISO/IEC 9834 describes procedures related to OID registration (see 2.2).

3.1.102 one: The logical true condition of a variable.

3.1.103 organizationally unique identifier (OUI): A numeric identifier that is assigned by the IEEE such that no assigned identifiers are identical. OUI is equivalent to company_id or IEEE company_id. The IEEE prefers OUI for EUI-48 identifiers (see 3.1.56) and company_id for EUI-64 identifiers (see 3.1.57). However, the numeric identifier is called an OUI when it is assigned by the IEEE. The IEEE maintains a tutorial describing the OUI at http://standards.ieee.org/regauth/oui/.

3.1.104 page: A regular parameter structure (or format) used by several commands. These pages are identified with a value known as a page code.

3.1.105 peripheral device: An object that connects to a logical unit.

3.1.106 peripheral device identifying information: A value associated with a peripheral device accessed via the REPORT IDENTIFYING INFORMATION command (see 6.22) and SET IDENTIFYING INFORMATION command (see 6.33). See 5.17.

3.1.107 peripheral device text identifying information: A UTF-8 (see 3.1.189) format string associated with a peripheral device accessed via the REPORT IDENTIFYING INFORMATION command (see 6.22) and SET IDENTIFYING INFORMATION command (see 6.33). See 5.17.

3.1.108 peripheral device type: Synonym for device type (see 3.1.47).

3.1.109 persist through power loss: An optional capability associated with some features that allows an application client to request that a device server maintain information regarding that feature across power failures.

3.1.110 persistent reservation holder: The I_T nexus(es) that are allowed to release or change a persistent reservation without preempting it. See 5.9.10.

3.1.111 power cycle: Power being removed from and later applied to a SCSI device.

3.1.112 power on: A condition resulting from the events defined by SAM-4 in which the SCSI device performs the power on operations described in SAM-4, this standard, and the applicable command standards.

3.1.113 pre-shared key: A shared key (see 3.1.162) that is established by a method outside the scope of this standard prior to the initiation of the function that requires its use. Requirement on pre-shared keys that are particular to this standard are described in 5.16.4.3.2. Some of the RFCs referenced by this standard use the name pre-shared secret for the same information that this standard calls a pre-shared key.

3.1.114 primary target port asymmetric access state: The characteristic that defines the behavior of a target port and the allowable command set for a logical unit when commands and task management functions are routed through the target port maintaining that state. See 5.11.2.1.

3.1.115 primary target port group: A set of target ports that are in the same primary target port asymmetric access state (see 3.1.114) at all times. See 5.11.2.1.

3.1.116 primary target port group asymmetric access state: The primary target port asymmetric access state (see 3.1.114) common to the set of target ports in a primary target port group (see 3.1.115). See 5.11.2.1.

3.1.117 protection information: Fields appended to each logical block that contain a CRC, an application tag, and a reference tag. See SBC-3.

3.1.118 protocol identifier: A coded value used in various fields to identify the protocol to which other fields apply. See 7.6.1.

3.1.119 protocol specific: A requirement that is defined by a SCSI transport protocol standard (see clause 1). A detailed definition of protocol specific may be found in SAM-4.

3.1.120 protocol standard: A SCSI standard that defines SCSI transport protocol (e.g., SPL, SPI-5, SBP-3, or FCP-4). See clause 1.

3.1.121 proxy token: An identifier for a logical unit that may be used to gain temporary access to that logical unit in the presence of access controls (see 8.3.1.6.2).

3.1.122 random nonce: A secure random number (see 4.6) that has a negligible chance of repeating and whose uses include providing significant uniqueness and randomness in cryptographic calculations.

3.1.123 request for comment (RFC): The name given to standards developed by the Internet Engineering Task Force (see 2.5).

3.1.124 registered: The condition that exists for an I_T nexus following the successful completion of a PERSISTENT RESERVE OUT command with a REGISTER service action, REGISTER AND IGNORE EXISTING KEY service action (see 5.9.7), or REGISTER AND MOVE service action (see 5.9.8) and lasting until the registration is removed (see 5.9.11).

3.1.125 registrant: An I_T nexus that is registered (see 3.1.124).

3.1.126 right-aligned: A type of field containing ASCII data in which unused bytes are placed at the start of the field (i.e., lowest offset) and are filled with ASCII space (20h) characters. See 4.4.1.

3.1.127 relative port identifier: An identifier for a SCSI port (see 3.1.141) that is unique within a SCSI device (see 3.1.136). See SAM-4. Application clients may use the SCSI Ports VPD page (see 7.8.11) to determine relative port identifier values.

3.1.128 relative initiator port identifier: A relative port identifier (see 3.1.127) for a SCSI initiator port (see 3.1.140).

3.1.129 relative target port identifier: A relative port identifier (see 3.1.127) for a SCSI target port (see 3.1.146).

3.1.130 SA creation transaction: Any sequence of SECURITY PROTOCOL IN commands (see 6.30) and SECURITY PROTOCOL OUT commands (see 6.31), including but not limited a CCS (see 3.1.37), that is used to create an SA between an application client and device server.

3.1.131 SA generation: Computation and initialization of the SA parameter values (see 3.1.133) required to create an SA (see 3.1.155) that is performed separately by the application client and device server after all SCSI commands required to create an SA have been performed without error. See 5.16.4.9.

3.1.132 SA keys: Shared keys (see 3.1.162) that are maintained in the USAGE_DATA SA parameter (see 3.1.133) and used to provide security for the operations that use the SA (e.g., encryption of command parameter data). The shared keys names used by IKEv2-SCSI are listed in 5.16.4.4.

3.1.133 SA parameters: The parameters stored by both an application client and a device server that are associated with one SA (see 3.1.155) and identified by a pair of SAIs (see 3.1.156). See 5.16.2.2.

3.1.134 SA participant: An application client or device server that participates in the creation or use of an SA (see 3.1.155).

3.1.135 saturating counter: A counter that remains at its maximum value after reaching its maximum value.

3.1.136 SCSI device: A device that contains one or more SCSI ports that are each connected to a service delivery subsystem and supports a SCSI application protocol (see SAM-4).

3.1.137 SCSI device name: A name (see 3.1.94) of a SCSI device that is world wide unique within the protocol of a SCSI domain (see 3.1.138) in which the SCSI device has SCSI ports (see 3.1.141). The SCSI device name may be made available to other SCSI devices or SCSI ports in protocol specific ways. See SAM-4.

3.1.138 SCSI domain: The interconnection of two or more SCSI devices and a service delivery subsystem. A detailed definition of a SCSI Domain may be found in SAM-4.

3.1.139 SCSI initiator device: A SCSI device (see 3.1.136) containing application clients (see 3.1.10) and SCSI initiator ports (see 3.1.140) that originate device service and task management requests (see SAM-4) to be processed by a SCSI target device (see 3.1.145) and receives device service and task management responses from SCSI target devices.

3.1.140 SCSI initiator port: A SCSI initiator device (see SAM-4) object that acts as the connection between application clients and a service delivery subsystem through which requests and responses are routed.

3.1.141 SCSI port: A port of a SCSI device that connects the application client, device server or task manager to a service delivery subsystem (see SAM-4).

3.1.142 SCSI port identifier: A value by which a SCSI port is referenced within a domain. The SCSI port identifier is either an initiator port identifier (see 3.1.75) or a target port identifier (see 3.1.177).

3.1.143 SCSI port name: A name (see 3.1.94) of a SCSI port that is world wide unique within the protocol of the SCSI domain of that SCSI port (see 3.1.141). The name may be made available to other SCSI devices or SCSI ports in that SCSI domain in protocol specific ways. See SAM-4.

3.1.144 SCSI Ports VPD page: A VPD page that allows retrieval of information about all the SCSI ports in a SCSI target device or SCSI target/initiator device. See 7.8.11.

3.1.145 SCSI target device: A SCSI device (see 3.1.136) containing logical units (see 3.1.84) and SCSI target ports (see 3.1.146) that receives device service and task management requests (see SAM-4) for processing and sends device service and task management responses to SCSI initiator devices.

3.1.146 SCSI target port: A SCSI target device (see SAM-4) object that acts as the connection between device servers and task managers and a service delivery subsystem through which requests and responses are routed.

3.1.147 SCSI transport protocol standard: A SCSI standard that defines a SCSI transport protocol (e.g., FCP-4, SPL, SRP, or SBP-3). See clause 1.

3.1.148 secondary target port asymmetric access state: A characteristic indicating a condition that affects the way in which an individual target port participates in its assigned primary target port group (see 3.1.115). See 5.11.2.1.

3.1.149 secondary target port group: One or more target ports that are in the same secondary target port asymmetric access state (see 3.1.148). See 5.11.2.1.

3.1.150 Secure CDB Originator class: The command security (see 5.16.6) class whose objects are, or an object that is an application client that originates secure CDBs and performs any additional functions necessary to do so. See 5.16.6.2.

3.1.151 Secure CDB Processor class: The command security (see 5.16.6) class whose objects are, or an object that is a device server that processes secure CDBs and performs any additional functions necessary to do so. See 5.16.6.3.

3.1.152 Secure Digital Card Association (SD Card): An organization that establishes and maintains standards for secure digital memory card applications. See http://www.sdcard.org/

3.1.153 secure hash algorithm (SHA): A secure hash algorithm (e.g., SHA-1) specified in FIPS 180-2 with Change Notice 1 dated February 25, 2004 (see 2.4).

3.1.154 secure random number: A random number that is generated in ways that protect it from security attacks. See 4.6.

3.1.155 security association (SA): A relationship and associated security processing between an application client and device server that is used to apply security functions (e.g., data integrity checking, data encryption) to data that is transferred in either direction. See 5.16.2.

3.1.156 security association index (SAI): A number representing the parameters for a security association as stored internally by the application client or device server. In other security models, this value is called the security parameters index (SPI). See 5.16.2.

3.1.157 Security Manager class: The command security (see 5.16.6) class whose objects maintain, or an object that maintains information about which secure CDBs may be originated by which secure CDB originator (see 3.1.150) application clients to which secure CDB processor (see 3.1.151) device servers, responds to requests to allow the origination of secure CDBs from secure CDB originator application clients, and updates secure CDB permissions and prohibitions in appropriate enforcement manager (see 3.1.52) members. Equivalent to the Policy Decision Point in the policy model defined by RFC 2753 and similar policy-based authorization standards. See 5.16.6.5.

3.1.158 sense data: Data describing an error or exceptional condition that a device server delivers to an application client in the same I_T_L_Q nexus transaction (see 3.1.67) as the status or as parameter data in response to a REQUEST SENSE command (see 6.29). The format of sense data is defined in 4.5.

3.1.159 sense key: The contents of the SENSE KEY field in sense data (see 4.5).

3.1.160 service action: A request describing a unit of work to be performed by a device server. A service action is an extension of a command. See SAM-4.

3.1.161 service delivery subsystem: That part of a SCSI domain 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. See SAM-4.

3.1.162 shared key (SK): A cryptographically protected secret (e.g., with more randomness (see RFC 4089) than a password) that is known only to a defined and limited set of entities (e.g., one application client and one device server). The shared keys names used by IKEv2-SCSI are listed in 5.16.4.4.

3.1.163 SK_ai: Shared key (see 3.1.162) used for integrity checking data in a Data-Out Buffer (e.g., integrity checking the Encrypted payload in an IKEv2-SCSI Authentication step SECURITY PROTOCOL OUT command (see 5.16.4.7.2)).

3.1.164 SK_ar: Shared key (see 3.1.162) used for integrity checking data in a Data-In Buffer (e.g., integrity checking the Encrypted payload in an IKEv2-SCSI Authentication step SECURITY PROTOCOL IN command (see 5.16.4.7.3)).

3.1.165 SK_d: Shared key (see 3.1.162) KDF input material use to generate the shared keys stored in the KEYMAT SA Parameter (see 3.1.133).

3.1.166 SK_ei: Shared key (see 3.1.162) used for encrypting data in a Data-Out Buffer (e.g., encrypting the Encrypted payload in an IKEv2-SCSI Authentication step SECURITY PROTOCOL OUT command (see 5.16.4.7.2)).

3.1.167 SK_er: Shared key (see 3.1.162) used for encrypting data in a Data-In Buffer (e.g., encrypting the Encrypted payload in an IKEv2-SCSI Authentication step SECURITY PROTOCOL IN command (see 5.16.4.7.3)).

3.1.168 SK_pi: Shared key (see 3.1.162) used to construct the IKEv2-SCSI Data-Out Buffer Authentication payload (see 5.16.4.7.2).

3.1.169 SK_pr: Shared key (see 3.1.162) used to construct the IKEv2-SCSI Data-In Buffer Authentication payload (see 5.16.4.7.3).

3.1.170 standby power condition: A power condition in which a device server is capable of completing the processing of all its supported commands, except those that require the logical unit be in the idle power condition (see 3.1.68) or active power condition (see 3.1.5) to be capable of completing the command with GOOD status. See 5.12.4.

3.1.171 status: One byte of response information sent from a device server to an application client upon completion of each command. See SAM-4.

3.1.172 storage networking industry association (SNIA): An organization that develops and promotes standards for storage management and other storage-related functions, and which publishes standards via the INCITS Fast Track process. See http://www.snia.org/.

3.1.173 system: One or more SCSI domains operating as a single configuration.

3.1.174 target port: Synonymous with SCSI target port (see 3.1.146).

3.1.175 target port asymmetric access state: A primary target port asymmetric access state (see 3.1.114) or secondary target port asymmetric access state (see 3.1.148).

3.1.176 target port group: A primary target port group (see 3.1.115) or secondary target port group (see 3.1.149).

3.1.177 target port identifier: A value by which a SCSI target port (see 3.1.146) is referenced within a SCSI domain (see SAM-4).

3.1.178 target port name: A SCSI port name (see 3.1.143) of a SCSI target port or of a SCSI target/initiator port when operating as a SCSI target port (see SAM-4).

3.1.179 task set: A group of commands within a logical unit, whose interaction is dependent on the task management (queuing) and ACA rules. See SAM-4 and the Control mode page (see 7.5.7).

3.1.180 TCP port numbers: One of the data needed to establish a TCP connection. TCP port numbers may be assigned to protocols that layer on TCP by the Internet Assigned Numbers Authority. The Internet Assigned

Numbers Authority maintains a list of TCP port number assignments at http://www.iana.org/assignments/port-numbers.

3.1.181 third-party command: A command sent to one SCSI device requesting than an operation be performed involving two other SCSI devices (e.g., the EXTENDED COPY command may perform copy operations between two or more SCSI devices none of which are the SCSI device to which the EXTENDED COPY command was sent).

3.1.182 Trusted Computing Group (TCG): An organization that develops and promotes open standards for hardware-enabled trusted computing and security technologies. See https://www.trustedcomputinggroup.org.

3.1.183 unbounded data counter log parameter: a log parameter that contains one or more saturating counter values (see 3.1.135) or wrapping counter values (see 3.1.194) that do not affect other log parameters in the same log page and for which the establishment of thresholds is not allowed. See 7.3.2.2.2.3.

3.1.184 unit attention condition: Asynchronous status information that a logical unit establishes to report to the initiator ports associated with one or more I_T nexuses. See SAM-4.

3.1.185 universal time (UT): The time at longitude zero, colloquially known as Greenwich Mean Time. See http://www.usno.navy.mil/USNO/astronomical-applications/astronomical-information-center/universal-time.

3.1.186 URI Schemes: The Internet Assigned Numbers Authority maintains a list of schemes for URI and URL names at http://www.iana.org/assignments/uri-schemes.

3.1.187 user data: Data contained in logical blocks (see SBC-3) that is not protection information (see 3.1.117). Each command standard (see 3.1.31) may provide its own definition of user data.

3.1.188 user data segment: A contiguous sequence of logical blocks. See SBC-3.

3.1.189 UTF-8: A character set that is a transformation format of the character set defined by ISO 10646. See RFC 2279.

3.1.190 vendor specific (VS): Something (e.g., a bit, field, or code value) that is not defined by this standard and may be vendor defined.

3.1.191 volatile cache memory: Cache memory (see 3.1.19) that does not retain data through power cycles.

3.1.192 well known logical unit: A logical unit that only does specific functions (see clause 8). Well known logical units allow an application client to issue requests to receive and manage specific information usually relating to a SCSI target device.

3.1.193 well known logical unit number (W-LUN): The logical unit number that identifies a well known logical unit.

3.1.194 wrapping counter: A counter that wraps back to zero after reaching its maximum value.

3.1.195 zero: The logical false condition of a variable.

3.1.196 zero-padded: A type of field in which unused bytes are placed at the end of the field (i.e., highest offset) and are filled with zeros. See 4.4.2.

3.2 Symbols and acronyms

÷	divided by
×	multiplied by
<	less than
≤	less than or equal to
>	greater than
≥	greater than or equal to
≠	not equal to
||	concatenation
x, xx	any valid value for a bit or field
AAD	Additional Authenticated Data (see 3.1.6)
ACE	Access Control list Entry (see 3.1.2)
ACL	Access Control List (see 3.1.1)
ACA	Auto Contingent Allegiance (see 3.1.13)
ADC	Automation/Drive Interface - Commands
ADC-2	Automation/Drive Interface - Commands -2
ADC-3	Automation/Drive Interface - Commands -3
ADT	Automation/Drive Interface - Transport Protocol
ADT-2	Automation/Drive Interface - Transport Protocol -2
AES	Advanced Encryption Standards
AES-GCM	Advanced Encryption Standard - Galois Counter Mode (see RFC 4106)
ASC	Additional Sense Code (see 4.5)
ASCII	American Standard Code for Information Interchange (see 2.2)
ASCQ	Additional Sense Code Qualifier (see 4.5)
ASN.1	Abstract Syntax Notation One (see 2.2)
ATA	AT Attachment (see www.t13.org)
ATAPI	AT Attachment with Packet Interface (see www.t13.org)
CbCS	Capability-based Command Security (see 5.16.6.8)
CCS	Cryptographic Command Sequence (see 3.1.37)
CDB	Command Descriptor Block (see 3.1.30)
CFA	CompactFlash Association (see 2.7)
CRC	Cyclic Redundancy Check
D_ID	Destination Identifier (defined in FC-FS, see clause 2)
ECP	Elliptic Curve group modulo Prime (see RFC 4753)
ECDSA	Elliptic Curve Digital Signature Algorithm (see RFC 4754)
ESP-SCSI	Encapsulating Security Payload for SCSI (see 3.1.51)
EUI-48	Extended Unique Identifier, a 48-bit globally unique identifier (see 3.1.56)
EUI-64	Extended Unique Identifier, a 64-bit globally unique identifier (see 3.1.57)
FC-FS	Fibre Channel Framing and Signaling Interface
FCP-3	Fibre Channel Protocol for SCSI -3
FCP-4	Fibre Channel Protocol for SCSI -4
HMAC	Hashed Message Authentication Code (see 3.1.61)
HTTP	Hypertext Transfer Protocol (see RFC 2616)
IANA	Internet Assigned Numbers Authority (see 3.1.78)
ID	Identifier or Identification
IEC	International Electrotechnical Commission
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force (see 2.5)
IKEv2	Internet Key Exchange version 2 (see 2.5)
IP	Internet Protocol (see 2.5)
IPv4	Internet Protocol version 4
IPv6	Internet Protocol version 6

iSCSI	Internet SCSI (see 2.5)
ISO	International Organization for Standardization
IU	Information Unit
JEDEC	Joint Electronic Device Engineering Council
KDF	Key Derivation Function (see 3.1.81)
LBA	Logical Block Address
LSB	Least Significant Bit (see 3.1.82)
LUACD	Logical Unit Access Control Descriptor (see 3.1.85)
LUN	Logical Unit Number (see 3.1.87)
MAM	Medium Auxiliary Memory (see 3.1.91)
MMC-4	SCSI Multi-Media Commands -4
MMC-5	SCSI Multi-Media Commands -5
MMC-6	SCSI Multi-Media Commands -6
MODP	Modular exponential (see RFC 3526)
MSB	Most Significant Bit (see 3.1.93)
NAA	Network Address Authority (see 3.1.95)
n/a	not applicable
INCITS	InterNational Committee for Information Technology Standards
OCRW	SCSI Specification for Optical Card Reader/Writer
OID	Object Identifier (see 3.1.101)
OSD	Object-based Storage Devices Commands
OUI	Organizationally Unique Identifier (see 3.1.103)
PKI	Public Key Infrastructure (see RFC 5280)
PRF	Pseudo-Random Function (see RFC 4306)
RAID	Redundant Array of Independent Disks
RBC	SCSI Reduced Block Commands
RDMA	Remote Direct Memory Access (see SRP)
RFC	Request For Comments (see 3.1.123)
SA	Security Association (see 3.1.155)
SAI	Security Association Index (see 3.1.156)
SAM-2	SCSI Architecture Model -2
SAM-4	SCSI Architecture Model -4
SAM-5	SCSI Architecture Model -5
SAT	SCSI / ATA Translation
SAT-2	SCSI / ATA Translation -2
SAT-3	SCSI / ATA Translation -3
SAUT	SA Usage Type (see 7.7.3.5.14)
SBC-2	SCSI Block Commands -2
SBC-3	SCSI Block Commands -3
SBP-3	Serial Bus Protocol -3
SCC-2	SCSI Controller Commands -2
SCSI	The architecture defined by the family of standards
SES-2	SCSI Enclosure Services -2
SES-3	SCSI Enclosure Services -3
SHA-1	Secure Hash Algorithm, 160 bits (see 3.1.153)
SHA-256	Secure Hash Algorithm, 256 bits (see 3.1.153)
SHA-384	Secure Hash Algorithm, 384 bits (see 3.1.153)
SHA-512	Secure Hash Algorithm, 512 bits (see 3.1.153)
SK	Shared Key (see 3.1.162)
SMC-2	SCSI Media Changer Commands -2
SMC-3	SCSI Media Changer Commands -3
SNIA	Storage Networking Industry Association (see 3.1.172)
SPC-2	SCSI Primary Commands -2
SPC-3	SCSI Primary Commands -3

SPC-4	SCSI Primary Commands -4 (this standard)
SPI-5	SCSI Parallel Interface -5
SPL	SAS Protocol Layer
SRP	SCSI RDMA Protocol
SSC-2	SCSI Stream Commands -2
SSC-3	SCSI Stream Commands -3
SSC-4	SCSI Stream Commands -4
TCG	Trusted Computing Group (see 3.1.182)
TCP	Transmission Control Protocol (see RFC 793)
UFS	Universal Flash Storage (see 2.8)
UML	Unified Modeling Language (see 2.6)
URI	Uniform Resource Identifier (see RFC 2396, RFC 3305, and 3.1.186)
URL	Uniform Resource Locator (see RFC 2396, RFC 3305, and 3.1.186)
UT	Universal time (see 3.1.185)
USB	Universal Serial Bus (see www.usb.org)
VPD	Vital Product Data (see 7.8)
VPG1	CF Performance Guarantee Profile 1 (see 2.7)
VS	Vendor Specific (see 3.1.190)
W-LUN	Well known logical unit number (see 3.1.193)
XCDB	Extended CDB (see 3.1.55)

3.3 Keywords

3.3.1 invalid: A keyword used to describe an illegal or unsupported bit, byte, word, field or code value. Receipt of an invalid bit, byte, word, field or code value shall be reported as an error.

3.3.2 mandatory: A keyword indicating an item that is required to be implemented as defined in this standard.

3.3.3 may: A keyword that indicates flexibility of choice with no implied preference (equivalent to "may or may not").

3.3.4 may not: A keyword that indicates flexibility of choice with no implied preference (equivalent to "may or may not").

3.3.5 obsolete: A keyword indicating that an item was defined in prior SCSI standards but has been removed from this standard.

3.3.6 optional: A 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.

3.3.7 prohibited: A keyword used to describe a feature, function, or coded value that is defined in a non-SCSI standard (i.e., a standard that is not a member of the SCSI family of standards) to which this standard makes a normative reference where the use of said feature, function, or coded value is not allowed for implementations of this standard.

3.3.8 **reserved:** A keyword referring to bits, bytes, words, fields and code values that are set aside for future standardization. A reserved bit, byte, word or field shall be set to zero, or in accordance with a future extension to this standard. Recipients are not required to check reserved bits, bytes, words or fields for zero values. Receipt of reserved code values in defined fields shall be reported as an error.

3.3.9 restricted: A keyword referring to bits, bytes, words, and fields that are set aside for other identified standardization purposes. A restricted bit, byte, word, or field shall be treated as a reserved bit, byte, word or field in the context where the restricted designation appears.

A keyword referring to bits, bytes, words, and fields that are set aside for use in other SCSI standards. A restricted bit, byte, word, or field shall be treated as a reserved bit, byte, word or field for the purposes of the requirements defined in this standard.

3.3.10 shall: A 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.

3.3.11 should: A keyword indicating flexibility of choice with a strongly preferred alternative; equivalent to the phrase "it is strongly recommended".

3.3.12 x or **xx:** The value of the bit or field is not relevant.

3.4 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 3.1 or in the text where they first appear. Names of commands, statuses, sense keys, and additional sense codes are in all uppercase (e.g., REQUEST SENSE). Lowercase is used for words having the normal English meaning.

If there is more than one CDB length for a particular command (e.g., MODE SENSE(6) and MODE SENSE(10)) and the name of the command is used in a sentence without any CDB length descriptor (e.g., MODE SENSE), then the condition specified in the sentence applies to all CDB lengths for that command.

The names of fields are in small uppercase (e.g., ALLOCATION LENGTH). When a field name is a concatenation of acronyms, uppercase letters may be used for readability (e.g., NORMACA). Normal case is used when 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.

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

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).

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.

3.5 Numeric and character conventions

3.5.1 Numeric conventions

A binary number is represented in this standard by any sequence of digits comprised of only the Arabic numerals 0 and 1 immediately followed by a lower-case b (e.g., 0101b). Underscores or spaces may be included in binary number representations to increase readability or delineate field boundaries (e.g., 0 0101 1010b or 0_0101_1010b).

A hexadecimal number is represented in this standard by any sequence of digits comprised of only the Arabic numerals 0 through 9 and/or the upper-case English letters A through F immediately followed by a lower-case h (e.g., FA23h). Underscores or spaces may be included in hexadecimal number representations to increase readability or delineate field boundaries (e.g., B FD8C FA23h or B_FD8C_FA23h).

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

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

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;
c) the thousands separator is used in both the integer portion and the fraction portion of a number; and
d) the decimal representation for a year is 1999 not 1 999.

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

A decimal number represented in this standard with an overline over one or more digits following the decimal point is a number where the overlined digits are infinitely repeating (e.g., $666.\overline{6}$ means 666.666 666… or 666 2/3 and $12.\overline{142\ 857}$ means 12.142 857 142 857… or 12 1/7).

3.5.2 Units of measure

This standard represents values using both decimal units of measure and binary units of measure. Values are represented by the following formats:

a) for values based on decimal units of measure:
   1) numerical value (e.g., 100);
   2) space;
   3) prefix symbol and unit:
      1) decimal prefix symbol (e.g., M) (see table 2); and
      2) unit abbreviation (e.g., B);

   and
b) for values based on binary units of measure:
   1) numerical value (e.g., 1 024);
   2) space;
   3) prefix symbol and unit:
      1) binary prefix symbol (e.g., Gi) (see table 2); and
      2) unit abbreviation (e.g., b).

Table 2 compares the prefix, symbols, and power of the binary and decimal units.

Table 2 — Comparison of decimal prefixes and binary prefixes

Decimal			Binary		
Prefix name	**Prefix symbol**	**Power (base-10)**	**Prefix name**	**Prefix symbol**	**Power (base-2)**
kilo	k	10^3	kibi	Ki	2^{10}
mega	M	10^6	mebi	Mi	2^{20}
giga	G	10^9	gibi	Gi	2^{30}
tera	T	10^{12}	tebi	Ti	2^{40}
peta	P	10^{15}	pebi	Pi	2^{50}
exa	E	10^{18}	exbi	Ei	2^{60}
zetta	Z	10^{21}	zebi	Zi	2^{70}
yotta	Y	10^{14}	yobi	Yi	2^{80}

3.5.3 Byte encoded character strings conventions

When this standard requires one or more bytes to contain specific encoded characters, the specific characters are enclosed in single quotation marks. The single quotation marks identify the start and end of the characters that are required to be encoded but are not themselves to be encoded. The characters that are to be encoded are shown in the case that is to be encoded.

An ASCII space character (i.e., 20h) may be represented in a string by the character '¬' (e.g., 'SCSI¬device').

The encoded characters and the single quotation marks that enclose them are preceded by text that specifies the character encoding methodology and the number of characters required to be encoded.

EXAMPLE - Using the notation described in this subclause, stating that eleven ASCII characters 'SCSI device' are to be encoded would be the same writing out the following sequence of byte values: 53h 43h 53h 49h 20h 64h 65h 76h 69h 63h 65h.

3.6 Bit and byte ordering

This subclause describes the representation of fields in a table that defines the format of a SCSI structure (e.g., the format of a CDB).

If a field consists of more than one bit and contains a single value (e.g., a number), the least significant bit (LSB) is shown on the right and the most significant bit (MSB) is shown on the left (e.g., in a byte, bit 7 is the MSB and is shown on the left, and bit 0 is the LSB and is shown on the right). The MSB and LSB are not labeled if the field consists of 8 or fewer bits.

If a field consists of more than one byte and contains a single value, the byte containing the MSB is stored at the lowest address and the byte containing the LSB is stored at the highest address (i.e., big-endian byte ordering). The MSB and LSB are labeled.

If a field consists of more than one byte and always contains multiple fields each with their own values (e.g., a descriptor), there is no MSB and LSB of the field itself and thus there are no MSB and LSB labels. Each individual field has an MSB and LSB that are labeled as appropriate in the table, if any, that describes the format of the sub-structure having multiple fields.

If a field that consists of more than one byte contains either multiple fields each with their own values or a single value, then:

a) the MSB and LSB are labeled and they apply as described in this subclause whenever the field contains a single value; and
b) the labels on constituent fields supersede the single value labels whenever the field contains multiple fields.

If a field contains a text string (e.g., ASCII or UTF-8), the MSB label is the MSB of the first character and the LSB label is the LSB of the last character.

When required for clarity, multiple byte fields may be represented with only two rows in a table. This condition is represented by values in the byte number column not increasing by one in each subsequent table row, thus indicating the presence of additional bytes.

3.7 Notation conventions

3.7.1 Notation for procedure calls

In this standard, the model for functional interfaces between objects is a procedure call. Such interfaces are specified using the following notation:

[Result =] Procedure Name (IN ([input-1] [,input-2] …]), OUT ([output-1] [,output-2] …))

Where:

Result: A single value representing the outcome of the procedure call.

Procedure Name: A descriptive name for the function modeled by the procedure call. When the procedure call model is used to describe a SCSI transport protocol service, the procedure name is the same as the service name.

Input-1, Input-2, …: A comma-separated list of names identifying caller-supplied input arguments.

Output-1, Output-2, …: A comma-separated list of names identifying output arguments to be returned by the procedure call.

"[…]": Brackets enclosing optional or conditional arguments.

This notation allows arguments to be specified as inputs and outputs. The following is an example of a procedure call specification:

Found = Search (IN (Pattern, Item List), OUT ([Item Found]))

Where:

Found = Flag
Flag: if set to one, indicates that a matching item was located.

Input Arguments:

Pattern = … /* Definition of Pattern argument */
Argument containing the search pattern.

Item List = Item<NN> /* Definition of Item List as an array of NN Item arguments*/
Contains the items to be searched for a match.

Output Arguments:

Item Found = Item … /* Item located by the search procedure call */
This argument is only returned if the search succeeds.

3.7.2 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 subclauses describing the states and state transitions.

Each state and state 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.

A system specified in this manner has the following properties:

a) time elapses only within discrete states; and
b) state transitions are logically instantaneous.

3.7.3 Notation for UML figures

3.7.3.1 Introduction

This standard contains class diagram figures that use notation that is based on UML (see 2.6).

Some class diagrams contain constraints or notes that use the notion shown in table 3.

Table 3 — Class diagram constraints and notes notation

Notation	Description
{constraint text}	The presence of the curly brackets defines constraint that is a normative requirement. An example of a constraint is shown in figure 2.
note text	The absence of curly brackets defines a note that is informative. An example of a note is shown in figure 4.

The notation used to denote multiplicity in class diagrams is shown in table 4.

Table 4 — Class diagram multiplicity notation

Notation	Description
not specified	The number of instances of an attribute is not specified.
1	One instance of the class or attribute exists.
0..*	Zero or more instances of the class or attribute exist.
1..*	One or more instances of the class or attribute exist.
0..1	Zero or one instance of the class or attribute exists.
n..m	n to m instances of the class or attribute exist (e.g., 2..8).
x,n..m	Multiple disjoint instances of the class or attribute exist (e.g., 2, 8..15).

Class diagrams show:

a) two or more classes (see 3.7.3.2); and
b) one or more of the following relationships between them:
   A) association (see 3.7.3.3);
   B) aggregation (see 3.7.3.4);
   C) generalization (see 3.7.3.5); and
   D) dependency (see 3.7.3.6).

3.7.3.2 Class notation

The notation used for classes is shown in table 5.

Table 5 — Class diagram notation for classes (part 1 of 2)

Notation	Description
Class Name Class Name Class Name	A class with no attributes or operations
Class Name Attribute01[1] Attribute02[1] Class Name Attribute01[1] Attribute02[1]	A class with attributes and no operations
Class Name Operation01() Operation02()	A class with operations and no attributes

Table 5 — Class diagram notation for classes (part 2 of 2)

Notation	Description
Class Name Attribute01[1] Attribute02[1] Operation01() Operation02()	A class with attributes and operations
Class Name Attribute01[1..*] Attribute02[1] Operation01() Operation02()	A class with attributes that have a specified multiplicity (see table 4 in 3.7.3.1) and operations

3.7.3.3 Class association relationships notation

The notation used to denote association (i.e., "knows about") relationships between classes is shown in table 6. Unless the two classes in an association relationship also have an aggregation relationship (see 3.7.3.4), association relationships have multiplicity notation (see table 4 in 3.7.3.1) at each end of the relationship line.

Table 6 — Class diagram notation for associations

Notation	Description
Class A — association name — Class B 1..* 0..1 multiplicity notation (see table 4)	Class A knows about Class B (i.e., read as "Class A association name Class B") and Class B knows about Class A (i.e., read as "Class B association name Class A")
Class A — Class B 1 0..1	Class B knows about Class A (i.e., read as "Class B knows about Class A") but Class A does not know about Class B
Class A — role name — Class B 0..* 0..1	Class A knows about Class B (i.e., read as "Class A uses the role name attribute of Class B") but Class B does not know about Class A
Note: The use of role names and association names are optional.	

Several example association relationships between classes are shown in figure 2.

Figure 2 — Example class association relationships

3.7.3.4 Class aggregation relationships notation

The aggregation relationship is a specific type of association. The notation used to denote aggregation (i.e., "is a part of" or "contains") relationships between classes is shown in table 7. Aggregation relationships always include multiplicity notation (see table 4 in 3.7.3.1) at each end of the relationship line.

Table 7 — Class diagram notation for aggregations

Notation	Description
	The Part class is part of the Whole class and may continue to exist even if the Whole class is removed (i.e., read as "the Whole contains the Part.")
	The Part class is part of the Whole class, shall only belong to one Whole class, and shall not continue to exist if the Whole class is removed (i.e., read as "the Whole contains the Part.")

Several example aggregation relationships between classes are shown in figure 3.

Figure 3 — Example class aggregation relationships

3.7.3.5 Class generalization relationships notation

The notation used to denote generalization (i.e., "is a kind of") relationships between classes is shown in table 8.

Table 8 — Class diagram notation for generalizations

Notation	Description
	Subclass is a kind of superclass. A subclass shares all the attributes and operations of the superclass (i.e., the subclass inherits from the superclass).

Several example generalization relationships between classes are shown in figure 4.

Figure 4 — Example class generalization relationships

3.7.3.6 Class dependency relationships notation

The notation used to denote dependency (i.e., "depends on") relationships between classes is shown in table 9.

Table 9 — Class diagram notation for dependencies

Notation	Description
Class A ⇢ Class B	Class A depends on class B. A change in class B may cause a change in class A.

An example dependency relationship between classes is shown in figure 5.

Figure 5 — Example class dependency relationships

3.7.3.7 Object notation

The notation used for objects is shown in table 10.

Table 10 — Notation for objects

Notation	Description
label : Class Name	Notation for a named object with no attributes
label : Class Name Attribute01 = x Attribute02 = y	Notation for a named object with attributes
: Class Name	Notation for a anonymous object with no attributes
: Class Name Attribute01 = x Attribute02 = y	Notation for a anonymous object with attributes

4 General Concepts

4.1 Introduction

This standard defines behaviors that are common to all SCSI device models (see clause 5). This standard defines the SCSI commands that are basic to more than one device model and the SCSI commands that may apply to any device model (see clause 6). This standard defines the parameters that are basic to more than one device model (see clause 7).

4.2 The request-response model

The SCSI command set assumes an underlying request-response protocol. The fundamental properties of the request-response protocol are defined in SAM-4. Action on SCSI commands shall not be deemed completed until a response is received. The response shall include a status that indicates the final disposition of the command. As per SAM-4, the request-response protocol may be modeled as a procedure call, specifically:

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

SAM-4 defines all of the inputs and outputs in the procedure call above. As they may apply to any SCSI device, this standard defines the contents of the following procedure inputs and outputs; CDB, Data-Out Buffer, Data-In Buffer, and Sense Data. This standard does not define all possible instances of these procedure inputs and outputs. This standard defines only those instances that may apply to any SCSI device. Instances of the procedure inputs and outputs that apply to specific SCSI device models are defined in the applicable SCSI command standards (see 3.1.31).

This standard references values returned via the Status output parameter (e.g., CHECK CONDITION, RESERVATION CONFLICT). Status values are not defined by this standard. SAM-4 defines all Status values.

The entity that makes the procedure call is an application client (see SAM-4). The procedure call's representation arrives at the SCSI target device in the form of a device service request. The entity that performs the work of the procedure call is a device server (see SAM-4).

4.3 The Command Descriptor Block (CDB)

4.3.1 CDB usage and structure

A command is communicated by sending a command descriptor block (CDB) to the device server. For several commands, the CDB is accompanied by a list of parameters in the Data-Out Buffer. See the specific commands for detailed information.

If a logical unit validates reserved CDB fields and receives a reserved field within the CDB that is not zero, then the logical unit 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.

If a logical unit receives a reserved CDB code value in a field other than the OPERATION CODE field, then the logical unit shall terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB.

The fixed length CDB formats are described in 4.3.2. The variable length CDB formats are described in 4.3.3. The XCDB format is described in 4.3.4. The CDB fields that are common to most commands are described in 4.3.5. The fields shown in 4.3.2 and 4.3.3 and described in 4.3.5 are used consistently by most commands. However, the actual usage of any field (except the OPERATION CODE field and the CONTROL field) is described in the subclause defining that command. If a device server receives a CDB containing an operation code that is invalid or not supported, the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID COMMAND OPERATION CODE.

For all commands, if there is an invalid parameter in the CDB, the device server shall terminate the command without altering the medium.

The XCDB format (see 4.3.4) is a CDB in which additional information is appended to another CDB. The requirements in this subclause apply to the XCDB and the CDB contained in the XCDB.

4.3.2 The fixed length CDB formats

All fixed length CDBs shall have an OPERATION CODE field as their first byte and a CONTROL byte as their last byte. Table 11 shows the typical format of a 6-byte CDB. Table 12 shows the typical format of a 10-byte CDB. Table 13 shows the typical format of a 12-byte CDB. Table 14 shows the typical format of a 16-byte CDB. Table 15 shows the typical format of a 16-byte CDB for commands that provide for a long LBA.

Table 11 — Typical CDB for 6-byte commands

Bit / Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE							
1	miscellaneous CDB information			(MSB)				
2	LOGICAL BLOCK ADDRESS (if required)							
3								(LSB)
4	TRANSFER LENGTH (if required) PARAMETER LIST LENGTH (if required) ALLOCATION LENGTH (if required)							
5	CONTROL							

Table 12 — Typical CDB for 10-byte commands

Bit Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE							
1	miscellaneous CDB information			SERVICE ACTION (if required)				
2	(MSB)			LOGICAL BLOCK ADDRESS (if required)				
3								
4								
5								(LSB)
6	miscellaneous CDB information							
7	(MSB)			TRANSFER LENGTH (if required) PARAMETER LIST LENGTH (if required) ALLOCATION LENGTH (if required)				
8								(LSB)
9	CONTROL							

Table 13 — Typical CDB for 12-byte commands

Bit Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE							
1	miscellaneous CDB information			SERVICE ACTION (if required)				
2	(MSB)			LOGICAL BLOCK ADDRESS (if required)				
3								
4								
5								(LSB)
6	(MSB)			TRANSFER LENGTH (if required) PARAMETER LIST LENGTH (if required) ALLOCATION LENGTH (if required)				
7								
8								
9								(LSB)
10	miscellaneous CDB information							
11	CONTROL							

Table 14 — Typical CDB for 16-byte commands

Bit Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE							
1	miscellaneous CDB information			SERVICE ACTION (if required)				
2	(MSB)							
3				LOGICAL BLOCK ADDRESS (if required)				
4								
5								(LSB)
6								
7				Additional CDB data (if required)				
8								
9								
10	(MSB)							
11				TRANSFER LENGTH (if required) PARAMETER LIST LENGTH (if required) ALLOCATION LENGTH (if required)				
12								
13								(LSB)
14	miscellaneous CDB information							
15	CONTROL							

Table 15 — Typical CDB for long LBA 16-byte commands

Bit Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE							
1	miscellaneous CDB information							
2	(MSB)							
3								
4								
5				LOGICAL BLOCK ADDRESS				
6								
7								
8								
9								(LSB)
10	(MSB)							
11				TRANSFER LENGTH (if required) PARAMETER LIST LENGTH (if required) ALLOCATION LENGTH (if required)				
12								
13								(LSB)
14	miscellaneous CDB information							
15	CONTROL							

4.3.3 The variable length CDB formats

The first byte of a variable length CDB shall contain the operation code 7Fh. The CONTROL byte is the second byte in the variable length CDB (see table 16).

Table 16 — Typical variable length CDB

Bit / Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (7Fh)							
1	CONTROL							
2	miscellaneous CDB information							
3	miscellaneous CDB information							
4	miscellaneous CDB information							
5	miscellaneous CDB information							
6	miscellaneous CDB information							
7	ADDITIONAL CDB LENGTH (n-7)							
8	(MSB)	SERVICE ACTION						
9								(LSB)
10	Service action specific fields							
•••								
n								

The OPERATION CODE field is defined in 4.3.5.1 and shall be set as shown in table 16 for a variable length CDB.

The CONTROL field is defined in SAM-5.

The ADDITIONAL CDB LENGTH field specifies the number of additional CDB bytes. This value in the ADDITIONAL CDB LENGTH field shall be a multiple of 4. If the number of CDB bytes delivered by the service delivery subsystem is not sufficient to contain the number of bytes specified by the ADDITIONAL CDB LENGTH 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.

The SERVICE ACTION field specifies the action being requested by the application client. The SERVICE ACTION field is required in the variable length CDB format and is described in 4.3.5.2. Each service action code description defines a number of service action specific fields that are needed for that service action.

A 32-byte variable length CDB format (see table 17) is defined for long LBA operations.

Table 17 — Typical variable length CDB for long LBA 32-byte commands

Bit / Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (7Fh)							
1	CONTROL							
2	miscellaneous CDB information							
3	miscellaneous CDB information							
4	miscellaneous CDB information							
5	miscellaneous CDB information							
6	miscellaneous CDB information							
7	ADDITIONAL CDB LENGTH (18h)							
8	(MSB)	SERVICE ACTION						
9								(LSB)
10	miscellaneous CDB information			DPO	FUA	miscellaneous CDB information		
11	miscellaneous CDB information							
12	(MSB)	LOGICAL BLOCK ADDRESS						
•••								
19								(LSB)
20	miscellaneous CDB information							
•••								
27								
28	(MSB)	TRANSFER LENGTH (if required)						
•••		PARAMETER LIST LENGTH (if required)						
31		ALLOCATION LENGTH (if required)						(LSB)

4.3.4 Extended CDBs

4.3.4.1 XCDB model

With one exception, any SCSI CDB may be extended in an XCDB. An XCDB shall not be extended in another XCDB. Instead, additional XCDB descriptors shall be added to the existing XCDB.

XCDB descriptors may be:

a) added by application clients and removed by device servers, or
b) added and removed by entities outside the scope of this standard that transport or process CDBs and XCDBs during their transfer from an application client to a device server.

4.3.4.2 The XCDB format

The first byte of an XCDB shall contain the operation code 7Eh. The CONTROL byte is the CONTROL byte in the CDB field (see table 18).

Table 18 — XCDB format

Bit / Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (7Eh)							
1	Reserved							
2	(MSB)	LENGTH (n-3)						
3								(LSB)
4	CDB							
•••								
i								
	XCDB descriptors							
k	XCDB descriptor [first]							
•••								
	⋮							
	XCDB descriptor [last]							
•••								
n								

The OPERATION CODE field is defined in 4.3.5.1 and shall be set as shown in table 18 for an XCDB.

The LENGTH field contains the number of bytes that follow in the XCDB.

The CDB field contains the CDB to which XCDB descriptors are being appended.

Each XCDB descriptor (see table 19) contains fields associated with a specified extension type (see table 20). 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 XCDB, if:

a) an XCDB does not contain at least one XCDB descriptor;
b) more than one XCDB descriptor contains the same extension type; or
c) the order of extension types in the XCDB descriptors differs from that shown in table 20.

Table 19 — XCDB descriptor format

Bit / Byte	7	6	5	4	3	2	1	0
0	EXTENSION TYPE							
1	Extension parameters							
•••								
n								

The EXTENSION TYPE field (see table 20) specifies the size and format of the extension parameters that follow in the XCDB descriptor.

Table 20 — EXTENSION TYPE field

Code	Descriptor Order [a]	Description	Extension size (bytes)	Reference
40h	first	CbCS extension descriptor	140	5.16.6.8.16
all others	Reserved			

[a] The order in which XCDB descriptors appear in an XCDB is arranged so that all the XCDB descriptors that follow an XCDB descriptor defined in a future version of this standard are also XCDB descriptors defined in a future version of this standard (i.e., after encountering one unrecognized XCDB descriptor, all subsequent XCDB descriptors are also going to be unrecognized).

The extension parameters contain additional information whose processing is associated with the CDB in the CDB field. The number, content, and size of the extension parameters depends on the contents of the EXTENSION TYPE field.

4.3.5 Common CDB fields

4.3.5.1 Operation code

The first byte of a SCSI CDB shall contain an operation code identifying the operation being requested by the CDB. Some operation codes provide for modification of their operation based on a service action (see 4.3.5.2). In such cases, the operation code and service action code combine to identify the operation being requested. The location of the SERVICE ACTION field in the CDB varies depending on the operation code value.

The OPERATION CODE (see table 21) of the CDB has a GROUP CODE field and a COMMAND CODE field. The three-bit GROUP CODE field provides for eight groups of command codes. The five-bit COMMAND CODE field provides for thirty-two command codes in each group. A total of 256 possible operation codes exist. Operation codes are defined in this standard and other command standards (see 3.1.31). The group code value (see table 22) shall determine the length of the CDB.

Table 21 — OPERATION CODE byte

Bit	7	6	5	4	3	2	1	0
	GROUP CODE			COMMAND CODE				

The value in the GROUP CODE field specifies one of the groups shown in table 22.

Table 22 — Group Code values

Group Code	Meaning	Typical CDB format
000b	6 byte commands	see table 11 in 4.3.2
001b	10 byte commands	see table 12 in 4.3.2
010b	10 byte commands	see table 12 in 4.3.2
011b	reserved [a]	
100b	16 byte commands	see table 14 and table 15 in 4.3.2
101b	12 byte commands	see table 13 in 4.3.2
110b	vendor specific	
111b	vendor specific	

[a] The format of the commands using the group code 011b and operation code 7Fh is described in 4.3.3. The format of the commands using the group code 011b and operation code 7Eh is described in 4.3.4. With the exception of operation codes 7Fh and 7Eh, all group code 011b operation codes are reserved.

4.3.5.2 Service action

All CDB formats except the 6-byte format provide for a SERVICE ACTION field containing a coded value identifying a function to be performed under the more general command function specified in the OPERATION CODE field. While the SERVICE ACTION field is defined for CDB formats, it is used as described in this subclause only in those CDB formats that contain a SERVICE ACTION field. When the specific field SERVICE ACTION is not defined in a CDB format, the bits identified as the SERVICE ACTION field in a CDB shall be used or reserved as specified by the particular CDB format.

4.3.5.3 Logical block address

The logical block addresses on a logical unit or within a volume or partition shall begin with block zero and be contiguous up to the last logical block of that logical unit or within that volume or partition.

A six-byte CDB may contain a 21-bit LOGICAL BLOCK ADDRESS field. The ten-byte and the twelve-byte CDBs may contain 32-bit LOGICAL BLOCK ADDRESS fields. The sixteen-byte CDB has two formats: one allows a 32-bit LOGICAL BLOCK ADDRESS field (see table 14) and the other allows a 64-bit LOGICAL BLOCK ADDRESS field (see table 15). LOGICAL BLOCK ADDRESS fields in additional parameter data have their length specified for each occurrence. See the specific command descriptions.

4.3.5.4 Transfer length

The TRANSFER LENGTH field specifies the amount of data to be transferred, usually the number of blocks. Some commands use transfer length to specify the requested number of bytes to be sent as defined in the command description.

Commands that use one byte for the TRANSFER LENGTH field may allow up to 256 blocks or 256 bytes of data to be transferred by one command.

In commands that use multiple bytes for the TRANSFER LENGTH field, a transfer length of zero specifies that no data transfer shall take place. A value of one or greater specifies the number of blocks or bytes that shall be transferred.

Refer to the specific command description for further information.

4.3.5.5 Parameter list length

The PARAMETER LIST LENGTH field is used to specify the number of bytes sent from the Data-Out Buffer. This field is typically used in CDBs for parameters that are sent to a device server (e.g., mode parameters, diagnostic parameters, log parameters). A parameter length of zero specifies that no data shall be transferred. This condition shall not be considered as an error, unless otherwise specified.

4.3.5.6 Allocation length

The ALLOCATION LENGTH field specifies the maximum number of bytes or blocks that an application client has allocated in the Data-In Buffer. The ALLOCATION LENGTH field specifies bytes unless a different requirement is stated in the command definition.

An allocation length of zero specifies that no data shall be transferred. This condition shall not be considered as an error.

The device server shall terminate transfers to the Data-In Buffer when the number of bytes or blocks specified by the ALLOCATION LENGTH field have been transferred or when all available data have been transferred, whichever is less. The allocation length is used to limit the maximum amount of variable length data (e.g., mode data, log data, diagnostic data) returned to an application client. If the information being transferred to the Data-In Buffer includes fields containing counts of the number of bytes in some or all of the data, then the contents of these fields shall not be altered to reflect the truncation, if any, that results from an insufficient ALLOCATION LENGTH value, unless the standard that describes the Data-In Buffer format states otherwise.

If the amount of information to be transferred exceeds the maximum value that the ALLOCATION LENGTH field is capable of specifying, the device server shall transfer no data and 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.

4.3.5.7 Control

The contents of the CONTROL byte are defined in SAM-4. The CONTROL byte has the same definition for all commands.

4.4 Data field requirements

4.4.1 ASCII data field requirements

ASCII data fields shall contain only ASCII printable characters (i.e., code values 20h to 7Eh) and may be terminated with one or more ASCII null (00h) characters.

ASCII data fields described as being left-aligned shall have any unused bytes at the end of the field (i.e., highest offset) and the unused bytes shall be filled with ASCII space characters (20h).

ASCII data fields described as being right-aligned shall have any unused bytes at the start of the field (i.e., lowest offset) and the unused bytes shall be filled with ASCII space characters (20h).

4.4.2 Null data field termination and zero padding requirements

A data field that is described as being null-terminated shall have one byte containing an ASCII or UTF-8 null (00h) character in the last used byte (i.e., highest offset) of the field and no other bytes in the field shall contain the ASCII/UTF-8 null character.

A data field may be specified to be a fixed length. The length specified for a data field may be greater than the length required to contain the contents of the field. A data field may be specified to have a length that is a multiple of a given value (e.g., a multiple of four bytes). When such fields are described as being null-padded, the bytes at the end of the field that are not needed to contain the field data shall contain ASCII or UTF-8 null (00h) characters. When such fields are described as being zero-padded, the bytes at the end of the field that are not needed to contain the field data shall contain zeros.

NOTE 1 - There is no difference between the pad byte contents in null-padded and zero-padded fields. The difference is in the format of the other bytes in the field.

A data field that is described as being both null-terminated and null-padded shall have at least one byte containing an ASCII or UTF-8 null (00h) character in the end of the field (i.e., highest offset) and may have more than one byte containing ASCII or UTF-8 null characters, if needed, to meet the specified field length requirements. If more than one byte in a null-terminated, null-padded field contains the ASCII or UTF-8 null character, then all the bytes containing the ASCII or UTF-8 null character shall be at the end of the field (i.e., only the highest offsets).

4.4.3 Variable type data field requirements

Parameter lists may contain fields or descriptors in which data may be represented in different formats. To indicate which format is being used a field may be defined that contains a code set enumeration (see table 23).

Table 23 — Code set enumeration

Code	Description
0h	Reserved
1h	The associated fields or descriptors contain binary values
2h	The associated fields or descriptors contain ASCII printable characters (i.e., code values 20h to 7Eh)
3h	The associated fields or descriptors contain UTF-8 (see 3.1.189) codes
4h to Fh	Reserved

4.5 Sense data

4.5.1 Sense data introduction

Sense data shall be returned in the same I_T_L_Q nexus transaction as the status and as parameter data in response to the REQUEST SENSE command (see 6.29). Sense data returned in the same I_T_L_Q nexus transaction as the status shall be either fixed or descriptor format sense data format based on the value of the D_SENSE bit in the Control mode page (see 7.5.7). The REQUEST SENSE command may be used to request either the fixed format sense data or the descriptor format sense data.

The first byte of all sense data contains the RESPONSE CODE field (see table 24) that indicates the report type and format of the sense data.

Table 24 — Sense data response codes

Response Code	Report type		Sense data format	
	Description	Reference	Description	Reference
00h to 6Fh	Reserved			
70h	Current information	4.5.4	Fixed	4.5.3
71h	Deferred error	4.5.5	Fixed	4.5.3
72h	Current information	4.5.4	Descriptor	4.5.2
73h	Deferred error	4.5.5	Descriptor	4.5.2
74h to 7Eh	Reserved			
7Fh	Vendor specific			

The RESPONSE CODE field shall be set to 70h in all unit attention condition sense data in which:

a) the ADDITIONAL SENSE CODE field is set to 29h; or
b) the additional sense code is set to MODE PARAMETERS CHANGED.

4.5.2 Descriptor format sense data

4.5.2.1 Descriptor format sense data overview

The descriptor format sense data for response codes 72h (i.e., current information) and 73h (i.e., deferred errors) is defined in table 25.

Table 25 — Descriptor format sense data

Bit / Byte	7	6	5	4	3	2	1	0
0	Reserved	RESPONSE CODE (72h or 73h)						
1	Reserved				SENSE KEY			
2	ADDITIONAL SENSE CODE							
3	ADDITIONAL SENSE CODE QUALIFIER							
4 ••• 6	Reserved							
7	ADDITIONAL SENSE LENGTH (n-7)							
	Sense data descriptor(s)							
8 •••	Sense data descriptor 0 (see table 26)							
	⋮							
••• n	Sense data descriptor x (see table 26)							

The contents of the RESPONSE CODE field indicate the report type and format of the sense data (see 4.5.1). For descriptor format sense data, the RESPONSE CODE field shall be set to 72h or 73h.

The SENSE KEY, ADDITIONAL SENSE CODE and ADDITIONAL SENSE CODE QUALIFIER fields provide a hierarchy of information. The hierarchy provides a top-down approach for an application client to determine information relating to the reported condition.

The SENSE KEY field indicates generic information describing a reported condition. The sense keys are defined in 4.5.6.

The ADDITIONAL SENSE CODE field indicates further information related to the condition reported in the SENSE KEY field. Support of the additional sense codes not required by this standard is optional. For a list of additional sense codes see 4.5.6. If the device server does not have further information related to the reported condition, the ADDITIONAL SENSE CODE field shall be set to zero.

The ADDITIONAL SENSE CODE QUALIFIER field indicates detailed information related to the condition reported in the ADDITIONAL SENSE CODE field. If the condition is reported by the device server, the value returned shall be as defined in 4.5.6. If the device server does not have detailed information related to the reported condition, the ADDITIONAL SENSE CODE QUALIFIER field shall be set to zero.

The ADDITIONAL SENSE LENGTH field indicates the number of additional sense bytes that follow. The additional sense length shall be less than or equal to 244 (i.e., limiting the total length of the sense data to 252 bytes). If the sense data is being returned as parameter data by a REQUEST SENSE command, then the relationship between the ADDITIONAL SENSE LENGTH field and the CDB ALLOCATION LENGTH field is defined in 4.3.5.6.

Sense data descriptors (see table 26) provide specific sense information. A given type of sense data descriptor shall be included in the sense data only when the information it contains is valid.

Table 26 — Sense data descriptor format

Bit / Byte	7	6	5	4	3	2	1	0
0	DESCRIPTOR TYPE							
1	ADDITIONAL LENGTH (n-1)							
2	Sense data descriptor specific							
•••								
n								

The DESCRIPTOR TYPE field (see table 27) identifies the type of sense data descriptor.

Table 27 — DESCRIPTOR TYPE field

Code	Description	Number of descriptors allowed (maximum)	Reference
00h	Information	1	4.5.2.2
01h	Command specific information	1	4.5.2.3
02h	Sense key specific	1	4.5.2.4
03h	Field replaceable unit	1	4.5.2.5
04h	Stream commands	1	SSC-3
05h	Block commands	1	SBC-3
06h	OSD object identification	1	OSD
07h	OSD response integrity check value	1	OSD
08h	OSD attribute identification	1	OSD
09h	ATA Status Return	1	SAT
0Ah	Progress indication	32	4.5.2.6
0Bh	User data segment referral	1	SBC-3
0Ch	Forwarded sense data	2	4.5.2.7
0Dh to 7Fh	Reserved		
80h to FFh	Vendor specific		4.5.2.8

The ADDITIONAL LENGTH field indicates the number of sense data descriptor specific bytes that follow in the sense data descriptor.

4.5.2.2 Information sense data descriptor

The information sense data descriptor (see table 28) provides information that is device-type or command specific and is defined in a command standard (see 3.1.31).

Table 28 — Information sense data descriptor format

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	(MSB)	INFORMATION						
•••								
11								(LSB)

The DESCRIPTOR TYPE and ADDITIONAL LENGTH fields are described in 4.5.2.1 and shall be set as shown in table 28 for the information sense data descriptor.

The VALID bit shall be set to one.

NOTE 2 - In previous versions of this standard and in the fixed format sense data, the VALID bit indicates whether the contents of the INFORMATION field is valid as defined by a command standard. Since the contents of the INFORMATION field are valid whenever an information sense data descriptor is included in the sense data, the only legal value for the VALID bit is set to one.

The contents of the INFORMATION field are device-type or command specific and are defined in a command standard (see 3.1.31). When a four byte quantity is stored in the INFORMATION field, the first four bytes shall be zero.

4.5.2.3 Command-specific information sense data descriptor

The command-specific information sense data descriptor (see table 29) provides information that depends on the command for which the reported condition occurred.

Table 29 — Command-specific information sense data descriptor format

Bit / Byte	7	6	5	4	3	2	1	0
0	DESCRIPTOR TYPE (01h)							
1	ADDITIONAL LENGTH (0Ah)							
2	Reserved							
3	Reserved							
4	(MSB)	COMMAND-SPECIFIC INFORMATION						
•••								
11								(LSB)

The DESCRIPTOR TYPE and ADDITIONAL LENGTH fields are described in 4.5.2.1 and shall be set as shown in table 29 for the command-specific information sense data descriptor.

The COMMAND-SPECIFIC INFORMATION field contains information that depends on the command for which the reported condition occurred. When a four byte quantity is stored in the COMMAND-SPECIFIC INFORMATION field, the first four bytes shall be set to zero.

Further meaning for the COMMAND-SPECIFIC INFORMATION field is defined within the command description in the appropriate command standard (e.g., SBC-3 for the REASSIGN BLOCKS commands, or 6.3 for the EXTENDED COPY command).

4.5.2.4 Sense key specific sense data descriptor

4.5.2.4.1 Sense key specific sense data descriptor introduction

The sense key specific sense data descriptor (see table 30) provides additional information about the reported condition described in 4.5.2.1. The format and content of the sense-key specific data depends on the value in the SENSE KEY field (see 4.5.2.1).

Table 30 — Sense key specific sense data descriptor format

Bit / Byte	7	6	5	4	3	2	1	0
0	DESCRIPTOR TYPE (02h)							
1	ADDITIONAL LENGTH (06h)							
2	Reserved							
3	Reserved							
4	SKSV (1b)	SENSE KEY SPECIFIC						
5								
6								
7	Reserved							

The DESCRIPTOR TYPE and ADDITIONAL LENGTH fields are described in 4.5.2.1 and shall be set as shown in table 30 for the sense-key specific sense data descriptor.

The sense-key specific valid (SKSV) bit shall be set to one.

NOTE 3 - In SPC-2 and in the fixed format sense data, the SKSV bit indicates whether the contents of the SENSE KEY SPECIFIC field are valid as defined by a command standard. Since the contents of the SENSE KEY SPECIFIC field are valid whenever a sense key specific sense data descriptor is included in the sense data, the only legal value for the SKSV bit is set to one.

The definition of the SENSE KEY SPECIFIC field (see table 31) is determined by the value of the SENSE KEY field (see 4.5.2.1).

Table 31 — Sense key specific sense data descriptor definitions

Sense key	SENSE KEY SPECIFIC field definition	Reference
ILLEGAL REQUEST	Field pointer	4.5.2.4.2
HARDWARE ERROR, MEDIUM ERROR, or RECOVERED ERROR	Actual retry count	4.5.2.4.3
NO SENSE or NOT READY	Progress indication	4.5.2.4.4
COPY ABORTED	Segment pointer	4.5.2.4.5
UNIT ATTENTION	Unit attention condition queue overflow	4.5.2.4.6
All other sense keys	The sense key specific sense data descriptor shall not appear in the descriptor format sense data and the SKSV bit (see 4.5.3) shall be set to zero in the fixed format sense data.	

4.5.2.4.2 Field pointer sense key specific data

If the sense key is ILLEGAL REQUEST, then the SENSE KEY SPECIFIC field shall be as shown in table 32.

Table 32 — Field pointer sense key specific data

Bit / Byte	7	6	5	4	3	2	1	0
0	SKSV (1b)	C/D	Reserved		BPV	BIT POINTER		
1	(MSB)	FIELD POINTER						
2								(LSB)

The SKSV bit is described in 4.5.2.4.1 for descriptor format sense data and in 4.5.3 for fixed format sense data.

A command data (C/D) bit set to one indicates that the illegal parameter is in the CDB. A C/D bit set to zero indicates that the illegal parameter is in the data parameters sent by the application client in the Data-Out Buffer.

A bit pointer valid (BPV) bit set to zero indicates that the value in the BIT POINTER field is not valid. A BPV bit set to one indicates that the BIT POINTER field specifies which bit of the byte designated by the FIELD POINTER field is in error. When a multiple-bit field is in error, the BIT POINTER field shall point to the first bit (i.e., the left-most bit) of the field.

The FIELD POINTER field indicates which byte of the CDB or of the parameter data was in error. Bytes are numbered starting from zero, as shown in the tables describing the commands and parameters. When a multiple-byte field is in error, the field pointer shall point to the first byte (i.e., the left-most byte) of the field. If several consecutive bytes are reserved, each shall be treated as a single-byte field.

NOTE 4 - The bytes identified as being in error are not necessarily the bytes that need to be changed to correct the problem.

4.5.2.4.3 Actual retry count sense key specific data

If the sense key is HARDWARE ERROR, MEDIUM ERROR, or RECOVERED ERROR, then the SENSE KEY SPECIFIC field shall be as shown in table 33.

Table 33 — Actual retry count sense key specific data

Bit / Byte	7	6	5	4	3	2	1	0
0	SKSV (1b)	Reserved						
1	(MSB)	ACTUAL RETRY COUNT						
2								(LSB)

The SKSV bit is described in 4.5.2.4.1 for descriptor format sense data and in 4.5.3 for fixed format sense data.

The ACTUAL RETRY COUNT field returns vendor specific information on the number of retries of the recovery algorithm used in attempting to recover an error or exception condition.

NOTE 5 - This field should be computed in the same way as the retry count fields within the Read-Write Error Recovery mode page (see SBC-2, SSC-3, and MMC-4).

4.5.2.4.4 Progress indication sense key specific data

If the sense key is NO SENSE or NOT READY, the SENSE KEY SPECIFIC FIELD shall be as shown in table 34.

Table 34 — Progress indication sense key specific data

Bit / Byte	7	6	5	4	3	2	1	0
0	SKSV (1b)	Reserved						
1	(MSB)	PROGRESS INDICATION						
2								(LSB)

The SKSV bit is described in 4.5.2.4.1 for descriptor format sense data and in 4.5.3 for fixed format sense data.

The PROGRESS INDICATION field is a percent complete indication in which the returned value is a numerator that has 65 536 (10000h) as its denominator. The progress indication shall be based upon the total operation.

NOTE 6 - The progress indication should be time related, however this is not an absolute requirement. (E.g., since format time varies with the number of defects encountered, etc., it is reasonable for the device server to assign values to various steps within the process. The granularity of these steps should be small enough to provide reasonable assurances to the application client that progress is being made.)

4.5.2.4.5 Segment pointer sense key specific data

If the sense key is COPY ABORTED, the SENSE KEY SPECIFIC field shall be as shown in table 35.

Table 35 — Segment pointer sense key specific data

Bit / Byte	7	6	5	4	3	2	1	0
0	SKSV (1b)	Reserved	SD	Reserved	BPV	BIT POINTER		
1	(MSB)	FIELD POINTER						
2								(LSB)

The SKSV bit is described in 4.5.2.4.1 for descriptor format sense data and in 4.5.3 for fixed format sense data.

The segment descriptor (SD) bit indicates whether the field pointer is relative to the start of the parameter list or to the start of a segment descriptor. An SD bit set to zero indicates that the field pointer is relative to the start of the parameter list. An SD bit set to one indicates that the field pointer is relative to the start of the segment descriptor indicated by the third and fourth bytes of the COMMAND-SPECIFIC INFORMATION field (see 6.3.3).

A bit pointer valid (BPV) bit set to zero indicates that the value in the BIT POINTER field is not valid. A BPV bit set to one indicates that the BIT POINTER field specifies which bit of the byte designated by the FIELD POINTER field is in error. When a multiple-bit field is in error, the BIT POINTER field shall point to the most-significant (i.e., left-most) bit of the field.

The FIELD POINTER field indicates which byte of the parameter list or segment descriptor was in error.

If the parameter list is in excess of 65 528 bytes in length and SD is set to zero, the FIELD POINTER value may not fit in two bytes provided by the sense key specific sense data descriptor.

4.5.2.4.6 Unit attention condition queue overflow sense key specific data

If the sense key is UNIT ATTENTION, the SENSE KEY SPECIFIC field shall be as shown in table 36.

Table 36 — Unit attention condition queue overflow sense key specific data

Bit / Byte	7	6	5	4	3	2	1	0
0	SKSV (1b)	Reserved						OVERFLOW
1		Reserved						
2								

The SKSV bit is described in 4.5.2.4.1 for descriptor format sense data and in 4.5.3 for fixed format sense data.

An OVERFLOW bit set to one indicates that the unit attention condition queue has overflowed. An OVERFLOW bit set to zero indicates that the unit attention condition queue has not overflowed.

4.5.2.5 Field replaceable unit sense data descriptor

The field replaceable unit sense data descriptor (see table 37) provides information about a component associated with the sense data.

Table 37 — Field replaceable unit sense data descriptor format

Bit / Byte	7	6	5	4	3	2	1	0
0	DESCRIPTOR TYPE (03h)							
1	ADDITIONAL LENGTH (02h)							
2	Reserved							
3	FIELD REPLACEABLE UNIT CODE							

The DESCRIPTOR TYPE and ADDITIONAL LENGTH fields are described in 4.5.2.1 and shall be set as shown in table 37 for the field replaceable unit sense data descriptor.

Non-zero values in the FIELD REPLACEABLE UNIT CODE field are used to identify a component associated with the sense data. A value of zero in this field indicates that no specific component has been associated with the sense data or that the data is not available. The format of this information is not specified by this standard. Additional information about the field replaceable unit may be available in the ASCII Information VPD page (see 7.8.3), if supported by the device server.

4.5.2.6 Progress indication sense data descriptor

If the sense key is set to NO SENSE or NOT READY, the progress indication sense data descriptor (see table 38) may be used to provide a progress indication for one operation other than the one described by the non-descriptor fields in 4.5.2.1. The sense data should include one progress indication sense data descriptor for each operation for which the device server is able to report progress other than the operation described by the non-descriptor fields in 4.5.2.1.

Table 38 — Progress indication sense data descriptor format

Bit / Byte	7	6	5	4	3	2	1	0
0	DESCRIPTOR TYPE (0Ah)							
1	ADDITIONAL LENGTH (06h)							
2	SENSE KEY							
3	ADDITIONAL SENSE CODE							
4	ADDITIONAL SENSE CODE QUALIFIER							
5	Reserved							
6	(MSB)	PROGRESS INDICATION						
7								(LSB)

The DESCRIPTOR TYPE field and ADDITIONAL LENGTH field are described in 4.5.2.1 and shall be set as shown in table 38 for the progress indications sense data descriptor.

The SENSE KEY field indicates the operation for which this progress indication sense data descriptor provides a progress indication. A list of sense key values is in 4.5.6.

The ADDITIONAL SENSE CODE field indicates the operation for which this progress indication sense data descriptor provides a progress indication. A list of additional sense codes is in 4.5.6.

The ADDITIONAL SENSE CODE QUALIFIER field indicates detailed information related to the additional sense code for the operation for which this progress indication sense data descriptor provides a progress indication. The value returned in the ADDITIONAL SENSE CODE QUALIFIER field shall be as specified in 4.5.6.

The PROGRESS INDICATION field indicates a percent complete for the operation indicated by the SENSE KEY field, the ADDITIONAL SENSE CODE field, and the ADDITIONAL SENSE CODE QUALIFIER field. The value in the PROGRESS INDICATION field shall be as defined in 4.5.2.4.4.

4.5.2.7 Forwarded sense data

Forwarded sense data descriptors (see table 39) contain the status and sense data that is supplied by another device server as part of command completion (e.g., an exception condition reported by a copy target device in an EXTENDED COPY command (see 6.3) during segment descriptor processing).

Table 39 — Forwarded sense data descriptor format

Bit / Byte	7	6	5	4	3	2	1	0
0	DESCRIPTOR TYPE (0Ch)							
1	ADDITIONAL LENGTH (n-1)							
2	FSDT	Reserved			SENSE DATA SOURCE			
3	FORWARDED STATUS							
4	FORWARDED SENSE DATA							
•••								
n								

The DESCRIPTOR TYPE field is described in 4.5.2.1, and shall be set as shown in table 39 for the forwarded sense data descriptor.

The ADDITIONAL LENGTH field shall be set to:

a) a minimum of 22, if the FORWARDED SENSE DATA field contains fixed format sense data (i.e., the RESPONSE CODE field (see 4.5.1) in the forwarded sense data is 70h or 71h);
b) a minimum of 10, if the FORWARDED SENSE DATA field contains descriptor format sense data (i.e., the RESPONSE CODE field in the forwarded sense data is 72h or 73h); and
c) two less than a multiple of four.

A forwarded sense data truncated (FSDT) bit set to one indicates that the contents of the FORWARDED SENSE DATA field have been truncated (i.e., the FORWARDED SENSE DATA field does not contain all of the sense data that was supplied). An FSDT bit is set to zero indicates that the contents of the FORWARDED SENSE DATA field have not been truncated.

The SENSE DATA SOURCE field (see table 40) indicates the supplier of the forwarded sense data.

Table 40 — SENSE DATA SOURCE field

Code	Description
0h	Unknown
1h	EXTENDED COPY command (see 6.3) source device
2h	EXTENDED COPY command destination device
all others	Reserved

The FORWARDED STATUS field contains the status code (see SAM-5) returned by the supplier of the forwarded sense data at the same time that the forwarded sense data was returned.

The FORWARDED SENSE DATA field contains the forwarded sense data. The FORWARDED SENSE DATA field shall be zero-padded in a way that causes its length to be a multiple of four bytes.

4.5.2.8 Vendor specific sense data descriptors

Vendor specific sense data descriptors (see table 41) contain vendor specific data that further defines the nature of the reported condition.

Table 41 — Vendor specific sense data descriptor format

Bit / Byte	7	6	5	4	3	2	1	0
0	DESCRIPTOR TYPE (80h to FFh)							
1	ADDITIONAL LENGTH (n-1)							
2	Vendor specific							
•••								
n								

The DESCRIPTOR TYPE field and ADDITIONAL LENGTH field are described in 4.5.2.1. The DESCRIPTOR TYPE field shall be set as shown in table 41 for the vendor specific sense data descriptor.

4.5.3 Fixed format sense data

The fixed format sense data for response codes 70h (current information) and 71h (deferred errors) is defined in table 42.

Table 42 — Fixed format sense data

Bit / Byte	7	6	5	4	3	2	1	0
0	VALID	RESPONSE CODE (70h or 71h)						
1	Obsolete							
2	FILEMARK	EOM	ILI	Reserved	SENSE KEY			
3	(MSB)	INFORMATION						
•••								
6								(LSB)
7	ADDITIONAL SENSE LENGTH (n-7)							
8	(MSB)	COMMAND-SPECIFIC INFORMATION						
•••								
11								(LSB)
12	ADDITIONAL SENSE CODE							
13	ADDITIONAL SENSE CODE QUALIFIER							
14	FIELD REPLACEABLE UNIT CODE							
15	SKSV	SENSE KEY SPECIFIC						
16								
17								
18	Additional sense bytes							
•••								
n								

A VALID bit set to zero indicates that the INFORMATION field is not defined in this standard or any other command standard (see 3.1.31). A VALID bit set to one indicates the INFORMATION field contains valid information as defined in this standard or a command standard.

The contents of the RESPONSE CODE field indicate the report type and format of the sense data (see 4.5.1). For fixed format sense data, the RESPONSE CODE field shall be set to 70h or 71h.

The meaning of the FILEMARK bit is device-type or command specific (e.g., see the SSC-3 READ command and SPACE command for examples of FILEMARK bit usage) and the bit is defined in a command standard.

The meaning of the end-of-medium (EOM) bit is device-type or command specific (e.g., see the SSC-3 READ command, SPACE command, and WRITE command for examples of EOM bit usage) and the bit is defined in a command standard.

The meaning of the incorrect length indicator (ILI) bit is device-type or command specific (e.g., see the SBC-3 READ LONG command, SBC-3 WRITE LONG command, and SSC-3 READ command for examples of ILI bit usage) and the bit is defined in a command standard.

The SENSE KEY, ADDITIONAL SENSE CODE, and ADDITIONAL SENSE CODE QUALIFIER fields are described in 4.5.2.1.

The contents of the INFORMATION field are device-type or command specific and are defined in a command standard.

The ADDITIONAL SENSE LENGTH field indicates the number of additional sense bytes that follow. The additional sense length shall be less than or equal to 244 (i.e., limiting the total length of the sense data to 252 bytes). If the sense data is being returned as parameter data by a REQUEST SENSE command, then the relationship between the ADDITIONAL SENSE LENGTH field and the CDB ALLOCATION LENGTH field is defined in 4.3.5.6.

The COMMAND-SPECIFIC INFORMATION field contains information that depends on the command for which the reported condition occurred.

The FIELD REPLACEABLE UNIT CODE field is described in 4.5.2.5.

A sense-key specific valid (SKSV) bit set to one indicates the SENSE KEY SPECIFIC field contains valid information as defined in this standard. An SKSV bit set to zero indicates that the SENSE KEY SPECIFIC field is not as defined by this standard.

The SENSE KEY SPECIFIC field is described in 4.5.2.4.

The additional sense bytes may contain vendor specific data that further defines the nature of the reported condition.

4.5.4 Current information

Response codes 70h and 72h (i.e., current information) indicate that the sense data returned is:

a) the result of an error or exception condition on the command that returned the CHECK CONDITION status;
b) the result of a protocol specific failure condition; or
c) additional information returned with a status other than CHECK CONDITION.

Current information includes:

a) errors generated during processing of the command;
b) errors not related to any command that are detected during processing of a command (e.g., disk servo-mechanism failures, off-track errors, or power-up test errors); and
c) referral information (see SBC-3) returned in association with a GOOD status.

4.5.5 Deferred errors

Response codes 71h and 73h (deferred error) indicate that the sense data returned is the result of an error or exception condition that occurred during processing of a previous command for which GOOD status or CONDITION MET status has already been returned. Such commands are associated with the use of the immediate bit and with some forms of caching. Device servers that implement these features shall implement deferred error reporting.

The deferred error may be indicated by returning CHECK CONDITION status to an application client accessed through a defined I_T nexus as described in this subclause.

If the command terminates with CHECK CONDITION status and the sense data describes a deferred error, the terminated command shall not have been processed. After the device server detects a deferred error condition, it shall return a deferred error according to the following rules:

a) if no external intervention is necessary to recover a deferred error, a deferred error indication shall not be returned unless required by the error handling parameters of a MODE SELECT command. The occurrence of the error may be logged;
b) if it is possible to associate a deferred error with an I_T nexus and with a particular function or a particular subset of data, and the error is either unrecovered or required to be reported by the mode parameters, then a deferred error indication shall be returned for a command received on the I_T nexus associated with the deferred error. If an application client request received on an I_T nexus other than the I_T nexus associated with the deferred error attempts to access the particular function or subset of data associated with the deferred error and the TST field equals 000b (see 7.5.7), then the device server shall complete the command with BUSY status or ACA ACTIVE status according to the requirements in SAM-4. If an application client request received on an I_T nexus other than the I_T nexus associated with the deferred error attempts to access the particular function or subset of data associated with the deferred error and the TST field equals 001b, then the command attempting the access shall not be blocked by the deferred error and the cause of the deferred error may result in an error being reported for the command attempting the access;
c) if the device server is unable to associate a deferred error with an I_T nexus or with a particular subset of data, the device server shall return a deferred error for one command received on each I_T nexus. If multiple deferred errors have accumulated for an I_T nexus, only the last error shall be returned;
d) if the SCSI target device is unable to associate a deferred error with a particular logical unit, it shall establish a deferred error for every logical unit and shall return the deferred error for one command for each logical unit received on each appropriate I_T nexus; or
e) if a command has never entered the enabled command state, and a deferred error occurs, the command shall be terminated with CHECK CONDITION status and deferred error information returned in the sense data. If a deferred error occurs after a command has entered the enabled command state and the command is affected by the error, the command shall be terminated with CHECK CONDITION status and the current error information shall be returned in the sense data. In this case, if the current error information does not adequately define the deferred error condition, a deferred error may be returned after the current error information has been returned. If a deferred error occurs after a command has entered the enabled command state and the command completes successfully, the device server may choose to return the deferred error information after the completion of the current command in conjunction with a subsequent command that has not begun processing.

NOTE 7 - A deferred error may indicate that an operation was unsuccessful long after GOOD status was returned. If the application client is unable to replicate or recover from other sources the data that is being written using cached or buffered write operations, then synchronization commands should be performed before the critical data is destroyed. This is necessary for actions taken when deferred errors occur in the storing of the data. The synchronizing process should provide the necessary commands to allow returning CHECK CONDITION status and subsequent returning of deferred error sense information after all cached or buffered operations are completed.

4.5.6 Sense key and additional sense code definitions

The sense keys are defined in table 43.

Table 43 — Sense key descriptions (part 1 of 2)

Sense Key	Description
0h	**NO SENSE:** Indicates that there is no specific sense key information to be reported. This may occur for a successful command or for a command that receives CHECK CONDITION status because one of the FILEMARK, EOM, or ILI bits is set to one.
1h	**RECOVERED ERROR:** Indicates that the command completed successfully, with some recovery action performed by the device server. Details may be determined by examining the additional sense bytes and the INFORMATION field. When multiple recovered errors occur during one command, the choice of which error to report (e.g., first, last, most severe) is vendor specific.
2h	**NOT READY:** Indicates that the logical unit is not accessible. Operator intervention may be required to correct this condition.
3h	**MEDIUM ERROR:** Indicates that the command terminated with a non-recovered error condition that may have been caused by a flaw in the medium or an error in the recorded data. This sense key may also be returned if the device server is unable to distinguish between a flaw in the medium and a specific hardware failure (i.e., sense key 4h).
4h	**HARDWARE ERROR:** Indicates that the device server detected a non-recoverable hardware failure (e.g., controller failure, device failure, or parity error) while performing the command or during a self test.
5h	**ILLEGAL REQUEST:** Indicates that: a) the command was addressed to an incorrect logical unit number (see SAM-4); b) the command had an invalid task attribute (see SAM-4); c) the command was addressed to a logical unit whose current configuration prohibits processing the command; d) there was an illegal parameter in the CDB; or e) there was an illegal parameter in the additional parameters supplied as data for some commands (e.g., PERSISTENT RESERVE OUT). If the device server detects an invalid parameter in the CDB, it shall terminate the command without altering the medium. If the device server detects an invalid parameter in the additional parameters supplied as data, the device server may have already altered the medium.
6h	**UNIT ATTENTION:** Indicates that a unit attention condition has been established (e.g., the removable medium may have been changed, a logical unit reset occurred). See SAM-4.
7h	**DATA PROTECT:** Indicates that a command that reads or writes the medium was attempted on a block that is protected. The read or write operation is not performed.
8h	**BLANK CHECK:** Indicates that a write-once device or a sequential-access device encountered blank medium or format-defined end-of-data indication while reading or that a write-once device encountered a non-blank medium while writing.
9h	**VENDOR SPECIFIC:** This sense key is available for reporting vendor specific conditions.
Ah	**COPY ABORTED:** Indicates an EXTENDED COPY command was aborted due to an error condition on the source device, the destination device, or both (see 6.3.3).
Bh	**ABORTED COMMAND:** Indicates that the device server aborted the command. The application client may be able to recover by trying the command again.
Ch	Reserved

Table 43 — Sense key descriptions (part 2 of 2)

Sense Key	Description
Dh	**VOLUME OVERFLOW:** Indicates that a buffered SCSI device has reached the end-of-partition and data may remain in the buffer that has not been written to the medium. One or more RECOVER BUFFERED DATA command(s) may be issued to read the unwritten data from the buffer. (See SSC-2.)
Eh	**MISCOMPARE:** Indicates that the source data did not match the data read from the medium.
Fh	**COMPLETED:** Indicates there is completion sense data to be reported. This may occur for a successful command.

The additional sense codes (i.e., the ADDITIONAL SENSE CODE field and ADDITIONAL SENSE CODE QUALIFIER field values returned in sense data) are defined in table 44.

Table 44 — ASC and ASCQ assignments (part 1 of 17)

```
                 D – Direct Access Block Device (SBC-3)                    Device Column key
                 . T – Sequential Access Device (SSC-4)                    blank = code not used
                 .   L – Printer Device (SSC)                              not blank = code used
                 .     P – Processor Device (SPC-2)
                 .     . W – Write Once Block Device (SBC)
                 .     .   R – C/DVD Device (MMC-6)
                 .     .     O – Optical Memory Block Device (SBC)
                 .     .     . M – Media Changer Device (SMC-3)
                 .     .     .   A – Storage Array Device (SCC-2)
                 .     .     .     E – SCSI Enclosure Services device (SES-3)
                 .     .     .     . B – Simplified Direct-Access (Reduced Block) device (RBC)
                 .     .     .     .   K – Optical Card Reader/Writer device (OCRW)
                 .     .     .     .     V – Automation/Device Interface device (ADC-3)
                 .     .     .     .     . F – Object-based Storage Device (OSD-2)
                 .     .     .     .     .
ASC  ASCQ        DTLPWROMAEBKVF  Description
20h  0Bh         DT  PWROMAEBK   ACCESS DENIED - ACL LUN CONFLICT
20h  08h         DT  PWROMAEBK   ACCESS DENIED - ENROLLMENT CONFLICT
20h  01h         DT  PWROMAEBK   ACCESS DENIED - INITIATOR PENDING-ENROLLED
20h  09h         DT  PWROMAEBK   ACCESS DENIED - INVALID LU IDENTIFIER
20h  03h         DT  PWROMAEBK   ACCESS DENIED - INVALID MGMT ID KEY
20h  0Ah         DT  PWROMAEBK   ACCESS DENIED - INVALID PROXY TOKEN
20h  02h         DT  PWROMAEBK   ACCESS DENIED - NO ACCESS RIGHTS
4Bh  03h         DT  PWROMAEBK   ACK/NAK TIMEOUT
67h  02h                 A       ADD LOGICAL UNIT FAILED
13h  00h         D     W O   BK  ADDRESS MARK NOT FOUND FOR DATA FIELD
12h  00h         D     W O   BK  ADDRESS MARK NOT FOUND FOR ID FIELD
67h  08h                 A       ASSIGN FAILURE OCCURRED
27h  03h           T     R       ASSOCIATED WRITE PROTECT
2Ah  06h         DTLPWROMAEBKVF  ASYMMETRIC ACCESS STATE CHANGED
47h  04h         DTLPWROMAEBKVF  ASYNCHRONOUS INFORMATION PROTECTION ERROR DETECTED
44h  71h         DT          B   ATA DEVICE FAILED SET FEATURES
67h  0Bh         DT          B   ATA DEVICE FEATURE NOT ENABLED
00h  1Dh         DT          B   ATA PASS THROUGH INFORMATION AVAILABLE
67h  06h                 A       ATTACHMENT OF LOGICAL UNIT FAILED
00h  11h             R           AUDIO PLAY OPERATION IN PROGRESS
00h  12h             R           AUDIO PLAY OPERATION PAUSED
00h  14h             R           AUDIO PLAY OPERATION STOPPED DUE TO ERROR
```

A numeric ordered listing of the ASC and ASCQ assignments is provided in D.2.

Table 44 — ASC and ASCQ assignments (part 2 of 17)

D – Direct Access Block Device (SBC-3)
. T – Sequential Access Device (SSC-4)
. L – Printer Device (SSC)
. P – Processor Device (SPC-2)
. . W – Write Once Block Device (SBC)
. . R – C/DVD Device (MMC-6)
. . O – Optical Memory Block Device (SBC)
. . . M – Media Changer Device (SMC-3)
. . . A – Storage Array Device (SCC-2)
. . . E – SCSI Enclosure Services device (SES-3)
. . . . B – Simplified Direct-Access (Reduced Block) device (RBC)
. . . . K – Optical Card Reader/Writer device (OCRW)
. . . . V – Automation/Device Interface device (ADC-3)
. . . . . F – Object-based Storage Device (OSD-2)

Device Column key
blank = code not used
not blank = code used

ASC	ASCQ	D	T	L	P	W	R	O	M	A	E	B	K	V	F	Description
00h	13h						R									AUDIO PLAY OPERATION SUCCESSFULLY COMPLETED
74h	40h	D	T				R		M	A	E	B	K	V		AUTHENTICATION FAILED
66h	00h															AUTOMATIC DOCUMENT FEEDER COVER UP
66h	01h															AUTOMATIC DOCUMENT FEEDER LIFT UP
55h	06h	D	T			W	R	O	M			B				AUXILIARY MEMORY OUT OF SPACE
11h	12h	D	T			W	R	O	M			B				AUXILIARY MEMORY READ ERROR
0Ch	0Bh	D	T			W	R	O	M			B				AUXILIARY MEMORY WRITE ERROR
00h	04h		T													BEGINNING-OF-PARTITION/MEDIUM DETECTED
0Ch	06h	D	T			W		O				B				BLOCK NOT COMPRESSIBLE
14h	04h		T													BLOCK SEQUENCE ERROR
29h	03h	D	T	L	P	W	R	O	M	A	E	B	K	V	F	BUS DEVICE RESET FUNCTION OCCURRED
11h	0Eh	D	T			W	R	O				B				CANNOT DECOMPRESS USING DECLARED ALGORITHM
30h	06h	D	T			W	R	O				B				CANNOT FORMAT MEDIUM - INCOMPATIBLE MEDIUM
30h	02h	D	T			W	R	O				B	K			CANNOT READ MEDIUM - INCOMPATIBLE FORMAT
30h	01h	D	T			W	R	O				B	K			CANNOT READ MEDIUM - UNKNOWN FORMAT
30h	08h						R									CANNOT WRITE - APPLICATION CODE MISMATCH
30h	05h	D	T			W	R	O				B	K			CANNOT WRITE MEDIUM - INCOMPATIBLE FORMAT
30h	04h	D	T			W	R	O				B	K			CANNOT WRITE MEDIUM - UNKNOWN FORMAT
2Ah	09h	D														CAPACITY DATA HAS CHANGED
52h	00h		T													CARTRIDGE FAULT
73h	00h						R									CD CONTROL ERROR
24h	01h	D	T	L	P	W	R	O		A	E	B	K	V	F	CDB DECRYPTION ERROR
3Fh	02h	D	T	L	P	W	R	O	M			B	K			CHANGED OPERATING DEFINITION
11h	06h					W	R	O				B				CIRC UNRECOVERED ERROR
30h	03h	D	T				R		M				K			CLEANING CARTRIDGE INSTALLED
30h	07h	D	T	L		W	R	O	M	A	E	B	K	V	F	CLEANING FAILURE
30h	0Ah	D	T			W	R	O		A	E	B	K			CLEANING REQUEST REJECTED
00h	17h	D	T	L		W	R	O	M	A	E	B	K	V	F	CLEANING REQUESTED
30h	13h								M							CLEANING VOLUME EXPIRED
4Ah	00h	D	T	L	P	W	R	O	M	A	E	B	K	V	F	COMMAND PHASE ERROR
2Ch	00h	D	T	L	P	W	R	O	M	A	E	B	K	V	F	COMMAND SEQUENCE ERROR
6Eh	00h									A						COMMAND TO LOGICAL UNIT FAILED
2Fh	00h	D	T	L	P	W	R	O	M	A	E	B	K	V	F	COMMANDS CLEARED BY ANOTHER INITIATOR
2Fh	02h	D	T	L	P	W	R	O	M	A	E	B	K	V	F	COMMANDS CLEARED BY DEVICE SERVER
2Fh	01h	D														COMMANDS CLEARED BY POWER LOSS NOTIFICATION
3Fh	04h	D	T			W	R	O	M	A	E	B	K			COMPONENT DEVICE ATTACHED
0Ch	04h	D	T			W		O				B				COMPRESSION CHECK MISCOMPARE ERROR
27h	06h						R								F	CONDITIONAL WRITE PROTECT
67h	00h									A						CONFIGURATION FAILURE
67h	01h									A						CONFIGURATION OF INCAPABLE LOGICAL UNITS FAILED

A numeric ordered listing of the ASC and ASCQ assignments is provided in D.2.

Table 44 — ASC and ASCQ assignments (part 3 of 17)

D – Direct Access Block Device (SBC-3)
. T – Sequential Access Device (SSC-4)
. L – Printer Device (SSC)
. P – Processor Device (SPC-2)
. . W – Write Once Block Device (SBC)
. . R – C/DVD Device (MMC-6)
. . O – Optical Memory Block Device (SBC)
. . . M – Media Changer Device (SMC-3)
. . . A – Storage Array Device (SCC-2)
. . . E – SCSI Enclosure Services device (SES-3)
. . . . B – Simplified Direct-Access (Reduced Block) device (RBC)
. . . . K – Optical Card Reader/Writer device (OCRW)
. . . . V – Automation/Device Interface device (ADC-3)
. . . . . F – Object-based Storage Device (OSD-2)

Device Column key
blank = code not used
not blank = code used

ASC	ASCQ	D	T	L	P	W	R	O	M	A	E	B	K	V	F	Description
6Fh	07h						R									CONFLICT IN BINDING NONCE RECORDING
00h	1Eh	D	T				R		M	A	E	B	K	V		CONFLICTING SA CREATION REQUEST
4Bh	07h	D	T		P	W	R	O	M	A	E	B	K		F	CONNECTION LOST
5Dh	25h	D										B				CONTROLLER IMPENDING FAILURE ACCESS TIMES TOO HIGH
5Dh	27h	D										B				CONTROLLER IMPENDING FAILURE CHANNEL PARAMETRICS
5Dh	28h	D										B				CONTROLLER IMPENDING FAILURE CONTROLLER DETECTED
5Dh	22h	D										B				CONTROLLER IMPENDING FAILURE DATA ERROR RATE TOO HIGH
5Dh	2Ch	D										B				CONTROLLER IMPENDING FAILURE DRIVE CALIBRATION RETRY COUNT
5Dh	21h	D										B				CONTROLLER IMPENDING FAILURE DRIVE ERROR RATE TOO HIGH
5Dh	20h	D										B				CONTROLLER IMPENDING FAILURE GENERAL HARD DRIVE FAILURE
5Dh	23h	D										B				CONTROLLER IMPENDING FAILURE SEEK ERROR RATE TOO HIGH
5Dh	2Ah	D										B				CONTROLLER IMPENDING FAILURE SEEK TIME PERFORMANCE
5Dh	2Bh	D										B				CONTROLLER IMPENDING FAILURE SPIN-UP RETRY COUNT
5Dh	26h	D										B				CONTROLLER IMPENDING FAILURE START UNIT TIMES TOO HIGH
5Dh	29h	D										B				CONTROLLER IMPENDING FAILURE THROUGHPUT PERFORMANCE
5Dh	24h	D										B				CONTROLLER IMPENDING FAILURE TOO MANY BLOCK REASSIGNS
2Bh	00h	D	T	L	P	W	R	O					K			COPY CANNOT EXECUTE SINCE HOST CANNOT DISCONNECT
6Fh	00h						R									COPY PROTECTION KEY EXCHANGE FAILURE - AUTHENTICATION FAILURE
6Fh	02h						R									COPY PROTECTION KEY EXCHANGE FAILURE - KEY NOT ESTABLISHED
6Fh	01h						R									COPY PROTECTION KEY EXCHANGE FAILURE - KEY NOT PRESENT
26h	0Dh	D	T	L	P	W	R	O					K			COPY SEGMENT GRANULARITY VIOLATION
0Dh	05h	D	T	L	P	W	R	O		A			K			COPY TARGET DEVICE DATA OVERRUN
0Dh	04h	D	T	L	P	W	R	O		A			K			COPY TARGET DEVICE DATA UNDERRUN
0Dh	02h	D	T	L	P	W	R	O		A			K			COPY TARGET DEVICE NOT REACHABLE
67h	07h									A						CREATION OF LOGICAL UNIT FAILED
74h	04h		T													CRYPTOGRAPHIC INTEGRITY VALIDATION FAILED
73h	10h						R									CURRENT POWER CALIBRATION AREA ALMOST FULL
73h	11h						R									CURRENT POWER CALIBRATION AREA IS FULL
2Ch	04h						R									CURRENT PROGRAM AREA IS EMPTY
2Ch	03h						R									CURRENT PROGRAM AREA IS NOT EMPTY
30h	09h						R									CURRENT SESSION NOT FIXATED FOR APPEND
5Dh	35h	D										B				DATA CHANNEL IMPENDING FAILURE ACCESS TIMES TOO HIGH
5Dh	37h	D										B				DATA CHANNEL IMPENDING FAILURE CHANNEL PARAMETRICS
5Dh	38h	D										B				DATA CHANNEL IMPENDING FAILURE CONTROLLER DETECTED
5Dh	32h	D										B				DATA CHANNEL IMPENDING FAILURE DATA ERROR RATE TOO HIGH
5Dh	3Ch	D										B				DATA CHANNEL IMPENDING FAILURE DRIVE CALIBRATION RETRY COUNT
5Dh	31h	D										B				DATA CHANNEL IMPENDING FAILURE DRIVE ERROR RATE TOO HIGH
5Dh	30h	D										B				DATA CHANNEL IMPENDING FAILURE GENERAL HARD DRIVE FAILURE
5Dh	33h	D										B				DATA CHANNEL IMPENDING FAILURE SEEK ERROR RATE TOO HIGH
5Dh	3Ah	D										B				DATA CHANNEL IMPENDING FAILURE SEEK TIME PERFORMANCE

A numeric ordered listing of the ASC and ASCQ assignments is provided in D.2.

Table 44 — ASC and ASCQ assignments (part 4 of 17)

```
D – Direct Access Block Device (SBC-3)
. T – Sequential Access Device (SSC-4)
.   L – Printer Device (SSC)
.     P – Processor Device (SPC-2)
.     . W – Write Once Block Device (SBC)
.     .   R – C/DVD Device (MMC-6)
.     .     O – Optical Memory Block Device (SBC)
.     .     . M – Media Changer Device (SMC-3)
.     .     .   A – Storage Array Device (SCC-2)
.     .     .     E – SCSI Enclosure Services device (SES-3)
.     .     .     . B – Simplified Direct-Access (Reduced Block) device (RBC)
.     .     .     .   K – Optical Card Reader/Writer device (OCRW)
.     .     .     .     V – Automation/Device Interface device (ADC-3)
.     .     .     .     . F – Object-based Storage Device (OSD-2)
.     .     .     .     .
```

Device Column key
blank = code not used
not blank = code used

ASC	ASCQ	DTLPWROMAEBKVF	Description
5Dh	3Bh	D B	DATA CHANNEL IMPENDING FAILURE SPIN-UP RETRY COUNT
5Dh	36h	D B	DATA CHANNEL IMPENDING FAILURE START UNIT TIMES TOO HIGH
5Dh	39h	D B	DATA CHANNEL IMPENDING FAILURE THROUGHPUT PERFORMANCE
5Dh	34h	D B	DATA CHANNEL IMPENDING FAILURE TOO MANY BLOCK REASSIGNS
55h	0Ah	M	DATA CURRENTLY UNAVAILABLE
26h	05h	DTLPWRO A BK	DATA DECRYPTION ERROR
26h	10h	T	DATA DECRYPTION KEY FAIL LIMIT REACHED
2Ah	0Dh	T	DATA ENCRYPTION CAPABILITIES CHANGED
74h	21h	T	DATA ENCRYPTION CONFIGURATION PREVENTED
2Ah	13h	T	DATA ENCRYPTION KEY INSTANCE COUNTER HAS CHANGED
2Ah	11h	T	DATA ENCRYPTION PARAMETERS CHANGED BY ANOTHER I_T NEXUS
2Ah	12h	T	DATA ENCRYPTION PARAMETERS CHANGED BY VENDOR SPECIFIC EVENT
0Ch	05h	DT W O B	DATA EXPANSION OCCURRED DURING COMPRESSION
69h	00h	A	DATA LOSS ON LOGICAL UNIT
4Bh	05h	DT PWROMAEBK	DATA OFFSET ERROR
41h	00h	D	DATA PATH FAILURE (SHOULD USE 40 NN)
47h	01h	DTLPWROMAEBKVF	DATA PHASE CRC ERROR DETECTED
4Bh	00h	DTLPWROMAEBKVF	DATA PHASE ERROR
11h	07h	W O B	DATA RE-SYNCHRONIZATION ERROR
16h	03h	D W O BK	DATA SYNC ERROR - DATA AUTO-REALLOCATED
16h	01h	D W O BK	DATA SYNC ERROR - DATA REWRITTEN
16h	04h	D W O BK	DATA SYNC ERROR - RECOMMEND REASSIGNMENT
16h	02h	D W O BK	DATA SYNC ERROR - RECOMMEND REWRITE
16h	00h	D W O BK	DATA SYNCHRONIZATION MARK ERROR
3Bh	1Bh	M	DATA TRANSFER DEVICE INSERTED
3Bh	1Ah	M	DATA TRANSFER DEVICE REMOVED
11h	0Dh	DT WRO B	DE-COMPRESSION CRC ERROR
71h	00h	T	DECOMPRESSION EXCEPTION LONG ALGORITHM ID
70h	NNh	T	DECOMPRESSION EXCEPTION SHORT ALGORITHM ID OF NN
19h	00h	D O K	DEFECT LIST ERROR
19h	03h	D O K	DEFECT LIST ERROR IN GROWN LIST
19h	02h	D O K	DEFECT LIST ERROR IN PRIMARY LIST
19h	01h	D O K	DEFECT LIST NOT AVAILABLE
1Ch	00h	D O BK	DEFECT LIST NOT FOUND
32h	01h	D W O BK	DEFECT LIST UPDATE FAILURE
0Ch	0Fh	R	DEFECTS IN ERROR WINDOW
3Fh	05h	DT WROMAEBK	DEVICE IDENTIFIER CHANGED
29h	04h	DTLPWROMAEBKVF	DEVICE INTERNAL RESET
40h	NNh	DTLPWROMAEBKVF	DIAGNOSTIC FAILURE ON COMPONENT NN (80h-FFh)
74h	08h	DT R M E VF	DIGITAL SIGNATURE VALIDATION FAILURE

A numeric ordered listing of the ASC and ASCQ assignments is provided in D.2.

Table 44 — ASC and ASCQ assignments (part 5 of 17)

D – Direct Access Block Device (SBC-3)
. T – Sequential Access Device (SSC-4)
. L – Printer Device (SSC)
. P – Processor Device (SPC-2)
. . W – Write Once Block Device (SBC)
. . R – C/DVD Device (MMC-6)
. . O – Optical Memory Block Device (SBC)
. . . M – Media Changer Device (SMC-3)
. . . A – Storage Array Device (SCC-2)
. . . E – SCSI Enclosure Services device (SES-3)
. . . . B – Simplified Direct-Access (Reduced Block) device (RBC)
. . . . K – Optical Card Reader/Writer device (OCRW)
. . . . V – Automation/Device Interface device (ADC-3)
. . . . . F – Object-based Storage Device (OSD-2)

Device Column key
blank = code not used
not blank = code used

ASC	ASCQ	D	T	L	P	W	R	O	M	A	E	B	K	V	F	Description
66h	02h															DOCUMENT JAM IN AUTOMATIC DOCUMENT FEEDER
66h	03h															DOCUMENT MISS FEED AUTOMATIC IN DOCUMENT FEEDER
6Fh	05h						R									DRIVE REGION MUST BE PERMANENT/REGION RESET COUNT ERROR
53h	07h								M							DUPLICATE VOLUME IDENTIFIER
3Fh	0Fh	D	T	L	P	W	R	O	M	A	E	B	K	V	F	ECHO BUFFER OVERWRITTEN
3Bh	18h								M							ELEMENT DISABLED
3Bh	19h								M							ELEMENT ENABLED
53h	08h								M							ELEMENT STATUS UNKNOWN
72h	04h						R									EMPTY OR PARTIALLY WRITTEN RESERVED TRACK
34h	00h	D	T	L	P	W	R	O	M	A	E	B	K	V	F	ENCLOSURE FAILURE
35h	05h	D	T	L		W	R	O	M	A	E	B	K	V	F	ENCLOSURE SERVICES CHECKSUM ERROR
35h	00h	D	T	L	P	W	R	O	M	A	E	B	K	V	F	ENCLOSURE SERVICES FAILURE
35h	03h	D	T	L	P	W	R	O	M	A	E	B	K	V	F	ENCLOSURE SERVICES TRANSFER FAILURE
35h	04h	D	T	L	P	W	R	O	M	A	E	B	K	V	F	ENCLOSURE SERVICES TRANSFER REFUSED
35h	02h	D	T	L	P	W	R	O	M	A	E	B	K	V	F	ENCLOSURE SERVICES UNAVAILABLE
74h	0Ah		T													ENCRYPTED BLOCK NOT RAW READ ENABLED
74h	0Dh		T													ENCRYPTION ALGORITHM DISABLED
74h	09h		T													ENCRYPTION MODE MISMATCH ON READ
74h	07h		T													ENCRYPTION PARAMETERS NOT USEABLE
3Bh	0Fh						R									END OF MEDIUM REACHED
63h	00h						R									END OF USER AREA ENCOUNTERED ON THIS TRACK
00h	05h		T	L												END-OF-DATA DETECTED
14h	03h		T													END-OF-DATA NOT FOUND
00h	02h		T													END-OF-PARTITION/MEDIUM DETECTED
51h	00h		T				R	O								ERASE FAILURE
51h	01h						R									ERASE FAILURE - INCOMPLETE ERASE OPERATION DETECTED
00h	18h		T													ERASE OPERATION IN PROGRESS
74h	05h		T													ERROR DECRYPTING DATA
0Dh	00h	D	T	L	P	W	R	O		A			K			ERROR DETECTED BY THIRD PARTY TEMPORARY INITIATOR
2Ah	0Ah	D	T													ERROR HISTORY I_T NEXUS CLEARED
2Ah	0Bh	D	T													ERROR HISTORY SNAPSHOT RELEASED
0Ah	00h	D	T	L	P	W	R	O	M	A	E	B	K	V	F	ERROR LOG OVERFLOW
11h	10h						R									ERROR READING ISRC NUMBER
11h	0Fh						R									ERROR READING UPC/EAN NUMBER
2Ah	0Ch														F	ERROR RECOVERY ATTRIBUTES HAVE CHANGED
11h	02h	D	T			W	R	O				B	K			ERROR TOO LONG TO CORRECT
38h	06h											B				ESN - DEVICE BUSY CLASS EVENT
38h	04h											B				ESN - MEDIA CLASS EVENT
38h	02h											B				ESN - POWER MANAGEMENT CLASS EVENT
38h	00h											B				EVENT STATUS NOTIFICATION

A numeric ordered listing of the ASC and ASCQ assignments is provided in D.2.

Table 44 — ASC and ASCQ assignments (part 6 of 17)

D – Direct Access Block Device (SBC-3)
. T – Sequential Access Device (SSC-4)
. L – Printer Device (SSC)
. P – Processor Device (SPC-2)
. . W – Write Once Block Device (SBC)
. . R – C/DVD Device (MMC-6)
. . O – Optical Memory Block Device (SBC)
. . . M – Media Changer Device (SMC-3)
. . . A – Storage Array Device (SCC-2)
. . . E – SCSI Enclosure Services device (SES-3)
. . . . B – Simplified Direct-Access (Reduced Block) device (RBC)
. . . . K – Optical Card Reader/Writer device (OCRW)
. . . . V – Automation/Device Interface device (ADC-3)
. . . . . F – Object-based Storage Device (OSD-2)

Device Column key
blank = code not used
not blank = code used

ASC	ASCQ	D	T	L	P	W	R	O	M	A	E	B	K	V	F	Description
03h	02h		T													EXCESSIVE WRITE ERRORS
67h	04h									A						EXCHANGE OF LOGICAL UNIT FAILED
74h	6Fh		T													EXTERNAL DATA ENCRYPTION CONTROL ERROR
74h	6Eh		T													EXTERNAL DATA ENCRYPTION CONTROL TIMEOUT
74h	61h													V		EXTERNAL DATA ENCRYPTION KEY MANAGER ACCESS ERROR
74h	62h													V		EXTERNAL DATA ENCRYPTION KEY MANAGER ERROR
74h	63h													V		EXTERNAL DATA ENCRYPTION KEY NOT FOUND
74h	64h													V		EXTERNAL DATA ENCRYPTION REQUEST NOT AUTHORIZED
3Bh	07h			L												FAILED TO SENSE BOTTOM-OF-FORM
3Bh	06h			L												FAILED TO SENSE TOP-OF-FORM
5Dh	00h	D	T	L	P	W	R	O	M	A	E	B	K	V	F	FAILURE PREDICTION THRESHOLD EXCEEDED
5Dh	FFh	D	T	L	P	W	R	O	M	A	E	B	K	V	F	FAILURE PREDICTION THRESHOLD EXCEEDED (FALSE)
00h	01h		T													FILEMARK DETECTED
14h	02h		T													FILEMARK OR SETMARK NOT FOUND
5Dh	65h	D										B				FIRMWARE IMPENDING FAILURE ACCESS TIMES TOO HIGH
5Dh	67h	D										B				FIRMWARE IMPENDING FAILURE CHANNEL PARAMETRICS
5Dh	68h	D										B				FIRMWARE IMPENDING FAILURE CONTROLLER DETECTED
5Dh	62h	D										B				FIRMWARE IMPENDING FAILURE DATA ERROR RATE TOO HIGH
5Dh	6Ch	D										B				FIRMWARE IMPENDING FAILURE DRIVE CALIBRATION RETRY COUNT
5Dh	61h	D										B				FIRMWARE IMPENDING FAILURE DRIVE ERROR RATE TOO HIGH
5Dh	60h	D										B				FIRMWARE IMPENDING FAILURE GENERAL HARD DRIVE FAILURE
5Dh	63h	D										B				FIRMWARE IMPENDING FAILURE SEEK ERROR RATE TOO HIGH
5Dh	6Ah	D										B				FIRMWARE IMPENDING FAILURE SEEK TIME PERFORMANCE
5Dh	6Bh	D										B				FIRMWARE IMPENDING FAILURE SPIN-UP RETRY COUNT
5Dh	66h	D										B				FIRMWARE IMPENDING FAILURE START UNIT TIMES TOO HIGH
5Dh	69h	D										B				FIRMWARE IMPENDING FAILURE THROUGHPUT PERFORMANCE
5Dh	64h	D										B				FIRMWARE IMPENDING FAILURE TOO MANY BLOCK REASSIGNS
09h	02h					W	R	O					K			FOCUS SERVO FAILURE
31h	01h	D		L			R	O				B				FORMAT COMMAND FAILED
28h	02h						R									FORMAT-LAYER MAY HAVE CHANGED
58h	00h							O								GENERATION DOES NOT EXIST
1Ch	02h	D						O				B	K			GROWN DEFECT LIST NOT FOUND
5Dh	15h	D										B				HARDWARE IMPENDING FAILURE ACCESS TIMES TOO HIGH
5Dh	17h	D										B				HARDWARE IMPENDING FAILURE CHANNEL PARAMETRICS
5Dh	18h	D										B				HARDWARE IMPENDING FAILURE CONTROLLER DETECTED
5Dh	12h	D										B				HARDWARE IMPENDING FAILURE DATA ERROR RATE TOO HIGH
5Dh	1Ch	D										B				HARDWARE IMPENDING FAILURE DRIVE CALIBRATION RETRY COUNT
5Dh	11h	D										B				HARDWARE IMPENDING FAILURE DRIVE ERROR RATE TOO HIGH
5Dh	10h	D										B				HARDWARE IMPENDING FAILURE GENERAL HARD DRIVE FAILURE
5Dh	13h	D										B				HARDWARE IMPENDING FAILURE SEEK ERROR RATE TOO HIGH

A numeric ordered listing of the ASC and ASCQ assignments is provided in D.2.

Table 44 — ASC and ASCQ assignments (part 7 of 17)

D – Direct Access Block Device (SBC-3)
. T – Sequential Access Device (SSC-4)
. L – Printer Device (SSC)
. P – Processor Device (SPC-2)
. . W – Write Once Block Device (SBC)
. . R – C/DVD Device (MMC-6)
. . O – Optical Memory Block Device (SBC)
. . . M – Media Changer Device (SMC-3)
. . . A – Storage Array Device (SCC-2)
. . . E – SCSI Enclosure Services device (SES-3)
. . . . B – Simplified Direct-Access (Reduced Block) device (RBC)
. . . . K – Optical Card Reader/Writer device (OCRW)
. . . . V – Automation/Device Interface device (ADC-3)
. . . . . F – Object-based Storage Device (OSD-2)

Device Column key
blank = code not used
not blank = code used

ASC	ASCQ	D	T	L	P	W	R	O	M	A	E	B	K	V	F	Description
5Dh	1Ah	D										B				HARDWARE IMPENDING FAILURE SEEK TIME PERFORMANCE
5Dh	1Bh	D										B				HARDWARE IMPENDING FAILURE SPIN-UP RETRY COUNT
5Dh	16h	D										B				HARDWARE IMPENDING FAILURE START UNIT TIMES TOO HIGH
5Dh	19h	D										B				HARDWARE IMPENDING FAILURE THROUGHPUT PERFORMANCE
5Dh	14h	D										B				HARDWARE IMPENDING FAILURE TOO MANY BLOCK REASSIGNS
27h	01h	D	T			W	R	O				B	K			HARDWARE WRITE PROTECTED
09h	04h	D	T			W	R	O				B				HEAD SELECT FAULT
00h	06h	D	T	L	P	W	R	O	M	A	E	B	K	V	F	I/O PROCESS TERMINATED
10h	00h	D				W		O				B	K			ID CRC OR ECC ERROR
5Eh	03h	D	T	L	P	W	R	O		A			K			IDLE CONDITION ACTIVATED BY COMMAND
5Eh	01h	D	T	L	P	W	R	O		A			K			IDLE CONDITION ACTIVATED BY TIMER
5Eh	06h	D	T	L	P	W	R	O		A			K			IDLE_B CONDITION ACTIVATED BY COMMAND
5Eh	05h	D	T	L	P	W	R	O		A			K			IDLE_B CONDITION ACTIVATED BY TIMER
5Eh	08h	D	T	L	P	W	R	O		A			K			IDLE_C CONDITION ACTIVATED BY COMMAND
5Eh	07h	D	T	L	P	W	R	O		A			K			IDLE_C CONDITION ACTIVATED BY TIMER
20h	0Ch		T													ILLEGAL COMMAND WHEN NOT IN APPEND-ONLY MODE
20h	06h		T													ILLEGAL COMMAND WHILE IN EXPLICIT ADDRESS MODE
20h	07h		T													ILLEGAL COMMAND WHILE IN IMPLICIT ADDRESS MODE
20h	04h		T													ILLEGAL COMMAND WHILE IN WRITE CAPABLE STATE
22h	00h	D														ILLEGAL FUNCTION (USE 20 00, 24 00, OR 26 00)
64h	00h						R									ILLEGAL MODE FOR THIS TRACK
2Ch	05h											B				ILLEGAL POWER CONDITION REQUEST
2Ah	07h	D	T	L	P	W	R	O	M	A	E	B	K	V	F	IMPLICIT ASYMMETRIC ACCESS STATE TRANSITION FAILED
28h	01h	D	T			W	R	O	M			B				IMPORT OR EXPORT ELEMENT ACCESSED
28h	03h								M							IMPORT/EXPORT ELEMENT ACCESSED, MEDIUM CHANGED
30h	00h	D	T			W	R	O	M			B	K			INCOMPATIBLE MEDIUM INSTALLED
30h	12h								M							INCOMPATIBLE VOLUME QUALIFIER
30h	11h								M							INCOMPATIBLE VOLUME TYPE
11h	08h		T													INCOMPLETE BLOCK READ
26h	11h		T													INCOMPLETE KEY-ASSOCIATED DATA SET
0Dh	03h	D	T	L	P	W	R	O		A			K			INCORRECT COPY TARGET DEVICE TYPE
74h	03h		T													INCORRECT DATA ENCRYPTION KEY
74h	0Bh		T													INCORRECT ENCRYPTION PARAMETERS
0Eh	02h	D	T		P	W	R	O	M	A	E	B	K		F	INFORMATION UNIT TOO LONG
0Eh	01h	D	T		P	W	R	O	M	A	E	B	K		F	INFORMATION UNIT TOO SHORT
47h	03h	D	T	L	P	W	R	O	M	A	E	B	K	V	F	INFORMATION UNIT iuCRC ERROR DETECTED
6Ah	00h									A						INFORMATIONAL, REFER TO LOG
48h	00h	D	T	L	P	W	R	O	M	A	E	B	K	V	F	INITIATOR DETECTED ERROR MESSAGE RECEIVED
4Bh	06h	D	T		P	W	R	O	M	A	E	B	K			INITIATOR RESPONSE TIMEOUT
26h	0Bh	D	T	L	P	W	R	O					K			INLINE DATA LENGTH EXCEEDED

A numeric ordered listing of the ASC and ASCQ assignments is provided in D.2.

Table 44 — ASC and ASCQ assignments (part 8 of 17)

```
                D – Direct Access Block Device (SBC-3)                    Device Column key
                . T – Sequential Access Device (SSC-4)                    blank = code not used
                .   L – Printer Device (SSC)                              not blank = code used
                .     P – Processor Device (SPC-2)
                .     . W – Write Once Block Device (SBC)
                .     .   R – C/DVD Device (MMC-6)
                .     .     O – Optical Memory Block Device (SBC)
                .     .     . M – Media Changer Device (SMC-3)
                .     .     .   A – Storage Array Device (SCC-2)
                .     .     .     E – SCSI Enclosure Services device (SES-3)
                .     .     .     . B – Simplified Direct-Access (Reduced Block) device (RBC)
                .     .     .     .   K – Optical Card Reader/Writer device (OCRW)
                .     .     .     .     V – Automation/Device Interface device (ADC-3)
                .     .     .     .     . F – Object-based Storage Device (OSD-2)
                .     .     .     .     .
ASC ASCQ        DTLPWROMAEBKVF  Description
3Fh 03h         DTLPWROMAEBKVF  INQUIRY DATA HAS CHANGED
55h 05h         DT PWROMAEBK    INSUFFICIENT ACCESS CONTROL RESOURCES
6Fh 06h             R           INSUFFICIENT BLOCK COUNT FOR BINDING NONCE RECORDING
55h 0Bh         DTLPWROMAEBKVF  INSUFFICIENT POWER FOR OPERATION
55h 04h         DTLPWROMAE K    INSUFFICIENT REGISTRATION RESOURCES
55h 02h         DTLPWROMAE K    INSUFFICIENT RESERVATION RESOURCES
55h 03h         DTLPWROMAE K    INSUFFICIENT RESOURCES
2Eh 00h             R           INSUFFICIENT TIME FOR OPERATION
44h 00h         DTLPWROMAEBKVF  INTERNAL TARGET FAILURE
21h 02h             R           INVALID ADDRESS FOR WRITE
3Dh 00h         DTLPWROMAE K    INVALID BITS IN IDENTIFY MESSAGE
2Ch 02h                         INVALID COMBINATION OF WINDOWS SPECIFIED
20h 00h         DTLPWROMAEBKVF  INVALID COMMAND OPERATION CODE
26h 0Fh                       F INVALID DATA-OUT BUFFER INTEGRITY CHECK VALUE
21h 01h         DT  WROM    BK  INVALID ELEMENT ADDRESS
24h 00h         DTLPWROMAEBKVF  INVALID FIELD IN CDB
0Eh 03h         DT P R MAEBK F  INVALID FIELD IN COMMAND INFORMATION UNIT
26h 00h         DTLPWROMAEBKVF  INVALID FIELD IN PARAMETER LIST
0Eh 00h         DT PWROMAEBK F  INVALID INFORMATION UNIT
49h 00h         DTLPWROMAEBKVF  INVALID MESSAGE ERROR
26h 0Ch         DTLPWRO       K INVALID OPERATION FOR COPY SOURCE OR DESTINATION
64h 01h             R           INVALID PACKET SIZE
26h 0Eh         DT PWROMAEBK    INVALID PARAMETER WHILE PORT IS ENABLED
26h 04h         DTLPWROMAEBKVF  INVALID RELEASE OF PERSISTENT RESERVATION
74h 12h         DT    R MAEBKV  INVALID SA USAGE
4Bh 01h         DT PWROMAEBK    INVALID TARGET PORT TRANSFER TAG RECEIVED
23h 00h         DT P R      B   INVALID TOKEN OPERATION, CAUSE NOT REPORTABLE
21h 03h             R           INVALID WRITE CROSSING LAYER JUMP
24h 08h         DT    R MAEBKV  INVALID XCDB
29h 07h         DTLPWROMAEBKVF  I_T NEXUS LOSS OCCURRED
11h 05h           WRO       B   L-EC UNCORRECTABLE ERROR
60h 00h                         LAMP FAILURE
14h 07h           T             LOCATE OPERATION FAILURE
00h 19h           T             LOCATE OPERATION IN PROGRESS
5Bh 02h         DTLPWROM      K LOG COUNTER AT MAXIMUM
5Bh 00h         DTLPWROM      K LOG EXCEPTION
5Bh 03h         DTLPWROM      K LOG LIST CODES EXHAUSTED
2Ah 02h         DTL WROMAE    K LOG PARAMETERS CHANGED
21h 00h         DT  WRO     BK  LOGICAL BLOCK ADDRESS OUT OF RANGE
10h 02h         DT  W O         LOGICAL BLOCK APPLICATION TAG CHECK FAILED
```

A numeric ordered listing of the ASC and ASCQ assignments is provided in D.2.

Table 44 — ASC and ASCQ assignments (part 9 of 17)

D – Direct Access Block Device (SBC-3)
. T – Sequential Access Device (SSC-4)
. L – Printer Device (SSC)
. P – Processor Device (SPC-2)
. . W – Write Once Block Device (SBC)
. . R – C/DVD Device (MMC-6)
. . O – Optical Memory Block Device (SBC)
. . . M – Media Changer Device (SMC-3)
. . . A – Storage Array Device (SCC-2)
. . . E – SCSI Enclosure Services device (SES-3)
. . . . B – Simplified Direct-Access (Reduced Block) device (RBC)
. . . . K – Optical Card Reader/Writer device (OCRW)
. . . . V – Automation/Device Interface device (ADC-3)
. . . . . F – Object-based Storage Device (OSD-2)

Device Column key
blank = code not used
not blank = code used

ASC	ASCQ	D	T	L	P	W	R	O	M	A	E	B	K	V	F	Description
10h	01h	D	T			W		O								LOGICAL BLOCK GUARD CHECK FAILED
10h	04h		T													LOGICAL BLOCK PROTECTION ERROR ON RECOVER BUFFERED DATA
10h	05h		T													LOGICAL BLOCK PROTECTION METHOD ERROR
10h	03h	D	T			W		O								LOGICAL BLOCK REFERENCE TAG CHECK FAILED
74h	71h	D	T				R		M		E			V		LOGICAL UNIT ACCESS NOT AUTHORIZED
08h	03h	D	T				R	O	M			B	K			LOGICAL UNIT COMMUNICATION CRC ERROR (ULTRA-DMA/32)
08h	00h	D	T	L		W	R	O	M	A	E	B	K	V	F	LOGICAL UNIT COMMUNICATION FAILURE
08h	02h	D	T	L		W	R	O	M	A	E	B	K	V	F	LOGICAL UNIT COMMUNICATION PARITY ERROR
08h	01h	D	T	L		W	R	O	M	A	E	B	K	V	F	LOGICAL UNIT COMMUNICATION TIME-OUT
05h	00h	D	T	L		W	R	O	M	A	E	B	K	V	F	LOGICAL UNIT DOES NOT RESPOND TO SELECTION
4Ch	00h	D	T	L	P	W	R	O	M	A	E	B	K	V	F	LOGICAL UNIT FAILED SELF-CONFIGURATION
3Eh	03h	D	T	L	P	W	R	O	M	A	E	B	K	V	F	LOGICAL UNIT FAILED SELF-TEST
3Eh	01h	D	T	L	P	W	R	O	M	A	E	B	K	V	F	LOGICAL UNIT FAILURE
5Dh	02h						R									LOGICAL UNIT FAILURE PREDICTION THRESHOLD EXCEEDED
3Eh	00h	D	T	L	P	W	R	O	M	A	E	B	K	V	F	LOGICAL UNIT HAS NOT SELF-CONFIGURED YET
04h	01h	D	T	L	P	W	R	O	M	A	E	B	K	V	F	LOGICAL UNIT IS IN PROCESS OF BECOMING READY
04h	0Ah	D	T	L	P	W	R	O	M	A	E	B	K	V	F	LOGICAL UNIT NOT ACCESSIBLE, ASYMMETRIC ACCESS STATE TRANSITION
04h	0Bh	D	T	L	P	W	R	O	M	A	E	B	K	V	F	LOGICAL UNIT NOT ACCESSIBLE, TARGET PORT IN STANDBY STATE
04h	0Ch	D	T	L	P	W	R	O	M	A	E	B	K	V	F	LOGICAL UNIT NOT ACCESSIBLE, TARGET PORT IN UNAVAILABLE STATE
68h	00h									A						LOGICAL UNIT NOT CONFIGURED
04h	18h								M							LOGICAL UNIT NOT READY, A DOOR IS OPEN
04h	1Ch	D	T						M	A	E	B				LOGICAL UNIT NOT READY, ADDITIONAL POWER USE NOT YET GRANTED
04h	10h	D	T			W	R	O	M			B				LOGICAL UNIT NOT READY, AUXILIARY MEMORY NOT ACCESSIBLE
04h	17h								M							LOGICAL UNIT NOT READY, CALIBRATION REQUIRED
04h	00h	D	T	L	P	W	R	O	M	A	E	B	K	V	F	LOGICAL UNIT NOT READY, CAUSE NOT REPORTABLE
04h	16h								M							LOGICAL UNIT NOT READY, CONFIGURATION REQUIRED
04h	04h	D	T	L			R	O				B				LOGICAL UNIT NOT READY, FORMAT IN PROGRESS
04h	02h	D	T	L	P	W	R	O	M	A	E	B	K	V	F	LOGICAL UNIT NOT READY, INITIALIZING COMMAND REQUIRED
04h	08h						R									LOGICAL UNIT NOT READY, LONG WRITE IN PROGRESS
04h	03h	D	T	L	P	W	R	O	M	A	E	B	K	V	F	LOGICAL UNIT NOT READY, MANUAL INTERVENTION REQUIRED
04h	11h	D	T			W	R	O		A	E	B		V	F	LOGICAL UNIT NOT READY, NOTIFY (ENABLE SPINUP) REQUIRED
04h	12h								M					V		LOGICAL UNIT NOT READY, OFFLINE
04h	19h								M							LOGICAL UNIT NOT READY, OPERATING IN SEQUENTIAL MODE
04h	07h	D	T	L	P	W	R	O	M	A	E	B	K	V	F	LOGICAL UNIT NOT READY, OPERATION IN PROGRESS
04h	05h	D	T			W		O		A		B	K		F	LOGICAL UNIT NOT READY, REBUILD IN PROGRESS
04h	06h	D	T			W		O		A		B	K			LOGICAL UNIT NOT READY, RECALCULATION IN PROGRESS
04h	15h								M							LOGICAL UNIT NOT READY, ROBOTICS DISABLED
04h	13h	D	T				R		M	A	E	B	K	V		LOGICAL UNIT NOT READY, SA CREATION IN PROGRESS
04h	1Bh	D										B				LOGICAL UNIT NOT READY, SANITIZE IN PROGRESS

A numeric ordered listing of the ASC and ASCQ assignments is provided in D.2.

Table 44 — ASC and ASCQ assignments (part 10 of 17)

```
                 D – Direct Access Block Device (SBC-3)                    Device Column key
                 . T – Sequential Access Device (SSC-4)                    blank = code not used
                 .   L – Printer Device (SSC)                              not blank = code used
                 .     P – Processor Device (SPC-2)
                 .     . W – Write Once Block Device (SBC)
                 .     .   R – C/DVD Device (MMC-6)
                 .     .     O – Optical Memory Block Device (SBC)
                 .     .     . M – Media Changer Device (SMC-3)
                 .     .     .   A – Storage Array Device (SCC-2)
                 .     .     .     E – SCSI Enclosure Services device (SES-3)
                 .     .     .     . B – Simplified Direct-Access (Reduced Block) device (RBC)
                 .     .     .     .   K – Optical Card Reader/Writer device (OCRW)
                 .     .     .     .     V – Automation/Device Interface device (ADC-3)
                 .     .     .     .     . F – Object-based Storage Device (OSD-2)
                 .     .     .     .     .
ASC  ASCQ        DTLPWROMAEBKVF  Description
```

ASC	ASCQ	DTLPWROMAEBKVF	Description
04h	09h	DTLPWROMAEBKVF	LOGICAL UNIT NOT READY, SELF-TEST IN PROGRESS
04h	14h	D B	LOGICAL UNIT NOT READY, SPACE ALLOCATION IN PROGRESS
04h	1Ah	DT B	LOGICAL UNIT NOT READY, START STOP UNIT COMMAND IN PROGRESS
04h	0Dh	F	LOGICAL UNIT NOT READY, STRUCTURE CHECK REQUIRED
25h	00h	DTLPWROMAEBKVF	LOGICAL UNIT NOT SUPPORTED
27h	02h	DT WRO BK	LOGICAL UNIT SOFTWARE WRITE PROTECTED
00h	1Fh	DT B	LOGICAL UNIT TRANSITIONING TO ANOTHER POWER CONDITION
3Eh	04h	DTLPWROMAEBKVF	LOGICAL UNIT UNABLE TO UPDATE SELF-TEST LOG
5Eh	00h	DTLPWRO A K	LOW POWER CONDITION ON
55h	08h	T	MAXIMUM NUMBER OF SUPPLEMENTAL DECRYPTION KEYS EXCEEDED
15h	01h	DTL WROM BK	MECHANICAL POSITIONING ERROR
3Bh	16h	R	MECHANICAL POSITIONING OR CHANGER ERROR
5Dh	01h	R B	MEDIA FAILURE PREDICTION THRESHOLD EXCEEDED
53h	00h	DTL WROM BK	MEDIA LOAD OR EJECT FAILED
6Fh	04h	R	MEDIA REGION CODE IS MISMATCHED TO LOGICAL UNIT REGION
3Fh	11h	DT WROM B	MEDIUM AUXILIARY MEMORY ACCESSIBLE
55h	09h	M	MEDIUM AUXILIARY MEMORY NOT ACCESSIBLE
3Bh	0Dh	DT WROM BK	MEDIUM DESTINATION ELEMENT FULL
31h	00h	DT WRO BK	MEDIUM FORMAT CORRUPTED
3Fh	10h	DT WROM B	MEDIUM LOADABLE
3Bh	13h	DT WROM BK	MEDIUM MAGAZINE INSERTED
3Bh	14h	DT WROM BK	MEDIUM MAGAZINE LOCKED
3Bh	11h	DT WROM BK	MEDIUM MAGAZINE NOT ACCESSIBLE
3Bh	12h	DT WROM BK	MEDIUM MAGAZINE REMOVED
3Bh	15h	DT WROM BK	MEDIUM MAGAZINE UNLOCKED
30h	10h	R	MEDIUM NOT FORMATTED
3Ah	00h	DTL WROM BK	MEDIUM NOT PRESENT
3Ah	03h	DT WROM B	MEDIUM NOT PRESENT - LOADABLE
3Ah	04h	DT WRO B	MEDIUM NOT PRESENT - MEDIUM AUXILIARY MEMORY ACCESSIBLE
3Ah	01h	DT WROM BK	MEDIUM NOT PRESENT - TRAY CLOSED
3Ah	02h	DT WROM BK	MEDIUM NOT PRESENT - TRAY OPEN
53h	02h	DT WROM BK	MEDIUM REMOVAL PREVENTED
53h	03h	M	MEDIUM REMOVAL PREVENTED BY DATA TRANSFER ELEMENT
3Bh	0Eh	DT WROM BK	MEDIUM SOURCE ELEMENT EMPTY
53h	04h	T	MEDIUM THREAD OR UNTHREAD FAILURE
43h	00h	DTLPWROMAEBKVF	MESSAGE ERROR
3Fh	01h	DTLPWROMAEBKVF	MICROCODE HAS BEEN CHANGED
1Dh	00h	DT WRO BK	MISCOMPARE DURING VERIFY OPERATION
1Dh	01h	D B	MISCOMPARE VERIFY OF UNMAPPED LBA
11h	0Ah	DT O BK	MISCORRECTED ERROR

A numeric ordered listing of the ASC and ASCQ assignments is provided in D.2.

Table 44 — ASC and ASCQ assignments (part 11 of 17)

```
                D – Direct Access Block Device (SBC-3)                    Device Column key
                . T – Sequential Access Device (SSC-4)                    blank = code not used
                .   L – Printer Device (SSC)                              not blank = code used
                .     P – Processor Device (SPC-2)
                .     . W – Write Once Block Device (SBC)
                .     .   R – C/DVD Device (MMC-6)
                .     .     O – Optical Memory Block Device (SBC)
                .     .     . M – Media Changer Device (SMC-3)
                .     .     .   A – Storage Array Device (SCC-2)
                .     .     .     E – SCSI Enclosure Services device (SES-3)
                .     .     .     . B – Simplified Direct-Access (Reduced Block) device (RBC)
                .     .     .     .   K – Optical Card Reader/Writer device (OCRW)
                .     .     .     .     V – Automation/Device Interface device (ADC-3)
                .     .     .     .     . F – Object-based Storage Device (OSD-2)
                .     .     .     .     .
ASC  ASCQ  DTLPWROMAEBKVF  Description
2Ah  01h   DTL  WROMAEBKVF  MODE PARAMETERS CHANGED
67h  03h                A   MODIFICATION OF LOGICAL UNIT FAILED
69h  01h                A   MULTIPLE LOGICAL UNIT FAILURES
07h  00h   DTL  WROM   BK   MULTIPLE PERIPHERAL DEVICES SELECTED
11h  03h   DT   W O    BK   MULTIPLE READ ERRORS
67h  09h                A   MULTIPLY ASSIGNED LOGICAL UNIT
4Bh  04h   DT  PWROMAEBK    NAK RECEIVED
00h  00h   DTLPWROMAEBKVF   NO ADDITIONAL SENSE INFORMATION
00h  15h          R         NO CURRENT AUDIO STATUS TO RETURN
32h  00h   D    W O    BK   NO DEFECT SPARE LOCATION AVAILABLE
11h  09h     T               NO GAP FOUND
01h  00h   D    W O    BK   NO INDEX/SECTOR SIGNAL
72h  07h          R         NO MORE TEST ZONE EXTENSIONS ARE ALLOWED
72h  05h          R         NO MORE TRACK RESERVATIONS ALLOWED
06h  00h   D    WROM   BK   NO REFERENCE POSITION FOUND
02h  00h   D    WRO    BK   NO SEEK COMPLETE
03h  01h     T               NO WRITE CURRENT
24h  06h                  F NONCE NOT UNIQUE
24h  07h                  F NONCE TIMESTAMP OUT OF RANGE
28h  00h   DTLPWROMAEBKVF   NOT READY TO READY CHANGE, MEDIUM MAY HAVE CHANGED
2Ch  0Bh     T               NOT RESERVED
00h  16h   DTLPWROMAEBKVF   OPERATION IN PROGRESS
5Ah  01h   DT   WROM   BK   OPERATOR MEDIUM REMOVAL REQUEST
5Ah  00h   DTLPWRO     BK   OPERATOR REQUEST OR STATE CHANGE INPUT
5Ah  03h   DT   WRO  A BK   OPERATOR SELECTED WRITE PERMIT
5Ah  02h   DT   WRO  A BK   OPERATOR SELECTED WRITE PROTECT
2Ch  0Ch   D                ORWRITE GENERATION DOES NOT MATCH
61h  02h                    OUT OF FOCUS
4Eh  00h   DTLPWROMAEBKVF   OVERLAPPED COMMANDS ATTEMPTED
2Dh  00h     T               OVERWRITE ERROR ON UPDATE IN PLACE
20h  05h     T               Obsolete
24h  02h     T               Obsolete
24h  03h     T               Obsolete
63h  01h          R         PACKET DOES NOT FIT IN AVAILABLE SPACE
3Bh  05h       L             PAPER JAM
1Ah  00h   DTLPWROMAEBKVF   PARAMETER LIST LENGTH ERROR
26h  01h   DTLPWROMAEBKVF   PARAMETER NOT SUPPORTED
26h  02h   DTLPWROMAEBKVF   PARAMETER VALUE INVALID
2Ah  00h   DTL  WROMAEBKVF  PARAMETERS CHANGED
69h  02h                A   PARITY/DATA MISMATCH
```

A numeric ordered listing of the ASC and ASCQ assignments is provided in D.2.

Table 44 — ASC and ASCQ assignments (part 12 of 17)

D – Direct Access Block Device (SBC-3)
. T – Sequential Access Device (SSC-4)
. L – Printer Device (SSC)
. P – Processor Device (SPC-2)
. . W – Write Once Block Device (SBC)
. . R – C/DVD Device (MMC-6)
. . O – Optical Memory Block Device (SBC)
. . . M – Media Changer Device (SMC-3)
. . . A – Storage Array Device (SCC-2)
. . . E – SCSI Enclosure Services device (SES-3)
. . . . B – Simplified Direct-Access (Reduced Block) device (RBC)
. . . . K – Optical Card Reader/Writer device (OCRW)
. . . . V – Automation/Device Interface device (ADC-3)
. . . . . F – Object-based Storage Device (OSD-2)
. . . . .

Device Column key
blank = code not used
not blank = code used

ASC	ASCQ	D	T	L	P	W	R	O	M	A	E	B	K	V	F	Description
1Fh	00h	D						O					K			PARTIAL DEFECT LIST TRANSFER
2Ch	0Ah														F	PARTITION OR COLLECTION CONTAINS USER OBJECTS
03h	00h	D	T	L		W		O				B	K			PERIPHERAL DEVICE WRITE FAULT
27h	05h		T				R									PERMANENT WRITE PROTECT
2Ch	06h						R									PERSISTENT PREVENT CONFLICT
27h	04h		T				R									PERSISTENT WRITE PROTECT
47h	06h	D	T						M	A	E	B	K	V	F	PHY TEST FUNCTION IN PROGRESS
50h	02h		T													POSITION ERROR RELATED TO TIMING
3Bh	0Ch		T													POSITION PAST BEGINNING OF MEDIUM
3Bh	0Bh															POSITION PAST END OF MEDIUM
15h	02h	D	T			W	R	O				B	K			POSITIONING ERROR DETECTED BY READ OF MEDIUM
73h	01h						R									POWER CALIBRATION AREA ALMOST FULL
73h	03h						R									POWER CALIBRATION AREA ERROR
73h	02h						R									POWER CALIBRATION AREA IS FULL
29h	01h	D	T	L	P	W	R	O	M	A	E	B	K	V	F	POWER ON OCCURRED
29h	00h	D	T	L	P	W	R	O	M	A	E	B	K	V	F	POWER ON, RESET, OR BUS DEVICE RESET OCCURRED
5Eh	41h											B				POWER STATE CHANGE TO ACTIVE
5Eh	47h											B	K			POWER STATE CHANGE TO DEVICE CONTROL
5Eh	42h											B				POWER STATE CHANGE TO IDLE
5Eh	45h											B				POWER STATE CHANGE TO SLEEP
5Eh	43h											B				POWER STATE CHANGE TO STANDBY
42h	00h	D														POWER-ON OR SELF-TEST FAILURE (SHOULD USE 40 NN)
2Ch	07h	D	T	L	P	W	R	O	M	A	E	B	K	V	F	PREVIOUS BUSY STATUS
2Ch	09h	D	T	L	P	W	R	O	M		E	B	K	V	F	PREVIOUS RESERVATION CONFLICT STATUS
2Ch	08h	D	T	L	P	W	R	O	M	A	E	B	K	V	F	PREVIOUS TASK SET FULL STATUS
1Ch	01h	D						O				B	K			PRIMARY DEFECT LIST NOT FOUND
2Ah	08h	D	T			W	R	O	M	A	E	B	K	V	F	PRIORITY CHANGED
73h	05h						R									PROGRAM MEMORY AREA IS FULL
73h	04h						R									PROGRAM MEMORY AREA UPDATE FAILURE
00h	07h		T													PROGRAMMABLE EARLY WARNING DETECTED
47h	05h	D	T	L	P	W	R	O	M	A	E	B	K	V	F	PROTOCOL SERVICE CRC ERROR
55h	07h														F	QUOTA ERROR
40h	00h	D														RAM FAILURE (SHOULD USE 40 NN)
15h	00h	D	T	L		W	R	O	M			B	K			RANDOM POSITIONING ERROR
73h	17h						R									RDZ IS FULL
11h	13h	D	T	L	P	W	R	O		A	E	B	K	V	F	READ ERROR - FAILED RETRANSMISSION REQUEST
11h	14h	D														READ ERROR - LBA MARKED BAD BY APPLICATION CLIENT
11h	11h						R									READ ERROR - LOSS OF STREAMING
6Fh	03h						R									READ OF SCRAMBLED SECTOR WITHOUT AUTHENTICATION
3Bh	0Ah															READ PAST BEGINNING OF MEDIUM

A numeric ordered listing of the ASC and ASCQ assignments is provided in D.2.

Table 44 — ASC and ASCQ assignments (part 13 of 17)

D – Direct Access Block Device (SBC-3)
. T – Sequential Access Device (SSC-4)
. L – Printer Device (SSC)
. P – Processor Device (SPC-2)
. . W – Write Once Block Device (SBC)
. . R – C/DVD Device (MMC-6)
. . O – Optical Memory Block Device (SBC)
. . . M – Media Changer Device (SMC-3)
. . . A – Storage Array Device (SCC-2)
. . . E – SCSI Enclosure Services device (SES-3)
. . . . B – Simplified Direct-Access (Reduced Block) device (RBC)
. . . . K – Optical Card Reader/Writer device (OCRW)
. . . . V – Automation/Device Interface device (ADC-3)
. . . . . F – Object-based Storage Device (OSD-2)

Device Column key
blank = code not used
not blank = code used

ASC	ASCQ	D	T	L	P	W	R	O	M	A	E	B	K	V	F	Description
3Bh	09h															READ PAST END OF MEDIUM
3Bh	17h														F	READ PAST END OF USER OBJECT
11h	01h	D	T			W	R	O				B	K			READ RETRIES EXHAUSTED
6Ch	00h									A						REBUILD FAILURE OCCURRED
6Dh	00h									A						RECALCULATE FAILURE OCCURRED
14h	01h	D	T			W	R	O				B	K			RECORD NOT FOUND
14h	06h	D	T			W		O				B	K			RECORD NOT FOUND - DATA AUTO-REALLOCATED
14h	05h	D	T			W		O				B	K			RECORD NOT FOUND - RECOMMEND REASSIGNMENT
14h	00h	D	T	L		W	R	O				B	K			RECORDED ENTITY NOT FOUND
18h	02h	D				W	R	O				B	K			RECOVERED DATA - DATA AUTO-REALLOCATED
18h	05h	D				W	R	O				B	K			RECOVERED DATA - RECOMMEND REASSIGNMENT
18h	06h	D				W	R	O				B	K			RECOVERED DATA - RECOMMEND REWRITE
17h	05h	D				W	R	O				B	K			RECOVERED DATA USING PREVIOUS SECTOR ID
18h	03h						R									RECOVERED DATA WITH CIRC
18h	07h	D				W		O				B	K			RECOVERED DATA WITH ECC - DATA REWRITTEN
18h	01h	D				W	R	O				B	K			RECOVERED DATA WITH ERROR CORR. & RETRIES APPLIED
18h	00h	D	T			W	R	O				B	K			RECOVERED DATA WITH ERROR CORRECTION APPLIED
18h	04h						R									RECOVERED DATA WITH L-EC
18h	08h						R									RECOVERED DATA WITH LINKING
17h	03h	D	T			W	R	O				B	K			RECOVERED DATA WITH NEGATIVE HEAD OFFSET
17h	00h	D	T			W	R	O				B	K			RECOVERED DATA WITH NO ERROR CORRECTION APPLIED
17h	02h	D	T			W	R	O				B	K			RECOVERED DATA WITH POSITIVE HEAD OFFSET
17h	01h	D	T			W	R	O				B	K			RECOVERED DATA WITH RETRIES
17h	04h					W	R	O				B				RECOVERED DATA WITH RETRIES AND/OR CIRC APPLIED
17h	06h	D				W		O				B	K			RECOVERED DATA WITHOUT ECC - DATA AUTO-REALLOCATED
17h	09h	D				W	R	O				B	K			RECOVERED DATA WITHOUT ECC - DATA REWRITTEN
17h	07h	D				W	R	O				B	K			RECOVERED DATA WITHOUT ECC - RECOMMEND REASSIGNMENT
17h	08h	D				W	R	O				B	K			RECOVERED DATA WITHOUT ECC - RECOMMEND REWRITE
1Eh	00h	D				W		O				B	K			RECOVERED ID WITH ECC CORRECTION
3Fh	06h	D	T			W	R	O	M	A	E	B				REDUNDANCY GROUP CREATED OR MODIFIED
3Fh	07h	D	T			W	R	O	M	A	E	B				REDUNDANCY GROUP DELETED
6Bh	01h									A						REDUNDANCY LEVEL GOT BETTER
6Bh	02h									A						REDUNDANCY LEVEL GOT WORSE
2Ah	05h	D	T	L	P	W	R	O	M	A	E					REGISTRATIONS PREEMPTED
67h	05h									A						REMOVE OF LOGICAL UNIT FAILED
3Fh	0Eh	D	T	L	P	W	R	O	M	A	E					REPORTED LUNS DATA HAS CHANGED
3Bh	08h		T													REPOSITION ERROR
2Ah	03h	D	T	L	P	W	R	O	M	A	E		K			RESERVATIONS PREEMPTED
2Ah	04h	D	T	L	P	W	R	O	M	A	E					RESERVATIONS RELEASED
00h	1Ah		T													REWIND OPERATION IN PROGRESS

A numeric ordered listing of the ASC and ASCQ assignments is provided in D.2.

Table 44 — ASC and ASCQ assignments (part 14 of 17)

D – Direct Access Block Device (SBC-3)
. T – Sequential Access Device (SSC-4)
. L – Printer Device (SSC)
. P – Processor Device (SPC-2)
. . W – Write Once Block Device (SBC)
. . R – C/DVD Device (MMC-6)
. . O – Optical Memory Block Device (SBC)
. . . M – Media Changer Device (SMC-3)
. . . A – Storage Array Device (SCC-2)
. . . E – SCSI Enclosure Services device (SES-3)
. . . . B – Simplified Direct-Access (Reduced Block) device (RBC)
. . . . K – Optical Card Reader/Writer device (OCRW)
. . . . V – Automation/Device Interface device (ADC-3)
. . . . . F – Object-based Storage Device (OSD-2)

Device Column key
blank = code not used
not blank = code used

ASC	ASCQ	D	T	L	P	W	R	O	M	A	E	B	K	V	F	Description
36h	00h			L												RIBBON, INK, OR TONER FAILURE
73h	06h						R									RMA/PMA IS ALMOST FULL
72h	06h						R									RMZ EXTENSION IS NOT ALLOWED
37h	00h	D	T	L		W	R	O	M	A	E	B	K	V	F	ROUNDED PARAMETER
5Ch	00h	D						O								RPL STATUS CHANGE
2Ah	14h	D	T				R		M	A	E	B	K	V		SA CREATION CAPABILITIES DATA HAS CHANGED
74h	30h	D	T				R		M	A	E	B	K	V		SA CREATION PARAMETER NOT SUPPORTED
74h	10h	D	T				R		M	A	E	B	K	V		SA CREATION PARAMETER VALUE INVALID
74h	11h	D	T				R		M	A	E	B	K	V		SA CREATION PARAMETER VALUE REJECTED
31h	03h	D										B				SANITIZE COMMAND FAILED
39h	00h	D	T	L		W	R	O	M	A	E		K			SAVING PARAMETERS NOT SUPPORTED
62h	00h															SCAN HEAD POSITIONING ERROR
29h	02h	D	T	L	P	W	R	O	M	A	E	B	K	V	F	SCSI BUS RESET OCCURRED
47h	00h	D	T	L	P	W	R	O	M	A	E	B	K	V	F	SCSI PARITY ERROR
47h	02h	D	T	L	P	W	R	O	M	A	E	B	K	V	F	SCSI PARITY ERROR DETECTED DURING ST DATA PHASE
54h	00h				P											SCSI TO HOST SYSTEM INTERFACE FAILURE
24h	04h														F	SECURITY AUDIT VALUE FROZEN
74h	79h	D														SECURITY CONFLICT IN TRANSLATED DEVICE
74h	00h		T													SECURITY ERROR
24h	05h														F	SECURITY WORKING KEY FROZEN
45h	00h	D	T	L	P	W	R	O	M	A	E	B	K	V	F	SELECT OR RESELECT FAILURE
3Bh	00h		T	L												SEQUENTIAL POSITIONING ERROR
5Dh	45h	D										B				SERVO IMPENDING FAILURE ACCESS TIMES TOO HIGH
5Dh	47h	D										B				SERVO IMPENDING FAILURE CHANNEL PARAMETRICS
5Dh	48h	D										B				SERVO IMPENDING FAILURE CONTROLLER DETECTED
5Dh	42h	D										B				SERVO IMPENDING FAILURE DATA ERROR RATE TOO HIGH
5Dh	4Ch	D										B				SERVO IMPENDING FAILURE DRIVE CALIBRATION RETRY COUNT
5Dh	41h	D										B				SERVO IMPENDING FAILURE DRIVE ERROR RATE TOO HIGH
5Dh	40h	D										B				SERVO IMPENDING FAILURE GENERAL HARD DRIVE FAILURE
5Dh	43h	D										B				SERVO IMPENDING FAILURE SEEK ERROR RATE TOO HIGH
5Dh	4Ah	D										B				SERVO IMPENDING FAILURE SEEK TIME PERFORMANCE
5Dh	4Bh	D										B				SERVO IMPENDING FAILURE SPIN-UP RETRY COUNT
5Dh	46h	D										B				SERVO IMPENDING FAILURE START UNIT TIMES TOO HIGH
5Dh	49h	D										B				SERVO IMPENDING FAILURE THROUGHPUT PERFORMANCE
5Dh	44h	D										B				SERVO IMPENDING FAILURE TOO MANY BLOCK REASSIGNS
72h	00h						R									SESSION FIXATION ERROR
72h	03h						R									SESSION FIXATION ERROR - INCOMPLETE TRACK IN SESSION
72h	01h						R									SESSION FIXATION ERROR WRITING LEAD-IN
72h	02h						R									SESSION FIXATION ERROR WRITING LEAD-OUT
00h	1Bh		T													SET CAPACITY OPERATION IN PROGRESS

A numeric ordered listing of the ASC and ASCQ assignments is provided in D.2.

Table 44 — ASC and ASCQ assignments (part 15 of 17)

```
                D – Direct Access Block Device (SBC-3)                    Device Column key
                . T – Sequential Access Device (SSC-4)                    blank = code not used
                .   L – Printer Device (SSC)                              not blank = code used
                .     P – Processor Device (SPC-2)
                .     . W – Write Once Block Device (SBC)
                .     .   R – C/DVD Device (MMC-6)
                .     .     O – Optical Memory Block Device (SBC)
                .     .     . M – Media Changer Device (SMC-3)
                .     .     .   A – Storage Array Device (SCC-2)
                .     .     .     E – SCSI Enclosure Services device (SES-3)
                .     .     .     . B – Simplified Direct-Access (Reduced Block) device (RBC)
                .     .     .     .   K – Optical Card Reader/Writer device (OCRW)
                .     .     .     .     V – Automation/Device Interface device (ADC-3)
                .     .     .     .     . F – Object-based Storage Device (OSD-2)
                .     .     .     .     .
ASC  ASCQ  DTLPWROMAEBKVF  Description
```

ASC	ASCQ	DTLPWROMAEBKVF	Description
67h	0Ah	DTLPWROMAEBKVF	SET TARGET PORT GROUPS COMMAND FAILED
00h	03h	T	SETMARK DETECTED
3Bh	04h	L	SLEW FAILURE
47h	7Fh	DT PWROMAEBK	SOME COMMANDS CLEARED BY ISCSI PROTOCOL EVENT
27h	07h	D B	SPACE ALLOCATION FAILED WRITE PROTECT
5Dh	03h	R	SPARE AREA EXHAUSTION PREDICTION THRESHOLD EXCEEDED
3Fh	08h	DT WROMAEB	SPARE CREATED OR MODIFIED
3Fh	09h	DT WROMAEB	SPARE DELETED
5Dh	55h	D B	SPINDLE IMPENDING FAILURE ACCESS TIMES TOO HIGH
5Dh	57h	D B	SPINDLE IMPENDING FAILURE CHANNEL PARAMETRICS
5Dh	58h	D B	SPINDLE IMPENDING FAILURE CONTROLLER DETECTED
5Dh	52h	D B	SPINDLE IMPENDING FAILURE DATA ERROR RATE TOO HIGH
5Dh	5Ch	D B	SPINDLE IMPENDING FAILURE DRIVE CALIBRATION RETRY COUNT
5Dh	51h	D B	SPINDLE IMPENDING FAILURE DRIVE ERROR RATE TOO HIGH
5Dh	50h	D B	SPINDLE IMPENDING FAILURE GENERAL HARD DRIVE FAILURE
5Dh	53h	D B	SPINDLE IMPENDING FAILURE SEEK ERROR RATE TOO HIGH
5Dh	5Ah	D B	SPINDLE IMPENDING FAILURE SEEK TIME PERFORMANCE
5Dh	5Bh	D B	SPINDLE IMPENDING FAILURE SPIN-UP RETRY COUNT
5Dh	56h	D B	SPINDLE IMPENDING FAILURE START UNIT TIMES TOO HIGH
5Dh	59h	D B	SPINDLE IMPENDING FAILURE THROUGHPUT PERFORMANCE
5Dh	54h	D B	SPINDLE IMPENDING FAILURE TOO MANY BLOCK REASSIGNS
09h	03h	WRO	SPINDLE SERVO FAILURE
5Ch	02h	D O	SPINDLES NOT SYNCHRONIZED
5Ch	01h	D O	SPINDLES SYNCHRONIZED
5Eh	04h	DTLPWRO A K	STANDBY CONDITION ACTIVATED BY COMMAND
5Eh	02h	DTLPWRO A K	STANDBY CONDITION ACTIVATED BY TIMER
5Eh	0Ah	DTLPWRO A K	STANDBY_Y CONDITION ACTIVATED BY COMMAND
5Eh	09h	DTLPWRO A K	STANDBY_Y CONDITION ACTIVATED BY TIMER
6Bh	00h	A	STATE CHANGE HAS OCCURRED
1Bh	00h	DTLPWROMAEBKVF	SYNCHRONOUS DATA TRANSFER ERROR
55h	01h	D O BK	SYSTEM BUFFER FULL
55h	00h	P	SYSTEM RESOURCE FAILURE
4Dh	NNh	DTLPWROMAEBKVF	TAGGED OVERLAPPED COMMANDS (NN = TASK TAG)
33h	00h	T	TAPE LENGTH ERROR
3Bh	03h	L	TAPE OR ELECTRONIC VERTICAL FORMS UNIT NOT READY
3Bh	01h	T	TAPE POSITION ERROR AT BEGINNING-OF-MEDIUM
3Bh	02h	T	TAPE POSITION ERROR AT END-OF-MEDIUM
3Fh	00h	DTLPWROMAEBKVF	TARGET OPERATING CONDITIONS HAVE CHANGED
38h	07h	D	THIN PROVISIONING SOFT THRESHOLD REACHED
0Dh	01h	DTLPWRO A K	THIRD PARTY DEVICE FAILURE

A numeric ordered listing of the ASC and ASCQ assignments is provided in D.2.

Table 44 — ASC and ASCQ assignments (part 16 of 17)

```
D – Direct Access Block Device (SBC-3)
. T – Sequential Access Device (SSC-4)
.   L – Printer Device (SSC)
.     P – Processor Device (SPC-2)
.     . W – Write Once Block Device (SBC)
.     .   R – C/DVD Device (MMC-6)
.     .     O – Optical Memory Block Device (SBC)
.     .     . M – Media Changer Device (SMC-3)
.     .     .   A – Storage Array Device (SCC-2)
.     .     .     E – SCSI Enclosure Services device (SES-3)
.     .     .     . B – Simplified Direct-Access (Reduced Block) device (RBC)
.     .     .     .   K – Optical Card Reader/Writer device (OCRW)
.     .     .     .     V – Automation/Device Interface device (ADC-3)
.     .     .     .     . F – Object-based Storage Device (OSD-2)
.     .     .     .     .
```

Device Column key
blank = code not used
not blank = code used

ASC	ASCQ	DTLPWROMAEBKVF	Description
5Bh	01h	DTLPWROM K	THRESHOLD CONDITION MET
26h	03h	DTLPWROMAE K	THRESHOLD PARAMETERS NOT SUPPORTED
3Eh	02h	DTLPWROMAEBKVF	TIMEOUT ON LOGICAL UNIT
2Ah	10h	DT M E V	TIMESTAMP CHANGED
26h	08h	DTLPWRO K	TOO MANY SEGMENT DESCRIPTORS
26h	06h	DTLPWRO K	TOO MANY TARGET DESCRIPTORS
2Ch	01h		TOO MANY WINDOWS SPECIFIED
4Bh	02h	DT PWROMAEBK	TOO MUCH WRITE DATA
09h	00h	DT WRO B	TRACK FOLLOWING ERROR
09h	01h	WRO K	TRACKING SERVO FAILURE
29h	06h	DTLPWROMAEBKVF	TRANSCEIVER MODE CHANGED TO LVD
29h	05h	DTLPWROMAEBKVF	TRANSCEIVER MODE CHANGED TO SINGLE-ENDED
61h	01h		UNABLE TO ACQUIRE VIDEO
74h	01h	T	UNABLE TO DECRYPT DATA
74h	0Ch	DT R MAEBKV	UNABLE TO DECRYPT PARAMETER LIST
57h	00h	R	UNABLE TO RECOVER TABLE-OF-CONTENTS
74h	02h	T	UNENCRYPTED DATA ENCOUNTERED WHILE DECRYPTING
26h	0Ah	DTLPWRO K	UNEXPECTED INEXACT SEGMENT
74h	06h	T	UNKNOWN SIGNATURE VERIFICATION KEY
53h	01h	T	UNLOAD TAPE FAILURE
08h	04h	DTLPWRO K	UNREACHABLE COPY TARGET
11h	00h	DT WRO BK	UNRECOVERED READ ERROR
11h	04h	D W O BK	UNRECOVERED READ ERROR - AUTO REALLOCATE FAILED
11h	0Bh	D W O BK	UNRECOVERED READ ERROR - RECOMMEND REASSIGNMENT
11h	0Ch	D W O BK	UNRECOVERED READ ERROR - RECOMMEND REWRITE THE DATA
46h	00h	DTLPWROM BK	UNSUCCESSFUL SOFT RESET
35h	01h	DTLPWROMAEBKVF	UNSUPPORTED ENCLOSURE FUNCTION
26h	09h	DTLPWRO K	UNSUPPORTED SEGMENT DESCRIPTOR TYPE CODE
26h	07h	DTLPWRO K	UNSUPPORTED TARGET DESCRIPTOR TYPE CODE
59h	00h	O	UPDATED BLOCK READ
26h	12h	T	VENDOR SPECIFIC KEY REFERENCE NOT FOUND
00h	1Ch	T	VERIFY OPERATION IN PROGRESS
61h	00h		VIDEO ACQUISITION ERROR
65h	00h	DTLPWROMAEBKVF	VOLTAGE FAULT
53h	05h	M	VOLUME IDENTIFIER INVALID
53h	06h	M	VOLUME IDENTIFIER MISSING
3Fh	0Ah	DT WROMAEBK	VOLUME SET CREATED OR MODIFIED
3Fh	0Ch	DT WROMAEBK	VOLUME SET DEASSIGNED
3Fh	0Bh	DT WROMAEBK	VOLUME SET DELETED
3Fh	0Dh	DT WROMAEBK	VOLUME SET REASSIGNED

A numeric ordered listing of the ASC and ASCQ assignments is provided in D.2.

Table 44 — ASC and ASCQ assignments (part 17 of 17)

```
                D – Direct Access Block Device (SBC-3)                    Device Column key
                . T – Sequential Access Device (SSC-4)                    blank = code not used
                .   L – Printer Device (SSC)                              not blank = code used
                .     P – Processor Device (SPC-2)
                .     . W – Write Once Block Device (SBC)
                .     .   R – C/DVD Device (MMC-6)
                .     .     O – Optical Memory Block Device (SBC)
                .     .     . M – Media Changer Device (SMC-3)
                .     .     .   A – Storage Array Device (SCC-2)
                .     .     .     E – SCSI Enclosure Services device (SES-3)
                .     .     .     . B – Simplified Direct-Access (Reduced Block) device (RBC)
                .     .     .     .   K – Optical Card Reader/Writer device (OCRW)
                .     .     .     .     V – Automation/Device Interface device (ADC-3)
                .     .     .     .     . F – Object-based Storage Device (OSD-2)
                .     .     .     .     .
ASC  ASCQ   DTLPWROMAEBKVF  Description
0Bh  00h    DTLPWROMAEBKVF  WARNING
0Bh  05h    DTLPWRO  AEBKVF WARNING - BACKGROUND MEDIUM SCAN DETECTED MEDIUM ERROR
0Bh  04h    DTLPWRO  AEBKVF WARNING - BACKGROUND PRE-SCAN DETECTED MEDIUM ERROR
0Bh  03h    DTLPWROMAEBKVF  WARNING - BACKGROUND SELF-TEST FAILED
0Bh  07h    DTLPWROMAEBKVF  WARNING - DEGRADED POWER TO NON-VOLATILE CACHE
0Bh  02h    DTLPWROMAEBKVF  WARNING - ENCLOSURE DEGRADED
0Bh  06h    DTLPWROMAEBKVF  WARNING - NON-VOLATILE CACHE NOW VOLATILE
0Bh  08h    DTLPWROMAEBKVF  WARNING - POWER LOSS EXPECTED
0Bh  01h    DTLPWROMAEBKVF  WARNING - SPECIFIED TEMPERATURE EXCEEDED
30h  0Dh      T             WORM MEDIUM - INTEGRITY CHECK
30h  0Ch      T             WORM MEDIUM - OVERWRITE ATTEMPTED
50h  00h      T             WRITE APPEND ERROR
50h  01h      T             WRITE APPEND POSITION ERROR
0Ch  00h      T     R       WRITE ERROR
0Ch  02h    D     W  O     BK WRITE ERROR - AUTO REALLOCATION FAILED
0Ch  09h            R       WRITE ERROR - LOSS OF STREAMING
0Ch  0Dh    DTLPWRO  AEBKVF WRITE ERROR - NOT ENOUGH UNSOLICITED DATA
0Ch  0Ah            R       WRITE ERROR - PADDING BLOCKS ADDED
0Ch  03h    D     W  O     BK WRITE ERROR - RECOMMEND REASSIGNMENT
0Ch  01h                   K WRITE ERROR - RECOVERED WITH AUTO REALLOCATION
0Ch  08h            R       WRITE ERROR - RECOVERY FAILED
0Ch  07h            R       WRITE ERROR - RECOVERY NEEDED
0Ch  0Ch    DTLPWRO  AEBKVF WRITE ERROR - UNEXPECTED UNSOLICITED DATA
27h  00h    DT    WRO     BK WRITE PROTECTED
31h  02h            R       ZONED FORMATTING FAILED DUE TO SPARE LINKING
3Fh  12h    DTLPWR  MAEBK F iSCSI IP ADDRESS ADDED
3Fh  14h    DTLPWR  MAEBK F iSCSI IP ADDRESS CHANGED
3Fh  13h    DTLPWR  MAEBK F iSCSI IP ADDRESS REMOVED

80h  xxh            \
  Through             >     Vendor specific
FFh  xxh            /

xxh  80h            \
  Through             >     Vendor specific qualification of standard ASC
xxh  FFh            /
                          All codes not shown are reserved.
A numeric ordered listing of the ASC and ASCQ assignments is provided in D.2.
```

4.6 Secure random numbers

Secure Random numbers should be generated as specified by RFC 4086 (e.g., see FIPS 140-2 Annex C: Approved Random Number Generators).

If the same random number source is used generate random numbers for multiple purposes (e.g., nonces and secret keys), interactions between the two shall not be allowed to compromise secrecy. If the value sequence generated by the common random number source is predictable to any degree, then the random number values that are transmitted outside the SCSI device may provide information about the random number values that the SCSI device maintains internally, based on the reasonable assumption that an adversary knows the order in which the random numbers are obtained from the common random number source. SCSI devices shall eliminate sources of such predictability.

Compliance with RFC 4086 is one method for achieving the required independence between random number values.

5 Model common to all device types

5.1 Introduction to the model common to all device types

This model describes some of the general characteristics expected of most SCSI devices. It is not intended to alter any requirements defined elsewhere in SCSI. Devices conforming to this standard also shall conform to SAM-4.

5.2 Important commands for all SCSI device servers

5.2.1 Commands implemented by all SCSI device servers

This standard defines the following mandatory commands (see 6.1):

a) the INQUIRY command (see 5.2.3);
b) the REPORT LUNS command (see 5.2.4); and
c) the TEST UNIT READY command (see 5.2.5).

These commands are used to discover a logical unit's capabilities, to discover the system configuration, and to determine whether a logical unit is ready.

If the device server is able to process commands after an error occurs that prohibits normal command completion, then the device server may include a field replaceable unit code in the sense data when returning CHECK CONDITION status for commands other than the INQUIRY command (see 6.4), the REPORT LUNS command (see 6.23), and the REQUEST SENSE command (see 6.29). If the sense data includes a field replaceable unit code, then an application client may use the INQUIRY command to request the corresponding ASCII Information VPD page (see 7.8.3), if any, which contains ASCII information about the field replaceable unit causing the error.

5.2.2 Commands recommended for all SCSI device servers

Support for the REQUEST SENSE command is recommended to provide compatibility with application clients designed to use previous versions of this standard or status polling features defined by command standards (see 3.1.31).

5.2.3 Using the INQUIRY command

The INQUIRY command (see 6.4) may be used by an application client to determine the configuration of a logical unit. Device servers respond with information that includes their device type and standard version and may include the vendor's identification, model number and other information.

The Device Identification VPD page (see 7.8.6) returned in response to an INQUIRY command with the EVPD bit set to one and the PAGE CODE field set to 83h contains identifying information for the logical unit, the target port, and the SCSI target device.

The INQUIRY command may return incomplete information after power on (see 6.4.1).

5.2.4 Using the REPORT LUNS command

The REPORT LUNS command (see 6.23) may be used by an application client to discover the logical unit inventory (see 3.1.86) that is accessible to the I_T nexus on which the command is sent.

5.2.5 Using the TEST UNIT READY command

The TEST UNIT READY command (see 6.37) allows an application client to poll a logical unit until it is ready without the need to allocate space for returned data. The TEST UNIT READY command may be used to check the media status of logical units with removable media. Device servers should respond promptly to indicate the current status of the SCSI device.

NOTE 8 - Delays to achieve GOOD status from a TEST UNIT READY command may adversely affect initiator device performance.

5.2.6 Using the REQUEST SENSE command

The REQUEST SENSE command (see 6.29) may be used by an application client to poll the status of some background operations and to clear interlocked unit attention conditions (see 7.5.7).

5.3 Implicit head of queue

Each of the following commands may be processed by the task manager as if it has a task attribute of HEAD OF QUEUE (see SAM-4) if it is received with a SIMPLE task attribute, an ORDERED task attribute, or no task attribute:

a) INQUIRY; and
b) REPORT LUNS.

See SAM-5 for additional rules on implicit head of queue processing.

5.4 Parameter rounding

Certain parameters sent to a device server with various commands contain a range of values. Device servers may choose to implement only selected values from this range. When the device server receives a value that it does not support, it either rejects the command (i.e., CHECK CONDITION status with ILLEGAL REQUEST sense key) or it rounds the value received to a supported value.

When parameter rounding is implemented, a device server that receives a parameter value that is not an exact supported value shall adjust the value to one that it supports and shall return CHECK CONDITION status, with the sense key set to RECOVERED ERROR, and the additional sense code set to ROUNDED PARAMETER. The application client should send an appropriate command to learn what value the device server has selected.

The device server shall reject unsupported values unless rounding is permitted in the description of the parameter. When the description of a parameter states that rounding is permitted, the device server should adjust maximum-value fields down to the next lower supported value than the one specified by the application client. Minimum-value fields should be rounded up to the next higher supported value than the one specified by the application client. In some cases, the type of rounding (i.e., up or down) is specified in the description of the parameter.

5.5 Parsing variable-length parameter lists and parameter data

Parameter lists and parameter data (e.g., diagnostic pages, mode pages, log pages, and VPD pages) often include length fields indicating the size of the parameter list or parameter data (e.g., the MODE DATA LENGTH field in the mode parameter header (see 7.5.4)). Parameter lists and parameter data often include descriptor lists and

descriptor length fields containing the length of the descriptors in the descriptor lists (e.g., the DESIGNATOR LENGTH field in the designation descriptor used in the Device Identification VPD page (see 7.8.6.1)).

An application client or device server shall not assume that any length field contains the value defined in a SCSI standard.

If a device server receives a parameter list containing a length field (e.g., a PAGE LENGTH field) and containing more bytes than are defined in the standard to which it was designed (e.g., the device server complies with a version of a SCSI standard defining that a parameter list has 24 bytes, but receives a parameter list containing 36 bytes), 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.

For parameter lists containing a descriptor length field and a descriptor list, if a device server receives more bytes in a descriptor than are defined in the standard to which it was designed (e.g., the device server complies with a version of a SCSI standard defining that a descriptor is 12 bytes, but receives a parameter list containing a 16 byte form of that descriptor), 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.

An application client should ignore any bytes of parameter data beyond those defined in the standard to which it was designed (e.g., if the application client complies with a version of a SCSI standard defining 24 bytes of parameter data, but receives 36 bytes of parameter data, then the application client should ignore the last 12 bytes or the parameter data).

For additional response bytes containing a descriptor length field and a descriptor list, an application client should ignore any bytes in each descriptor beyond those defined in the standard to which it was designed (e.g., if the application client complies with a version of a SCSI standard defining that a descriptor has 24 bytes, but receives parameter data containing a descriptor list with a 36 byte form of that descriptor, then the application client should ignore the last 12 bytes of the descriptor).

5.6 Pollable condition information

5.6.1 Overview

A device server may have information about a condition that does not represent an exception condition. This information is not reported with a CHECK CONDITION status. Instead, this information is reported:

a) as sense data format parameter data in response to a REQUEST SENSE command as described in 5.6.2; or
b) as log parameter data as described in 5.6.3.

5.6.2 REQUEST SENSE pollable sense data

5.6.2.1 Overview

SCSI target devices are required to make specified information used to prepare the parameter data that is returned by the REQUEST SENSE command (see 6.29) available whenever that information is applicable. Which sense data is returned in a REQUEST SENSE parameter data is determined at the time the REQUEST SENSE command is processed.

Application clients have no way to control which pollable sense data is returned by a REQUEST SENSE command. Mechanisms that are specialized to a particular function (e.g., log pages, mode pages) should be used to obtain information about that function.

5.6.2.2 Selecting pollable sense data to return

Conditions that are not related to the availability of pollable sense data (e.g., a pending unit attention condition) may cause the device server to ignore all available pollable sense data.

When pollable sense data is available to be returned by a REQUEST SENSE command (see 6.29), the choice of which sense key and additional sense code to return shall be made as follows:

1) sense data with the sense key set to NOT READY and the additional sense code set to:
   1) LOGICAL UNIT NOT READY, INITIALIZING COMMAND REQUIRED (see 5.12.5);
   2) LOGICAL UNIT NOT READY, FORMAT IN PROGRESS (see applicable command standard);
   3) LOGICAL UNIT NOT READY, SANITIZE IN PROGRESS (see SBC-3);
   4) LOGICAL UNIT NOT READY, SELF-TEST IN PROGRESS (see 5.8.4.4);
   5) LOGICAL UNIT NOT READY, LONG WRITE IN PROGRESS (see MMC-6);
   6) LOGICAL UNIT NOT READY, OPERATION IN PROGRESS (see MMC-6); and
   7) LOGICAL UNIT NOT READY, SA CREATION IN PROGRESS (see 5.16.5);
2) sense data with the sense key set to NO SENSE and the additional sense code set:
   1) as defined for the informational exceptions sense data described in the Informational Exceptions Control mode page (see applicable command standard);
   2) to LOGICAL UNIT TRANSITIONING TO ANOTHER POWER CONDITION (see 5.12.6 and SBC-3); and
   3) as defined for any power condition related sense data described in 5.12.5;

   and
3) sense data with the sense key set to NOT READY or NO SENSE and the additional sense code set as defined for sequential-access device progress indication reporting (see SSC-4).

5.6.2.3 Returning one or more progress indications

If the sense key is set to NOT READY or NO SENSE in the header for the sense data being returned by the REQUEST SENSE command (see 6.29) as parameter data, and progress indication information is being provided as part of any pollable sense data available for return by the REQUEST SENSE command (see 5.6.2.1), then:

a) if the DESC bit is set to zero in the REQUEST SENSE command, then the device server shall set the SKSV bit in the fixed-format sense data to one only if progress indication information is available for the additional sense code being returned in the ADDITIONAL SENSE CODE field and ADDITIONAL SENSE CODE QUALIFIER field; or
b) if the DESC bit is set to one in the REQUEST SENSE command, then the device server places progress indications in the sense data being returned as follows:
   A) if progress indication information is available for the additional sense code being returned in the ADDITIONAL SENSE CODE field and ADDITIONAL SENSE CODE QUALIFIER field in the sense data header, then the device shall include one sense key specific sense data descriptor (see 4.5.2.4) in which that progress indication is returned; and
   B) if progress indication information is available for one or more additional sense codes that are not being returned in the ADDITIONAL SENSE CODE field and ADDITIONAL SENSE CODE QUALIFIER field in the sense data header, then should include in the sense data one progress indication sense data descriptor (see 4.5.2.6) for each available instance of progress indication information.

5.6.3 Log parameter pollable device condition information

The following background testing functions report their condition information using log page parameters:

a) self-test operations report whether a test is in progress, but not how far it has progressed using the Self-Test Results log page (see 7.3.16); and
b) direct-access block devices report progress for background pre-scan operations and background medium scan operations using the Background Scan Results log page (see SBC-3).

5.7 Device specific background functions

5.7.1 Overview

Device specific background functions are SCSI target device specific functions that a SCSI target device may perform when there are no other application client-initiated operations in progress. A device specific background function is initiated when:

a) a device specific event associated with the function occurs (e.g., a timer expires or a counter reaches its limit); and
b) the function is enabled.

If conditions are met to initiate more than one device specific background function (e.g., more than one device specific event occurs), then the order of performing the functions is device specific. Other device server processes (e.g., processing a command) may or may not have any impact on events associated with device specific background functions.

Device specific background functions do not include self-test operations (see 5.8) or background scan operations (see SBC-3).

Device specific background functions:

a) may include background functions for informational exceptions;
b) may include regular, device specific self testing or saving of device specific data;
c) may require the logical unit to be in a different power condition to be performed (e.g. a logical unit may require being in the active power condition to access the medium for some functions);
d) may be enabled or disabled via the EBF bit in the Informational Exceptions Control mode page (see applicable command standard);
e) shall be affected by the PERF bit and the LOGERR bit in the Informational Exceptions Control mode page, if the background function is associated with informational exceptions;
f) should be processed relative to power conditions based on the setting in the PM_BG_PRECEDENCE field in the Power Condition mode page (see 7.5.12);
g) may have impact on the SCSI target device's performance (e.g., if a logical unit is performing a background function, and the device server receives a command from an application client that requires access to the logical unit, then the logical unit may take a short period of time (e.g., two seconds) to suspend the background function before the logical unit is able to process the command);
h) shall not affect power condition timers as defined in the Power Condition mode page;
i) shall not affect timers defined in the Background Control mode page (see SBC-3); and
j) shall have no negative impact on the reliability of the logical unit.

5.7.2 Suspending and resuming device-specific background functions

The SCSI target device shall suspend a device specific background function in progress when:

a) any of the following are true:
   A) a command or task management function is processed that requires the device-specific background function to be suspended; or
   B) a SCSI event (e.g., a reset event) (see SAM-5) occurs that requires the device-specific background function to be suspended;

   or
b) all of the following are true:
   A) a power condition timer defined in the Power Condition mode page (see 7.5.12) expires;
   B) the PM_BG_PRECEDENCE field in the Power Condition mode page is set to 00b or 10b; and
   C) the SCSI target device is unable to continue performing the device-specific background function in the power condition associated with the timer that expired.

When a device-specific background function is suspended, the device server shall not stop any process that causes a device-specific background function to be initiated (e.g., not stop any timers or counters associated with device-specific background functions).

A suspended device specific background function may be resumed when:

a) there are no commands in the task set to be processed;
b) there are no task management functions to be processed;
c) there are no SCSI events to be processed;
d) no ACA condition (see SAM-5) exists;
e) the PM_BG_PRECEDENCE field in the Power Condition mode page (see 7.5.12) is set to 00b or 10b; or
f) the PM_BG_PRECEDENCE field in the Power Condition mode page is set to 10b, and no power condition timer defined in the Power Condition mode page has expired.

5.8 Self-test operations

5.8.1 Overview

The SEND DIAGNOSTIC command (see 6.32) provides methods for an application client to request that a SCSI target device perform self test operations. This standard defines the following self-tests:

a) the default self-test (see 5.8.2);
b) the short self-test (see 5.8.3); and
c) the extended self-test (see 5.8.3).

5.8.2 Default self-test

The default self-test is mandatory for all SCSI target device types that support the SEND DIAGNOSTIC command. The operations performed for the default self-test are not defined by this standard. An application client requests that a SCSI target device perform a default self-test by setting the SELFTEST bit to one in the SEND DIAGNOSTIC command (see 6.32).

An application client may use the DEVOFFL bit and the UNITOFFL bit in the SEND DIAGNOSTIC command to allow the device server to perform operations during a default self-test that affect conditions for one or more logical units in the SCSI target device (e.g., if the DEVOFFL bit is set to one, then the device server may clear established reser-

vations while performing the test, and if the UNITOFFL bit is set to one, then the logical unit may alter its medium while performing the test).

While a SCSI target device is performing a default self-test, the device server shall terminate all commands received while the self-test is in progress, except INQUIRY, REPORT LUNS, and REQUEST SENSE, with CHECK CONDITION status with the sense key set to NOT READY and the additional sense code set to LOGICAL UNIT NOT READY, SELF-TEST IN PROGRESS. If the device server receives an INQUIRY command, a REPORT LUNS command, or a REQUEST SENSE command while performing a default self-test, then the device server shall process the command.

If the SCSI target device detects no errors during a default self-test, then the device server shall complete the command with GOOD status. If the SCSI target device detects an error during the test, then the device server shall terminate the command with CHECK CONDITION status with the sense key set to HARDWARE ERROR and the additional sense code set to indicate the cause of the error.

5.8.3 The short self-test and extended self-test

The short self-test and the extended self-test are optional. An application client requests that a SCSI target device perform one of these self-tests by the SELFTEST bit to zero and specifying an appropriate value in the SELF-TEST CODE field in the SEND DIAGNOSTIC command (see 6.32).

The criteria for the short self-test are that the test has one or more segments and completes in two minutes or less. The criteria for the extended self-test are that the test is has one or more segments and that the completion time required by the SCSI target device to complete the extended self-test is reported in the EXTENDED SELF-TEST COMPLETION MINUTES field in the Extended INQUIRY Data VPD page (see 7.8.7), the EXTENDED SELF-TEST COMPLETION TIME field in the Control mode page (see 7.5.7), or both.

The tests performed in the segments are vendor specific and may be the same for the short self-test and the extended self-test.

The following are examples of segments:

a) an electrical segment wherein the logical unit tests its own electronics. The tests in this segment are vendor specific, but some examples of tests that may be included are:
   A) a buffer RAM test;
   B) a read/write circuitry test; or
   C) a test of the read/write heads;
b) a seek/servo segment wherein a device tests it capability to find and servo on data tracks; and
c) a read/verify scan segment wherein a device performs read scanning of some or all of the medium surface.

5.8.4 Self-test modes

5.8.4.1 Self-test modes overview

A foreground mode (see 5.8.4.2) and a background mode (see 5.8.4.3) are defined for the short self-test and the extended self-test. An application client specifies the self-test mode by the value in the SELF-TEST CODE field in the SEND DIAGNOSTIC command (see 6.32).

5.8.4.2 Foreground mode

When an application client specifies a self-test to be performed in the foreground mode, the device server shall return status for the command after the self-test has been completed.

While a SCSI target device is performing a self-test in the foreground mode, the device server shall terminate all commands received while the self-test is in progress, except INQUIRY, REPORT LUNS, and REQUEST SENSE, with CHECK CONDITION status, with the sense key set to NOT READY, and the additional sense code set to LOGICAL UNIT NOT READY, SELF-TEST IN PROGRESS. If the device server receives an INQUIRY command, a REPORT LUNS command, or a REQUEST SENSE command while performing a self-test in the foreground mode, then the device server shall process the command.

If a SCSI target device is performing a self-test in the foreground mode and an test occurs during the test, then:

a) the SCSI target device shall abort the self-test; and
b) the device server shall update the Self-Test Results log page (see 7.3.16):
   A) if the device server is able to update the Self-Test Results log page, then the device server shall terminate the SEND DIAGNOSTIC command with CHECK CONDITION status, with the sense key set to HARDWARE ERROR, and the additional sense code set to LOGICAL UNIT FAILED SELF-TEST. The application client may obtain additional information about the failure by reading the Self-Test Results log page; or
   B) if the device server is unable to update the Self-Test Results log page, then the device server shall terminate the SEND DIAGNOSTIC command with CHECK CONDITION status, with the sense key set to HARDWARE ERROR, and the additional sense code set to LOGICAL UNIT UNABLE TO UPDATE SELF-TEST LOG.

An application client may cause a SCSI target device to abort a self-test that is being performed in the foreground mode by using a task management function (see SAM-4) (e.g., an ABORT TASK task management function, a CLEAR TASK SET task management function) or a transport specific reset (e.g., a hard reset event) (see SAM-4). In addition, a self-test being performed in the foreground mode shall be terminated by an I_T nexus loss event or a power loss expected event (see SAM-4). If a SCSI target device aborts a self-test that is being performed in the foreground mode based on the SCSI target device receiving a task management function or a transport specific event, then the device server shall update the Self-Test Results log page (see 7.3.16).

5.8.4.3 Background mode

When a device server receives a SEND DIAGNOSTIC command specifying a self-test to be performed in the background mode:

a) the device server shall terminate the command if the CDB is invalid; or
b) the device server shall:
   1) complete the command with GOOD status;
   2) initialize the next self-test results log parameter in the Self-Test Results log page (see 7.3.16) by setting:
      a) the SELF-TEST CODE field to the self-test code from the SEND DIAGNOSTIC command; and
      b) the SELF-TEST RESULTS field to Fh;
      and
   3) begin the self-test.

An application client may request that a device server abort a self-test that is being performed in the background mode by sending a SEND DIAGNOSTIC command with the SELF-TEST CODE field set to 100b (i.e., abort background self-test function). A SCSI target device shall not abort a self-test being performed in the background mode as the result of an I_T nexus loss event (see SAM-4). A SCSI target device shall abort a self-test being performed in the background mode as the result of a power loss expected event (see SAM-4).

While the SCSI target device is performing a self-test in the background mode, the device server shall terminate with CHECK CONDITION status any received SEND DIAGNOSTIC command that meets any of the following criteria:

a) the SELFTEST bit is set to one; or
b) the SELF-TEST CODE field contains a value other than 000b or 100b.

When terminating a received SEND DIAGNOSTIC command that meets the criteria described in this subclause, the device server shall set the sense key to NOT READY and the additional sense code to LOGICAL UNIT NOT READY, SELF-TEST IN PROGRESS.

If the SCSI target device is performing a self-test in the background mode, and the device server receives any command that requires suspension of the self-test to process, except those listed in table 45, then:

a) the logical unit shall suspend the self-test;
b) the device server shall begin processing the command within two seconds after the CDB has been validated; and
c) after the command completes, the logical unit shall resume the self-test.

If the device server receives one of the commands listed in table 45, then the device server shall:

a) abort the self-test;
b) update the self-test log; and
c) begin processing the command within two seconds after the CDB has been validated.

Table 45 — Exception commands for background self-tests

Device type	Command	Reference
All device types	SEND DIAGNOSTIC (with SELF-TEST CODE field set to 100b) WRITE BUFFER (with the mode set to any download microcode option)	6.32 6.39
Direct access block	FORMAT UNIT START STOP UNIT	SBC-3
Sequential access	ERASE FORMAT MEDIUM LOAD UNLOAD LOCATE READ READ POSITION READ REVERSE REWIND SPACE VERIFY WRITE WRITE BUFFER WRITE FILEMARKS	SSC-3
Media changer	EXCHANGE MEDIUM INITIALIZE ELEMENT STATUS MOVE MEDIUM POSITION TO ELEMENT READ ELEMENT STATUS (if CURDATA=0 and device motion is required) WRITE BUFFER	SMC-3
Object-based storage	Any command with operation code 7Fh (i.e., all commands defined by the OSD standard)	OSD

NOTE Device types not listed in this table do not have commands that are exceptions for background self-tests, other than those listed above for all device types.

5.8.4.4 Features common to foreground and background self-test modes

An application client may use a REQUEST SENSE command (see 6.29) to poll for progress indication at any time during a self-test. In response to the REQUEST sense command, the device server returns parameter data containing sense data with the sense key set to NOT READY, the additional sense code set to LOGICAL UNIT NOT READY, SELF-TEST IN PROGRESS, and the PROGRESS INDICATION field set to indicate the progress of the self-test.

An application client may use the EBACKERR bit and the MRIE field in the Informational Exceptions Control mode page (see applicable command standard) to control the reporting of errors that occur during a background self-test operation.

An application client may obtain information about the 20 most recent self-tests, including the self-test in progress, if any, by reading the Self-Test Results log page (see 7.3.16). With the exception of progress indication, this is the only method for an application client to obtain information about self-tests performed in the background mode unless an error occurs during the self-test.

Table 46 summarizes:

a) when a device server returns status after receipt of a self-test command;
b) how an application client may abort a self-test;
c) how a device server processes commands that are entered into the task set while a self-test is in progress; and

d) how a self-test failure is reported.

Table 46 — Self-test mode summary

Self-test mode	When status is returned	How to abort the self-test	Processing of commands while a self-test is in progress	Self-test failure reporting
Foreground	After the self-test is complete	A task management function or reset event that causes a self-test to be aborted (see 5.8.4.2)	If the command is INQUIRY, REPORT LUNS or REQUEST SENSE, process normally. Otherwise, terminate with CHECK CONDITION status, with the sense key set to NOT READY, and the additional sense code set to LOGICAL UNIT NOT READY, SELF-TEST IN PROGRESS.	The device server terminates the SEND DIAGONISTIC command with CHECK CONDITION status, with the sense key set to HARDWARE ERROR, and the additional sense code set to LOGICAL UNIT FAILED SELF-TEST or LOGICAL UNIT UNABLE TO UPDATE SELF-TEST LOG (see 5.8.4.2). [a]
Background	After the CDB is validated	A SEND DIAGNOSTIC command with the SELF-TEST CODE field set to 100b	Process the command, except as described in 5.8.4.3.	An application client: a) checks the Self-Test Results log page (see 7.3.16) after the PROGRESS INDICATION field returned in response to a REQUEST SENSE command indicates that the self-test is complete; or b) uses the EBACKERR bit and the MRIE field (see applicable command standard) to specify a method of indicating that a failure occurred. If a failure occurs, then an additional sense code of WARNING - BACKGROUND SELF-TEST FAILED shall be returned using the method defined in the MRIE field.

[a] The device server shall not report an error until after the Self-Test Results log page is updated.

5.9 Reservations

5.9.1 Persistent Reservations overview

Reservations may be used to allow a device server to process commands from a selected set of I_T nexuses (i.e., combinations of initiator ports accessing target ports) and reject commands from I_T nexuses outside the selected set. The device server uniquely identifies I_T nexuses using protocol specific mechanisms.

Application clients may add or remove I_T nexuses from the selected set using reservation commands. If the application clients do not cooperate in the reservation protocol, data may be unexpectedly modified and deadlock conditions may occur.

The persistent reservations mechanism allows multiple application clients communicating through multiple I_T nexuses to preserve reservation operations across SCSI initiator device failures, which usually involve logical unit resets and involve I_T nexus losses. Persistent reservations persist across recovery actions. Persistent reservations are not reset by hard reset, logical unit reset, or I_T nexus loss.

The persistent reservation held by a failing I_T nexus may be preempted by another I_T nexus as part of its recovery process. Persistent reservations shall be retained by the device server until released, preempted, or cleared by mechanisms specified in this standard. Optionally, persistent reservations may be retained when power to the SCSI target device is removed.

The PERSISTENT RESERVE OUT and PERSISTENT RESERVE IN commands provide the basic mechanism for dynamic contention resolution in systems with multiple initiator ports accessing a logical unit.

Before a persistent reservation may be established, the application client shall register a reservation key for each I_T nexus with the device server. Reservation keys are necessary to allow:

a) authentication of subsequent PERSISTENT RESERVE OUT commands;
b) identification of other I_T nexuses that are registered;
c) identification of the reservation key(s) that have an associated persistent reservation;
d) preemption of a persistent reservation from a failing or uncooperative I_T nexus; and
e) multiple I_T nexuses to participate in a persistent reservation.

The reservation key provides a method for the application client to associate a protocol-independent identifier with a registered I_T nexus. The reservation key is used in the PERSISTENT RESERVE IN command to identify which I_T nexuses are registered and which I_T nexus, if any, holds the persistent reservation. The reservation key is used in the PERSISTENT RESERVE OUT command to register an I_T nexus, to verify the I_T nexus being used for the PERSISTENT RESERVE OUT command is registered, and to specify which registrations or persistent reservation to preempt.

Reservation key values may be used by application clients to identify registered I_T nexuses, using application specific methods that are outside the scope of this standard. This standard provides the ability to register no more than one reservation key per I_T nexus. Multiple initiator ports may use the same reservation key value for a logical unit accessed through the same target ports. An initiator port may use the same reservation key value for a logical unit accessed through different target ports. The logical unit shall maintain a separate reservation key for each I_T nexus, regardless of the reservation key's value.

An application client may register an I_T nexus with multiple logical units in a SCSI target device using any combination of unique or duplicate reservation keys. These rules provide the ability for an application client to preempt multiple I_T nexuses with a single PERSISTENT RESERVE OUT command, but they do not provide the ability for the application client to uniquely identify the I_T nexuses using the PERSISTENT RESERVE commands.

See table 179 in 6.14.2 for a list of PERSISTENT RESERVE OUT service actions. See table 167 in 6.13.1 for a list of PERSISTENT RESERVE IN service actions.

The scope (see 6.13.3.3) of a persistent reservation shall be the entire logical unit.

The type (see 6.13.3.4) of a persistent reservation defines the selected set of I_T nexuses for which the persistent reservation places restrictions on commands.

The details of which commands are allowed under what types of reservations are described in table 47.

In table 47 and table 48 the following key words are used:

allowed: Commands received from I_T nexuses not holding the reservation or from I_T nexuses not registered when a registrants only or all registrants type persistent reservation is present should complete normally.

conflict: Commands received from I_T nexuses not holding the reservation or from I_T nexuses not registered when a registrants only or all registrants type persistent reservation is present shall not be performed and the device server shall complete the command with RESERVATION CONFLICT status.

Commands from I_T nexuses holding a reservation should complete normally. The behavior of commands from registered I_T nexuses when a registrants only or all registrants type persistent reservation is present is specified in table 47 and table 48.

A command shall be checked for reservation conflicts when processing of the command begins. After that check succeeds, the command shall not be completed with RESERVATION CONFLICT status due to a subsequent reservation.

The time at which a reservation is established with respect to other commands being managed by the device server is vendor specific. Successful completion of a reservation command indicates that the new reservation is established. A reservation may apply to some or all of the commands in the task set before the completion of the reservation command. The reservation shall apply to all commands received by the device server after successful completion of the reservation command. Any persistent reserve service action shall be performed as a single indivisible event.

Multiple persistent reserve service actions may be present in the task set at the same time. The order of processing of such service actions is defined by the task set management requirements defined in SAM-4, but each is processed as a single indivisible command without any interleaving of actions that may be required by other reservation commands.

For each command, this standard or a command standard (see 3.1.31) defines the conditions that result in the command being completed with RESERVATION CONFLICT. Command standards define the conditions either in the device model or in the descriptions of each specific command.

Table 47 — SPC-4 commands that are allowed in the presence of various reservations (part 1 of 2)

Command	Addressed logical unit 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 not registered I_T nexus	
	Write Excl	Excl Access		Write Excl RR	Excl Access – RR
ACCESS CONTROL IN	Allowed	Allowed	Allowed	Allowed	Allowed
ACCESS CONTROL OUT	Allowed	Allowed	Allowed	Allowed	Allowed
CHANGE ALIASES	Conflict	Conflict	Allowed	Conflict	Conflict
EXTENDED COPY	Conflict	Conflict	Allowed	Conflict	Conflict
INQUIRY	Allowed	Allowed	Allowed	Allowed	Allowed
LOG SELECT	Conflict	Conflict	Allowed	Conflict	Conflict
LOG SENSE	Allowed	Allowed	Allowed	Allowed	Allowed
MANAGEMENT PROTOCOL IN	Allowed	Conflict	Allowed	Allowed	Conflict
MANAGEMENT PROTOCOL OUT	Conflict	Conflict	Allowed	Conflict	Conflict
MODE SELECT(6) / MODE SELECT(10)	Conflict	Conflict	Allowed	Conflict	Conflict
MODE SENSE(6) / MODE SENSE(10)	Allowed[b]	Conflict	Allowed	Allowed[b]	Conflict
PERSISTENT RESERVE IN	Allowed	Allowed	Allowed	Allowed	Allowed
PERSISTENT RESERVE OUT	see table 48				
READ ATTRIBUTE	Allowed[b]	Conflict	Allowed	Allowed[b]	Conflict
READ BUFFER	Allowed[b]	Conflict	Allowed	Allowed[b]	Conflict
READ MEDIA SERIAL NUMBER	Allowed	Allowed	Allowed	Allowed	Allowed
RECEIVE CREDENTIAL	Conflict	Conflict	Allowed	Conflict	Conflict
RECEIVE COPY RESULTS	Allowed[b]	Allowed[b]	Allowed	Allowed[b]	Allowed[b]
RECEIVE DIAGNOSTIC RESULTS	Allowed[b]	Conflict	Allowed	Allowed[b]	Conflict
RELEASE(6)/ RELEASE(10)	As defined in SPC-2 [a]				
REPORT ALIASES	Allowed	Allowed	Allowed	Allowed	Allowed

Key: **Excl**=Exclusive, **RR**=Registrants Only or All Registrants

[a] Exceptions to the behavior of the RESERVE and RELEASE commands described in SPC-2 are defined in 5.9.3.

[b] Logical units claiming compliance with previous versions of this standard (e.g., SPC-2, SPC-3) may return RESERVATION CONFLICT in this case. Logical units may report whether certain commands are allowed in ALLOW COMMANDS field of the parameter data returned by the PERSISTENT RESERVE IN command with REPORT CAPABILITIES service action (see 6.13.4).

Table 47 — SPC-4 commands that are allowed in the presence of various reservations (part 2 of 2)

Command	Addressed logical unit 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 not registered I_T nexus	
	Write Excl	Excl Access		Write Excl RR	Excl Access – RR
REPORT IDENTIFYING INFORMATION	Allowed	Allowed	Allowed	Allowed	Allowed
REPORT LUNS	Allowed	Allowed	Allowed	Allowed	Allowed
REPORT PRIORITY	Allowed	Allowed	Allowed	Allowed	Allowed
REPORT SUPPORTED OPERATION CODES	Allowed[b]	Conflict	Allowed	Allowed[b]	Conflict
REPORT SUPPORTED TASK MANAGEMENT FUNCTIONS	Allowed[b]	Conflict	Allowed	Allowed[b]	Conflict
REPORT TARGET PORT GROUPS	Allowed	Allowed	Allowed	Allowed	Allowed
REPORT TIMESTAMP	Allowed	Allowed	Allowed	Allowed	Allowed
REQUEST SENSE	Allowed	Allowed	Allowed	Allowed	Allowed
RESERVE(6) / RESERVE(10)	As defined in SPC-2 [a]				
SECURITY PROTOCOL IN	Allowed	Conflict	Allowed	Allowed	Conflict
SECURITY PROTOCOL OUT	Conflict	Conflict	Allowed	Conflict	Conflict
SEND DIAGNOSTIC	Conflict	Conflict	Allowed	Conflict	Conflict
SET IDENTIFYING INFORMATION	Conflict	Conflict	Allowed	Conflict	Conflict
SET PRIORITY	Conflict	Conflict	Allowed	Conflict	Conflict
SET TARGET PORT GROUPS	Conflict	Conflict	Allowed	Conflict	Conflict
SET TIMESTAMP	Conflict	Conflict	Allowed	Conflict	Conflict
TEST UNIT READY	Allowed[b]	Allowed[b]	Allowed	Allowed [b]	Allowed [b]
WRITE ATTRIBUTE	Conflict	Conflict	Allowed	Conflict	Conflict
WRITE BUFFER	Conflict	Conflict	Allowed	Conflict	Conflict

Key: **Excl**=Exclusive, **RR**=Registrants Only or All Registrants

[a] Exceptions to the behavior of the RESERVE and RELEASE commands described in SPC-2 are defined in 5.9.3.

[b] Logical units claiming compliance with previous versions of this standard (e.g., SPC-2, SPC-3) may return RESERVATION CONFLICT in this case. Logical units may report whether certain commands are allowed in ALLOW COMMANDS field of the parameter data returned by the PERSISTENT RESERVE IN command with REPORT CAPABILITIES service action (see 6.13.4).

Table 48 — PERSISTENT RESERVE OUT service actions allowed in the presence of various reservations

Command Service Action	Addressed logical unit has a persistent reservation held by another I_T nexus	
	Command is from a registered I_T nexus	Command is from a not registered I_T nexus
CLEAR	Allowed	Conflict
PREEMPT	Allowed	Conflict
PREEMPT AND ABORT	Allowed	Conflict
REGISTER	Allowed	Allowed
REGISTER AND IGNORE EXISTING KEY	Allowed	Allowed
REGISTER AND MOVE	Conflict	Conflict
RELEASE	Allowed [a]	Conflict
RESERVE	Conflict	Conflict

[a] The reservation is not released (see 5.9.11.2).

5.9.2 Third party persistent reservations

Except for all registrants type reservations, a reservation holder (see 5.9.10) may move the persistent reservation to a third party (e.g., a copy manager supporting the EXTENDED COPY command) using the REGISTER AND MOVE service action (see 5.9.8). A copy manager supporting the EXTENDED COPY command may be instructed to move the persistent reservation to a specified I_T nexus using the third party persistent reservations source I_T nexus segment descriptor (see 6.3.7.19).

5.9.3 Exceptions to SPC-2 RESERVE and RELEASE behavior

This subclause defines exceptions to the behavior of the RESERVE and RELEASE commands defined in SPC-2. The RESERVE and RELEASE commands are obsolete in this standard, except for the behavior defined in this subclause. Device servers that operate using the exceptions described in this subclause shall set the CRH bit to one in the parameter data returned by the REPORT CAPABILITIES service action of the PERSISTENT RESERVE IN command (see 6.13.4).

A RELEASE(6) or RELEASE(10) command shall complete with GOOD status, but the persistent reservation shall not be released, if the command is received from:

a) an I_T nexus that is a persistent reservation holder (see 5.9.10); or
b) an I_T nexus that is registered if a registrants only or all registrants type persistent reservation is present.

A RESERVE(6) or RESERVE(10) command shall complete with GOOD status, but no reservation shall be established and the persistent reservation shall not be changed, if the command is received from:

a) an I_T nexus that is a persistent reservation holder; or
b) an I_T nexus that is registered if a registrants only or all registrants type persistent reservation is present.

In all other cases, a RESERVE(6) command, RESERVE(10) command, RELEASE(6) command, or RELEASE(10) command shall be processed as defined in SPC-2.

5.9.4 Persistent reservations interactions with IKEv2-SCSI SA creation

If a PERSISTENT RESERVE OUT command is received while an IKEv2-SCSI CCS is in progress (see 5.16.4), the command shall be terminated with CHECK CONDITION status, with the sense key NOT READY, and the additional sense code set to LOGICAL UNIT NOT READY, SA CREATION IN PROGRESS. The sense key specific additional sense data may be set as described in 5.16.5.

5.9.5 Preserving persistent reservations and registrations

5.9.5.1 Preserving persistent reservations and registrations through power loss

The application client may request activation of the persist through power loss device server capability to preserve the persistent reservation and registrations across power cycles by setting the APTPL bit to one in the PERSISTENT RESERVE OUT parameter data sent with a REGISTER service action, REGISTER AND IGNORE EXISTING KEY service action, or REGISTER AND MOVE service action.

After the application client enables the persist through power loss capability the device server shall preserve the persistent reservation, if any, and all current and future registrations associated with the logical unit to which the REGISTER service action, the REGISTER AND IGNORE EXISTING KEY service action, or REGISTER AND MOVE service action was addressed until an application client disables the persist through power loss capability. The APTPL value from the most recent successfully completed REGISTER service action, REGISTER AND IGNORE EXISTING KEY service action, or REGISTER AND MOVE service action from any application client shall determine the logical unit's behavior in the event of a power loss.

The device server shall preserve the following information for each existing registration across any hard reset, logical unit reset, or I_T nexus loss, and if the persist through power loss capability is enabled, across any power cycle:

a) for SCSI transport protocols where initiator port names (see 3.1.76) are required, the initiator port name; otherwise, the initiator port identifier (see 3.1.75);
b) reservation key; and
c) indication of the target port to which the registration was applied.

The device server shall preserve the following information about the existing persistent reservation across any hard reset, logical unit reset, or I_T nexus loss, and if the persist through power loss capability is enabled, across any power cycle:

a) for SCSI transport protocols where initiator port names are required, the initiator port name; otherwise, the initiator port identifier;
b) reservation key;
c) scope;
d) type; and
e) indication of the target port through which the reservation was established.

NOTE 9 - The scope of a persistent reservation is always LU_SCOPE (see 6.13.3.3). For an all registrants type persistent reservation, only the scope and type need to be preserved.

5.9.5.2 Nonvolatile memory considerations for preserving persistent reservations and registrations

The capability of preserving persistent reservations and registrations across power cycles requires logical units to use nonvolatile memory within the SCSI device. Any logical unit that supports the persist through power loss capability of persistent reservation and has nonvolatile memory that is not ready shall allow the following commands into the task set:

a) INQUIRY;
b) LOG SENSE;
c) READ BUFFER;
d) REPORT LUNS;
e) REQUEST SENSE;
f) START STOP UNIT with the START bit set to one and POWER CONDITION field value of 0h (see SBC-3); and
g) WRITE BUFFER.

When nonvolatile memory has not become ready since a power cycle, commands other than those listed in this subclause shall be terminated with CHECK CONDITION status, with the sense key set to NOT READY, and the additional sense code set as described in table 284 (see 6.37).

5.9.6 Finding persistent reservations and reservation keys

5.9.6.1 Summary of commands for finding persistent reservations and reservation keys

The application client may obtain information about the persistent reservation and the reservation keys (i.e., registrations) that are present within a device server by issuing a PERSISTENT RESERVE IN command with a READ RESERVATION service action, a READ KEYS service action, or a READ FULL STATUS service action.

5.9.6.2 Reporting reservation keys

An application client may send a PERSISTENT RESERVE IN command with READ KEYS service action to determine if any I_T nexuses have been registered with a logical unit through any target port.

In response to a PERSISTENT RESERVE IN with READ KEYS service action the device server shall report the following:

a) the current PRgeneration value (see 6.13.2); and
b) the reservation key for every I_T nexus that is currently registered regardless of the target port through which the registration occurred.

The PRgeneration value allows the application client to verify that the configuration of the I_T nexuses registered with a logical unit has not been modified.

Duplicate reservation keys shall be reported if multiple I_T nexuses are registered using the same reservation key.

If an application client uses a different reservation key for each I_T nexus, the application client may use the reservation key to uniquely identify an I_T nexus.

5.9.6.3 Reporting the persistent reservation

An application client may send a PERSISTENT RESERVE IN command with READ RESERVATION service action to receive the persistent reservation information.

In response to a PERSISTENT RESERVE IN command with READ RESERVATION service action the device server shall report the following information for the persistent reservation, if any:

a) the current PRgeneration value (see 6.13.2);
b) the registered reservation key, if any, associated with the I_T nexus that holds the persistent reservation (see 5.9.10). If the persistent reservation is an all registrants type, the registered reservation key reported shall be zero; and
c) the scope and type of the persistent reservation, if any.

If an application client uses a different reservation key for each I_T nexus, the application client may use the reservation key to associate the persistent reservation with the I_T nexus that holds the persistent reservation. This association is done using techniques that are outside the scope of this standard.

5.9.6.4 Reporting full status

An application client may send a PERSISTENT RESERVE IN command with READ FULL STATUS service action to receive all information about registrations and the persistent reservation, if any.

In response to a PERSISTENT RESERVE IN command with READ FULL STATUS service action the device server shall report the current PRgeneration value (see 6.13.2) and, for every I_T nexus that is currently registered, the following information:

a) the registered reservation key;
b) whether the I_T nexus is a persistent reservation holder;
c) if the I_T nexus is a persistent reservation holder, the scope and type of the persistent reservation;
d) the relative target port identifier identifying the target port of the I_T nexus; and
e) a TransportID identifying the initiator port of the I_T nexus.

5.9.7 Registering

To establish a persistent reservation the application client shall first register an I_T nexus with the device server. An application client registers with a logical unit by issuing a PERSISTENT RESERVE OUT command with REGISTER service action or REGISTER AND IGNORE EXISTING KEY service action.

If the I_T nexus has an established registration, an application client may remove the reservation key (see 5.9.11.3). This is accomplished by issuing a PERSISTENT RESERVE OUT command with a REGISTER service action or a REGISTER AND IGNORE EXISTING KEY service action as shown in table 49 and table 50, respectively.

If an I_T nexus has not yet established a reservation key or the reservation key and registration have been removed, an application client may register that I_T nexus and zero or more specified unregistered I_T nexuses by issuing a PERSISTENT RESERVE OUT command with REGISTER service action as defined in table 49.

If the I_T nexus has an established registration, the application client may change the reservation key by issuing a PERSISTENT RESERVE OUT command with REGISTER service action as defined in table 49.

Table 49 — Register behaviors for a REGISTER service action

Command I_T nexus status	Parameter list fields [a]			Results
	RESERVATION KEY	SERVICE ACTION RESERVATION KEY	SPEC_I_PT	
received on an unregistered I_T nexus	zero	zero	ignore	Do nothing except return GOOD status.
		non-zero	zero	Register the I_T nexus on which the command was received with the value specified in the SERVICE ACTION RESERVATION KEY field.
			one	Register the I_T nexus on which the command was received and each unregistered I_T nexus specified in the parameter list with the value specified in the SERVICE ACTION RESERVATION KEY field. [b]
	non-zero	ignore	ignore	Return RESERVATION CONFLICT status.
received on a registered I_T nexus	Not equal to I_T nexus reservation key	ignore	ignore	Return RESERVATION CONFLICT status.
	Equal to I_T nexus reservation key	zero	zero	Unregister the I_T nexus on which the command was received (see 5.9.11.3).
			one	Return CHECK CONDITION status. [c]
		non-zero	zero	Change the reservation key of the I_T nexus on which the command was received to the value specified in the SERVICE ACTION RESERVATION KEY field.
			one	Return CHECK CONDITION status. [c]

[a] For requirements regarding the parameter list fields not shown in this table see 6.14.3.

[b] If any I_T nexus specified in the parameter list is registered, 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. Devices compliant with previous versions of this standard may return an additional sense code set to INVALID FIELD IN CDB.

[c] The sense key shall be set to ILLEGAL REQUEST, and the additional sense code shall be set to INVALID FIELD IN PARAMETER LIST. Devices compliant with previous versions of this standard may return an additional sense code set to INVALID FIELD IN CDB.

Alternatively, an application client may establish a reservation key for an I_T nexus without regard for whether one has previously been established by issuing a PERSISTENT RESERVE OUT command with REGISTER AND IGNORE EXISTING KEY service action as defined in table 50.

Table 50 — Register behaviors for a REGISTER AND IGNORE EXISTING KEY service action

Command I_T nexus status	Parameter list fields [a]	
	SERVICE ACTION RESERVATION KEY	Results
received on an unregistered I_T nexus	zero	Do nothing except return GOOD status.
	non-zero	Register the I_T nexus on which the command was received with the value specified in the SERVICE ACTION RESERVATION KEY field.
received on a registered I_T nexus	zero	Unregister the I_T nexus on which the command was received (see 5.9.11.3).
	non-zero	Change the reservation key of the I_T nexus on which the command was received to the value specified in the SERVICE ACTION RESERVATION KEY field.

[a] The RESERVATION KEY field is ignored when processing a REGISTER AND IGNORE EXISTING KEY service action. For requirements regarding other parameter list fields not shown in this table see 6.14.3.

If a PERSISTENT RESERVE OUT command with a REGISTER service action or a REGISTER AND IGNORE EXISTING KEY service action is attempted, but there are insufficient device server resources to complete the operation, 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 INSUFFICIENT REGISTRATION RESOURCES.

In response to a PERSISTENT RESERVE OUT command with a REGISTER service action or a REGISTER AND IGNORE EXISTING KEY service action the device server shall perform a registration for each specified I_T nexus by doing the following as an uninterrupted series of actions:

a) process the registration request regardless of any persistent reservations;
b) process the APTPL bit;
c) ignore the contents of the SCOPE and TYPE fields;
d) associate the reservation key specified in the SERVICE ACTION RESERVATION KEY field with the I_T nexus being registered, where:
   A) the I_T nexus(es) being registered are shown in table 51; and
   B) regardless of how the I_T nexus initiator port is specified, the association for the initiator port is based on either the initiator port name (see 3.1.76) on SCSI transport protocols where port names are required or the initiator port identifier (see 3.1.75) on SCSI transport protocols where port names are not required;
e) register the reservation key specified in the SERVICE ACTION RESERVATION KEY field without changing any persistent reservation that may exist; and
f) retain the reservation key specified in the SERVICE ACTION RESERVATION KEY field and associated information.

Table 51 — I_T Nexuses being registered

SPEC_I_PT [a]	ALL_TG_PT	I_T nexus(es) being registered	
		Initiator port	Target port
0	0	The port's names or identifiers to be registered are determined from the I_T nexus on which the PERSISTENT RESERVE OUT command was received	
0	1	The port's name or identifier to be registered is determined from the I_T nexus on which the PERSISTENT RESERVE OUT command was received	Register all of the target ports in the SCSI target device
1	0	a) The port's name or identifier to be registered is determined from the I_T nexus on which the PERSISTENT RESERVE OUT command was received; and b) Specified by each TransportID in the additional parameter data (see 6.14.3)	The port's name or identifier to be registered is determined from the I_T nexus on which the PERSISTENT RESERVE OUT command was received
1	1	a) The port's name or identifier to be registered is determined from the I_T nexus on which the PERSISTENT RESERVE OUT command was received; and b) Specified by each TransportID in the additional parameter data	Register all of the target ports in the SCSI target device

[a] If the SPEC_I_PT bit is set to one and the service action is REGISTER AND IGNORE EXISTING KEY, 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 PARAMETER LIST.

After the registration request has been processed, the device server shall then allow other PERSISTENT RESERVE OUT commands from the registered I_T nexus to be processed. The device server shall retain the reservation key until the key is changed as described in this subclause or removed as described in 5.9.11.

Any PERSISTENT RESERVE OUT command service action received from an unregistered I_T nexus, other than the REGISTER or the REGISTER AND IGNORE EXISTING KEY service action, shall be completed with RESERVATION CONFLICT status.

It is not an error for an I_T nexus that is registered to be registered again with the same reservation key or a new reservation key. A registration shall have no effect on any other registrations (e.g., when more than one I_T nexus is registered with the same reservation key and one of those I_T nexuses registers again it has no effect on the other I_T nexus' registrations). A registration that contains a non-zero value in the SERVICE ACTION RESERVATION KEY field shall have no effect on any persistent reservations (i.e., the reservation key for an I_T nexus may be changed without affecting any previously created persistent reservation).

Multiple I_T nexuses may be registered with the same reservation key. An application client may use the same reservation key for other I_T nexuses and logical units.

5.9.8 Registering and moving the reservation

The PERSISTENT RESERVE OUT command REGISTER AND MOVE service action is used to register a specified I_T nexus (see table 52) and move the reservation to establish that I_T nexus as the reservation holder.

Table 52 — Register behaviors for a REGISTER AND MOVE service action

Command I_T nexus status	Parameter list fields [a]			Results
	RESERVATION KEY	SERVICE ACTION RESERVATION KEY	UNREG	
received on an unregistered I_T nexus	ignore	ignore	ignore	Return RESERVATION CONFLICT status. [d]
received on the registered I_T nexus of reservation holder	Not equal to I_T nexus reservation key	ignore	ignore	Return RESERVATION CONFLICT status.
	Equal to I_T nexus reservation key	zero	ignore	Return CHECK CONDITION status. [b]
		non-zero [c]	zero	The I_T nexus on which PERSISTENT RESERVE OUT command was received shall remain registered. See this subclause for the registration and the move specifications.
			one	The I_T nexus on which PERSISTENT RESERVE OUT command was received shall be unregistered (see 5.9.11.3) upon completion of command processing. See this subclause for the registration and the move specifications.
received on a registered I_T nexus that is not the reservation holder	ignore	ignore	ignore	Return RESERVATION CONFLICT status.

[a] For requirements regarding other parameter list fields not shown in this table see 6.14.4.
[b] The sense key shall be set to ILLEGAL REQUEST, and the additional sense code shall be set to INVALID FIELD IN PARAMETER LIST. Devices compliant with previous versions of this standard may return an additional sense code set to INVALID FIELD IN CDB.
[c] The application client and backup application should use the same reservation key.
[d] Devices compliant with previous versions of this standard may return CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.

If a PERSISTENT RESERVE OUT command with a REGISTER AND MOVE service action is attempted, but there are insufficient device server resources to complete the operation, 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 INSUFFICIENT REGISTRATION RESOURCES.

If a PERSISTENT RESERVE OUT command with a REGISTER AND MOVE service action is received and the established persistent reservation is a Write Exclusive - All Registrants type or Exclusive Access - All Registrants type reservation, then the command shall be completed with RESERVATION CONFLICT status.

If a PERSISTENT RESERVE OUT command with a REGISTER AND MOVE service action is received and there is no persistent reservation established, 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 a PERSISTENT RESERVE OUT command with a REGISTER AND MOVE service action specifies a TransportID that is the same as the initiator port of the I_T nexus on which the command received, 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 PARAMETER LIST.

In response to a PERSISTENT RESERVE OUT command with a REGISTER AND MOVE service action the device server shall perform a register and move by doing the following as an uninterrupted series of actions:

a) process the APTPL bit;
b) ignore the contents of the SCOPE and TYPE fields;
c) associate the reservation key specified in the SERVICE ACTION RESERVATION KEY field with the I_T nexus specified as the destination of the register and move, where:
   A) the I_T nexus is specified by the TransportID and the RELATIVE TARGET PORT IDENTIFIER field (see 6.14.4); and
   B) regardless of the TransportID format used, the association for the initiator port is based on either the initiator port name (see 3.1.76) on SCSI transport protocols where port names are required or the initiator port identifier (see 3.1.75) on SCSI transport protocols where port names are not required;
d) register the reservation key specified in the SERVICE ACTION RESERVATION KEY field;
e) retain the reservation key specified in the SERVICE ACTION RESERVATION KEY field and associated information;
f) release the persistent reservation for the persistent reservation holder (i.e., the I_T nexus on which the command was received);
g) move the persistent reservation to the specified I_T nexus using the same scope and type as the persistent reservation released in item f); and
h) if the UNREG bit is set to one, unregister (see 5.9.11.3) the I_T nexus on which PERSISTENT RESERVE OUT command was received.

It is not an error for a REGISTER AND MOVE service action to register an I_T nexus that is already registered with the same reservation key or a different reservation key.

5.9.9 Reserving

An application client creates a persistent reservation by issuing a PERSISTENT RESERVE OUT command with RESERVE service action through a registered I_T nexus with the following parameters:

a) RESERVATION KEY set to the value of the reservation key that is registered with the logical unit for the I_T nexus; and
b) TYPE field and SCOPE field set to the persistent reservation being created.

Only one persistent reservation is allowed at a time per logical unit and that persistent reservation has a scope of LU_SCOPE.

If the device server receives a PERSISTENT RESERVE OUT command from an I_T nexus other than a persistent reservation holder (see 5.9.10) that attempts to create a persistent reservation when a persistent reservation already exists for the logical unit, then the command shall be completed with RESERVATION CONFLICT status.

If a persistent reservation holder attempts to modify the type or scope of an existing persistent reservation, the command shall be completed with RESERVATION CONFLICT status.

If the device server receives a PERSISTENT RESERVE OUT command with RESERVE service action where the TYPE field and the SCOPE field contain the same values as the existing type and scope from a persistent reservation holder, it shall not make any change to the existing persistent reservation and shall complete the command with GOOD status.

See 5.9.1 for information on when a persistent reservation takes effect.

5.9.10 Persistent reservation holder

The persistent reservation holder is determined by the type of the persistent reservation as follows:

a) for a persistent reservation of the type Write Exclusive – All Registrants or Exclusive Access – All Registrants, the persistent reservation holder is any registered I_T nexus; or
b) for all other persistent reservation types, the persistent reservation holder is the I_T nexus:
   A) for which the reservation was established with a PERSISTENT RESERVE OUT command with RESERVE service action, PREEMPT service action, or PREEMPT AND ABORT service action; or
   B) to which the reservation was moved by a PERSISTENT RESERVE OUT command with REGISTER AND MOVE service action.

A persistent reservation holder has its reservation key returned in the parameter data from a PERSISTENT RESERVE IN command with READ RESERVATION service action as follows:

a) for a persistent reservation of the type Write Exclusive – All Registrants or Exclusive Access – All Registrants, the reservation key shall be set to zero; or
b) for all other persistent reservation types, the reservation key shall be set to the registered reservation key for the I_T nexus that holds the persistent reservation.

It is not an error for a persistent reservation holder to send a PERSISTENT RESERVE OUT command with RESERVE service action to the reserved logical unit with TYPE and SCOPE fields that match those of the persistent reservation (see 5.9.9).

A persistent reservation holder is allowed to release the persistent reservation using the PERSISTENT RESERVE OUT command with RELEASE service action (see 5.9.11.2).

If the registration of the persistent reservation holder is removed (see 5.9.11.1), the reservation shall be released. If the persistent reservation holder is more than one I_T nexus, the reservation shall not be released until the registrations for all persistent reservation holder I_T nexuses are removed.

5.9.11 Releasing persistent reservations and removing registrations

5.9.11.1 Summary of service actions that release persistent reservations and remove registrations

An application client may release or preempt the persistent reservation by issuing one of the following commands through a registered I_T nexus with the RESERVATION KEY field set to the reservation key value that is registered with the logical unit for that I_T nexus:

a) a PERSISTENT RESERVE OUT command with RELEASE service action from a persistent reservation holder (see 5.9.11.2);
b) a PERSISTENT RESERVE OUT command with PREEMPT service action specifying the reservation key of the persistent reservation holder or holders (see 5.9.11.4);
c) a PERSISTENT RESERVE OUT command with PREEMPT AND ABORT service action specifying the reservation key of the persistent reservation holder or holders (see 5.9.11.5);
d) a PERSISTENT RESERVE OUT command with CLEAR service action (see 5.9.11.6); or

e) if the I_T nexus is the persistent reservation holder and the persistent reservation is not an all registrants type, then a PERSISTENT RESERVE OUT command with REGISTER service action or REGISTER AND IGNORE EXISTING KEY service action with the SERVICE ACTION RESERVATION KEY field set to zero (see 5.9.11.3).

Table 53 defines processing for a persistent reservation released or preempted by an application client based on the reservation type.

Table 53 — Processing for a released or preempted persistent reservation

Reservation Type	Processing
Write Exclusive – Registrants Only or Exclusive Access – Registrants Only	When the persistent reservation holder (see 5.9.10) of this reservation type becomes unregistered the persistent reservation shall be released.
Write Exclusive – All Registrants or Exclusive Access – All Registrants	This persistent reservation shall be released when the registration for the last registered I_T nexus is removed or when the type or scope is changed.
Write Exclusive or Exclusive Access	When the persistent reservation holder of this reservation type becomes unregistered the persistent reservation shall be released.

An application client may remove registrations by issuing one of the following commands through a registered I_T nexus with the RESERVATION KEY field set to the reservation key value that is registered with the logical unit for that I_T nexus:

a) a PERSISTENT RESERVE OUT command with PREEMPT service action with the SERVICE ACTION RESERVATION KEY field set to the reservation key (see 5.9.11.4) to be removed;
b) a PERSISTENT RESERVE OUT command with PREEMPT AND ABORT service action with the SERVICE ACTION RESERVATION KEY field set to the reservation key (see 5.9.11.5) to be removed;
c) a PERSISTENT RESERVE OUT command with CLEAR service action (see 5.9.11.6); or
d) a PERSISTENT RESERVE OUT command with REGISTER service action or REGISTER AND IGNORE EXISTING KEY service action with the SERVICE ACTION RESERVATION KEY field set to zero (see 5.9.11.3).

When a reservation key (i.e., registration) has been removed, no information shall be reported for that unregistered I_T nexus in subsequent READ KEYS service actions until the I_T nexus is registered again (see 5.9.7).

If the persist through power loss capability is not enabled, loss of power also causes persistent reservations to be released and registrations to be removed. When the most recent APTPL value received by the device server is zero (see 6.14.3), a power cycle:

a) releases all persistent reservations; and
b) removes all registered reservation keys (see 5.9.7).

5.9.11.2 Releasing

Only the persistent reservation holder (see 5.9.10) is allowed to release a persistent reservation.

An application client releases the persistent reservation by issuing a PERSISTENT RESERVE OUT command with RELEASE service action through an I_T nexus that is a persistent reservation holder with the following parameters:

a) RESERVATION KEY field set to the value of the reservation key that is registered with the logical unit for the I_T nexus; and
b) TYPE field and SCOPE field set to match the persistent reservation being released.

In response to a persistent reservation release request from the persistent reservation holder the device server shall perform a release by doing the following as an uninterrupted series of actions:

a) release the persistent reservation;
b) not remove any registration(s);
c) if the NUAR bit (see 7.5.7) is set to zero and the released persistent reservation is either a registrants only type or an all registrants type persistent reservation, then the device server shall establish a unit attention condition for the initiator port associated with every registered I_T nexus other than I_T nexus on which the PERSISTENT RESERVE OUT command with RELEASE service action was received, with the additional sense code set to RESERVATIONS RELEASED;
d) if the NUAR bit is set to one and the released persistent reservation is either a registrants only type or an all registrants type persistent reservation, then the device server shall not establish a unit attention condition; and
e) if the persistent reservation is of any other type, the device server shall not establish a unit attention condition.

The established persistent reservation shall not be altered and 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 RELEASE OF PERSISTENT RESERVATION, for a PERSISTENT RESERVE OUT command that specifies the release of a persistent reservation if:

a) the requesting I_T nexus is a persistent reservation holder (see 5.9.10); and
b) the SCOPE and TYPE fields do not match the scope and type of the established persistent reservation.

If there is no persistent reservation or in response to a persistent reservation release request from a registered I_T nexus that is not a persistent reservation holder (see 5.9.10), the device server shall do the following:

a) not release the persistent reservation, if any;
b) not remove any registrations; and
c) complete the command with GOOD status.

5.9.11.3 Unregistering

An application client may remove a registration for an I_T nexus by issuing a PERSISTENT RESERVE OUT command with REGISTER service action or a REGISTER AND IGNORE EXISTING KEY service action with the SERVICE ACTION RESERVATION KEY field set to zero through that I_T nexus.

If the I_T nexus is a reservation holder, the persistent reservation is of an all registrants type, and the I_T nexus is the last remaining registered I_T nexus, then the device server shall also release the persistent reservation.

If the I_T nexus is the reservation holder and the persistent reservation is of a type other than all registrants, the device server shall also release the persistent reservation. If the persistent reservation is a registrants only type, the device server shall establish a unit attention condition for the initiator port associated with every registered I_T nexus except for the I_T nexus on which the PERSISTENT RESERVE OUT command was received, with the additional sense code set to RESERVATIONS RELEASED.

5.9.11.4 Preempting

5.9.11.4.1 Overview

A PERSISTENT RESERVE OUT command with PREEMPT service action or PREEMPT AND ABORT service action is used to:

a) preempt (i.e., replace) the persistent reservation and remove registrations; or
b) remove registrations.

Table 54 lists the actions taken based on the current persistent reservation type and the SERVICE ACTION RESERVATION KEY field in the PERSISTENT RESERVE OUT command.

Table 54 — Preempting actions

Reservation Type	Service Action Reservation Key	Action	Reference
All Registrants	Zero	Preempt the persistent reservation and remove registrations.	5.9.11.4.3
	Not Zero	Remove registrations.	5.9.11.4.4
All other types	Zero	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.	
	Reservation holder's reservation key	Preempt the persistent reservation and remove registrations.	5.9.11.4.3
	Any other, non-zero reservation key	Remove registrations.	5.9.11.4.4

See figure 6 for a description of how a device server interprets a PREEMPT service action to determine its actions (e.g., preempt the persistent reservation, remove registration, or both preempt the persistent reservation and remove registration).

Figure 6 — Device server interpretation of PREEMPT service action

5.9.11.4.2 Failed persistent reservation preempt

If the preempting I_T nexus' PREEMPT service action or PREEMPT AND ABORT service action fails (e.g., repeated TASK SET FULL status, repeated BUSY status, SCSI transport protocol time-out, or time-out due to the task set being blocked due to failed initiator port or failed SCSI initiator device), the application client may send a LOGICAL UNIT RESET task management function to the failing logical unit to remove blocking commands and then resend the preempting service action.

5.9.11.4.3 Preempting persistent reservations and registration handling

An application client may preempt the persistent reservation with another persistent reservation by issuing a PERSISTENT RESERVE OUT command with PREEMPT service action or PREEMPT AND ABORT service action through a registered I_T nexus with the following parameters:

a) RESERVATION KEY field set to the value of the reservation key that is registered with the logical unit for the I_T nexus;
b) SERVICE ACTION RESERVATION KEY field set to the value of the reservation key of the persistent reservation to be preempted; and
c) TYPE field and SCOPE field set to define a new persistent reservation. The SCOPE and TYPE of the persistent reservation created by the preempting I_T nexus may be different than those of the persistent reservation being preempted.

If the SERVICE ACTION RESERVATION KEY field identifies a persistent reservation holder (see 5.9.10), the device server shall perform a preempt by doing the following as an uninterrupted series of actions:

a) release the persistent reservation for the holder identified by the SERVICE ACTION RESERVATION KEY field;
b) remove the registrations for all I_T nexuses identified by the SERVICE ACTION RESERVATION KEY field, except the I_T nexus that is being used for the PERSISTENT RESERVE OUT command. If an all registrants persistent reservation is present and the SERVICE ACTION RESERVATION KEY field is set to zero, then all registrations shall be removed except for that of the I_T nexus that is being used for the PERSISTENT RESERVE OUT command;
c) establish a persistent reservation for the preempting I_T nexus using the contents of the SCOPE and TYPE fields;
d) process commands as defined in 5.9.1;
e) establish a unit attention condition for the initiator port associated with every I_T nexus that lost its persistent reservation and/or registration, with the additional sense code set to REGISTRATIONS PREEMPTED; and
f) if the type or scope has changed, then for every I_T nexus whose reservation key was not removed, except for the I_T nexus on which the PERSISTENT RESERVE OUT command was received, the device server shall establish a unit attention condition for the initiator port associated with that I_T nexus, with the additional sense code set to RESERVATIONS RELEASED. If the type or scope have not changed, then no unit attention condition(s) shall be established for this reason.

After the PERSISTENT RESERVE OUT command has been completed with GOOD status, new commands are subject to the persistent reservation restrictions established by the preempting I_T nexus.

The following commands shall be subjected in a vendor specific manner either to the restrictions established by the persistent reservation being preempted or to the restrictions established by the preempting I_T nexus:

a) a command received after the arrival, but before the completion of the PERSISTENT RESERVE OUT command with the PREEMPT service action or the PREEMPT AND ABORT service action; or
b) a command in the dormant command state, blocked command state, or enabled command state (see SAM-4) at the time the PERSISTENT RESERVE OUT command with the PREEMPT service action or the PREEMPT AND ABORT service action is received.

Completion status shall be returned for each command unless it was aborted by a PERSISTENT RESERVE OUT command with the PREEMPT AND ABORT service action and TAS bit set to zero in the Control mode page (see 7.5.7).

If an all registrants persistent reservation is not present, it is not an error for the persistent reservation holder to preempt itself (i.e., a PERSISTENT RESERVE OUT with a PREEMPT service action or a PREEMPT AND ABORT service action with the SERVICE ACTION RESERVATION KEY value equal to the persistent reservation holder's reservation key that is received from the persistent reservation holder). In that case, the device server shall establish the new persistent reservation and maintain the registration.

5.9.11.4.4 Removing registrations

When a registered reservation key does not identify a persistent reservation holder (see 5.9.10), an application client may remove the registration(s) without affecting any persistent reservations by issuing a PERSISTENT RESERVE OUT command with PREEMPT service action through a registered I_T nexus with the following parameters:

a) RESERVATION KEY field set to the value of the reservation key that is registered for the I_T nexus; and
b) SERVICE ACTION RESERVATION KEY field set to match the reservation key of the registration or registrations being removed.

If the SERVICE ACTION RESERVATION KEY field does not identify a persistent reservation holder or there is no persistent reservation holder (i.e., there is no persistent reservation), then the device server shall perform a preempt by doing the following in an uninterrupted series of actions:

a) remove the registrations for all I_T nexuses specified by the SERVICE ACTION RESERVATION KEY field;
b) ignore the contents of the SCOPE and TYPE fields;
c) process commands as defined in 5.9.1; and
d) establish a unit attention condition for the initiator port associated with every I_T nexus that lost its registration other than the I_T nexus on which the PERSISTENT RESERVE OUT command was received, with the additional sense code set to REGISTRATIONS PREEMPTED.

If a PERSISTENT RESERVE OUT with a PREEMPT service action or a PREEMPT AND ABORT service action sets the SERVICE ACTION RESERVATION KEY field to a value that does not match any registered reservation key, then the device server shall complete the command with RESERVATION CONFLICT status.

It is not an error for a PERSISTENT RESERVE OUT with a PREEMPT service action or a PREEMPT AND ABORT service action to set the RESERVATION KEY and the SERVICE ACTION RESERVATION KEY to the same value, however, no unit attention condition is established for the I_T nexus on which the PERSISTENT RESERVE OUT command was received. The registration is removed.

5.9.11.5 Preempting and aborting

The application client's request for and the device server's responses to a PERSISTENT RESERVE OUT command PREEMPT AND ABORT service action are identical to the responses to a PREEMPT service action (see 5.9.11.4) except for the additions described in this subclause. If no reservation conflict occurred, the device server shall perform the following uninterrupted series of actions:

a) if the persistent reservation is not an all registrants type then:
   A) if the TST field is 000b (see 7.5.7) and the faulted I_T nexus (see 3.1.58), if any, is not the I_T nexus associated with the persistent reservation or registration being preempted, then the task set ACA condition shall be processed as defined in SAM-4;

B) if the TST field contains 000b and the faulted I_T nexus, if any, is the I_T nexus associated with the persistent reservation or registration being preempted, then the PERSISTENT RESERVE OUT command shall be processed without regard for the task set ACA condition; or
C) if the TST field contains 001b, then the ACA condition shall be processed as defined in SAM-4;

b) perform the uninterrupted series of actions described for the PREEMPT service action (see 5.9.11.4);
c) all commands from the I_T nexus(es) associated with the persistent reservations or registrations being preempted (i.e., preempted commands) except the PERSISTENT RESERVE OUT command itself shall be aborted as defined in SAM-4. If an aborted command is a command that causes the device server or copy manager to generate additional commands and data transfers (e.g., the EXTENDED COPY command (see 6.3)), then all commands and data transfers generated by the command shall be aborted before the ABORT TASK SET task management function is considered completed. After the ABORT TASK SET function has completed, all new commands are subject to the persistent reservation restrictions established by the preempting I_T nexus;
d) if the persistent reservation is not an all registrants type, then the device server shall clear any ACA condition associated with an I_T nexus being preempted and shall abort any commands with an ACA attribute received on that I_T nexus;
e) if the persistent reservation is an all registrants type, then:
   A) if the service action reservation key is set to zero, the device server shall clear any ACA condition and shall abort any commands with an ACA attribute; or
   B) if the service action reservation key is not set to zero, the device server shall do the following for any I_T nexus registered using the specified reservation key:
      a) clear any ACA condition; and
      b) abort any commands with an ACA attribute;

   and
f) for logical units that implement the PREVENT ALLOW MEDIUM REMOVAL command (see SBC-3, SSC-3, and SMC-3), the device server shall perform an action equivalent to the processing of a PREVENT ALLOW MEDIUM REMOVAL command with the PREVENT field equal to zero received on the I_T nexuses associated with the persistent reservation being preempted.

The actions described in this subclause shall be performed for all I_T nexuses that are registered with the non-zero SERVICE ACTION RESERVATION KEY value, without regard for whether the preempted I_T nexuses hold the persistent reservation. If the SERVICE ACTION RESERVATION KEY value is zero and an all registrants persistent reservation is present, the device server shall abort all commands for all registered I_T nexuses.

5.9.11.6 Clearing

Any application client may release the persistent reservation and remove all registrations from a device server by issuing a PERSISTENT RESERVE OUT command with CLEAR service action through a registered I_T nexus with the following parameter:

a) RESERVATION KEY field set to the value of the reservation key that is registered with the logical unit for the I_T nexus.

In response to this request the device server shall perform a clear by doing the following as part of an uninterrupted series of actions:

a) release the persistent reservation, if any;
b) remove all registration(s) (see 5.9.7);
c) ignore the contents of the SCOPE and TYPE fields;
d) continue normal processing of any commands from any I_T nexus that have been accepted by the device server as allowed (i.e., nonconflicting); and
e) establish a unit attention condition for the initiator port associated with every registered I_T nexus other than the I_T nexus on which the PERSISTENT RESERVE OUT command with CLEAR service action was received, with the additional sense code set to RESERVATIONS PREEMPTED.

NOTE 10 - Application clients should not use the CLEAR service action except during recovery operations that are associated with a specific initiator port, since the effect of the CLEAR service action defeats the persistent reservations features that protect data integrity.

5.10 Multiple target port and initiator port behavior

SAM-4 specifies the behavior of logical units being accessed by application clients through more than one initiator port and/or through more than one target port. Additional initiator ports and target ports allow the definition of multiple I_T nexuses through which the device server may be reached. Multiple I_T nexuses may be used to improve the availability of logical units in the presence of certain types of failures and to improve the performance between an application client and logical unit when some I_T nexuses may be busy.

If one target port is being used by an initiator port, accesses attempted through other target port(s) may:

a) receive a status of BUSY; or
b) be accepted as if the other target port(s) were not in use.

The device server shall indicate the presence of multiple target ports by setting the MULTIP bit to one in its standard INQUIRY data.

Only the following operations allow one I_T nexus to interact with the commands of other I_T nexuses:

a) the PERSISTENT RESERVE OUT with PREEMPT service action preempts persistent reservations (see 5.9.11.4);
b) the PERSISTENT RESERVE OUT with PREEMPT AND ABORT service action preempts and aborts persistent reservations (see 5.9.11.5);
c) The PERSISTENT RESERVE OUT with CLEAR service action releases persistent reservations for all I_T nexuses (see 5.9.11.6); and
d) commands and task management functions that allow one I_T nexus to abort commands received on a different I_T nexus (see SAM-4).

5.11 Target port group access states

5.11.1 Target port group access overview

Logical units may be connected to one or more service delivery subsystems via multiple target ports (see SAM-4). The access to logical units through the multiple target ports may be symmetrical (see 5.11.3) or asymmetrical (see 5.11.2).

If referrals are supported (see SBC-3), then a logical unit accessed through a target port group may have different target port group access states based on the user data segments being accessed.

5.11.2 Asymmetric logical unit access

5.11.2.1 Introduction to asymmetric logical unit access

Asymmetric logical unit access occurs when the access characteristics of one port may differ from those of another port. SCSI target devices with target ports implemented in separate physical units may need to designate differing levels of access for the target ports associated with each logical unit. While commands and task management functions (see SAM-4) may be routed to a logical unit through any target port, the performance may not be optimal, and the allowable command set may be less complete than when the same commands and task management

functions are routed through a different target port. In addition, some target ports may be in a state (e.g., offline) that is unique to that target port. When a failure on the path to one target port is detected, the SCSI target device may perform automatic internal reconfiguration to make a logical unit accessible from a different set of target ports or may be instructed by the application client to make a logical unit accessible from a different set of target ports.

A target port characteristic called primary target port asymmetric access state (see 5.11.2.4) defines properties of a target port and the allowable command set for a logical unit when commands and task management functions are routed through the target port maintaining that state.

A primary target port group is defined as a set of target ports that are in the same primary target port asymmetric access state at all times (i.e., a change in one target port's primary target port asymmetric access state implies an equivalent change in the primary target port asymmetric access state of all target ports in the same primary target port group). A primary target port group asymmetric access state is defined as the primary target port asymmetric access state common to the set of target ports in a primary target port group. One target port is a member of at most one primary target port group for a logical unit group (see 7.8.6.9). The grouping of target ports in a primary target port group is vendor specific.

A logical unit may have commands and task management functions routed through multiple primary target port groups. Logical units support asymmetric logical unit access if different primary target port groups may be in different primary target port group asymmetric access states.

An example of asymmetric logical unit access is a SCSI controller device with two separated controllers where all target ports on one controller are in the same primary target port asymmetric access state with respect to a logical unit and are members of the same primary target port group. Target ports on the other controller are members of another primary target port group. The behavior of each primary target port group may be different with respect to a logical unit, but all members of a single primary target port group are always in the same primary target port group asymmetric access state with respect to a logical unit.

An example of primary target port groups is shown in figure 7.

Figure 7 — Primary target port group example

Another target port characteristic called secondary target port asymmetric access state (see 5.11.2.4) indicates a condition that affects the way in which an individual target port participates in its assigned primary target port group. All target ports, if any, in one secondary target port asymmetric access state are grouped into a secondary target port group. Secondary target port groups have the following properties:

a) a target port in any secondary target port group also shall be in one primary target port group;

b) a change of secondary target port asymmetric access state for one target port shall not cause changes in the secondary target port asymmetric access state of other target ports, if any, in the same secondary target port group; and
c) a target port may be a member of zero or more secondary target port groups.

The term target port asymmetric access state represents both primary target port asymmetric access states and secondary target port asymmetric access states. The term target port group represents both primary target port groups and secondary target port groups.

5.11.2.2 Explicit and implicit asymmetric logical unit access

Asymmetric logical unit access may be managed explicitly by an application client using the REPORT TARGET PORT GROUPS (see 6.27) and SET TARGET PORT GROUPS (see 6.35) commands.

Alternatively, asymmetric logical unit access may be managed implicitly by the SCSI target device based on the type of transactions being routed through each target port and the internal configuration capabilities of the primary target port group(s) through which the logical unit may be accessed. The logical units may attempt to maintain full performance across the primary target port groups that are busiest and that show the most reliable performance, allowing other primary target port groups to select a lower performance primary target port asymmetric access state.

Implicit management of secondary target port asymmetric access states is based on the condition of an individual target port and how such conditions affect that target port's ability to participate in its assigned primary target port group.

If both explicit and implicit asymmetric logical unit access management methods are implemented, the precedence of one over the other is vendor specific.

5.11.2.3 Discovery of asymmetric logical unit access behavior

SCSI logical units with asymmetric logical unit access may be identified using the INQUIRY command. The value in the target port group support (TPGS) field (see 6.4.2) indicates whether or not the logical unit supports asymmetric logical unit access and if so whether implicit or explicit management is supported. The target port asymmetric access states supported by a logical unit may be determined by the REPORT TARGET PORT GROUPS command parameter data (see 6.27).

5.11.2.4 Target port asymmetric access states

5.11.2.4.1 Target port asymmetric access states overview

For all SCSI target devices that report in the INQUIRY data that they support asymmetric logical unit access, all of the target ports in a primary target port group (see 5.11.2.1) shall be in the same primary target port asymmetric access state with respect to the ability to route information to a logical unit. The primary target port asymmetric access states are:

a) active/optimized;
b) active/non-optimized;
c) standby;
d) unavailable; and
e) logical block dependent.

Individual target ports may be in secondary target port groups (see 5.11.2.1) that have the following secondary target port asymmetric access states:

a) offline.

5.11.2.4.2 Active/optimized state

The active/optimized state is a primary target port asymmetric access state (see 3.1.114). When commands and task management functions are being routed through a target port in the active/optimized primary target port asymmetric access state, the device server shall function (e.g., respond to commands) as specified in the appropriate command standards (see 3.1.31). All target ports within a primary target port group should be capable of immediately accessing the logical unit.

The SCSI target device shall participate in all task management functions as defined in SAM-4 and modified by the applicable SCSI transport protocol standards (see 3.1.147).

5.11.2.4.3 Active/non-optimized state

The active/non-optimized state is a primary target port asymmetric access state (see 3.1.114). When commands and task management functions are being routed through a target port in the active/non-optimized primary target port asymmetric access state, the device server shall function as specified in the appropriate command standards.

The processing of some task management functions and commands, especially those involving data transfer or caching, may operate with lower performance than they would if the target port were in the active/optimized primary target port asymmetric access state.

The SCSI target device shall participate in all task management functions as defined in SAM-4 and modified by the applicable SCSI transport protocol standards (see 3.1.147).

5.11.2.4.4 Standby state

The standby state is a primary target port asymmetric access state (see 3.1.114). When being accessed through a target port in the standby primary target port asymmetric access state, the device server shall support those of the following commands that it supports while in the active/optimized primary target port asymmetric access state:

a) INQUIRY;
b) LOG SELECT;
c) LOG SENSE;
d) MODE SELECT;
e) MODE SENSE;
f) REPORT LUNS;
g) RECEIVE DIAGNOSTIC RESULTS;
h) SEND DIAGNOSTIC;
i) REPORT TARGET PORT GROUPS;
j) SET TARGET PORT GROUPS;
k) REQUEST SENSE;
l) PERSISTENT RESERVE IN;
m) PERSISTENT RESERVE OUT;
n) echo buffer modes of READ BUFFER; and
o) echo buffer modes of WRITE BUFFER.

The device server may support other commands.

For those commands that are not supported, the device server shall terminate the command with CHECK CONDITION status, with the sense key set to NOT READY, and the additional sense code set to LOGICAL UNIT NOT ACCESSIBLE, TARGET PORT IN STANDBY STATE.

The SCSI target device shall participate in all task management functions as defined in SAM-4 and modified by the applicable SCSI transport protocol standards (see 3.1.147).

5.11.2.4.5 Unavailable state

The unavailable state is a primary target port asymmetric access state (see 3.1.114). When being accessed through a target port in the unavailable primary target port asymmetric access state, the device server shall accept only a limited set of commands. The unavailable primary target port asymmetric access state is intended for situations when the target port accessibility to a logical unit may be severely constrained due to SCSI target device limitations (e.g., hardware errors). Therefore it may not be possible to transition from this state to either the active/optimized, active/non-optimized or standby states. The unavailable primary target port asymmetric access state is also intended for minimizing any disruption when using the downloading microcode mode of the WRITE BUFFER command.

While in the unavailable primary target port asymmetric access state, the device server shall support those of the following commands that it supports while in the active/optimized state:

a) INQUIRY (the peripheral qualifier (see 6.4.2) shall be set to 001b);
b) REPORT LUNS;
c) REPORT TARGET PORT GROUPS;
d) SET TARGET PORT GROUPS;
e) REQUEST SENSE;
f) echo buffer modes of READ BUFFER;
g) echo buffer modes of WRITE BUFFER; and
h) download microcode mode of WRITE BUFFER.

The device server may support other commands.

For those commands that are not supported, the device server shall terminate the command with CHECK CONDITION status, with the sense key set to NOT READY, and the additional sense code set to LOGICAL UNIT NOT ACCESSIBLE, TARGET PORT IN UNAVAILABLE STATE.

The SCSI target device is not required to participate in all task management functions (see SAM-4 and the applicable SCSI transport protocol standards).

5.11.2.4.6 Offline state

The offline state is a secondary target port asymmetric access state (see 3.1.148). Target ports in the offline secondary target port asymmetric access state are not accessible via the service delivery subsystem (e.g. during maintenance, port replacement, port disabled, hot swap, or power failures impacting only some target port). While in the offline secondary target port asymmetric state, the target port is not capable of receiving or responding to any commands or task management functions.

The offline secondary target port asymmetric access state allows a device server to report that some target ports are not capable of being accessed.

After access to the service delivery subsystem is enabled, the target port shall transition out of the offline secondary target port asymmetric access state.

5.11.2.4.7 Logical block dependent state

The logical block dependent state only occurs if the device server supports referrals (see 7.8.7).

The target port asymmetric access state for a user data segment shall be one of the following target port asymmetric access states:

a) active/optimized;
b) active/non-optimized;
c) transitioning; or
d) unavailable.

An application client may determine the target port asymmetric access state for user data segments by issuing a REPORT REFERRALS command (see SBC-3).

5.11.2.5 Transitions between target port asymmetric access states

The movement from one target port asymmetric access state to another is called a transition.

During a transition between target port asymmetric access states the device server shall respond to a command in one of the following ways:

a) if during the transition the logical unit is inaccessible, then the transition is performed as a single indivisible event and the device server shall respond by either returning BUSY status, or returning CHECK CONDITION status, with the sense key set to NOT READY, and the sense code set to LOGICAL UNIT NOT ACCESSIBLE, ASYMMETRIC ACCESS STATE TRANSITION; or
b) if during the transition the target ports in a primary target port group are able to access the requested logical unit, then the device server shall support those of the following commands that it supports while in the active/optimized primary target port asymmetric access state:
   A) INQUIRY;
   B) REPORT LUNS;
   C) REPORT TARGET PORT GROUPS;
   D) REQUEST SENSE;
   E) echo buffer modes of READ BUFFER; and
   F) echo buffer modes of WRITE BUFFER.

   The device server may support other commands when those commands are routed though a target port that is transitioning between primary target port asymmetric access states.

   For those commands that are not supported during a transition, the device server shall terminate the command with CHECK CONDITION status, with the sense key set to NOT READY, and the additional sense code set to LOGICAL UNIT NOT ACCESSIBLE, ASYMMETRIC ACCESS STATE TRANSITION.

   The SCSI target device is not required to participate in all task management functions.

If the transition was explicit to a supported target port asymmetric access state and it failed, then the command shall be terminated with CHECK CONDITION status, with the sense key set to HARDWARE ERROR, and the additional sense code set to SET TARGET PORT GROUPS COMMAND FAILED. If the transition was to a primary target port asymmetric access stated, the primary target port group that encountered the error should complete a transition to the unavailable primary target port asymmetric access state.

If a target port asymmetric access state change occurred as a result of the failed transition, then the device server shall establish a unit attention condition for the initiator port associated with every I_T nexus other than the I_T nexus on which the SET TARGET PORT GROUPS command was received with the additional sense code set to ASYMMETRIC ACCESS STATE CHANGED.

If the transition was implicit and it failed, then the device server shall establish a unit attention condition for the initiator port associated with every I_T nexus with the additional sense code set to IMPLICIT ASYMMETRIC ACCESS STATE TRANSITION FAILED.

An implicit CLEAR TASK SET task management function may be performed following a transition failure.

Once a transition is completed, the new target port asymmetric access state may apply to some or all commands entered into the task set before the completion of the transition. The new target port asymmetric access state shall apply to all commands received by the device server after completion of a transition.

If a transition is to the offline secondary target port asymmetric access state, communication with the service delivery subsystem shall be terminated. This may result in commands being terminated and may cause command timeouts to occur on the initiator.

After an implicit target port asymmetric access state change, a device server shall establish a unit attention condition for the initiator port associated with every I_T nexus with the additional sense code set to ASYMMETRIC ACCESS STATE CHANGED.

After an explicit target port asymmetric access state change, a device server shall establish a unit attention condition with the additional sense code set to ASYMMETRIC ACCESS STATE CHANGED for the initiator port associated with every I_T nexus other than the I_T nexus on which the SET TARGET PORT GROUPS command was received.

5.11.2.6 Preference Indicator

A device server may indicate one or more primary target port groups is a preferred primary target port group for accessing a logical unit by setting the PREF bit to one in the target port group descriptor (see 6.27). The preference indication is independent of the primary target port asymmetric access state.

An application client may use the PREF bit value in the target port group descriptor to influence the path selected to a logical unit (e.g., a primary target port group in the standby primary target port asymmetric access state with the PREF bit set to one may be chosen over a primary target port group in the active/optimized primary target port asymmetric access state with the PREF bit set to zero).

The value of the PREF bit for a primary target port group may change whenever an primary target port asymmetric access state changes.

5.11.2.7 Implicit asymmetric logical units access management

SCSI target devices with implicit asymmetric logical units access management are capable using mechanisms other than the SET TARGET PORT GROUPS command to set the:

a) primary target port asymmetric access state of a primary target port group; or
b) secondary target port asymmetric access state of a target port that is a member of a primary target port group.

All logical units that report in the standard INQUIRY data (see 6.4.2) that they support asymmetric logical units access and support implicit asymmetric logical unit access (i.e., the TPGS field contains 01b or 11b):

a) shall implement the INQUIRY command Device Identification VPD page designator types 4h (see 7.8.6.7) and 5h (see 7.8.6.8);
b) shall support the REPORT TARGET PORT GROUPS command as described in 6.27; and
c) may implement the INQUIRY command Device Identification VPD page designator type 6h (see 7.8.6.9).

Implicit logical unit access state changes between primary target port asymmetric access states (see 3.1.114) may be disabled with the IALUAE bit in the Control Extension mode page (see 7.5.8).

5.11.2.8 Explicit asymmetric logical units access management

All logical units that report in the standard INQUIRY data (see 6.4.2) that they support asymmetric logical units access and support explicit asymmetric logical unit access (i.e., the TPGS field contains 10b or 11b):

a) shall implement the INQUIRY command Device Identification VPD page (see 7.8.6) designator types 4h and 5h;
b) shall support the REPORT TARGET PORT GROUPS command as described in 6.27;
c) shall support the SET TARGET PORT GROUPS command as described in 6.35; and
d) may implement the INQUIRY command Device Identification VPD page designator type 6h (see 7.8.6.9).

5.11.2.9 Behavior after power on, hard reset, logical unit reset, and I_T nexus loss

For all SCSI target devices that report in the standard INQUIRY data (see 6.4.2) that they support only explicit asymmetric logical unit access (i.e., the TPGS field contains 10b), the target port shall preserve the primary target port asymmetric access state during any power on, hard reset, logical unit reset, and I_T nexus loss.

5.11.2.10 Behavior of target ports that are not accessible from the service delivery subsystem

If the offline secondary target port asymmetric access state is supported and a subset of the target ports in a primary target port group are not accessible via the service delivery subsystem (e.g. power failure), then those ports may be reported in a primary target port group consistent with their primary target port asymmetric access state and in the secondary target port group with the offline secondary target port asymmetric access state.

5.11.3 Symmetric logical unit access

A device server that provides symmetrical access to a logical unit may use a subset of the asymmetrical logical access features (see 5.11.2) to indicate this ability to an application client, providing an application client a common set of commands to determine how to manage target port access to a logical unit.

Symmetrical logical unit access should be represented as follows:

a) the TPGS field in the standard INQUIRY data (see 6.4.2) indicates that implicit asymmetric access is supported;
b) the REPORT TARGET PORT GROUPS command is supported; and
c) the REPORT TARGET PORT GROUPS parameter data indicates that the same state (e.g., active/optimized state) is in effect for all primary target port groups.

5.12 Power management

5.12.1 Power management overview

The Power Condition mode page (see 7.5.12) allows an application client to manage the power condition of a logical unit in a manner that may reduce power consumption of the SCSI target device. This control is invoked by enabling and initializing one or more idle condition timers and/or standby condition timers based on Power Condition mode page values. A change in the power condition of any logical unit in a SCSI target device may result in a change in the SCSI target device's power consumption. If a SCSI target device contains multiple logical units, then the SCSI target device’s power consumption may not decrease until a group of the logical units have changed

to a lower power condition. Any grouping or groupings of logical units for power condition management is outside the scope of this standard.

Command standards may define the following additional power management features:

a) power conditions (e.g., the stopped power condition in SBC-3);
b) changes to the power conditions state machine (e.g., the SSU_PC8:Stopped state in SBC-3);
c) commands that manage power conditions (e.g., a START STOP UNIT command in SBC-3 or MMC-6);
d) changes to commands defined in this standard; or
e) mode pages, log pages, or VPD pages that are associated with managing power conditions.

Transport protocol standards (e.g., SPL) may define additional requirements on the states in the power condition state machine defined in this standard or a command standard.

There shall be no notification to the application client that a logical unit has changed from one power condition to another. The response to a REQUEST SENSE command (see 6.29) may indicate whether a logical unit is in a low power condition and which low power condition.

The current power condition of a logical unit may be decreased by:

a) the expiration of a power condition timer;
b) the completion of background functions; or
c) power condition activities described in a command standard.

The current power condition of a logical unit is increased by:

a) the processing of a command that the device server is unable to continue processing while in the current power condition;
b) the processing of a background function that the device server is unable to process while in the current power condition; or
c) power condition activities described in a command standard.

If a device server processes a command that the device server is capable of completing while the logical unit is in a low power condition, then the device server shall not stop any enabled power condition timers, regardless of which power condition the logical unit was in when the device server began processing the command.

If a device server processes a command that the device server is not capable of completing while the logical unit is in a low power condition, then the device server shall stop any running power condition timers. On completion of the command, the device server shall reinitialize all enabled power condition timers based on their values in the Power Condition mode page (see 7.5.12) and start the timers, regardless of which power condition the logical unit was in when the device server began processing the command.

The device server shall process any task management function (see SAM-5), except LOGICAL UNIT RESET, regardless of current power condition, without changing to a different power condition. The power condition timers shall not be affected by task management functions, except LOGICAL UNIT RESET.

The device server may change power conditions or power condition timers when processing a LOGICAL UNIT RESET.

No power condition defined in this standard shall affect the supply of any power required for proper operation of a service delivery subsystem.

Logical units that contain cache memory shall write all cached data to the medium for the logical unit (e.g., as a logical unit would do in response to a SYNCHRONIZE CACHE command as described in SBC-3) prior to entering into any power condition that prevents accessing the media (e.g., before a hard drive stops its spindle motor during a change to the standby power condition).

5.12.2 Active power condition

While in the active power condition:

a) the device server is capable of completing the processing of its supported commands, including those that require media access, without the logical unit changing power condition;
b) the device server completes processing of a command in the shortest time when compared to the time required for completion of that command if command processing began while the logical unit was in any of the idle power conditions or standby power conditions; and
c) the SCSI target device may consume more power than when the logical unit is in any of the idle power conditions or standby power conditions (e.g., a disk drive's spindle motor may be active).

5.12.3 Idle power conditions

A device server may support more than one idle power condition (i.e., idle_a, idle_b, and idle_c) to provide progressively lower power consumption (i.e., the following power consumption relationship: idle_a ≥ idle_b ≥ idle_c).

NOTE 11 - The idle_a power condition was referenced as the idle power condition in SPC-3.

While in one of the idle power conditions:

a) the device server is capable of completing the processing of its supported commands, except those that require the logical unit to be in the active power condition to be capable of completing the command with GOOD status (e.g., commands that require media access to complete processing);
b) the device server may take longer to complete processing a command than when the logical unit is in the active power condition (e.g., the device may have to activate some circuitry before completing processing of a command);
c) the power consumed by the SCSI target device while in an idle power condition should be less than the power consumed when the logical unit is in the active power condition and may be greater than the power consumed when the logical unit is in a standby power condition; and
d) the peak power consumption during a change from an idle power condition to the active power condition shall be no more than the typical peak power consumption in the active power condition.

5.12.4 Standby power conditions

A device server may support more than one standby power condition (i.e., standby_y and standby_z) to provide progressively lower power consumption (i.e., the following power consumption relationship: standby_y ≥ standby_z).

NOTE 12 - The standby_z power condition was referenced as the standby power condition in SPC-3.

While in one of the standby power conditions:

a) the device server is not capable of completing the processing of commands that require media access without the logical unit changing to the active power condition. SCSI transport protocol standards may impose additional requirements on command processing when changing to a higher power condition (e.g., the response may be CHECK CONDITION status with sense key set to NOT READY and additional sense

bytes set to LOGICAL UNIT NOT READY, NOTIFY (ENABLE SPINUP) REQUIRED instead of GOOD status described in SPL);

b) the device server may take longer to complete processing a command than when the logical unit is in the active power condition or one of the idle power conditions (e.g., a disk drive's spindle motor may need to be started);

c) the power consumed by the SCSI target device while in one of the standby power conditions should be less than the power consumed when the logical unit is in the active power condition or any of the idle power conditions; and

d) the peak power consumption during a change from a standby power condition to the active power condition or an idle power condition is not limited by this standard.

5.12.5 Power management pollable sense data

If the logical unit is in any power condition other than active, the following data shall be available for use by the REQUEST SENSE command when returning pollable sense data (see 5.6.2) and:

a) if the logical unit is in an idle power condition (see 5.12.3), then the sense key shall be set to NO SENSE and the additional sense code set to one of the following:
   A) LOW POWER CONDITION ON if the reason for entry into the idle power condition is unknown;
   B) IDLE CONDITION ACTIVATED BY TIMER if the logical unit entered the idle_a power condition due to the idle_a condition timer (see 5.12.6.4);
   C) IDLE CONDITION ACTIVATED BY COMMAND if the logical unit entered the idle_a power condition due to processing of a command;
   D) IDLE_B CONDITION ACTIVATED BY TIMER if the logical unit entered the idle_b power condition due to the idle_b condition timer (see 5.12.6.4);
   E) IDLE_B CONDITION ACTIVATED BY COMMAND if the logical unit entered the idle_b power condition due to processing of a command;
   F) IDLE_C CONDITION ACTIVATED BY TIMER if the logical unit entered the idle_c power condition due to the idle_c condition timer (see 5.12.6.4); or
   G) IDLE_C CONDITION ACTIVATED BY COMMAND if the logical unit entered the idle_c power condition due to processing of a command;

b) if the logical unit is in a standby power condition (see 5.12.4), then the sense key shall be set to NO SENSE and the additional sense code set to one of the following:
   A) LOW POWER CONDITION ON if the reason for entry into the standby power condition is unknown;
   B) STANDBY_Y CONDITION ACTIVATED BY TIMER if the logical unit entered the standby_y power condition due to the standby_y condition timer (see 5.12.6.5);
   C) STANDBY_Y CONDITION ACTIVATED BY COMMAND if the logical unit entered the standby_y power condition due to processing of a command;
   D) STANDBY CONDITION ACTIVATED BY TIMER if the logical unit entered the standby_z power condition due to the standby_z condition timer (see 5.12.6.5); or
   E) STANDBY_Z CONDITION ACTIVATED BY COMMAND if the logical unit entered the standby_y power condition due to processing of a command;

   or

c) if the logical unit is in the stopped power condition (see SBC-3), then the sense key shall be set to NOT READY and the additional sense code shall be set to LOGICAL UNIT NOT READY, INITIALIZING COMMAND REQUIRED.

NOTE 13 - Device servers that conform to SBC-2 may not provide the pollable sense data described in c).

5.12.6 Power condition state machine

5.12.6.1 Power condition state machine overview

The PC (power condition) state machine describes the logical unit power states and transitions resulting from command processing and Power Condition mode page values (see 7.5.12).

The PC states are shown in table 55.

Table 55 — PC state machine states

State	Reference
PC0:Powered_On [a]	5.12.6.2
PC1:Active	5.12.6.3
PC2:Idle	5.12.6.4
PC3:Standby	5.12.6.5
PC4:Active_Wait	5.12.6.6
PC5:Wait_Idle	5.12.6.7
PC6:Wait_Standby	5.12.6.8

[a] PC0:Powered_On is the initial state.

While in the following PC states the logical unit may be increasing power usage to enter a higher power condition:

a) PC4:Active_Wait.

While in the following PC states the logical unit may be decreasing power usage to enter a lower power condition:

a) PC5:Wait_Idle; and
b) PC6:Wait_Standby.

The state machine maintains the timers listed in table 56.

Table 56 — PC state machine timers

Timer	Initial value [a]	Enable bit [a, b]
idle_a condition	IDLE_A CONDITION TIMER field	IDLE_A bit
idle_b condition	IDLE_B CONDITION TIMER field	IDLE_B bit
idle_c condition	IDLE_C CONDITION TIMER field	IDLE_C bit
standby_y condition	STANDBY_Y CONDITION TIMER field	STANDBY_Y bit
standby_z condition	STANDBY_Z CONDITION TIMER field	STANDBY_Z bit

[a] These fields and bits are in the Power Condition mode page (see 7.5.12).
[b] The enabled state of these bits may be overridden by a START STOP UNIT command (see SBC-3).

If more than one of the timers listed in table 56 expire at the same time, then only one timer is processed. The processing priority order shall be as follows:

1) standby_z condition timer;
2) standby_y condition timer;
3) idle_c condition timer;
4) idle_b condition timer; and
5) idle_a condition timer.

The PC state machine shall start in the PC0:Powered_On state after power on.

Figure 8 describes the PC state machine.

Figure 8 — Power condition state machine

5.12.6.2 PC0:Powered_On state

5.12.6.2.1 PC0:Powered_On state description

The logical unit shall enter this state upon power on.

5.12.6.2.2 Transition PC0:Powered_On to PC4:Active_Wait

This transition shall occur after:

a) the logical unit is ready to begin power on initialization.

The transition shall include a Transitioning From Powered On argument.

5.12.6.3 PC1:Active state

5.12.6.3.1 PC1:Active state description

While in this state, if power on initialization is not complete, then the logical unit shall complete its power on initialization.

While in this state, if power on initialization is complete, then:

a) the logical unit is in the active power condition (see 5.12.2);
b) each enabled idle condition timer is running; and
c) each enabled standby condition timer is running.

5.12.6.3.2 Transition PC1:Active to PC5:Wait_Idle

This transition shall occur if:

a) an idle condition timer is enabled; and
b) that idle condition timer has expired.

The transition shall include a:

a) Transitioning To Idle_a argument, if the highest priority timer (see 5.12.6.1) that expired is the idle_a condition timer;
b) Transitioning To Idle_b argument, if the highest priority timer that expired is the idle_b condition timer; or
c) Transitioning To Idle_c argument, if the highest priority timer that expired is the idle_c condition timer.

5.12.6.3.3 Transition PC1:Active to PC6:Wait_Standby

This transition shall occur if:

a) a standby condition timer is enabled; and
b) that standby condition timer has expired.

The transition shall include a:

a) Transitioning To Standby_z argument, if the highest priority timer (see 5.12.6.1) that expired is the standby_z condition timer; or
b) Transitioning To Standby_y argument, if the highest priority timer that expired is the standby_y condition timer.

5.12.6.4 PC2:Idle state

5.12.6.4.1 PC2:Idle state description

While in this state:

a) the logical unit is in an idle power condition (see 5.12.3);
b) the device server shall provide pollable sense data (see 5.6.2);
c) each enabled idle condition timer that has not expired is running; and
d) each enabled standby condition timer that has not expired is running.

If a lower priority (see 5.12.6.1) idle condition timer is enabled and expires, then that timer is ignored.

5.12.6.4.2 Transition PC2:Idle to PC4:Active_Wait

This transition shall occur if:

a) the device server processes a command that requires the logical unit to be in the PC1:Active state to continue processing that command.

The transition shall include a:

a) Transitioning From Idle argument; and
b) Transitioning From Idle_c argument, if the current power condition is the idle_c power condition.

5.12.6.4.3 Transition PC2:Idle to PC5:Wait_Idle

This transition shall occur if:

a) an idle condition timer is enabled;
b) that idle condition timer has expired; and
c) the priority (see 5.12.6.1) of that idle condition timer is greater than the priority of the idle condition timer associated with the current idle power condition.

The transition shall include a:

a) Transitioning To Idle_b argument, if the highest priority timer that expired is the idle_b condition timer; or
b) Transitioning To Idle_c argument, if the highest priority timer that expired is the idle_c condition timer.

5.12.6.4.4 Transition PC2:Idle to PC6:Wait_Standby

This transition shall occur if:

a) a standby condition timer is enabled; and
b) that standby condition timer has expired.

The transition shall include a:

a) Transitioning To Standby_z argument, if the highest priority timer (see 5.12.6.1) that expired is the standby_z condition timer; or
b) Transitioning To Standby_y argument, if the highest priority timer that expired is the standby_y condition timer.

5.12.6.5 PC3:Standby state

5.12.6.5.1 PC3:Standby state description

While in this state:

a) the logical unit is in an standby power condition (see 5.12.4);
b) the device server shall provide pollable sense data (see 5.6.2);
c) each enabled idle condition timer that has not expired is running; and
d) each enabled standby condition timer that has not expired is running.

If an idle condition timer or a lower priority (see 5.12.6.1) standby condition timer is enabled and expires, then that timer is ignored.

5.12.6.5.2 Transition PC3:Standby to PC4:Active_Wait

This transition shall occur if:

a) the device server processes a command that requires the logical unit to be in the PC1:Active state to continue processing that command.

The transition shall include a Transitioning From Standby argument.

5.12.6.5.3 Transition PC3:Standby to PC6:Wait_Standby

This transition shall occur if:

a) the standby_z condition timer is enabled;
b) the standby_z condition timer expires; and
c) the current power condition is the standby_y power condition.

The transition shall include a Transitioning To Standby_z argument.

5.12.6.6 PC4:Active_Wait state

5.12.6.6.1 PC4:Active_Wait state description

While in this state:

a) each idle condition timer that is enabled and not expired is running;
b) each standby condition timer that is enabled and not expired is running;
c) the device server shall provide pollable sense data (see 5.6.2) with the sense key set to NO SENSE and the additional sense code set to LOGICAL UNIT TRANSITIONING TO ANOTHER POWER CONDITION for return by a REQUEST SENSE command (see 6.29); and
d) the logical unit is performing the operations required for it to be in the PC1:Active state (e.g., a disk drive spins up its media).

If this state was entered with a Transitioning From Idle argument, then:

a) the device server is capable of processing and completing the same commands that the device server is able to process and complete while in the PC2:Idle state;
b) the peak power consumed in this state shall be no more than the typical peak power consumed in the PC1: Active state; and
c) the device server shall terminate any command that requires the logical unit be in the PC1:Active state to continue processing, with CHECK CONDITION status, with the sense key set to NOT READY and the additional sense code set to LOGICAL UNIT IS IN PROCESS OF BECOMING READY if all of the following are true:
   A) this state was entered with an Transitioning From Idle_c argument; and
   B) from idle command processing control is enabled (i.e., the FIDCPC field in the Power Condition mode page (see 7.5.12) is set to 10b).

If this state was entered with a Transitioning From Standby argument, then:

a) the device server is capable of processing and completing the same commands that the device server is able to process and complete while in the PC3:Standby state;
b) the peak power consumption in this state is not limited by this standard; and
c) if from standby command processing control is enabled (i.e., the FSBCPC field in the Power Condition mode page (see 7.5.12) is set to 10b), then the device server shall terminate any command that requires the logical unit be in the PC1:Active state or PC2:Idle state to continue processing, with CHECK CONDITION status, with the sense key set to NOT READY and the additional sense code set to LOGICAL UNIT IN THE PROCESS OF BECOMING READY.

If this state was entered with a Transitioning From Powered On argument, then:

a) the device server is capable of processing and completing the same commands (except a TEST UNIT READY command) that the device server is able to process and complete while in the PC3:Standby state; and
b) the device server shall terminate with CHECK CONDITION status, with the sense key set to NOT READY and the additional sense code set to LOGICAL UNIT IN THE PROCESS OF BECOMING READY any of the following commands:
   A) any command that requires the logical unit be in the PC1:Active state or PC2:Idle state to continue processing; and
   B) all TEST UNIT READY commands (see 6.37).

If an idle condition timer or a standby condition timer is enabled and expires, then that timer is ignored in this state.

5.12.6.6.2 Transition PC4:Active_Wait to PC1:Active

This transition shall occur when:

a) the logical unit meets the requirements for being in the PC1:Active state.

5.12.6.7 PC5:Wait_Idle state

5.12.6.7.1 PC5:Wait_Idle state description

While in this state:

a) each idle condition timer that is enabled and not expired is running;
b) each standby condition timer that is enabled and not expired is running;
c) the device server shall provide pollable sense data (see 5.6.2) with the sense key set to NO SENSE and the additional sense code set to LOGICAL UNIT TRANSITIONING TO ANOTHER POWER CONDITION for return by a REQUEST SENSE command (see 6.29); and
d) the logical unit is performing the operations required for it to be in the PC2:Idle state (e.g., reducing power usage); and
e) the device server is capable of processing and completing the same commands, except a START STOP UNIT command with the IMMED bit set to zero (see SBC-3), that the device server is able to process and complete in the PC2:Idle state.

If an idle condition timer or a standby condition timer is enabled and expires, then that timer is ignored in this state.

5.12.6.7.2 Transition PC5:Wait_Idle to PC2:Idle

This transition shall occur when:

a) the logical unit meets the requirements for being in the:
   A) idle_a power condition, if this state was entered with a Transitioning To Idle_a argument;
   B) idle_b power condition, if this state was entered with a Transitioning To Idle_b argument; or
   C) idle_c power condition, if this state was entered with a Transitioning To Idle_c argument.

5.12.6.8 PC6:Wait_Standby state

5.12.6.8.1 PC6:Wait_Standby state description

While in this state:

a) each idle condition timer that is enabled and not expired is running;
b) each standby condition timer that is enabled and not expired is running;
c) the device server shall provide pollable sense data (see 5.6.2) with the sense key set to NO SENSE and the additional sense code set to LOGICAL UNIT TRANSITIONING TO ANOTHER POWER CONDITION for return by a REQUEST SENSE command (see 6.29); and
d) the logical unit is performing the operations required for it to be in the PC3:Standby state (e.g., reducing power usage); and
e) the device server is capable of processing and completing the same commands, except a START STOP UNIT command with the IMMED bit set to zero (see SBC-3), that the device server is able to process and complete in the PC3:Standby state.

If an idle condition timer or a standby condition timer is enabled and expires, then that timer is ignored in the state.

5.12.6.8.2 Transition PC6:Wait_Standby to PC3:Standby

This transition shall occur when:

a) the logical unit meets the requirements for being in the:
   A) standby_y power condition, if this state was entered with a Transitioning To Standby_y argument; or
   B) standby_z power condition, if this state was entered with a Transitioning To Standby_z argument.

5.13 Medium auxiliary memory

Some types of media, especially removable media, include a non-volatile memory referred to as MAM (medium auxiliary memory). Medium auxiliary memory is used to store data describing the media and its contents. This standard supports medium auxiliary memory with the READ ATTRIBUTE command (see 6.15) and the WRITE ATTRIBUTE command (see 6.38). These commands are used to retrieve and store information in the medium auxiliary memory in the form of MAM attributes.

A MAM attribute is represented in a format described in 7.4 and is composed of:

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

There are three types of MAM attributes (see table 57).

Table 57 — Types of MAM attributes

Attribute Type	Attribute Source	Example	Readable with READ ATTRIBUTE	Writable with WRITE ATTRIBUTE
Medium	Permanently stored in the medium auxiliary memory during manufacture.	Media Serial Number	Yes	No
Device	Maintained by the device server.	Load Count	Yes	No
Host	Maintained by the application client.	Backup Date	Yes	Yes

Depending on that attribute type, MAM attributes have the states shown in table 58.

Table 58 — MAM attribute states

Attribute Type	Attribute State	Description
Medium or Device	Read Only	An application client may read the contents of the MAM attribute with the READ ATTRIBUTE command, but an attempt to clear or change the MAM attribute using the WRITE ATTRIBUTE command shall result in the command being terminated with CHECK CONDITION status. When the READ ONLY bit (see 7.4.1) is one, the attribute is in the read only state.
	Unsupported	The device server does not support the MAM attribute and shall not return it in response to a READ ATTRIBUTE command.
	Unavailable	The MAM attribute exists but it is not available at this time. The device server shall not return it in response to a READ ATTRIBUTE command.
Host	Nonexistent	A host attribute does not exist in the medium auxiliary memory until a WRITE ATTRIBUTE command creates it.
	Read/Write	The MAM attribute has been created using the WRITE ATTRIBUTE command. After the MAM attribute has been created, the contents may be altered using subsequent WRITE ATTRIBUTE commands. A read/write MAM attribute may be returned to the nonexistent state using a WRITE ATTRIBUTE command with the attribute length set to zero. When the READ ONLY bit (see 7.4.1) is zero, the MAM attribute is in the read/write state.
	Unsupported	The device server does not support the MAM attribute and shall not return it in response to a READ ATTRIBUTE command.

5.14 Error history

5.14.1 Error history overview

Error history is optional data collected by a logical unit to aid in troubleshooting errors.

The READ BUFFER command (see 6.16.9) provides a method for retrieving error history from the logical unit (see 5.14.2).

The WRITE BUFFER command (see 6.39.14) provides a method for inserting application client error history into the error history (see 5.14.3) and for clearing the error history (see 5.14.4).

The format of the application client error history is defined by the manufacturer of the application client. The format of the error history, including the format of any application client error history included in the error history, is defined by the manufacturer of the logical unit.

5.14.2 Retrieving error history with the READ BUFFER command

Device servers may allow the error history to be retrieved using a sequence of READ BUFFER commands on one I_T nexus.

Error history is returned using error history snapshots. An error history snapshot is the contents of the error history at a specific point in time, created by the device server at vendor specific times or requested by the application client using the READ BUFFER command with certain buffer IDs.

The I_T nexus being used to retrieve error history snapshot is called the error history I_T nexus. Only one I_T nexus is allowed to retrieve an error history snapshot at a time.

To retrieve the complete error history, an application client uses one I_T nexus to:

1) create an error history snapshot if one does not already exist, establish the I_T nexus as the error history I_T nexus, and retrieve the error history directory by sending a READ BUFFER command (see 6.16.9.2) with:
   A) the MODE field set to 1Ch (i.e., error history);
   B) the BUFFER ID field set to one of the following:
      a) If the error history I_T nexus is expected to be valid:
         A) 00h (i.e., return error history directory);
         B) 01h (i.e., return error history directory and create new snapshot);
      b) if the application client has knowledge obtained by means outside the scope of this standard that the error history I_T nexus is no longer valid:
         A) 02h (i.e., return error history directory and establish new error history I_T nexus); or
         B) 03h (i.e., return error history directory, establish new error history I_T nexus, and create new snapshot);
   C) the BUFFER OFFSET field set to 000000h; and
   D) the ALLOCATION LENGTH field set to at least 2 088 (i.e., large enough to transfer the complete error history directory);
2) retrieve the error history. The application client uses a Data-In Buffer size that is a multiple of the offset boundary indicated in the READ BUFFER descriptor (see 6.16.5). For each buffer ID indicated in the error history directory in the range of 10h to EFh, the application client sends one or more READ BUFFER commands (see 6.16.9.3) as follows:
   1) send the first READ BUFFER command with:
      a) the MODE field set to 1Ch (i.e., error history);
      b) the BUFFER ID field set to the buffer ID (i.e., an error history data buffer);
      c) the BUFFER OFFSET field set to 000000h; and
      d) the ALLOCATION LENGTH field set to the size of the Data-In Buffer;
   2) until the number of bytes returned by the previous READ BUFFER command does not equal the specified allocation length and/or the total number of bytes returned from the buffer ID equals the maximum available length indicated in the error history directory, send zero or more additional READ BUFFER commands with:
      a) the MODE field set to 1Ch (i.e., error history);
      b) the BUFFER ID field set to the buffer ID (i.e., an error history data buffer);
      c) the BUFFER OFFSET field set to the previous buffer offset plus the previous allocation length; and
      d) the ALLOCATION LENGTH field set to the size of the Data-In Buffer;

and

3) clear the error history I_T nexus and, depending on the buffer ID, release the error history snapshot by sending a READ BUFFER command with:
   A) the MODE field set to 1Ch (i.e., error history);
   B) the BUFFER ID field set to:
      a) FEh (i.e., clear error history I_T nexus) (see 6.16.9.4); or
      b) FFh (i.e., clear error history I_T nexus and release snapshot) (see 6.16.9.5);
   C) the BUFFER OFFSET field set to any value allowed by table 197 (see 6.16.9.1) (e.g., 000000h); and
   D) the ALLOCATION LENGTH field set to any value allowed for the chosen BUFFER ID field value (see 6.16.9.4 or 6.16.9.5) (e.g., 000000h).

While an error history snapshot exists, the device server:

a) shall not modify the error history snapshot to reflect any changes to the error history;
b) may or may not record events that it detects into the error history; and
c) if it supports the WRITE BUFFER command download application client error history mode (see 6.39.14), shall record any application client error history received into the error history.

The device server shall clear the established error history I_T nexus and not release the error history snapshot when:

a) upon processing of a READ BUFFER command on the error history I_T nexus with:
   A) the MODE field set to 1Ch (i.e., error history); and
   B) the BUFFER ID field set to FEh (i.e., clear error history I_T nexus) (see 6.16.9.4);
   or
b) an I_T nexus loss occurs on the error history I_T nexus.

The device server shall clear the established error history I_T nexus and release the error history snapshot when:

a) upon processing of a READ BUFFER command using the same I_T nexus that was used to establish the snapshot with:
   A) the MODE field set to 1Ch (i.e., error history); and
   B) the BUFFER ID field set to FFh (i.e., clear error history I_T nexus and release snapshot) (see 6.16.9.5);
b) a power on occurs;
c) a hard reset occurs; or
d) a logical unit reset occurs.

The device server shall not replace or release the error history snapshot while the error history I_T nexus is established.

The device server shall implement a vendor specific timer for error history snapshot retrieval. If the vendor specific timer expires:

a) the device server shall:
   A) clear the error history I_T nexus; and
   B) establish a unit attention condition for the error history I_T nexus with the additional sense code set to ERROR HISTORY I_T NEXUS CLEARED;
   or
b) the device server shall:
   A) clear the error history I_T nexus;
   B) release the error history snapshot; and
   C) establish a unit attention condition for the error history I_T nexus with the additional sense code set to ERROR HISTORY SNAPSHOT RELEASED.

After an error history snapshot is released, the device server shall resume recording error history for events that it detects.

Error history may also be retrieved by vendor specific methods or other READ BUFFER command sequences that are outside the scope of this standard.

5.14.3 Adding application client error history with the WRITE BUFFER command

An application client adds application client detected error history to the error history collected by a logical unit by sending a WRITE BUFFER command. The application client error history may be recovered as part of the error history (see 5.14.2) or by means outside the scope of this standard and is not used for any logical unit related error recovery.

Error history that contains a mix of application client error history and logical unit error history may be used to correlate an application client-detected error with errors detected internally by the logical unit.

Application clients should minimize the amount of error history they store to prevent error history overflows (see 6.39.14).

5.14.4 Clearing error history with the WRITE BUFFER command

An application client clears the portions of the error history that the device server allows to be cleared by sending a WRITE BUFFER command (see 6.39.14) with:

a) the MODE field set to 1Ch (i.e., download error history);
b) the BUFFER OFFSET field set to any value allowed by 6.39.14 (e.g., 000000h);
c) the PARAMETER LIST LENGTH field set to 00001Ah;
d) in the parameter list, the CLR bit set to one; and
e) all other fields in the parameter list as 6.39.14 allows.

Clearing error history shall not:

a) clear the error history I_T nexus, if any, if it was created with the READ BUFFER command (see 5.14.2); or
b) release the error history snapshot, if any, if it was created with the READ BUFFER command (see 5.14.2).

5.15 Device clocks

A timestamp may be included in data logged or recorded by a device server. There shall be one timestamp per logical unit.

The timestamp origin shall be one of those specified in table 59.

Table 59 — TIMESTAMP ORIGIN field

Code	Description
000b	Timestamp initialized to zero at power on or as the result of a hard reset
001b	Reserved
010b	Timestamp initialized by the SET TIMESTAMP command (see 6.36)
011b	Timestamp initialized by methods outside the scope of this standard
100b to 111b	Reserved

The timestamp shall not be affected by an I_T nexus loss or a logical unit reset.

Once a timestamp is initialized it shall begin counting from that time forward. Once the timestamp is initialized it shall remain in effect until one of the following occurs:

a) a hard reset occurs;
b) a SET TIMESTAMP command is processed; or
c) a method outside the scope of this standard affects the timestamp.

The methods by which a timestamp may be changed are indicated in the Control Extension mode page (see 7.5.8).

If the timestamp is changed by means other than the SET TIMESTAMP command then the device server shall establish a unit attention condition for the initiator port associated with every I_T nexus (see SAM-4), with the additional sense code set to TIMESTAMP CHANGED.

The TIMESTAMP field format is shown in table 60.

Table 60 — TIMESTAMP field format

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)							
•••				TIMESTAMP				
5								(LSB)

The TIMESTAMP field contains the value established at the last action that set the timestamp incremented by one for every millisecond that has elapsed since the timestamp was set, within vendor specific constraints.

5.16 Security Features

5.16.1 Security goals and threat model

5.16.1.1 Overview

In many cases, the security goals and threat model used for the Internet are applicable to SCSI commands. The Internet security goals and threat model found in RFC 3552 as they apply to SCSI are summarized in 5.16.1. Terms, concepts, and classes of security techniques that are defined in RFC 3552 are discussed based on their RFC 3552 definitions without modification in this standard.

The security goals and threat model described in 5.16.1 are valid for all SCSI device types. SCSI command standards may modify this model to deal with threats appropriate to specific device types.

5.16.1.2 Security goals

The overall goals of security may be divided into two broad categories:

a) communications security (i.e., protecting communications); and
b) system security.

These goals interact as a result of communications being carried out by systems, with access to those systems provided through communications channels. It is possible to provide security services that independently meet these goals. A common methodology is to secure the communications first and then provide secure access to systems over the secured communication channels.

Communication security is subdivided into the following primary areas of protection:

a) **Confidentiality:** preventing unintended entities from seeing the data;
b) **Cryptographic Data Integrity:** ensuring that the data that arrives is identical to the data that was sent; and
c) **Peer Entity Authentication:** ensuring that the communicating endpoints are the intended peer entities.

Data Origin Authentication (i.e., ensuring that the received data was sent by the authenticated peer) is the combination of Peer Entity Authentication and Cryptographic Data Integrity.

Non Repudiation enhances Data Origin Authentication with the ability to prove to a third party that the sender sent the data that the receiver received.

Cryptographic Data Integrity is called Data Integrity in RFC 3552. The term Cryptographic is added in this standard to distinguish the class of integrity protection required to counter malicious adversaries from the class of integrity protection required to deal with random data corruption (e.g., caused by cosmic rays or electrical noise). Mechanisms used to deal with random data corruption (e.g., parity bits and CRCs) have minimal value against malicious adversaries that are able to modify integrity checks to conceal their modifications to the data. Cryptographic Data Integrity requires knowledge of a secret key in order to modify an integrity check without that modification being detectable. A well designed system provides a high level of assurance that an attacker is unable to learn, guess, discover, or otherwise obtain the required secret key.

In addition to the primary areas, there is another area of control:

a) **Authorization:** controlling what an entity is allowed to do. For communications security this is control of the entities with which an entity is allowed to communicate.

Access Control (i.e., controlling what an entity is allowed to access) is a form of Authorization.

Systems security consists of protecting systems from unauthorized usage, inappropriate usage, and denial of service.

5.16.1.3 Threat model

Most secured systems are vulnerable to an attacker equipped with sufficient resources, time, and skills. In order to make designing a security system practical, a threat model is defined to describe the capabilities that an attacker is assumed to be able to deploy (e.g., knowledge, computing capability, and ability to control the system).

The main purposes of a threat model are as follows:

a) to identify the threats of concern; and
b) to rule some threats explicitly out of scope.

Most security measures do not provide absolute assurance that an attack has not occurred. Rather, security measures raise the difficulty of accomplishing the attack to well beyond the attacker's assumed capabilities and/or resources. Design of security measures that resist attackers with essentially unlimited capabilities (e.g., certain nation-states) is outside the scope of this standard. Security measures that are susceptible to a level of capability available to some attackers may still be useful for deterring attackers who lack that level of capability, especially when combined with non-technical security measures such as physical access controls.

The computational capability of an attacker is treated as a variable because that capability is inherently a moving target as a result of more powerful processors. The computational capability of an attacker influences design aspects (e.g., key length). Well designed security systems are agile in that they are able to operate not only with different key lengths, but also with different cryptographic algorithms.

The Internet threat model described in RFC 3552 is generally applicable to SCSI, and is specifically applicable when Internet Protocols are used by the SCSI transport (e.g., iSCSI, Fibre Channel via FCIP or iFCP). Its basic assumptions can be summarized as:

a) end systems engaging in communication are not under the control of the attacker.
b) the attacker is able to read any communicated IU and undetectably remove, change, or inject forged IUs, including injection of IUs that appear to be from a known and/or trusted system.

Communications security designs are based on an additional assumption that secrets (e.g., keys) used to secure the communications are protected so that an attacker is unable to learn, guess, discover, or otherwise obtain them. A consequence of this assumption is that attacks against secured communications are assumed to begin without with advance knowledge of the secrets used to secure the communications.

5.16.1.4 Types of attacks

The following types of attacks are considered:

a) passive attacks (i.e., attacks that only require reading IUs), and
b) active attacks (i.e., attacks that require the attacker to change communication and/or engage in communication).

More information on attack types is available in RFC 3552.

Simple passive attacks involve reading communicated data that the attacker was not intended to see (e.g., password, credit card number). More complex passive attacks involve post-processing the communicated data (e.g., checking a challenge-response pair against a dictionary to see if a common word was used as a password).

There are a wide variety of active attacks (e.g., spoofing, replay, insertion, deletion, known plaintext, and modification of communications). Man-in-the-middle attacks are a class of active attacks that involve the attacker inserting itself in the middle of communication, enabling it to intercept all communications without the knowledge of the communicating parties for various purposes (e.g., insertion, deletion, replay, modification and/or inspection via decryption of the communications).

5.16.1.5 SCSI security considerations

The application of communication security techniques (see RFC 3552) is defined by command standards. This subclause describes specific design considerations in applying the threat model (see 5.16.1.3) to all SCSI device types.

SCSI environments tend not to be fully connected (i.e., there are restrictions on the SCSI device servers with which a SCSI application client is able to communicate) due to the following mechanisms:

a) physical and logical connectivity restrictions (e.g., in SCSI to SCSI gateways across different transports);
b) LUN mapping and masking; and
c) transport zoning.

The resulting connectivity is more limited than the Internet security assumption that an off-path attacker is able to transmit to an arbitrary victim (see RFC 3552).

SCSI security designs are also influenced by SCSI being a client-server distributed service model (see SAM-4) that is realized over a number of different SCSI transport protocols and interconnects.

Security functionality may be defined as part of a command set or at the SCSI transport level. Some SCSI transport protocols (e.g., Fibre Channel and iSCSI) define security functionality that provides confidentiality, cryptographic integrity, and peer entity authentication for all communicated data. However, there are situations in which some or all of those mechanisms are not used and there are SCSI communications whose scope spans more than one SCSI Transport Protocol (e.g., via a gateway between iSCSI and FCP). Security that is defined by a command set is appropriate for such situations.

5.16.2 Security associations

5.16.2.1 Principles of SAs

Before an application client and device server begin applying security functions (e.g., data integrity checking, data encryption) to messages (i.e., data that is transferred in either direction between them), they perform a security protocol to create at least one SA (see 5.16.2.3). The result of the SA creation protocol is two sets of SA parameters (see 5.16.2.2), one that is maintained by the application client and one that is maintained by the device server.

In this model, SAs decouple the process of creating a security relationship from its usage in processing security functions. This decoupling allows either the creation or the usage of an SA to be upgraded in response to changing security threats without requiring both processes to be upgraded concurrently.

Figure 9 shows the relationship between application clients and device servers with respect to SAs.

Figure 9 — SA relationships

In both the application client and the device server, the SA parameters are modelled as being stored in an indexed array and the SAI identifies one set of SA parameters within that array. The application client and device server are not required to store the parameters for any given SA in the same array locations. In order to support this implementation flexibility, a single SA is modelled as having two different SAI values (i.e., one for the application client and one for the device server).

The device server shall maintain a single SA parameters table for all I_T nexuses.

SAs shall not be preserved across a power cycle, hard reset, or logical unit reset. SAs shall not be affected by an I_T nexus loss.

5.16.2.2 SA parameters

Each SAI shall identify at least the SA parameters defined in table 61. Individual security protocols define how the SA parameters are generated and/or used by that security protocol.

Table 61 — Minimum SA parameters (part 1 of 3)

Name	Description	Size (bytes) [a]		Scope [b]
		Min.	Max.	
SA parameters that identify and manage the SA.				
AC_SAI	The SAI used by the application client to identify the SA. [c]	4	4	Public
DS_SAI	The SAI used by the device server to identify the SA. [c]	4	4	Public
TIMEOUT	The number of seconds that may elapse after the completion of an SA access operation (i.e., SA creation or SA usage by a command) before the device server should discard the state associated with this SA (e.g., the SA parameters). If SA state is discarded because no SA access operations are received during the specified interval, the device server shall respond to further attempts to access the SA as if the SA had never been created. This parameter shall not be set to zero.	4	4	Public
SA parameters that are incorporated in messages to prevent message replay attacks.				
AC_SQN	A sequence number that is incremented for each response message received by an application client on which a security function is performed and used to detect replay attacks (see 5.16.1.4).	8	8	Public
DS_SQN	A sequence number that is incremented for each request message received by a device server on which a security function is performed and used to detect replay attacks.	8	8	Public

[a] These size values are guidelines. Specific security protocols may place more exacting size requirements on SA parameters.
[b] Public SA parameters may be transferred outside a SCSI device unencrypted. Secret SA parameters shall not be transferred outside a SCSI device. Fields within a protocol specific SA parameter are Shared or Secret as defined by the applicable SA creation protocol.
[c] SAI values between 0 and 255, inclusive, are reserved.
[d] Nonce SA parameters shall be at least half the size of KEY_SEED SA parameter.
[e] The number of bits of entropy in the KEY_SEED should be as close to the number of bits in the KEY_SEED as possible (see RFC 3766).

Table 61 — Minimum SA parameters (part 2 of 3)

Name	Description	Size (bytes) [a]		Scope [b]
		Min.	Max.	
SA parameters that are used by security functions to derive the secret keys that are applied to messages (e.g., for encryption).				
AC_NONCE [d]	A random nonce (see 3.1.122) value that is generated by the application client and used as an input to the key derivation security algorithm specified by the KDF_ID SA parameter during the derivation of an encryption key.	16	64	Public
DS_NONCE [d]	A random nonce value that is generated by the device server and used as an input to the key derivation security algorithm specified by the KDF_ID SA parameter during the derivation of an encryption key.	16	64	Public
KEY_SEED [e]	A value that is known only to the application client and device server that are creating this SA that in combination with the applicable nonce is used to derive the KEYMAT value. The KEY_SEED shall be set to zero as part of completing the SA creation function.	16	64	Secret
KDF_ID	A security algorithm (see 5.16.8) coded value that identifies the KDF used by the application client and device server.	4	4	Public
SA parameters that are used by security functions to secure messages between the application client and device server.				
KEYMAT	A value that is known only to the application client and device server that are participating in this SA that may be subdivided into one or more key values that are used in security functions that secure messages. The contents of KEYMAT depend on the USAGE_TYPE SA parameter value.	14	1 024	Secret

[a] These size values are guidelines. Specific security protocols may place more exacting size requirements on SA parameters.

[b] Public SA parameters may be transferred outside a SCSI device unencrypted. Secret SA parameters shall not be transferred outside a SCSI device. Fields within a protocol specific SA parameter are Shared or Secret as defined by the applicable SA creation protocol.

[c] SAI values between 0 and 255, inclusive, are reserved.

[d] Nonce SA parameters shall be at least half the size of KEY_SEED SA parameter.

[e] The number of bits of entropy in the KEY_SEED should be as close to the number of bits in the KEY_SEED as possible (see RFC 3766).

Table 61 — Minimum SA parameters (part 3 of 3)

Name	Description	Size (bytes) [a]		Scope [b]
		Min.	Max.	
SA parameters that are used by SA management functions.				
USAGE_TYPE	A coded value (see table 62) that indicates the how the SA is used.	2	2	Public
USAGE_DATA	Information associated with how the SA is used (e.g., cryptograph algorithms and key sizes). The contents of USAGE_DATA depend on the USAGE_TYPE SA parameter value.	0	1 024	Public
MGMT_DATA	SA data that is used in ways defined by the SA creation protocol to perform SA management functions (e.g., deletion of the SA).	0	1 024	Protocol specific

[a] These size values are guidelines. Specific security protocols may place more exacting size requirements on SA parameters.
[b] Public SA parameters may be transferred outside a SCSI device unencrypted. Secret SA parameters shall not be transferred outside a SCSI device. Fields within a protocol specific SA parameter are Shared or Secret as defined by the applicable SA creation protocol.
[c] SAI values between 0 and 255, inclusive, are reserved.
[d] Nonce SA parameters shall be at least half the size of KEY_SEED SA parameter.
[e] The number of bits of entropy in the KEY_SEED should be as close to the number of bits in the KEY_SEED as possible (see RFC 3766).

The USAGE_TYPE SA parameter shall be one of the values shown in table 62.

Table 62 — USAGE_TYPE SA parameter values

Value [a]	Description	Usage model	Usage data description	Reference
0000h to 0080h	Reserved			
0081h	Tape data encryption	ESP-SCSI [b]	None [c]	SSC-3
0082h to 8000h	Reserved			
8001h	CbCS authentication and credential encryption	ESP-SCSI [b]	None [c]	5.16.6.8
8002h to FFFFh	Reserved			

[a] USAGE_TYPE values between 8000h and CFFFh inclusive place additional constraints on how an SA is to be created as described in 7.7.3.5.14.
[b] ESP-SCSI usage is defined in 5.16.7.
[c] The usage data length field in the IKEv2-SCSI SAUT Cryptographic Algorithms payload (see 7.7.3.5.14) shall contain zero.

5.16.2.3 Creating an SA

The SECURITY PROTOCOL IN command (see 6.30) and SECURITY PROTOCOL OUT command (see 6.31) security protocols shown in table 63 are used to create SAs. The process of creating an SA establishes the SA parameter (see 5.16.2.2) values as follows:

a) initial values for:
   A) both (i.e., application client and device server) sequence numbers set to one;
b) unchanging values for the lifetime of the SA:
   A) both SAIs;
   B) TIMEOUT;
   C) KDF_ID;
   D) KEYMAT;
   E) USAGE_TYPE;
   F) USAGE_DATA; and
   G) MGMT_DATA;
   and
c) values that are zero upon completion of SA creation:
   A) KEY_SEED; and
   B) both nonces.

Table 63 — Security protocols that create SAs

Security Protocol Code	Description	References
40h	SA creation capabilities	7.7.2
41h	IKEv2-SCSI	5.16.4 and 7.7.3

5.16.3 Key derivation functions

5.16.3.1 Overview

Table 64 summarizes the key derivation functions defined by this standard.

Table 64 — Key derivation functions summary

Security Algorithm Code (see table 92)	Description	Reference
8002 0002h	IKEv2-based iterative HMAC KDF based on SHA-1	5.16.3.3
8002 0005h	IKEv2-based iterative HMAC KDF based on SHA-256	5.16.3.3
8002 0006h	IKEv2-based iterative HMAC KDF based on SHA-384	5.16.3.3
8002 0007h	IKEv2-based iterative HMAC KDF based on SHA-512	5.16.3.3
8002 0004h	IKEv2-based iterative KDF based on AES-128 in XCBC mode	5.16.3.4

5.16.3.2 IKEv2-based iterative KDF

In principle, KEYMAT is generated by applying IFUNC to STRING using KEY_SEED to produce the requested number of KEYMAT bits. In equation notation, the operation is as follows:

KEYMAT = IFUNC(KEY_SEED, STRING)

However, accomplishing this may require multiple applications of IFUNC because the number of bits output by IFUNC is not sufficient to meet the number of KEYMAT bits that are to be produced.

The IKEv2-based (see RFC 4306) iterative technique for applying IFUNC is as follows:

1) initialize PREV_OUTPUT to a null string (i.e., a string that contains no bits);
2) repeat the following function for values of N that increment from one by one to a maximum of 255 or until the total number of bits returned by all invocations of IFUNC equals or exceeds the number of KEYMAT bits that are to be produced, whichever occurs first:
   T_N = IFUNC(KEY_SEED, (PREV_OUTPUT || STRING || a byte containing the value N))
   and
3) concatenate the T_N values (e.g., T_1 || T_2 || T_3) and return as many of the resulting bits as specified by the number of KEYMAT bits that are to be produced input parameter, starting with the first bit in T_1.

Protocols that call the IFUNC function to generate KEYMAT should ensure that the number of KEYMAT bits requested does not cause N to exceed 255. If N reaches 256, then:

1) the requested number of KEYMAT bits is not returned by IFUNC; and
2) the request to produce KEYMAT shall be terminated with an error.

5.16.3.3 HMAC-based KDFs

If the KDF_ID is one of those shown in table 65, the key derivation function is a combination of the:

a) the HMAC function defined in FIPS 198a (see 2.4);
b) the secure hash function shown in table 65 for the specified KDF_ID value; and
c) IKEv2-based iterative KDF technique (see 5.16.3.2).

The technique requires the following inputs from or related to the SA parameters:

a) AC_SAI;
b) DS_SAI;
c) AC_NONCE;
d) DS_NONCE;
e) KEY_SEED; and
f) KEYMAT size, in bits.

The USAGE_TYPE SA parameter and USAGE_DATA SA parameter (see 5.16.2.2) specify the KEYMAT size as part of the security protocol that performs SA creation (see 5.16.2.3).

The IKEv2-based iterative KDF technique is applied with the following inputs:

a) IFUNC is the HMAC function defined in FIPS 198a with the translation of inputs names shown in table 65;
b) KEY_SEED is the KEY_SEED SA parameter; and
c) STRING store the concatenated contents of the following SA parameters:
   1) AC_NONCE;

2) DS_NONCE;
3) AC_SAI; and
4) DS_SAI.

Table 65 — HMAC-based key derivation functions

FIPS 198a inputs selected by KDF_ID	KDF_ID (see table 64)			
	8002 0002h	8002 0005h	8002 0006h	8002 0007h
H (i.e., hash function)	SHA-1 (see table 66)	SHA-256 (see table 66)	SHA-384 (see table 66)	SHA-512 (see table 66)
B (i.e., hash input block size) [a]	64	64	128	128
L (i.e., hash output block size) [a]	20	32	48	64
t (i.e., MAC size) [a]				
K (i.e., key)	KEY_SEED SA parameter			
text	STRING as defined in this subclause and used in 5.16.3.2			

[a] In accordance with FIPS 198a, all sizes are shown in bytes.

Details of the hash functions that act as inputs to the FIPS 198a HMAC function are shown in table 66.

Table 66 — Hash functions used by HMAC based on KDF_ID

KDF_ID (see table 64)	Function	Description
8002 0002h	SHA-1	HMAC input *H* is the SHA-1 secure hash function defined in FIPS 180-2 with Change Notice 1 (see 2.4).
8002 0005h	SHA-256	HMAC input *H* is the SHA-256 secure hash function defined in FIPS 180-2 with Change Notice 1.
8002 0006h	SHA-384	HMAC input *H* is the SHA-384 secure hash function defined in FIPS 180-2 with Change Notice 1.
8002 0007h	SHA-512	HMAC input *H* is the SHA-512 secure hash function defined in FIPS 180-2 with Change Notice 1.

5.16.3.4 AES-XCBC-PRF-128 IKEv2-based iterative KDF

If the KDF_ID is 8002 0004h, the key derivation function is a combination of the:

a) AES-XCBC-PRF-128 secure hash function defined in RFC 4434 (see 2.5) and RFC 3566; and
b) IKEv2-based iterative KDF technique (see 5.16.3.2).

The technique requires the following inputs from or related to the SA parameters:

a) AC_SAI;
b) DS_SAI;
c) AC_NONCE;

d) DS_NONCE;
e) KEY_SEED; and
f) the number of KEYMAT bits that are to be produced.

The IKEv2-based iterative KDF technique is applied with the following inputs:

a) IFUNC is the AES-XCBC-PRF-128 secure hash function with the translation of inputs names shown in table 67;
b) KEY_SEED is the KEY_SEED SA parameter; and
c) STRING store the concatenated contents of the following SA parameters:
   1) AC_NONCE;
   2) DS_NONCE;
   3) AC_SAI; and
   4) DS_SAI.

Table 67 — RFC 3566 parameter translations KDF based on AES-XCBC-PRF-128

RFC 3566 Parameter	Translation
K (i.e., key)	KEY_SEED SA parameter
M (i.e., message)	STRING as defined in this subclause and used in 5.16.3.2

5.16.4 Using IKEv2-SCSI to create a security association

5.16.4.1 Overview

The IKEv2-SCSI protocol is a subset of the IKEv2 protocol (see RFC 4306) that this standard defines for use in the creation and maintenance of an SA (see 3.1.155).

An IKEv2-SCSI SA creation transaction (see 3.1.130) shall only be initiated by the application client.

The IKEv2-SCSI protocol creates the following pair of IKE SAs (see RFC 4306):

a) an SA that protects data sent from the application client to the device server; and
b) an SA that protects data sent from the device server to the application client.

An IKEv2-SCSI SA creation transaction consists of the following steps:

1) Device Server Capabilities step (see 5.16.4.5): The application client determines the device server's cryptographic capabilities;
2) Key Exchange step (see 5.16.4.6): The application client and device server:
   A) perform a key exchange;
   B) determine SAIs (see 3.1.156);
   C) generate the shared keys used for SA management (e.g., SA creation and deletion) (see 5.16.4.8); and
   D) may complete the generation of the SA (see 5.16.4.9);
   and
3) Authentication step (see 5.16.4.7): Unless omitted by application client and device server negotiations in the previous steps:
   A) the application client and device server authenticate:
      a) each other;
      b) the key exchange; and
      c) the capability selection;

and
B) complete the generation of the SA (see 5.16.4.9).

The values in the SECURITY PROTOCOL field and the SECURITY PROTOCOL SPECIFIC field in the SECURITY PROTOCOL IN command (see 6.30) and SECURITY PROTOCOL OUT command (see 6.31) identify the step of the IKEv2-SCSI protocol (see 7.7.3.2).

The Key Exchange step and the Authentication step depend on the results from the Device Capabilities step in order to create an SA. During the IKEv2-SCSI Key Exchange step, the application client and device server perform independent computations to construct the following sets of shared keys:

a) shared keys that are used by the IKEv2-SCSI Authentication step;
b) shared keys that are used by the IKEv2-SCSI Authentication step and to delete the SA; and
c) shared keys are used by SCSI the usage type specific operations that obtain security from the generated SA.

More details about these shared keys are provided in 5.16.4.4.

An application client may or may not:

a) proceed to the Key Exchange step after the Device Server Capabilities step; or
b) perform a separate Device Server Capabilities step for each IKEv2-SCSI SA creation transaction.

If the device server's capabilities have changed, the Authentication step returns an error, and the Key Exchange step may return an error.

Changes in the device server’s capabilities do not take effect until at least one application client has been notified of the new capabilities via the parameter data returned by the Device Server Capabilities step.

After a Device Capabilities step, the application client performs SA creation by sending a sequence of two or four IKEv2-SCSI commands over a single I_T_L nexus to the device server. The following commands constitute an IKEv2-SCSI CCS (see 3.1.71):

a) if the Authentication step is skipped (see 5.16.4.3.4):
   1) a Key Exchange step SECURITY PROTOCOL OUT command (see 5.16.4.6.2); and
   2) a Key Exchange step SECURITY PROTOCOL IN command (see 5.16.4.6.3);
   or
b) if the Authentication step is performed:
   1) a Key Exchange step SECURITY PROTOCOL OUT command (see 5.16.4.6.2);
   2) a Key Exchange step SECURITY PROTOCOL IN command (see 5.16.4.6.3);
   3) an Authentication step SECURITY PROTOCOL OUT command (see 5.16.4.7.2); and
   4) an Authentication step SECURITY PROTOCOL IN command (see 5.16.4.7.3).

The device server shall process each command in the IKEv2-SCSI CCS to completion before returning status. While a command in the IKEv2-SCSI CCS is being processed by the device server, the application client may use the REQUEST SENSE command (see 5.16.5) to ascertain the device server’s progress for the command.

When an error is encountered, the device server or application client may abandon the IKEv2-SCSI CCS before the SA is created (see 5.16.4.10).

The device server shall maintain state for the IKEv2-SCSI CCS on a given I_T_L nexus from the time the Key Exchange step SECURITY PROTOCOL OUT command is completed with GOOD status until one of the following occurs:

a) the IKEv2-SCSI CCS completes successfully;
b) the IKEv2-SCSI CCS is abandoned as described in 5.16.4.10;
c) the SA being created by the IKEv2-SCSI CCS is deleted as described in 5.16.4.11;
d) the number of seconds specified in the IKEV2-SCSI PROTOCOL TIMEOUT field of the IKEv2-SCSI Timeout Values payload (see 7.7.3.5.15) in the Key Exchange step SECURITY PROTOCOL OUT parameter data elapses and none of the following commands have been received:
   A) the next command in the IKEv2-SCSI CCS; or
   B) a REQUEST SENSE command;
   or
e) one of the following event-related SCSI device conditions (see SAM-4) occurs:
   A) power cycle;
   B) hard reset;
   C) logical unit reset; or
   D) I_T nexus loss.

If the device server receives a SECURITY PROTOCOL OUT or SECURITY PROTOCOL IN command with the SECURITY PROTOCOL field set to IKEv2-SCSI (i.e., 41h) on an I_T_L nexus other than the one for which IKEv2-SCSI CCS state is being maintained, then:

a) an additional IKEv2-SCSI CCS may be started; or
b) the command may be terminated with CHECK CONDITION status, with the sense key set to ABORTED COMMAND, and the additional sense code set to CONFLICTING SA CREATION REQUEST.

Except for the PERSISTENT RESERVE OUT command (see 5.9.4) and the cases described in this subclause, a device server that is maintaining a IKEv2-SCSI CCS state on a particular I_T_L nexus shall not alter its processing of new commands received on that I_T_L nexus.

If all of the following conditions are true:

a) the device server includes the following algorithm descriptors in the IKEv2-SCSI SA Creation Capabilities payload (see 7.7.3.5.12) in the parameter data returned by the Device Server Capabilities step SECURITY PROTOCOL IN command (see 5.16.4.5):
   A) an SA_AUTH_OUT algorithm descriptor (see 7.7.3.6.6) with the ALGORITHM IDENTIFIER field set to SA_AUTH_NONE; and
   B) an SA_AUTH_IN algorithm descriptor (see 7.7.3.6.6) with the ALGORITHM IDENTIFIER field set to SA_AUTH_NONE;
   and
b) the application client sends the following algorithm descriptors to the device server in the IKEv2-SCSI SA Cryptographic Algorithms payload (see 7.7.3.5.13) in the Key Exchange step SECURITY PROTOCOL OUT command (see 5.16.4.6.2):
   A) an SA_AUTH_OUT algorithm descriptor with the ALGORITHM IDENTIFIER field set to SA_AUTH_NONE; and
   B) an SA_AUTH_IN algorithm descriptor with the ALGORITHM IDENTIFIER field set to SA_AUTH_NONE;

then:

a) the Authentication step is skipped;
b) the IKEv2-SCSI CCS consists of the two Key Exchange step commands;
c) the device server requires the IKEv2-SCSI SAUT Cryptographic Algorithms payload (see 7.7.3.5.14) to be present in the parameter data sent by the Key Exchange step SECURITY PROTOCOL OUT command;

d) the device server returns the IKEv2-SCSI SAUT Cryptographic Algorithms payload in the parameter data returned by the Key Exchange step SECURITY PROTOCOL IN command; and
e) SA creation occurs upon the completion of the Key Exchange step.

Proper operation of an IKEv2-SCSI CCS depends on SA_AUTH_NONE being used in both the Authentication step SECURITY PROTOCOL OUT command and the Authentication step SECUIRTY PROTOCOL IN command, or SA_AUTH_NONE not being used by either command. Processing requirements placed on the Key Exchange step SECURITY PROTOCOL OUT command (see 7.7.3.6.6) ensure that this dependency is satisfied.

If no other errors are detected and any of the following conditions are true:

a) the device server does not include an SA_AUTH_OUT algorithm descriptor with the ALGORITHM IDENTIFIER field set to SA_AUTH_NONE in the IKEv2-SCSI SA Creation Capabilities payload in the parameter data returned by the Device Server Capabilities step SECURITY PROTOCOL IN command;
b) the device server does not include an SA_AUTH_IN algorithm descriptor with the ALGORITHM IDENTIFIER field set to SA_AUTH_NONE in the IKEv2-SCSI SA Creation Capabilities payload in the parameter data returned by the Device Server Capabilities step SECURITY PROTOCOL IN command; or
c) the application client does not set the ALGORITHM IDENTIFIER field to SA_AUTH_NONE in the SA_AUTH_OUT algorithm descriptor and the SA_AUTH_IN algorithm descriptor in the IKEv2-SCSI SA Cryptographic Algorithms payload sent to the device server in the Key Exchange step SECURITY PROTOCOL OUT command;

then:

a) the Authentication step is processed;
b) the IKEv2-SCSI CCS consists of the two Key Exchange step commands and two Authentication step commands;
c) the device server requires the IKEv2-SCSI SAUT Cryptographic Algorithms payload to be absent from the parameter data sent by the Key Exchange step SECURITY PROTOCOL OUT command;
d) the device server omits the IKEv2-SCSI SAUT Cryptographic Algorithms payload from the parameter data returned by the Key Exchange step SECURITY PROTOCOL IN command;
e) the device server requires the IKEv2-SCSI SAUT Cryptographic Algorithms payload to be present in the parameter data sent by the Authentication step SECURITY PROTOCOL OUT command;
f) the device server returns the IKEv2-SCSI SAUT Cryptographic Algorithms payload in the parameter data returned by the Authentication step SECURITY PROTOCOL IN command; and
g) SA creation occurs upon the completion of the Authentication step.

SA participants should perform the Authentication step unless man-in-the-middle attacks (see 5.13.1.4) are not of concern or are prevented by a means outside the scope of this standard (e.g., physical security of the transport).

NOTE 14 - Omission of the Authentication step provides no defense against a man-in-the-middle adversary that is capable of modifying SCSI commands. Such an adversary is able to insert itself as an intermediary on the created SA without knowledge of the SA participants, thereby completely subverting the intended security. Omission of the Authentication step is only appropriate in environments where the absence of such adversaries is assured by other means (e.g., a direct physical connection between the systems on which the application client and device server or use of end-to-end security in the SCSI transport security such as FC-SP).

5.16.4.2 IKEv2-SCSI Protocol summary

This subclause summarizes the IKE-v2-SCSI payloads (see 7.7.3.5) that are exchanged between an application client and a device server during all steps of an IKEv2-SCSI SA creation transaction (see 3.1.130) using message diagrams. Each IKEv2-SCSI step (see 5.16.4.1) is shown in a separate figure. The contents of a payload (e.g., Key Exchange) may not be the same in both directions of transfer.

Figure 10 shows the Device Server Capabilities step (see 5.16.4.5). The Device Server Capabilities step consists of a SECURITY PROTOCOL IN command carrying an IKEv2-SCSI SA Creation Capabilities payload (see 7.7.3.5.12). The IKEv2-SCSI header is not used.

Figure 10 — IKEv2-SCSI Device Server Capabilities step

The IKEv2-SCSI SA Creation Capabilities payload indicates the device server's capabilities for SA creation.

Figure 11 shows the Key Exchange step (see 5.16.4.6). The Key Exchange step consists of a SECURITY PROTOCOL OUT command followed by a SECURITY PROTOCOL IN command.

Figure 11 — IKEv2-SCSI Key Exchange step

The IKEv2-SCSI Timeout Values payload (see 7.7.3.5.15) contains timeouts for SA creation and usage.

The IKEv2-SCSI SA Cryptographic Algorithms payloads (see 7.7.3.5.13) are used to select and agree on the cryptographic algorithms used for creating the SA.

If the Authentication step is skipped (see 5.16.4.1), the IKEv2-SCSI SAUT Cryptographic Algorithms payloads (see 7.7.3.5.14) are used to select and agree on usage of the SA and the cryptographic algorithms used by the created

SA. If the Authentication step is processed (i.e., not skipped), the SA usage and algorithms selection is performed there.

The Key Exchange payload (see 7.7.3.5.3) and Nonce payload (see 7.7.3.5.8) are part of the key and nonce exchanges that are used to generate the IKEv2-SCSI keys (see 3.1.70) and SA keys (see 3.1.132).

The optional Certificate Request payload or payloads (see 7.7.3.5.6) enables the device server to request a certificate from the application client. If the Authentication step is being skipped (see 5.16.4.1), the device server shall not include any Certificate Request payloads in the parameter data. Use of the Certificate Request payload is described in 5.16.4.3.3.4.

Figure 12 shows the Authentication step (see 5.16.4.7). The Authentication step consists of a SECURITY PROTOCOL OUT command followed by a SECURITY PROTOCOL IN command.

Figure 12 — IKEv2-SCSI Authentication step

An Encrypted payload (see 7.7.3.5.11) contains all other Authentication step payloads that are protected using the cryptographic algorithms determined by the IKEv2-SCSI SA Cryptographic Algorithms payloads (see 7.7.3.5.13) in the Key Exchange step.

The Identification payloads (see 7.7.3.5.4) contain the identities to be authenticated. These identities are not required to be SCSI names or identifiers.

The IKEv2-SCSI SAUT Cryptographic Algorithms payloads (see 7.7.3.5.14) are used to select and agree on usage of the SA and the cryptographic algorithms used by the created SA.

The optional Certificate payload or payloads (see 7.7.3.5.5) respond to the Certificate Request payload(s) sent by the device server in the Key Exchange step SECURITY PROTOCOL IN command.

The optional Certificate Request payload or payloads (see 7.7.3.5.6) allows an application client to request the delivery of a Certificate payload (see 7.7.3.5.5) in the parameter data for the Authentication step SECURITY PROTOCOL IN command (see 5.16.4.3.3.4).

The optional Notify payload (see 7.7.3.5.9) provides a means for the application client to inform the device server that this is the only SA being used between them, and that the device server should discard state for any other SAs created by the same application client.

The Authenticate payloads (see 7.7.3.5.7) authenticate not only the SA participants, but also the entire protocol sequence (e.g., the Authenticate payloads prevent a man-in-the-middle attack from succeeding).

Figure 13 shows the Delete operation (see 5.16.4.11). The Delete operation consists of a SECURITY PROTOCOL OUT command.

Figure 13 — IKEv2-SCSI Delete operation

An Encrypted payload (see 7.7.3.5.11) contains the Delete payload that is protected using the cryptographic algorithms determined by the IKEv2-SCSI SA Cryptographic Algorithms payloads (see 7.7.3.5.13) in the Key Exchange step that was used to create the SA.

The Delete payload (see 7.7.3.5.10) specifies the SA to be deleted.

5.16.4.3 IKEv2-SCSI Authentication

5.16.4.3.1 Overview

IKEv2-SCSI authentication includes these security functions:

a) the application client and device server each establish an identity by demonstrating knowledge of a secret authentication key associated with that identity;
b) the application client demonstrates knowledge of the current device server capability information; and
c) the application client and device server check the integrity of the current IKEv2-SCSI CCS.

An IKEv2 SCSI authentication algorithm accomplishes these functions by generating and verifying authentication data based on a concatenation of bytes that includes device server capability information and an appropriate portion of the IKEv2-SCSI parameter data from the IKEv2-SCSI Key Exchange step (see 5.16.4.6).

An authentication key associated with an identity is used to generate authentication data. IKEv2-SCSI transfers the authentication data in the AUTHENTICATION DATA field of the IKEv2-SCSI Authentication payload (see 7.7.3.5.7). The recipient of an IKEv2-SCSI Authentication payload uses a verification key associated with the identity to verify the authentication data. The identity is:

a) transferred in the IDENTIFICATION DATA field of the appropriate IKEv2-SCSI Identification payload (see 7.7.3.5.4); or
b) obtained from a certificate transferred in the IKEv2-SCSI Certificate payload (see 7.7.3.5.5).

IKEv2-SCSI Authentication is bidirectional (i.e., both the application client and the device server authenticate). IKEv2-SCSI Authentication is skipped when the application client and device server agree to do so during the Key Exchange step (see 5.16.4.3.4).

The following IKEv2-SCSI Authentication methods are defined:

a) Pre-Shared Key (see 5.16.4.3.2): the authentication key is also used as the verification key; and
b) Digital Signature (see 5.16.4.3.3): the verification key and authentication key form a public/private key pair. The authentication data is a digital signature based on asymmetric cryptography.

Certificates and the IKEv2-SCSI Certificate payload may be used to provide verification keys for digital signatures to application clients and device servers.

5.16.4.3.2 Pre-shared key authentication

Pre-shared key authentication uses a single cryptographic algorithm to both generate and verify authentication data. A pre-shared key is associated with an identity that is transferred in the IDENTIFICATION DATA field of the appropriate Identification payload (see 7.7.3.5.4). The pre-shared key serves as both the authentication key and the verification key for the identity.

NOTE 15 - A pre-shared key that is not kept secret may compromise the security properties of IKEv2-SCSI.

If pre-shared key authentication is used, then the pre-shared key is the key for the cryptographic algorithm. Authentication data is generated by applying the cryptographic algorithm with this key to the input data (e.g., the applicable concatenation of bytes described in 7.7.3.5.7).

Verification of the authentication data shall consist of:

1) computing the expected contents of the AUTHENTICATION DATA field of the Authentication payload (see 7.7.3.5.7) using the input data and a verification key associated with the identity received in the Identification payload (see 7.7.3.5.4); and
2) comparing the expected contents to the actual contents of the AUTHENTICATION DATA field.

Verification is successful if the expected contents match the actual contents, otherwise verification is not successful.

The pre-shared key requirements in RFC 4306 shall apply to IKEv2-SCSI pre-shared keys, including the following requirements on interfaces for provisioning pre-shared keys:

a) ASCII strings of at least 64 bytes shall be supported;
b) a null terminator shall not be added to any input before it is used as a pre-shared key;
c) a hexadecimal ASCII encoding of the pre-shared key shall be supported; and
d) ASCII encodings other than hexadecimal may be supported. Support for any such encoding shall include specification of the algorithm for translating the encoding to a binary string as part of the interface.

The following requirements for pre-shared keys apply in addition to those found in RFC 4306:

a) a pre-shared key shall be associated with one identity;
b) the same pre-shared key shall not be used to authenticate both an application client and a device server;
c) the same pre-shared key should not be used for a group of application clients or a group of device servers;
d) the means for provisioning pre-shared keys are outside the scope of this standard; and
e) information about the size of the pre-shared key shall be stored at the same time that the pre-shared key is stored.

5.16.4.3.3 Digital signature authentication

5.16.4.3.3.1 Overview

Digital signature authentication uses a matched pair of signature and verification cryptographic algorithms to generate and verify authentication data that is a digital signature. A public/private key pair is associated with an identity. The private key is used as the authentication key for the identity. The public key is used as the verification key for the identity.

NOTE 16 - A private authentication key that is not kept secret may compromise the security properties of IKEv2-SCSI.

If digital signature authentication is used, then the private key is the key for the signature algorithm. A digital signature is generated by applying the signature algorithm with this private key to the input data (e.g., the applicable concatenation of bytes described in 7.7.3.5.7).

Verification of the digital signature shall consist of using the public verification key associated with the identity and the input data to verify the digital signature received as the contents of the AUTHENTICATION DATA field of the Authentication payload (see 7.7.3.5.7). Verification is successful if the digital signature is a valid digital signature over the input data, otherwise verification is not successful.

The means by which an application client or device server obtains a private authentication key are outside the scope of this standard. An identity and associated public verification key are obtained as follows:

a) if certificates are used for digital signature authentication, then the identity and the associated public verification key are obtained from a certificate transferred in the first the IKEv2-SCSI Certificate payload (see 5.16.4.3.3.4); or
b) if certificates are not used for digital signature authentication, then the identity is transferred in the IDENTIFICATION DATA field of the appropriate IKEv2-SCSI Identification payload (see 7.7.3.5.4) and the public verification key may be:
   A) transferred as a raw RSA key in an IKEv2-SCSI Certificate payload; or
   B) obtained by means that are outside the scope of this standard.

If certificates are not used for digital signature authentication, the association between the identity and the public key should be verified by means outside the scope of this standard.

5.16.4.3.3.2 Certificates and digital signature authentication

A certificate (see RFC 5280) is a data structure that contains:

a) an identity;
b) a public key for that identity;
c) additional relevant information that may constrain use of the public key;
d) the identity of a certification authority (see RFC 5280); and
e) a digital signature generated by that certification authority.

If the identity and associated public key used to verify a digital signature are obtained from a certificate, the certification path from the certificate to a trust anchor should be validated (see RFC 5280). If certification path validation is not successful, verification of the digital signature for that identity shall fail independent of whether the digital signature is valid.

The means by which an application client or device server obtains a trust anchor are outside the scope of this standard.

5.16.4.3.3.3 Example of certificate use for digital signature authentication

An example of certificate use involves an application client or device server that trusts a certification authority. Based on this trust, the public key of that certification authority is used to validate a certificate presented as part of authentication. Successful validation of that certificate establishes that the public key in that certificate is associated with the identity in the certificate. That public key is then used to verify the digital signature in the Authentication payload (see 7.7.3.5.7).

In this example, providing a certificate as part of the IKEv2-SCSI Authentication step (see 5.16.4.7) allows a single certification authority public key to serve as a trust anchor for verification of digital signatures for any identity that has been issued a certificate by that certification authority, avoiding the need to obtain a public key for each identity by other means.

Validating a certificate includes multiple checks beyond verifying the signature, and the validation may traverse a certification path composed of multiple certificates (see RFC 5280).

5.16.4.3.3.4 Handling of the Certificate Request payload and the Certificate payload

As detailed in this subclause, a Certificate Request payload (see 7.7.3.5.6) in one set of parameter data requests the delivery of a Certificate payload (see 7.7.3.5.5) in the next set of parameter data transferred. The purpose of these IKEv2-SCSI protocol elements is as follows:

a) each SA participant is allowed to require the delivery of a Certificate payload by the other SA participant for use in authentication; and
b) each Certificate Request payload indicates the trust anchors (see RFC 4306) list used by the device server or application client when PKI-based Authentication is being used with certificates that are not self signed (see RFC 5280).

The presence of one or more Certificate Request payloads in the Key Exchange step SECURITY PROTOCOL IN command (see 5.16.4.6.3) parameter data indicates that the device server requires the application client to send a Certificate payload in the Authentication step SECURITY PROTOCOL OUT command (see 5.16.4.7.2).

The presence of one or more Certificate Request payloads in the Authentication step SECURITY PROTOCOL OUT command parameter list specifies that the application client requires the device server to send a Certificate payload in the Authentication step SECURITY PROTOCOL IN command (see 5.16.4.7.3).

If any Certificate payloads are included in the parameter data, the first Certificate payload shall contain the public key used to verify the Authentication payload. Additional Certificate payloads may be sent to assist in establishing a certification path from the certificate in the first payload to a trust anchor (see RFC 4306 and RFC 5280).

The application client and device server may use different authentication methods that require or do not require the use of Certificate payloads. The presence or absence of Certificate Request payloads and Certificate payloads may vary in any of the commands described in this subclause.

5.16.4.3.4 Constraints on skipping the Authentication step

In the Device Server Capabilities step (see 5.16.4.5), the parameter data returned by the SECURITY PROTOCOL IN command (see 7.7.2.3.2) contains the IKEv2-SCSI SA Creation Algorithms payload (see 7.7.3.5.12) that contains one or more SA_AUTH_OUT IKEv2-SCSI cryptographic algorithm descriptors (see 7.7.3.6.6) and one or more SA_AUTH_IN IKEv2-SCSI cryptographic algorithm descriptors.

The device server shall permit the Authentication step to be omitted (see 5.16.4.1) if:

a) the ALGORITHM IDENTIFIER field is set to SA_AUTH_NONE (see 7.7.3.6.6) in one of the SA_AUTH_OUT IKEv2-SCSI cryptographic algorithm descriptors returned in the Device Server Capabilities step; and
b) the ALGORITHM IDENTIFIER field is set to SA_AUTH_NONE in one of the SA_AUTH_IN IKEv2-SCSI cryptographic algorithm descriptors returned in the Device Server Capabilities step.

The methods for configuring a device server to return SA_AUTH_NONE are outside the scope of this standard. Device servers shall not be manufactured to return SA_AUTH_NONE as an Authentication payload authentication algorithm type in the Device Server Capabilities step.

In the Key Exchange step SECURITY PROTOCOL OUT command (see 5.16.4.6.2), the application client requests that the Authentication step be omitted by setting the ALGORITHM IDENTIFIER field to SA_AUTH_NONE in:

a) the SA_AUTH_OUT cryptographic algorithm descriptor in the IKEv2-SCSI SA Cryptographic Algorithms payload (see 7.7.3.5.13); and
b) the SA_AUTH_IN cryptographic algorithm descriptor in the IKEv2-SCSI SA Cryptographic Algorithms payload.

To ensure adequate SA security, the application client should not select the SA_AUTH_NONE value as an Authentication payload authentication algorithm type unless:

a) an SA_AUTH_OUT IKEv2-SCSI cryptographic algorithm descriptor and an SA_AUTH_IN IKEv2-SCSI cryptographic algorithm descriptor from the Device Server Capabilities step indicates SA_AUTH_NONE availability; and
b) the application client is configured to omit the Authentication step.

NOTE 17 - When SA_AUTH_NONE is used, IKEv2-SCSI has no protection against any man-in-the-middle attacks. Enabling return of the SA_AUTH_NONE authentication algorithm type in the Device Capabilities step, and allowing an application client to select SA_AUTH_NONE in the Key Exchange step are administrative security policy decisions that absence of authentication is acceptable. Such decisions should only be made in situations where active attacks on IKEv2-SCSI are not of concern (e.g., direct attachment of a SCSI initiator device and a SCSI target device, or an end-to-end secure service delivery subsystem such as Fibre Channel secured by an end-to-end FC-SP SA).

5.16.4.4 Summary of IKEv2-SCSI shared keys nomenclature and shared key sizes

To facilitate use of the normative references made by this standard, the IKEv2-SCSI shared keys (see 3.1.162) are named as shown in table 68.

Table 68 — IKEv2-SCSI shared key names and SA shared key names

Name	Description	SA parameter (see 3.1.133) that stores this shared key
Shared keys used only during Authentication step		
SK_pi	The shared key used to construct the Authentication payload (see 7.7.3.5.7) for the SECURITY PROTOCOL OUT parameter list in the Authentication step (see 5.16.4.7.2).	shall not be stored in any SA parameter
SK_pr	The shared key used to construct the Authentication payload for the SECURITY PROTOCOL IN parameter data in the Authentication step.	
Shared keys used during IKEv2-SCSI SA creation and management		
SK_ai [a]	The shared key used to integrity check the Encrypted payload in the SECURITY PROTOCOL OUT parameter list in the: a) Authentication step; and b) IKEv2-SCSI Delete operation (see 5.16.4.11).	MGMT_DATA
SK_ar [a]	The shared key used to integrity check the Encrypted payload (see 7.7.3.5.11) in the SECURITY PROTOCOL IN parameter data in the Authentication step (see 5.16.4.7.3).	
SK_ei [a]	The shared key used to encrypt the Encrypted payload in the SECURITY PROTOCOL OUT parameter list in the: a) Authentication step; and b) IKEv2-SCSI Delete operation.	
SK_er [a]	The shared key used to encrypt the Encrypted payload in the SECURITY PROTOCOL IN parameter data in the Authentication step.	
Shared key used to construct the SA keys (see 3.1.132)		
SK_d	The shared key material that is used as input to the KDF that generates the KEYMAT SA parameter (see 3.1.133) bytes for the SA.	KEY_SEED

[a] The SA keys (see 3.1.132) SK_ai, SK_ar, SK_ei, and SK_er are stored in the KEYMAT SA parameter. SK_ai is intended to be used for integrity checking data in a Data-Out Buffer. SK_ar is intended to be used for integrity checking data in a Data-In Buffer. SK_ei is intended to be used for encrypting data in a Data-Out Buffer. SK_er is intended to be used for encrypting data in a Data-In Buffer.

The sizes of the shared keys are determined as shown in table 69.

Table 69 — Shared key size determination

Name	Shared key size determination	
	Separate encryption and integrity checking [a, b]	Combined mode encryption and integrity checking [a, b]
SK_ai and SK_ar [c]	The shared key length shown in table 492 for the value in the ALGORITHM IDENTIFIER field of the INTEG IKEv2-SCSI cryptographic algorithm descriptor (see 7.7.3.6.4) [b]	0
SK_er and SK_ei [c]	The shared key length shown in the KEY LENGTH field of the ENCR IKEv2-SCSI cryptographic algorithm descriptor (see 7.7.3.6.2) [b]	The shared key length shown in the KEY LENGTH field of the ENCR IKEv2-SCSI cryptographic algorithm descriptor plus the number of salt bytes shown in table 488 (see 7.7.3.6.2) [b]
SK_pi and SK_pr [c]	The shared key length is equal to the PRF output length (see table 490) associated with the value in the ALGORITHM IDENTIFIER field of the PRF IKEv2-SCSI cryptographic algorithm descriptor (see 7.7.3.6.3) in the IKEv2-SCSI SA Cryptographic Algorithms payload (see 7.7.3.5.13)	
SK_d		

[a] The use of combined mode encryption and integrity checking is indicated by the AUTH_COMBINED value in the algorithm identifier field of the INTEG IKEv2-SCSI cryptographic algorithm descriptor (see 7.7.3.6.4).

[b] For the shared keys used to create an SA, the algorithm descriptor is located in an IKEv2-SCSI SA Cryptographic Algorithms payload (see 7.7.3.5.13). For the shared keys used after the SA is created (i.e., the KEYMAT SA parameter), the algorithm descriptor is located in an IKEv2-SCSI SAUT Cryptographic Algorithms payload (see 7.7.3.5.14).

[c] To accommodate the need for two shared keys of the specified size, the shared key length shown is doubled for the purposes of shared key generation.

5.16.4.5 Device Server Capabilities step

In the Device Server Capabilities step, the application client sends a SECURITY PROTOCOL IN command (see 6.30) with the SECURITY PROTOCOL field set to SA creation capabilities (i.e., 40h) and the SECURITY PROTOCOL SPECIFIC field set to 0101h.

The device server returns the SECURITY PROTOCOL IN parameter data specified by the SECURITY PROTOCOL SPECIFIC field (see 7.7.2.2) and the parameter data (see 7.7.2.3.2) contains an IKEv2-SCSI SA Creation Capabilities payload (see 7.7.3.5.12).

In the Device Server Capabilities step, the device server shall return parameter data containing the IKEv2-SCSI cryptographic algorithm descriptors (see 7.7.3.6) in at least one complete row shown in table 70.

Table 70 — Device Server Capabilities step parameter data requirements

IKEv2-SCSI cryptographic algorithm descriptor [a]					
ENCR (see 7.7.3.6.2)	**PRF (see 7.7.3.6.3)**	**INTEG (see 7.7.3.6.4)**	**D-H (see 7.7.3.6.5)**	**SA_AUTH_OUT (see 7.7.3.6.6)**	**SA_AUTH_IN (see 7.7.3.6.6)**
The ALGORITHM IDENTIFIER field set to 8001 0014h (i.e., AES-GCM) and the KEY LENGTH field set to 0010h	The ALGORITHM IDENTIFIER field set to 8002 0005h (i.e., IKEv2-use based on SHA-256)	The ALGORITHM IDENTIFIER field set to F003 0001h (i.e., AUTH_ COMBINED)	The ALGORITHM IDENTIFIER field set to 8004 000Eh (i.e., 2 048-bit MODP group (finite field D-H))	The ALGORITHM IDENTIFIER field set to 00F9 0001h (i.e., RSA Digital Signature)	The ALGORITHM IDENTIFIER field set to 00F9 0001h (i.e., RSA Digital Signature)
The ALGORITHM IDENTIFIER field set to 8001 000Ch (i.e., AES-CBC) and the KEY LENGTH field set to 0020h	The ALGORITHM IDENTIFIER field set to 8002 0007h (i.e., IKEv2-use based on SHA-512)	The ALGORITHM IDENTIFIER field set to 8003 000Eh (i.e., AUTH_HMAC_ SHA2_ 512_256)	The ALGORITHM IDENTIFIER field set to 8004 0015h (i.e., 521-bit random ECP group)	The ALGORITHM IDENTIFIER field set to 00F9 000Bh (i.e., ECDSA with SHA-512 on the P-521 curve)	The ALGORITHM IDENTIFIER field set to 00F9 000Bh (i.e., ECDSA with SHA-512 on the P-521 curve)

[a] The IKEv2-SCSI cryptographic algorithm descriptors shown in all the columns in at least one row in this table shall be returned in the Device Server Capabilities step parameter data.

In the Device Server Capabilities step, the device server shall return parameter data containing one SA_AUTH_OUT IKEv2-SCSI cryptographic algorithm descriptor with the ALGORITHM IDENTIFIER field is set to SA_AUTH_NONE and one SA_AUTH_IN IKEv2-SCSI cryptographic algorithm descriptor with the ALGORITHM IDENTIFIER field is set to SA_AUTH_NONE if any of the following are true:

a) the ALGORITHM IDENTIFIER field is set to SA_AUTH_NONE in one of the SA_AUTH_OUT IKEv2-SCSI cryptographic algorithm descriptors returned in the Device Server Capabilities step; or
b) the ALGORITHM IDENTIFIER field is set to SA_AUTH_NONE in one of the SA_AUTH_IN IKEv2-SCSI cryptographic algorithm descriptors returned in the Device Server Capabilities step.

The device server capabilities returned in the SECURITY PROTOCOL IN parameter data may be changed at any time via interfaces that are outside the scope of this standard, however, such changes shall not take effect until at least one application client has been notified of the new capabilities via the parameter data returned by the Device Server Capabilities step SECURITY PROTOCOL IN command. Management applications may ensure that their device server capabilities changes to take effect by sending a Device Server Capabilities step SECURITY PROTOCOL IN command to the device server after the changes have been made.

When the device server capabilities change (i.e., upon completion of the processing for a Device Server Capabilities step SECURITY PROTOCOL IN command that reported changed information in its parameter data), the device server shall establish a unit attention condition for the initiator port associated with every I_T nexus except the I_T nexus on which the Device Server Capabilities step SECURITY PROTOCOL IN command was received (see SAM-4), with the additional sense code set to SA CREATION CAPABILITIES DATA HAS CHANGED.

The Device Server Capabilities step participates in the negotiation to skip the Authentication step as described in 5.16.4.3.4.

NOTE 18 - The Device Server Capabilities step has no IKEv2 exchange equivalent in RFC 4306. This step replaces most of IKEv2's negotiation by having the application client obtain the supported capabilities from the device server.

5.16.4.6 IKEv2-SCSI Key Exchange step

5.16.4.6.1 Overview

The Key Exchange step consists of a Diffie-Hellman key exchange with nonces (see RFC 4306) and is accomplished as follows:

1) a SECURITY PROTOCOL OUT command (see 5.16.4.6.2);
2) a SECURITY PROTOCOL IN command (see 5.16.4.6.3); and
3) key exchange completion (see 5.16.4.6.4)

NOTE 19 - The Key Exchange step corresponds to the IKEv2 IKE_SA_INIT exchange in RFC 4306, except that determination of device server capabilities has been moved to the Device Server Capabilities step.

5.16.4.6.2 Key Exchange step SECURITY PROTOCOL OUT command

To send its key exchange message to the device server, the application client sends a SECURITY PROTOCOL OUT command (see 6.31) with the SECURITY PROTOCOL field set to IKEv2-SCSI (i.e., 41h) and the SECURITY PROTOCOL SPECIFIC field set to 0102h. The parameter list consists of an IKEv2-SCSI header (see 7.7.3.4) and the following:

1) an IKEv2-SCSI Timeout Values payload (see 7.7.3.5.15);
2) an IKEv2-SCSI SA Cryptographic Algorithms payload (see 7.7.3.5.13);
3) if the Authentication step is skipped (see 5.16.4.1), an IKEv2-SCSI SAUT Cryptographic Algorithms payload (see 7.7.3.5.14);
4) a Key Exchange payload (see 7.7.3.5.3); and
5) a Nonce payload (see 7.7.3.5.8).

The IKEv2-SCSI Timeout Values payload contains the inactivity timeouts that apply to this IKEv2-SCSI SA creation transaction (see 3.1.130) and the SA (see 3.1.155) that is created.

The IKEv2-SCSI SA Cryptographic Algorithms payload selects the cryptographic algorithms from among those returned in the Device Server Capabilities step (see 5.16.4.5) to be used in the creation of the SA.

If the Authentication step is skipped, the IKEv2-SCSI SAUT Cryptographic Algorithms payload contains the following information about the SA to be created:

a) the cryptographic algorithms selected by the application client from among those returned in the Device Server Capabilities step; and
b) the usage data (see 7.7.3.5.14), if any, that is specific to the SA.

If the application client is unable to select a set of algorithms that are appropriate for the intended creation and usage of the SA, the application client should not perform the Key Exchange step to request the creation of an SA.

IKEv2-SCSI SA Cryptographic Algorithms payload error checking requirements that ensure a successful negotiation of SA creation algorithms are described in 7.7.3.5.13 and 7.7.3.6.

IKEv2-SCSI SAUT Cryptographic Algorithms payload error checking requirements needed to ensure a successful SA creation are described in 7.7.3.5.14 and 7.7.3.6.

The Key Exchange payload contains the application client's Diffie-Hellman value.

The Nonce payload contains the application client's random nonce (see 3.1.122).

5.16.4.6.3 Key Exchange step SECURITY PROTOCOL IN command

If the Key Exchange step SECURITY PROTOCOL OUT command (see 5.16.4.6.2) completes with GOOD status, the application client sends a SECURITY PROTOCOL IN command (see 6.30) with the SECURITY PROTOCOL field set to IKEv2-SCSI (i.e., 41h) and the SECURITY PROTOCOL SPECIFIC field set to 0102h to obtain the device server's key exchange message.

The parameter data returned by the device server in response to the SECURITY PROTOCOL IN command shall contain an IKEv2-SCSI header (see 7.7.3.4) and the following:

1) an IKEv2-SCSI SA Cryptographic Algorithms payload (see 7.7.3.5.13);
2) if the Authentication step is skipped (see 5.16.4.1), an IKEv2-SCSI SAUT Cryptographic Algorithms payload (see 7.7.3.5.14);
3) a Key Exchange payload (see 7.7.3.5.3);
4) a Nonce payload (see 7.7.3.5.8); and
5) zero or more Certificate Request payloads (see 7.7.3.5.6).

As part of processing of the Key Exchange step SECURITY PROTOCOL IN command, the device server shall:

a) associate the SECURITY PROTOCOL IN command to the last Key Exchange step SECURITY PROTOCOL OUT command received on the I_T_L nexus. If the device server is maintaining state for at least one IKEv2-SCSI CCS and the device server is unable to establish this association, then:
   A) the SECURITY PROTOCOL IN command shall be terminated with CHECK CONDITION status, with the sense key set to NOT READY, and the additional sense code set to LOGICAL UNIT NOT READY, SA CREATION IN PROGRESS; and
   B) the device server shall continue the IKEv2-SCSI CCS.

   If the device is not maintaining state for at least one IKEv2-SCSI CSS, then the SECURITY PROTOCOL IN command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to COMMAND SEQUENCE ERROR.
b) return the device server’s SAI in the IKEv2-SCSI header IKE_SA DEVICE SERVER SAI field;
c) return the IKEv2-SCSI SA Cryptographic Algorithms payload containing:
   A) the SA creation cryptographic algorithms supplied by the application client in the Key Exchange step SECURITY PROTOCOL OUT command parameter list; and
   B) the device server's SAI (see 3.1.156) in the SAI field (see 7.7.3.5.13);
d) if the Authentication step is skipped, return the IKEv2-SCSI SAUT Cryptographic Algorithms payload containing:
   A) the SA usage cryptographic algorithms supplied by the application client in the Key Exchange step SECURITY PROTOCOL OUT command parameter list; and
   B) the device server's SAI in the SAI field (see 7.7.3.5.14);
e) return information about the completed Diffie-Hellman exchange with the Key Exchange payload; and
f) return device server’s random nonce (see 3.1.122) in the Nonce payload.

If the Key Exchange step SECURITY PROTOCOL IN command (see 5.16.4.6.3) completes with GOOD status, the application client should copy the device server’s SAI from the IKE_SA DEVICE SERVER SAI field in the IKEv2-SCSI header to the state it is maintaining for the IKEv2-SCSI CCS.

Except for the SAI field, the application client should compare the fields in the IKEv2-SCSI SA Cryptographic Algorithms payload and the IKEv2-SCSI SAUT Cryptographic Algorithms payload, if any, to the values sent in the Key Exchange step SECURITY PROTOCOL OUT command (see 5.16.4.6.2). If the application client detects differences in the contents of the payloads other than in the SAI field, the application client should abandon the IKEv2-SCSI CCS and notify the device server that the IKEv2-SCSI CCS is being abandoned as described in 5.16.4.10.

5.16.4.6.4 Key Exchange step completion

Before completing the Key Exchange step SECURITY PROTOCOL IN command (see 5.16.4.6.3) with GOOD status the device server shall complete the Key Exchange step as described in this subclause.

Upon receipt of GOOD status for the Key Exchange step SECURITY PROTOCOL IN command the application client should complete the Key Exchange step as described in this subclause.

If the Key Exchange step does not end with the IKEv2-SCSI CCS being abandoned (see 5.16.4.10), then the contents of the SA_AUTH_OUT IKEv2-SCSI cryptographic algorithm descriptor and SA_AUTH_IN IKEv2-SCSI cryptographic algorithm descriptor (see 7.7.3.6.6) in the IKEv2-SCSI SA Cryptographic Algorithms payload specify how the shared key exchanged by the Key Exchange step SECURITY PROTOCOL OUT command and the Key Exchange step SECURITY PROTOCOL IN command is used to generate additional shared keys as follows:

a) if the ALGORITHM IDENTIFIER field in both descriptors contain SA_AUTH_NONE, then the SA participants generate the SA, including the generation of the shared keys used for SA management (e.g., SA creation and management) and the shared keys used by the created SA as defined in 5.16.4.9; or
b) if the ALGORITHM IDENTIFIER field in both descriptors contain a value other than SA_AUTH_NONE, then the SA participants generate shared keys (see 5.16.4.8.3) for the following:
   A) seeding the Authentication step generation of the shared keys used by the created SA; and
   B) SA creation and management.

5.16.4.6.5 After the Key Exchange step

Processing of the IKEv2-SCSI CCS subsequent to completion of the Key Exchange step (see 5.16.4.6.4) depends on the contents of the SA_AUTH_OUT IKEv2-SCSI cryptographic algorithm descriptor and SA_AUTH_IN IKEv2-SCSI cryptographic algorithm descriptor (see 7.7.3.6.6) in the IKEv2-SCSI SA Cryptographic Algorithms payload as follows:

a) if the ALGORITHM IDENTIFIER field in both descriptors contain SA_AUTH_NONE, then processing of the IKEv2-SCSI CCS is finished and the generated SA is ready for use; or
b) if the ALGORITHM IDENTIFIER field in both descriptors contain a value other than SA_AUTH_NONE, then processing of the IKEv2-SCSI CCS continues as follows:
   A) the Authentication step is performed (see 5.16.4.7); and
   B) the SA participants generate the SA, including the generation of the shared keys used for SA management and the shared keys used by the created SA as defined in 5.16.4.9.

5.16.4.7 IKEv2-SCSI Authentication step

5.16.4.7.1 Overview

The Authentication step performs the following functions:

a) authenticates both the application client and the device server;
b) protects the previous steps of the protocol; and
c) cryptographically binds the authentication and the previous steps to the created SA.

The Authentication step is accomplished as follows:

1) a SECURITY PROTOCOL OUT command (see 5.16.4.7.2); and
2) a SECURITY PROTOCOL IN command (see 5.16.4.7.3).

The parameter data for both commands shall be encrypted and integrity protected using the algorithms and keys determined in the Key Exchange step (see 5.16.4.6).

NOTE 20 - The Authentication step corresponds to the IKEv2 IKE_AUTH exchange in RFC 4306.

5.16.4.7.2 Authentication step SECURITY PROTOCOL OUT command

To send its authentication message to the device server, the application client sends a SECURITY PROTOCOL OUT command (see 6.31) with the SECURITY PROTOCOL field set to IKEv2-SCSI (i.e., 41h) and the SECURITY PROTOCOL SPECIFIC field set to 0103h. The parameter data consists of the IKEv2-SCSI header (see 7.7.3.4) and an Encrypted payload (see 7.7.3.5.11) that:

a) is integrity checked and encrypted as follows:
   A) using separate algorithms as follows:
      a) integrity checked using the following:
         A) the algorithm specified by the INTEG IKEv2-SCSI algorithm descriptor (see 7.7.3.6.4) in the IKEv2-SCSI SA Cryptographic Algorithms payload (see 7.7.3.5.13) from the Key Exchange step (see 5.16.4.6); and
         B) the SK_ai shared key (see 5.16.4.4);

         and
      b) encrypted using the following:
         A) the algorithm specified by the ENCR IKEv2-SCSI algorithm descriptor (see 7.7.3.6.2) in the IKEv2-SCSI SA Cryptographic Algorithms payload from the Key Exchange step; and
         B) the SK_ei shared key (see 5.16.4.4);

      or
   B) using a combined integrity check and encryption algorithm that uses the following (i.e., if the INTEG IKEv2-SCSI algorithm descriptor indicates AUTH_COMBINED):
      a) the algorithm specified by the ENCR IKEv2-SCSI algorithm descriptor in the IKEv2-SCSI SA Cryptographic Algorithms payload; and
      b) the SK_ei shared key with additional salt bytes as described in 5.16.4.8.1 and table 69 (see 5.16.4.4);

   and
b) contains the following:
   1) an Identification – Application Client payload (see 7.7.3.5.4);
   2) an IKEv2-SCSI SAUT Cryptographic Algorithms payload (see 7.7.3.5.14);
   3) zero or more Certificate payloads (see 7.7.3.5.5);
   4) zero or more Certificate Request payloads (see 7.7.3.5.6);
   5) zero or one Notify payload (see 7.7.3.5.9); and
   6) an Authentication payload (see 7.7.3.5.7).

Before performing any checks on data contained in the Encrypted payload, the device server shall validate the SECURITY PROTOCOL OUT command parameter data as follows:

a) the device server shall compare the IKE_SA APPLICATION CLIENT SAI field and the IKE_SA DEVICE SERVER SAI field in the IKEv2-SCSI header to the SAI values it is maintaining for the IKEv2-SCSI CCS, if any, being maintained for the I_T_L nexus on which the SECURITY PROTOCOL OUT command was received as described in 7.7.3.4; and
b) the device server shall decrypt and check the integrity of the Encrypted payload as described in 7.7.3.5.11.4.

Errors detected during the decryption and integrity checking of the Encrypted payload shall be handled as described in 7.7.3.8.2.

In the SECURITY PROTOCOL OUT command parameter list, the application client:

a) sends information about the SA usage and cryptographic algorithms;
b) sends its identity in the Identification – Application Client payload;
c) sends information proving its knowledge of the secret corresponding to its identity in the Authentication payload; and
d) integrity protects the Key Exchange step and the Authentication step using the Authentication payload.

The application client may include the Notify payload to send an initial contact notification to the device server. If sent, the initial contact notification specifies that the application client has no stored state for any SAs with the device server other than the SA that is being created.

In response to receipt of an initial contact notification, the device server should delete all other SAs that were authenticated with a SECURITY PROTOCOL OUT command that contained the same Identification - Application Client payload data as that which is present in the SECURITY PROTOCOL OUT command that the device server is processing.

If the device server deletes other SAs in response to an initial contact notification, it shall do so only after the successful completion of the Authentication step SECURITY PROTOCOL OUT command. If an error occurs during the Authentication SECURITY PROTOCOL OUT command, the device server shall ignore the initial contact notification.

If the device server is unable to proceed with SA creation for any reason (e.g., the verification of the Authentication payload fails), the SECURITY PROTOCOL OUT command shall be terminated with CHECK CONDITION status, with the sense key set to ABORTED COMMAND, and the additional sense code set to an appropriate value. The additional sense code AUTHENTICATION FAILED shall be used when verification of the Authentication payload fails, or when authentication fails for any other reason.

5.16.4.7.3 Authentication step SECURITY PROTOCOL IN command

If the Authentication step SECURITY PROTOCOL OUT command (see 5.16.4.7.2) completes with GOOD status, the application client sends a SECURITY PROTOCOL IN command (see 6.30) with the SECURITY PROTOCOL field set to IKEv2-SCSI (i.e., 41h) and the SECURITY PROTOCOL SPECIFIC field set to 0103h to obtain the device server's authentication message. The parameter data consists of the IKEv2-SCSI header (see 7.7.3.4) and an Encrypted payload (see 7.7.3.5.11) that:

a) is integrity checked using the following:
   A) using separate algorithms as follows:
      a) integrity checked using the following:
         A) the algorithm specified by the INTEG IKEv2-SCSI algorithm descriptor (see 7.7.3.6.4) in the IKEv2-SCSI SA Cryptographic Algorithms payload (see 7.7.3.5.13) from the Key Exchange step (see 5.16.4.6); and
         B) the SK_ar shared key (see 5.16.4.4);

         and
      b) encrypted using the following:
         A) the algorithm specified by the ENCR IKEv2-SCSI algorithm descriptor (see 7.7.3.6.2) in the IKEv2-SCSI SA Cryptographic Algorithms payload from the Key Exchange step; and
         B) the SK_er shared key (see 5.16.4.4);

      or
   B) using a combined integrity check and encryption algorithm that uses the following (i.e., if the INTEG IKEv2-SCSI algorithm descriptor indicates AUTH_COMBINED):

a) the algorithm specified by the ENCR IKEv2-SCSI algorithm descriptor in the IKEv2-SCSI SA Cryptographic Algorithms payload from the Key Exchange step; and
b) the SK_er shared key with additional salt bytes as described in 5.16.4.8.1 and table 69 (see 5.16.4.4);

and

b) contains the following:
1) an Identification – Device Server payload (see 7.7.3.5.4);
2) an IKEv2-SCSI SAUT Cryptographic Algorithms payload (see 7.7.3.5.14);
3) zero or more Certificate payloads (see 5.16.4.3.3.4); and
4) an Authentication payload (see 7.7.3.5.7).

In the SECURITY PROTOCOL IN parameter data, the device server:

a) confirms information about the SA usage and cryptographic algorithms;
b) sends its identity in the Identification – Device Server payload;
c) authenticates its identity; and
d) protects the integrity of the prior step messages using the Authentication payload.

Before completing the SECURITY PROTOCOL IN command with GOOD status, the device server shall generate the SA as described in 5.16.4.9.

The application client should verify the Authentication payload as described in 7.7.3.5.7. The Certificate payload(s) are used as part of this verification for PKI-based authentication. If the Authentication payload is verified and no other error occurs the application client should generate the SA as described in 5.16.4.9.

If the application client is unable to proceed with SA creation for any reason (e.g., the verification of the Authentication payload fails), the application client should:

a) not use the SA for any additional activities; and
b) notify the device server that the IKEv2-SCSI CCS is being abandoned as described in 5.16.4.10.

The application client should compare the fields in the IKEv2-SCSI SAUT Cryptographic Algorithms payload to the values sent in the Authentication step SECURITY PROTOCOL OUT command (see 5.16.4.7.2). If the application client detects differences in the contents of the payloads, the application client should abandon the IKEv2-SCSI CCS and notify the device server that the IKEv2-SCSI CCS is being abandoned as described in 5.16.4.10.

5.16.4.8 Generating shared keys

5.16.4.8.1 Overview

If the Authentication step is skipped (see 5.16.4.1), then shared key generation is performed as described in 5.16.4.8.2 and is summarized as follows:

a) the shared keys for SA management (e.g., SA creation and deletion) are generated at the same time as the shared keys used by the created SA; and
b) all the shared keys are generated during completion of the Key Exchange step (see 5.16.4.6.4).

If the Authentication step is processed (i.e., not skipped), then shared key generation is performed as described in 5.16.4.8.3 and is summarized as follows:

a) the shared keys for SA management (e.g., SA creation and deletion) are generated during the completion of the Key Exchange step (see 5.16.4.6.4); and
b) the shared keys used by the created SA are generated during SA generation (see 5.16.4.9).

Regardless of when the SA management shared keys (e.g., used for SA creation and deletion) and shared keys (see 5.16.4.4) used by the created SA are generated, the organization of the shared keys depends on the type of encryption and integrity checking algorithm being used as follows:

a) if an encryption algorithm that requires separate integrity checking is used, then separate shared keys are generated for each algorithm; or
b) if an encryption algorithm that includes integrity checking is used (i.e., if the ALGORITHM IDENTIFIER field in the INTEG IKEv2-SCSI cryptographic algorithm descriptor (see 7.7.3.6.4) contains AUTH_COMBINED), then no shared keys are generated for the integrity checking algorithm but additional key material is generated to act as a salt (see table 488 in 7.7.3.6.2) for the combined mode encryption algorithm.

5.16.4.8.2 Generating shared keys when the Authentication step is skipped

If the Authentication step is skipped (see 5.16.4.1), then the shared keys for SA management are generated at the same time as the shared keys for use by the created SA.

As part of completing the Key Exchange step (see 5.16.4.6.4), the SA participants generate all necessary shared keys as follows:

1) generate SKEYSEED (see RFC 4306) as described in 5.16.4.8.4;
2) generate the shared keys used for SA management as described in 5.16.4.8.5;
3) as part of generating the SA (see 5.16.4.9) (i.e., as part of completing the Key Exchange step as described in 5.16.4.6.4), generate the shared keys for use by the created SA as described in 5.16.4.8.6 and store them in the KEYMAT SA parameter (see 3.1.133).

5.16.4.8.3 Generating shared keys when the Authentication step is processed

If the Authentication step is not skipped (see 5.16.4.1), then:

1) the shared keys for SA management are generated during completion of the Key Exchange step (see 5.16.4.6.4) as follows;
   1) generate SKEYSEED (see RFC 4306) as described in 5.16.4.8.4; and
   2) generate the shared keys used for SA management as described in 5.16.4.8.5;

   and
2) the shared keys for use by the created SA are generated during SA generation (see 5.16.4.9), near the end of processing for the Authentication step (see 5.16.4.7) as described in 5.16.4.8.6 and store them in the KEYMAT SA parameter (see 3.1.133).

5.16.4.8.4 Initializing shared key generation

5.16.4.8.4.1 Initializing for SA creation shared key generation

The SA parameters (see 3.1.133) are initialized for the KDF function used to generate SA creation shared keys as follows:

1) generate the input to the PRF function by performing the last steps of the key exchange algorithm selected by the ALGORITHM IDENTIFIER field in the D-H IKEv2-SCSI algorithm descriptor (see 7.7.3.6.5) in the IKEv2-SCSI SA Cryptographic Algorithms payload (see 7.7.3.5.13) using at least the contents of one of the following fields as inputs to those last steps:
   A) the KEY EXCHANGE DATA field in the Key Exchange payload (see 7.7.3.5.3) in the Key Exchange SECURITY PROTOCOL OUT command (see 5.16.4.6.2) parameter data; and
   B) the KEY EXCHANGE DATA field in the Key Exchange payload in the Key Exchange SECURITY PROTOCOL IN command (see 5.16.4.6.3) parameter data;

2) generate SKEYSEED (see RFC 4306) using the output from step 1) and the PRF selected by the ALGORITHM IDENTIFIER field in the PRF IKEv2-SCSI cryptographic algorithm descriptor (see 7.7.3.6.3) in the IKEv2-SCSI SA Cryptographic Algorithms payload;
3) store the generated SKEYSEED value in the KEY_SEED SA parameter;
4) store the contents of the IKE_SA APPLICATION CLIENT SAI field from the IKEv2-SCSI Header (see 7.7.3.4) from the Key Exchange step SECURITY PROTOCOL IN command (see 5.16.4.6.3) in the AC_SAI SA parameter;
5) store the contents of the IKE_SA DEVICE SERVER SAI field from the IKEv2-SCSI Header (see 7.7.3.4) from the Key Exchange step SECURITY PROTOCOL IN command (see 5.16.4.6.3) in the DS_SAI SA parameter;
6) store the contents of the NONCE DATA field from the Nonce payload (see 7.7.3.5.8) from the parameter data for the Key Exchange step SECURITY PROTOCOL OUT command (see 5.16.4.6.2) in the AC_NONCE SA parameter; and
7) store the contents of the NONCE DATA field from the Nonce payload (see 7.7.3.5.8) from the parameter data for the Key Exchange step SECURITY PROTOCOL IN command (see 5.16.4.6.3) in the DS_NONCE SA parameter.

5.16.4.8.4.2 Initializing for generation of shared keys used by the created SA

The SA parameters (see 3.1.133) are initialized for the KDF function used to generate the shared keys that will be used by the created SA as follows:

1) store the SK_d value that was generated along with the other shared keys used in SA creation (see 5.16.4.8.5) in the KEY_SEED SA parameter;
2) store the contents of the IKE_SA APPLICATION CLIENT SAI field from the IKEv2-SCSI Header (see 7.7.3.4) from the Key Exchange step SECURITY PROTOCOL IN command (see 5.16.4.6.3) in the AC_SAI SA parameter;
3) store the contents of the IKE_SA DEVICE SERVER SAI field from the IKEv2-SCSI Header (see 7.7.3.4) from the Key Exchange step SECURITY PROTOCOL IN command (see 5.16.4.6.3) in the DS_SAI SA parameter;
4) store the contents of the NONCE DATA field from the Nonce payload (see 7.7.3.5.8) from the parameter data for the Key Exchange step SECURITY PROTOCOL OUT command (see 5.16.4.6.2) in the AC_NONCE SA parameter; and
5) store the contents of the NONCE DATA field from the Nonce payload (see 7.7.3.5.8) from the parameter data for the Key Exchange step SECURITY PROTOCOL IN command (see 5.16.4.6.3) in the DS_NONCE SA parameter.

5.16.4.8.5 Generating shared keys used for SA management

The shared keys used for SA management (e.g., SA creation and deletion) are generated using the SKEYSEED generated during initialization (see 5.16.4.8.4) and the steps described in this subclause.

Which shared keys are generated for SA management depends on:

a) whether the encryption algorithm includes integrity checking as indicated by the contents of the ALGORITHM IDENTIFIER field in the INTEG IKEv2-SCSI cryptographic algorithm descriptor (see 7.7.3.6.4) of the IKEv2-SCSI SA Cryptographic Algorithms payload (see 7.7.3.5.13), and
b) whether the Authentication step is skipped (see 5.16.4.1).

Using the contents of the initialized SA parameters (see 5.16.4.8.4.1), the INTEG ALGORITHM IDENTIFIER field, and the KDF selected by the ALGORITHM IDENTIFIER field in the PRF IKEv2-SCSI cryptographic algorithm descriptor in

the IKEv2-SCSI SA Cryptographic Algorithms payload the following shared keys (see 5.16.4.4) are generated in the order shown:

a) if the INTEG ALGORITHM IDENTIFIER field does not contain AUTH_COMBINED, then generate the following shared keys (see 5.16.4.4):
   A) if the Authentication step is not skipped, generate the following shared keys:
      1) SK_d;
      2) SK_ai;
      3) SK_ar;
      4) SK_ei;
      5) SK_er;
      6) SK_pi; and
      7) SK_pr;
      or
   B) if the Authentication step is skipped, then generate the following shared keys:
      1) SK_d;
      2) SK_ai;
      3) SK_ar;
      4) SK_ei; and
      5) SK_er.

   or

b) if the INTEG ALGORITHM IDENTIFIER field contains AUTH_COMBINED, then generate the following shared keys:
   A) if the Authentication step is not skipped, generate the following shared keys:
      1) SK_d;
      2) SK_ei with additional salt bytes as described in 5.16.4.8.1 and table 69 (see 5.16.4.4);
      3) SK_er with additional salt bytes as described in 5.16.4.8.1 and table 69 (see 5.16.4.4);
      4) SK_pi; and
      5) SK_pr;
      or
   B) if the Authentication step is skipped, then generate the following shared keys:
      1) SK_d;
      2) SK_ei with additional salt bytes as described in 5.16.4.8.1 and table 69 (see 5.16.4.4); and
      3) SK_er with additional salt bytes as described in 5.16.4.8.1 and table 69 (see 5.16.4.4).

The shared keys thus generated are combined with other data and stored in the MGMT_DATA SA parameter as described in 5.16.4.9.

How to determine the sizes of the shared keys to be generated is summarized in table 69 (see 5.16.4.4).

5.16.4.8.6 Generating shared keys for use by the created SA

As part of completing the Authentication step and generating the SA (see 5.16.4.9), the SA participants initialize the SA parameters for performing a KDF (see 5.16.4.8.4.2), and use the KDF selected by the ALGORITHM IDENTIFIER field in the PRF IKEv2-SCSI cryptographic algorithm descriptor in the Key Exchange step IKEv2-SCSI SA Cryptographic Algorithms payload (see 7.7.3.5.13) to generate the following shared keys (see 5.16.4.4) in the order shown and store them in the KEYMAT SA parameter:

a) if the ALGORITHM IDENTIFIER field in the ENCR IKEv2-SCSI cryptographic algorithm descriptor (see 7.7.3.6.2) contains ENCR_NULL in the IKEv2-SCSI SAUT Cryptographic Algorithms payload (see 7.7.3.5.14), then generate the following shared keys:
   1) SK_ai;
   2) SK_ar;
   and

b) if the ALGORITHM IDENTIFIER field in the ENCR IKEv2-SCSI cryptographic algorithm descriptor does not contain ENCR_NULL in the IKEv2-SCSI SAUT Cryptographic Algorithms payload, then generate the following shared keys:
   A) if the ALGORITHM IDENTIFIER field in the INTEG IKEv2-SCSI cryptographic algorithm descriptor (see 7.7.3.6.4) does not contain AUTH_COMBINED in the IKEv2-SCSI SAUT Cryptographic Algorithms payload (see 7.7.3.5.14), then generate the following shared keys:
      1) SK_ai;
      2) SK_ar;
      3) SK_ei; and
      4) SK_er;

   or

   B) if the ALGORITHM IDENTIFIER field in the INTEG IKEv2-SCSI cryptographic algorithm descriptor contains AUTH_COMBINED in the IKEv2-SCSI SAUT Cryptographic Algorithms payload, then generate the following shared keys:
      1) SK_ei with additional salt bytes as described in 5.16.4.8.1 and table 69 (see 5.16.4.4); and
      2) SK_er with additional salt bytes as described in 5.16.4.8.1 and table 69 (see 5.16.4.4).

How to determine the sizes of the shared keys to be generated is summarized in table 69 (see 5.16.4.4).

5.16.4.9 IKEv2-SCSI SA generation

Depending on whether or not the Authentication step was skipped (see 5.16.4.1), the SA participants shall generate shared keys as described in 5.16.4.8.

The SA participants shall initialize the SA parameters (see 3.1.133) as follows:

1) KEYMAT shall be set as follows:
   A) if the Authentication step is skipped, KEYMAT shall be set as described in 5.16.4.8.2; or
   B) if the Authentication step is processed, KEYMAT shall be set as described in 5.16.4.8.3;

   and

2) the other SA parameters shall be set as follows:
   A) AC_SAI shall be set to the value in the IKE_SA APPLICATION CLIENT SAI field in the IKEv2-SCSI header (see 7.7.3.4) in the Key Exchange step SECURITY PROTOCOL OUT command (see 5.16.4.6.2);
   B) DS_SAI shall be set to the value in the IKE_SA DEVICE SERVER SAI field in the IKEv2-SCSI header in the Key Exchange step SECURITY PROTOCOL IN command (see 5.16.4.6.3);
   C) TIMEOUT shall be set to the IKEV2-SCSI SA INACTIVITY TIMEOUT field in the IKEv2-SCSI Timeout Values payload (see 7.7.3.5.15) in the Key Exchange step SECURITY PROTOCOL OUT command;
   D) KDF_ID shall be set to the value in the ALGORITHM IDENTIFIER field in the PRF IKEv2-SCSI cryptographic algorithm descriptor (see 7.7.3.6.3) in the IKEv2-SCSI SA Cryptographic Algorithms payload (see 7.7.3.5.13);
   E) AC_SQN shall be set to one;
   F) DS_SQN shall be set to one;
   G) AC_NONCE shall be set to zero;
   H) DS_NONCE shall be set to zero;
   I) KEY_SEED shall be set to zero;
   J) USAGE_TYPE shall be set to the value in the SA TYPE field in the IKEv2-SCSI SAUT Cryptographic Algorithms payload (see 7.7.3.5.14) in the Key Exchange step SECURITY PROTOCOL OUT command;
   K) USAGE_DATA shall contain at least the following values from the IKEv2-SCSI SAUT Cryptographic Algorithms payload (see 7.7.3.5.14) in either the Key Exchange step SECURITY PROTOCOL OUT command or the Authentication step SECURITY PROTOCOL OUT command:
      a) the ALGORITHM IDENTIFIER field and KEY LENGTH field from the ENCR IKEv2-SCSI cryptographic algorithm descriptor (see 7.7.3.6.2);

b) the ALGORITHM IDENTIFIER field from the INTEG IKEv2-SCSI cryptographic algorithm descriptor (see 7.7.3.6.4); and
c) the contents, if any, of the USAGE DATA field;
and

L) MGMT_DATA shall contain at least the following values:
a) from the IKEv2-SCSI SA Cryptographic Algorithms payload (see 7.7.3.5.13) in the Key Exchange step SECURITY PROTOCOL OUT command:
A) the ALGORITHM IDENTIFIER field and KEY LENGTH field from the ENCR IKEv2-SCSI cryptographic algorithm descriptor (see 7.7.3.6.2); and
B) the ALGORITHM IDENTIFIER field from the INTEG IKEv2-SCSI cryptographic algorithm descriptor (see 7.7.3.6.4);
b) from the shared keys generated for SA management (see 5.16.4.8.5), the following shared keys and salt bytes:
A) the value of SK_ai, if any;
B) the value of SK_ar, if any;
C) the value of SK_ei, with additional salt bytes, if any; and
D) the value of SK_er, with additional salt bytes, if any;
and
c) the next value of the MESSAGE ID field in the IKEv2-SCSI header.

NOTE 21 - The inclusion of the algorithm identifiers and key length in MGMT_DATA SA parameter enables the SA to apply the same encryption and integrity algorithms that IKEv2-SCSI negotiated to future IKEv2-SCSI SECURITY PROTOCOL OUT commands and IKEv2-SCSI SECURITY PROTOCOL IN commands, if any.

5.16.4.10 Abandoning an IKEv2-SCSI CCS

The occurrence of errors in either the application client or the device server may require that an IKEv2-SCSI CCS (see 3.1.71) be abandoned.

A device server shall indicate that it has abandoned an IKEv2-SCSI CCS, if any, by terminating an IKEv2-SCSI CCS command (see 5.16.4.1) received on the I_T_L nexus for which the IKEv2-SCSI CCS state is being maintained with any combination of status and sense data other than those shown in table 71.

Table 71 — IKEv2-SCSI command terminations that do not abandon the CCS, if any

IKEv2-SCSI CCS Command	Status (Sense Key)	Additional Sense Code	Description
SECURITY PROTOCOL OUT SECURITY PROTOCOL IN	GOOD (n/a)	n/a	Indicates IKEv2-SCSI CCS is progressing as described in this standard
Key Exchange step SECURITY PROTOCOL OUT (see 5.16.4.6.2)	CHECK CONDITION (ABORTED COMMAND)	CONFLICTING SA CREATION REQUEST	At least one IKEv2-SCSI CCS is already active, and attempts to start another are blocked until the first CCS completes
SECURITY PROTOCOL OUT SECURITY PROTOCOL IN	CHECK CONDITION (NOT READY)	LOGICAL UNIT NOT READY, SA CREATION IN PROGRESS	Device server is busy processing another command in the IKEv2-SCSI CCS associated with this I_T_L nexus or a different IKEv2-SCSI CCS associated with a different I_T_L nexus
SECURITY PROTOCOL IN	CHECK CONDITION (ILLEGAL REQUEST)	INVALID FIELD IN CDB	Incorrect SECURITY PROTOCOL IN CDB format
Authentication step SECURITY PROTOCOL OUT (see 5.16.4.7.2)		UNABLE TO DECRYPT PARAMETER LIST	Device server is unable to decrypt the Encrypted payload (see 7.7.3.5.11) or the integrity check fails
SECURITY PROTOCOL OUT		SA CREATION PARAMETER VALUE REJECTED	To adapt to possible denial of service attacks, a condition for which the optimal response includes an additional sense code of SA CREATION PARAMETER VALUE INVALID and the abandonment of the CCS is not causing the CCS to be abandoned

As part of abandoning an IKEv2-SCSI CCS, the device server shall:

a) discard all maintained state (see 5.16.4.1); and
b) prepare to allow a future Key Exchange step SECURITY PROTOCOL OUT command received on any I_T_L nexus to start a new IKEv2-SCSI CCS.

After a device server abandons an IKEv2-SCSI CCS, the device server shall respond to all new IKEv2-SCSI protocol commands as if an IKEv2-SCSI CCS had never been started.

An application client should not abandon an IKEv2-SCSI CCS when the next command in the CCS is a SECURITY PROTOCOL IN command. Instead, the application client should send the appropriate SECURITY PROTOCOL IN command and then abandon the IKEv2-SCSI CCS.

An application client should specify that it has abandoned an IKEv2-SCSI CCS by sending an IKEv2-SCSI Delete operation (see 5.16.4.11) with application client SAI and device server SAI information that matches that of the IKEv2-SCSI CCS being abandoned.

5.16.4.11 Deleting an IKEv2-SCSI SA

When an SA is deleted, both sets of SA parameters (see 5.16.2.2) are deleted as follows:

1) the application client uses the information in its SA parameters to prepare an IKEv2-SCSI Delete operation that requests deletion of the device server's SA parameters;
2) the application client deletes its SA parameters and any associated data;
3) the application client sends the IKEv2-SCSI Delete operation prepared in step 1) to the device server;
4) in response to the IKEv2-SCSI Delete operation, the device server deletes its SA parameters and any associated data.

The IKEv2-SCSI Delete operation is a SECURITY PROTOCOL OUT command with the SECURITY PROTOCOL field set to IKEv2-SCSI (i.e., 41h) and the SECURITY PROTOCOL SPECIFIC field set to 0104h. The parameter data consists of the IKEv2-SCSI header (see 7.7.3.4) and an Encrypted payload (see 7.7.3.5.11) that contains the one Delete payload (see 7.7.3.5.10).

The Delete payload should conform to the requirements described in 7.7.3.5.10.

The device server shall process a valid Delete operation SECURITY PROTOCOL OUT command regardless of whether or not IKEv2-SCSI CCS state is being maintained for the I_T_L nexus on which the command is received.

The application client SAI and device server SAI information in a valid Delete operation may or may not identify an IKEv2-SCSI CCS for which state is being maintained for the I_T_L nexus on which the command is received (see 7.7.3.5.11).

5.16.5 Security progress indication

The cryptographic calculations required by some security protocols are capable of consuming significant amounts of time in the device server. While cryptographic security calculations are in progress, the device server shall provide pollable REQUEST SENSE data (see 5.6.2) with:

a) the sense key set to NOT READY;
b) the additional sense code set to LOGICAL UNIT NOT READY, SA CREATION IN PROGRESS; and
c) the PROGRESS INDICATION field set to indicate the progress of the device server in performing the necessary cryptographic calculations.

The device server shall not use the progress indication to report the detailed progress of cryptographic computations that take a variable amount of time based on their inputs. The device server may use the progress indication to report synthetic progress that does not reveal the detailed progress of the computation (e.g., divide a constant expected time for the computation by 10 and advance the progress indication in 10% increments based on the elapsed time as compared to the expected time).

The requirements in this subclause apply to implementations of Diffie-Hellman computations and operations involving any keys (e.g., RSA) that optimize operations on large numbers based on the values of inputs (e.g., a computational step may be skipped when a bit or set of bits in an input is zero). A progress indication that

advances based on the computation structure (e.g., count of computational steps) may reveal the time taken by content-dependent portions of the computation, and reveal information about the inputs.

5.16.6 Command security

5.16.6.1 Overview

SCSI command security defines techniques for protecting against inadvertent or malicious misuse of SCSI commands to gain unauthorized access to logical units.

The following classes are used to specify SCSI command security:

a) Secure CDB Originator class;
b) Security Manager class;
c) Enforcement Manager class; and
d) Secure CDB Processor class.

The relationship between those classes varies depending on the implemented security technique.

5.16.6.2 Secure CDB Originator class

The Secure CDB Originator class is a kind of application client that originates SCSI commands to which it has attached a security CDB extension (see 4.3.4) that allows an enforcement manager to determine if the SCSI command may be processed by the addressed logical unit.

The secure CDB originator interacts with the security manager to determine:

a) the types of the SCSI commands it is allowed to send to the Secure CDB processor; and
b) the content of the security CDB extension to be attached to the SCSI commands.

5.16.6.3 Secure CDB Processor class

The Secure CDB Processor class is a kind of device server that processes SCSI commands that have an attached security extension, if an enforcement manager allows that type of SCSI command from the originating application client to be processed.

The secure CDB processor determines if a SCSI command is allowed to be processed by communicating the following information to the enforcement manager:

a) the CDB of the SCSI command to be processed; and
b) the security CDB extension, if any, attached to the SCSI command to be processed.

The secure CDB processor shall always allow the processing of the following commands when they do not have an attached security CDB extension:

a) INQUIRY;
b) REPORT LUNS;
c) REPORT SUPPORTED OPERATION CODES;
d) REPORT SUPPORTED TASK MANAGEMENT FUNCTIONS;
e) REPORT TARGET PORT GROUPS; and
f) REQUEST SENSE.

5.16.6.4 Enforcement Manager class

The Enforcement Manager class is either contained within a:

a) device server (i.e., has the same LUN as the secure CDB processor); or
b) target device (e.g., has a W_LUN, or vendor specific presence in the SCSI target device).

The enforcement manager determines if the secure CDB processor is allowed to, or prohibited from, processing a SCSI command using security information received from the security manager.

If a secure CDB originator has not been authenticated, the enforcement manager shall not make determinations about whether a command from that secure CDB originator is permitted or prohibited.

5.16.6.5 Security Manager class

The Security Manager class contains communicates with the Secure CDB Originator class (see 5.16.6.2) and the Enforcement Manager class (see 5.16.6.4) as shown in table 72.

Table 72 — Security Manager class relationships

Security Manager location	Communications mechanism for …	
	Secure CDB Originator	Enforcement Manager
An application client located in the same SCSI device as the secure CDB originator	Outside the scope of this standard	Via the SCSI domain's service delivery sub-system (see SAM-4)
A device server located in the same SCSI device as the secure CDB processor	Via the SCSI domain's service delivery sub-system (see SAM-4)	Outside the scope of this standard
A SCSI device contained within the same SCSI domain as the secure CDB originator and the secure CDB processor [a]	Via the SCSI domain's service delivery sub-system (see SAM-4)	Via the SCSI domain's service delivery sub-system (see SAM-4)
Not a SCSI device, device server, or application client	Outside the scope of this standard	Outside the scope of this standard

[a] This SCSI device is required to contain an application client and a device server (i.e., to contain both a SCSI Initiator Port class (see SAM-4) and a SCSI Target Port class (see SAM-4)).

The security manager:

a) maintains SCSI command security information for the SCSI domain (e.g., authorization and authentication information);
b) delivers to the enforcement manager the security information required by the enforcement manager to determine if the secure CDB processor is allowed to, or prohibited from, processing a SCSI command; and
c) responds to requests from authenticated secure CDB originators to send SCSI commands to a secure CDB processor as follows:
   A) if the secure CDB originator sends its authentication and an authorization request, then the security manager responds with the authorization information necessary for the secure CDB originator to generate security information to be attached to any authorized CDB that is sent to the secure CDB processor; or

B) if the secure CDB originator sends its authentication, an authorization request, and the security information to be attached to CDBs, then the security manager shall only accept the request if the secure CDB originator is authorized to send the requested SCSI commands to the requested secure CDB processor.

5.16.6.6 The relationship between SAs and command security

As defined by this standard, SAs provide the following forms of secure communications for selected portions of selected CDB and command parameter data:

a) cryptographic data integrity provided by Message Authentication Code or Integrity Check Value; and
b) confidentially provided by data encryption.

The SAs defined by this standard do not apply to data communicated in the CDBs sent from the secure CDB originator to the secure CDB processor. The function of securing CDBs is performed by the command security features described in 5.16.6. Command security and SAs features may be used in concert to protect both the CDB data and the parameter data.

Authorization information (see 5.16.6.5) includes associations between:

a) permissions to use certain commands and command options; and
b) secure CDB originator identities.

SAs provide a mechanism for satisfying the secure CDB originator authentication requirements placed on enforcement managers (see 5.16.6.4). Every secure CDB originator that is authenticated using an SA is required to be authenticated using a unique SA.

Some command security techniques (e.g., the CbCS technique described in 5.16.6.8) depend on an established secure channel between a secure CDB originator and secure CDB processor. SAs provide a mechanism for establishing such secure channels. If SAs are used in this manner, multiple secure CDB originators may share a common SA, however, such SAs do not authenticate any specific secure CDB originator.

5.16.6.7 Command security techniques

This standard defines the following techniques for implementing command security:

a) capability-based command security (see 5.16.6.8).

5.16.6.8 Capability-based command security technique

5.16.6.8.1 Overview

CbCS is a credential-based system that manages access to a logical unit by the coordination between shared keys and security parameters set by the CbCS management application client (see 5.16.6.8.4) and credentials generated by the CbCS management device server (see 5.16.6.8.3). The mechanism for coordination between the CbCS management device server and the CbCS management application client is not defined in this standard.

The CbCS protocol enables centralized management of SCSI command security.

CbCS secures access to a logical unit or a volume (see SSC-3) by providing cryptographic integrity of credentials that are added to commands sent to the logical unit (see 3.1.84). This cryptographic integrity is based on mutual trust and key exchanges between the CbCS management device server, CbCS management application client, and the enforcement manager (see 3.1.52).

Different levels of protection and security are achieved by using different CbCS methods. The following CbCS methods are defined by this standard:

a) the BASIC CbCS method (see 5.16.6.8.8.2) provides protection against errors but does not prevent unauthorized access caused by means of malicious attacks (e.g., identity spoofing and network attacks); and
b) the CAPKEY CbCS method (see 5.16.6.8.8.3) enforces application client authentication and provides cryptographic integrity of credentials. It protects against the following types of unauthorized access attacks:
   A) Without cryptographic message integrity in the service delivery subsystem:
      a) illegal use of credentials beyond their original scope and life span;
      b) forging or stealing credentials; and
      c) using malformed credentials;

   and
   B) with cryptographic message integrity in the service delivery subsystem:
      a) network errors and malicious message modifications; and
      b) message replay attacks.

CbCS also supports rapid revocation of credentials, per SCSI target device and per logical unit.

CbCS does not define task management function security.

CbCS (see figure 14) is composed of the following class:

a) security Manager class (see 5.16.6.8.2) that contains the following classes:
   A) CbCS Management Device Server class (see 5.16.6.8.3); and
   B) CbCS Management Application Client class (see 5.16.6.8.4);
b) SCSI Initiator Device class (see SAM-4) that contains the following class:
   A) Secure CDB Originator class (see 5.16.6.8.5);

   and
c) SCSI Target Device class (see SAM-4) that contains the following classes:
   A) Secure CDB Processor class (see 5.16.6.8.6); and
   B) Enforcement Manager class (see 5.16.6.8.7).

Figure 14 shows the flow of transactions between the components of a CbCS capable SCSI domain.

Figure 14 — CbCS overview class diagram

Each instance of CbCS shall contain:

a) one security manager that shall contain:
   A) one CbCS management device server; and
   B) one CbCS management application;
b) one or more SCSI initiator devices that shall contain:
   A) one or more secure CDB originators;
   and
c) one or more SCSI target devices that shall contain:
   A) one secure CDB processor per logical unit; and
   B) the following:
      a) one enforcement manager per secure CDB processor;
      b) one enforcement manager per SCSI target device; or
      c) both.

5.16.6.8.2 Security Manager class

The Security Manager class for the CbCS technique manages access of secure CDB originators to logical units or volumes (see SSC-3). It uses a decision database to obtain the authorization information required for deciding the type and duration of access granted to secure CDB originator to a given logical unit (see 3.1.84) or volume (see SSC-3). It communicates with secure CDB originators to provide them CbCS credentials (see 5.16.6.8.12), and with enforcement managers (see 3.1.52) to exchange shared keys (see 5.16.6.8.11) and send CbCS parameters (see 5.16.6.8.15).

The security manager may be located and may communicate with secure CDB originators and enforcement managers as follows:

a) if it is a SCSI device contained within the same SCSI domain as the secure CDB originator and the enforcement manager, it shall contain an application client and use it to communicate to the enforcement manager, and it shall contain a device server and use it to communicate with secure CDB originators;
b) if it is an application client located in the same device as the secure CDB originators, it shall communicate to the enforcement manager via the SCSI domain's service delivery subsystem, and it may communicate with the secure CDB originators by means outside the scope of this standard; and
c) if it is a device server located in the same device as the secure CDB processor, it shall communicate to the secure CDB originators via the SCSI domain's service delivery subsystem, and it may communicate with the enforcement manager by means outside the scope of this standard.

The security manager's device server is called CbCS management device server (see 5.16.6.8.3). The security manager's application client is called CbCS management application client (see 5.16.6.8.4).

If the security manager is a SCSI device (see 3.1.136), it shall perform CbCS management using the CbCS management application client and the CbCS management device server as follows:

a) the CbCS management device server (see 5.16.6.8.3) provides access policy controls to secure CDB originators (see 3.1.150) using policy-coordinated CbCS capabilities; and
b) the CbCS management application client (see 5.16.6.8.4) prevents unsecured access to a logical unit (see 3.1.84) or a volume (see SSC-3) in concert with:
   A) the CbCS management device server (see 5.16.6.8.3);
   B) the enforcement manager (see 3.1.52); and
   C) the secure CDB processor (see 3.1.151).

CbCS management is confined to the CbCS management application client and CbCS management device server. The communication of CbCS management information may occur in a manner outside the scope of this standard.

5.16.6.8.3 CbCS Management Device Server class

5.16.6.8.3.1 CbCS Management Device Server class overview

The CbCS Management Device Server class returns a CbCS capability and a CbCS capability key (i.e., Capability-Key) with each CbCS credential giving the secure CDB originator access to a specific logical unit, and optionally to a volume (see SSC-3).

This standard defines the RECEIVE CREDENTIAL command (see 6.19) that the secure CDB originator may use to request a CbCS capability and a CbCS capability key from a CbCS management device server.

Commands to and responses from the CbCS management device server are protected by use of an SA whose creation included the Authentication Step (see 6.19).

The CbCS management device server shall set the CBCS bit to zero in any Extended INQUIRY Data VPD page (see 7.8.7) that it returns.

5.16.6.8.3.2 Decision Database attribute

The Decision Database attribute is used to obtain the authorization information required for deciding the type and duration of access granted to a secure CDB originator for a given logical unit (see 3.1.84) or volume (see SSC-3) within a SCSI target device. CbCS Credentials are prepared by the CbCS management device server based on the contents of that Decision Database attribute.

5.16.6.8.4 CbCS Management Application Client class

The CbCS Management Application Client class exchanges shared keys (see 5.16.6.8.11) with and sends CbCS parameters (see 5.16.6.8.15) to the enforcement manager using SECURITY PROTOCOL OUT commands (see 6.31) and SECURITY PROTOCOL IN commands (see 6.30) transferred over the SCSI domain's service delivery subsystem.

The CbCS capability keys are computed by the CbCS management device server using shared keys that are shared between the:

a) enforcement manager;
b) CbCS management application client; and
c) CbCS management device server.

The shared keys are managed by the security manager.

5.16.6.8.5 Secure CDB Originator class

The Secure CDB Originator class requests CbCS capabilities (see 3.1.20) and CbCS capability keys (see 3.1.21) from the CbCS management device server for a specific logical unit (see 3.1.84) or volume (see SSC-3). The secure CDB originator sends the CbCS capability and integrity check value to the logical unit's secure CDB processor as part of a CbCS extended CDB as described in 5.16.6.8.16.

For more information on the Secure CDB Originator class see 5.16.6.2.

5.16.6.8.6 Secure CDB Processor class

The Secure CDB Processor class:

a) receives a CbCS capability descriptor (see 6.19.2.3) from a secure CDB originator in a CbCS extension descriptor (see 5.16.6.8.16);
b) requests the SCSI command be validated by the enforcement manager; and
c) if the Enforcement Manager validates the SCSI command, then the secure CDB processor processes that SCSI command.

The secure CDB processor indicates that CbCS is applied to a logical unit by setting the CBCS bit to one in the Extended INQUIRY Data VPD page (see 7.8.7). If the CbCS bit is set to one, the logical unit shall support the following:

a) Extended CDBs (see 4.3.4);
b) CbCS extension type (see 5.16.6.8.16);
c) SECURITY PROTOCOL IN commands (see 6.30) specifying the CbCS security protocol (see 7.7.4); and
d) SECURITY PROTOCOL OUT commands (see 6.31) specifying the CbCS security protocol (see 7.7.4).

For more information on the Secure CDB Processor class see 5.16.6.3.

5.16.6.8.7 Enforcement Manager class

The Enforcement Manager class:

a) receives shared keys (see 5.16.6.8.11) and CbCS parameters (see 5.16.6.8.15) from the CbCS management application client;
b) authenticates the CbCS capability received from a secure CDB processor with an integrity check value as described in 5.16.6.8.13.2;
c) validates SCSI commands sent by secure CDB originators as described in 5.16.6.8.13.2; and
d) if the CAPKEY CbCS method (see 5.16.6.8.8.3) is supported, supplies one security token (see 5.16.6.8.10) for each active I_T nexus to the secure CDB processor for delivery to the secure CDB originator that is using that I_T nexus.

The enforcement manager may be contained within the secure CDB processor, or within the SCSI target device. If the enforcement manager is contained within the secure CDB processor then, the shared keys and CbCS parameters it uses pertain to that logical unit (see 3.1.84). If the enforcement manager is contained within the SCSI target device then, the shared keys and CbCS parameters it uses pertain to the SCSI target device, and the SECURITY PROTOCOL well-known logical unit (see 8.5) is used for the commands to exchange shared keys and set CbCS parameters.

If a shared key is stored in a well known logical unit then that key is shared between all logical units within the SCSI target device but shall only used by a logical unit if there has been no shared key assigned to that logical unit (i.e., a shared key assigned to a logical unit always overrides any shared key assigned to a well known logical unit).

For more information on the Enforcement Manager class see 5.16.6.4.

5.16.6.8.8 CbCS methods

5.16.6.8.8.1 Overview

The CbCS methods defined by this standard are summarized in table 73.

Table 73 — CbCS methods

CbCS method	Protection provided by the enforcement manager and secure CDB processor	Level of security provided, including I_T nexus security provided by SCSI transport (e.g., FC-SP)	
		Without I_T nexus security	With I_T nexus security
BASIC	Protection against errors provided by verifying that the CbCS capability allows processing, but no validation of the authenticity of the CbCS capability.	No protection against attacks	Same protection as is provided by the SCSI transport on the I_T nexus
CAPKEY	Protection against errors and some attacks by both verifying that the CbCS capability allows processing, and validating the authenticity of the CbCS capability.	CbCS capability authenticity assured, but still subject to network attacks (e.g., replay attacks)	CbCS capability authenticity assured and bound to an I_T nexus; other network attacks (e.g., data privacy) thwarted by SCSI transport security on the I_T nexus

If a secure CDB processor receives a command for a logical unit that has CbCS enabled, the enforcement manager shall validate the command as described in 5.16.6.8.13.2 before any other field in the CDB is validated, including the operation code.

5.16.6.8.8.2 The BASIC CbCS method

The BASIC CbCS method validates that the CbCS capability authorizes the encapsulated command for each CDB. It provides centrally-managed policy-driven command access control mechanism that enforces authorized access based on capabilities.

The BASIC CbCS method does not validate the authenticity of the CbCS capability.

Preparing CbCS credentials for the BASIC CbCS method does not require the knowledge of CbCS shared keys and may be done by the secure CDB originator without coordination with the CbCS management device server. In the CbCS extension descriptor (see 5.16.6.8.16):

a) the CbCS capability descriptor (see 6.19.2.3) CBCS METHOD field is set to BASIC;
b) the following CbCS capability descriptor fields are ignored:
   A) the KEY VERSION field; and
   B) the INTEGRITY CHECK VALUE ALGORITHM field;
   and
c) the INTEGRITY CHECK VALUE field is set to zero.

The BASIC CbCS method controls access between the secure CDB originator and the secure CDB processor without requiring authentication of the secure CDB originator. It is sufficient for the secure CDB processor to request that the enforcement manager verify the CbCS capabilities sent by the secure CDB originator.

5.16.6.8.8.3 The CAPKEY CbCS method

The CAPKEY CbCS method provides centrally-managed policy-driven command access control mechanism that enforces authorized access based on capabilities.

In addition, the CAPKEY CbCS method assures the integrity and authenticity of the CbCS capability transferred with each command.

The CAPKEY CbCS method provides for security of commands delivered to the secure CDB processor. When used in conjunction with a secure service delivery subsystem, it provides additional protection against network attacks (see 5.16.6.8.8.1).

Each capability (see 5.16.6.8.13) is cryptographically associated with a capability key, and the pair is returned to a secure CDB originator (see 5.16.6.8.5) in credential (see 5.16.6.8.12) in response to a RECEIVE CREDENTIAL command (see 6.19).

When the capability is associated with a specific command using a CbCS extension descriptor (see 5.16.6.8.16), an integrity check value is computed using the capability key and a CbCS security token (see 5.16.6.8.10). The enforcement manager (see 5.16.6.8.7) validates this integrity check value as described in 5.16.6.8.13.3.

5.16.6.8.9 CbCS trust assumptions

After the logical unit is trusted (i.e., after a secure CDB originator authenticates that it is communicating with a specific logical unit), the secure CDB originator trusts the secure CDB processor and the enforcement manager to do the following:

a) deny any access attempt from any application client that is not authorized by the security manager; and
b) deny access from any application client that does not perform the CbCS protocols and functions defined by this standard.

The CbCS management device server and the CbCS management application client are trusted after:

a) the CbCS management device server is authenticated by the secure CDB originator; and
b) the CbCS management application client is authenticated by the CbCS Enforcement Manager.

The CbCS management device server and the CbCS management application client are trusted to do the following:

a) securely store long-lived shared keys and capability keys;
b) grant credentials to secure CDB originators according to access control policies that are outside the scope of this standard; and
c) perform the defined security functions.

The service delivery subsystem between the secure CDB originator and the secure CDB processor is not trusted. However, the CbCS security model for CbCS methods other than BASIC is defined so that commands generated by the secure CDB originator are processed by the secure CDB processor only after the secure CDB originator interacts with both the CbCS management device server and the secure CDB processor as defined in this standard.

The communications trust requirements shown in table 74 provide a basis for the CbCS trust assumptions.

Table 74 — CbCS communications trust requirement

For connections between	CbCS cryptographic communications trust requirements	Requirement level
secure CDB originator and secure CDB processor	Message integrity [b]	Optional [a]
secure CDB originator and CbCS management application client	Message confidentiality [c] and integrity [b]	Mandatory
CbCS management application client and enforcement manager	Message integrity [b]	Mandatory
CbCS management application client and CbCS management device server	Message confidentiality [c] and integrity [b]	Mandatory

[a] If this requirement is not met, then the conditions that table 73 (see 5.16.6.8.8) show for an I_T nexus without security apply.
[b] Message integrity algorithms are those that table 79 (see 5.13.8) describes as integrity checking (i.e., AUTH) algorithms.
[c] Message confidentiality algorithms are those that table 79 (see 5.13.8) describes as encryption algorithms.

5.16.6.8.10 CbCS security tokens

A CbCS security token is a random nonce (see 3.1.122) that is at least eight bytes in length that is chosen by the enforcement manager (see 5.16.6.8.7) and returned to a secure CDB originator (see 5.16.6.8.5) by the secure CDB processor (see 5.16.6.8.6) in response to a SECURITY PROTOCOL IN command (see 6.30) with the SECURITY PROTOCOL field set to 07h (i.e., CbCS) and the SECURITY PROTOCOL SPECIFIC field set to 3Fh (i.e., the Security Token CbCS page) as described in 7.7.4.3.4.

The security token shall be unique to each instance of an I_T nexus known to the secure CDB processor and enforcement manager. Security tokens shall be reset and maintained as described in this subclause.

Each security token shall contain at least as many bytes as the largest cipher block size for all the integrity checking algorithms supported by the SCSI target device (see 7.7.4.3.3).

If a hard reset, logical unit reset, or I_T nexus loss is detected by the secure CDB processor, the enforcement manager shall be notified of the event once for each affected I_T nexus. In response to such a notification the enforcement manager shall discard the security token, if any, associated with the affected I_T nexus.

After a power on, the enforcement manager shall discard all security tokens, if any, that it had been maintaining before the power on.

In response to a request for a security token for a given I_T nexus from the secure CDB processor, the enforcement manager shall do one of the following:

a) return the security token value, if any, that is being maintained for the specified I_T nexus to the secure CDB processor; or
b) if no security token is being maintained for the specified I_T nexus, then:
   1) a security token shall be prepared;
   2) the security token shall be returned to the secure CDB processor; and

3) the security token shall be maintained in association with the specified I_T nexus until one of the events described in this subclause causes it to be discarded.

5.16.6.8.11 CbCS shared keys

5.16.6.8.11.1 Overview

Cryptographic integrity checking for CbCS capabilities depends on the following hierarchy of shared keys (see 3.1.162) that is specific to the CbCS model:

1) a master key that is composed of:
   A) an authentication key; and
   B) a generation key;
   and
2) up to 16 working keys whose values are generated from the generation key component of the master key.

Each CbCS shared key set shall support:

a) one master key; and
b) at least two working keys.

Each CbCS shared key in a set has an associated identifier (see 5.16.6.8.11.2) that describes the CbCS shared key to the objects that are managing it without revealing the CbCS shared key's value.

Coordinated sets of CbCS shared keys that conform to this hierarchy are maintained by:

a) the CbCS management application client (see 5.16.6.8.4);
b) the CbCS management device server (see 5.16.6.8.3); and
c) the enforcement manager (see 5.16.6.8.7).

The mechanism for coordinating CbCS shared key sets between the CbCS management application client and the CbCS management device server is outside the scope of this standard.

The following security protocols are provided by this standard for coordinating CbCS shared key sets between the CbCS management application client and the enforcement manager:

a) a Diffie-Hellman key exchange protocol for changing all the components of the master key (see 5.16.6.8.11.4); and
b) mechanisms that invalidate a working key or set a working key based on the generation key component of the current master key (see 5.16.6.8.11.5).

These key management protocols associate a key identifier with each shared key (i.e., master key or working key). The CbCS management application client may use these key identifiers to describe the shared key in some way (e.g., when the shared key was last refreshed or how the intended use of the shared key). Key identifiers shall not be used to contain shared key values.

Within any SCSI target device that contains an enforcement manager, sets of CbCS shared keys are maintained as follows:

a) a separate set of per-logical unit CbCS shared keys for a logical unit that has CbCS enabled;
b) one target-wide set of CbCS shared keys that are accessible to all logical units that have CbCS enabled; or
c) both a target-wide set of CbCS shared keys and per-logical unit sets of CbCS shared keys.

The enforcement manager in any logical unit that has CbCS enabled shall have access to at least one set of CbCS shared keys.

If an enforcement manager has access to both a target-wide set of CbCS shared keys and a per-logical unit set of CbCS shared keys, then a working key defined in the per-logical unit set of CbCS shared keys, if any, shall be used instead of the equivalent working key from the target-wide set of CbCS shared keys.

All CbCS shared keys should be retired from active use (i.e., set, changed, or discarded) often enough to thwart key attacks. How often to retire CbCS shared keys from active use is outside the scope of this standard.

The shared keys in a CbCS shared key set and the CbCS pages used to manage them between the CbCS management application client and the enforcement manager are summarized in table 75.

Table 75 — Summary of CbCS shared keys

CbCS shared key	Applicable CbCS page	Data direction	Capability protection	Reference
Master Authentication, and Generation	Current CbCS Parameters [a]	In	Working key	7.7.4.3.5
	Set Master Key – Seed Exchange Set Master Key – Seed Exchange Set Master Key – Change Master Key [b]	Out In Out	Authentication key	7.7.4.5.5 7.7.4.3.6 7.7.4.5.6
Working key	Current CbCS Parameters [a]	In	Working key	7.7.4.3.5
	Invalidate Key Set Key	Out	Authentication key	7.7.4.5.3 7.7.4.5.4

[a] Only key identifiers are returned, not shared key values.
[b] The new authentication key computed during the seed exchange is the authentication key used for this CbCS page.

5.16.6.8.11.2 CbCS shared key identifiers

CbCS shared key identifiers (see table 76) are set by the CbCS pages (see table 75 in 5.16.6.8.11.1) that change the values of a CbCS shared key and reported by the Current CbCS Parameters CbCS page (see 7.7.4.3.5).

Table 76 — CbCS shared key identifier values

Value	Description
0000 0000 0000 0000h [a]	The associated CbCS shared key has not been modified since the SCSI target device was manufactured
0000 0000 0000 0001h to FFFF FFFF FFFF FFFDh	The associated CbCS shared key has a valid value that has been set by the applicable CbCS page
FFFF FFFF FFFF FFFEh [a]	The associated CbCS shared key does not have a valid value
FFFF FFFF FFFF FFFFh [a]	The associated CbCS shared key is not supported

[a] The use of this value in the CbCS pages that change CbCS shared key values is reserved.

5.16.6.8.11.3 Specifying which CbCS shared key to change

The logical unit to which a SECURITY PROTOCOL IN command (see 6.30) or a SECURITY PROTOCOL OUT command (see 6.31) that specifies one of the CbCS pages shown in table 75 (see 5.16.6.8.11.1) is addressed determines which CbCS shared key is modified as follows:

a) if the command is addressed to the SECURITY PROTOCOL well known logical unit (see 8.5), then the CbCS the target-wide (see 5.16.6.8.11.1) set of CbCS shared keys is modified; or
b) if the command is addressed to a logical unit that is not a well known logical unit, then the per-logical unit (see 5.16.6.8.11.1) set of CbCS shared keys for the specified logical unit are modified.

5.16.6.8.11.4 Updating a CbCS master key

Both components of the CbCS master key (i.e., the authentication key and the generation key) are changed using a single Diffie-Hellman exchange CCS (see 3.1.37) as summarized in this subclause. Use of the Diffie-Hellman exchange ensures forward secrecy of the master key.

The CbCS master key update CCS is composed of the following commands:

1) a SECURITY PROTOCOL OUT command processing the Set Master Key – Seed Exchange CbCS page (see 7.7.4.5.5);
2) a SECURITY PROTOCOL IN command processing the Set Master Key – Seed Exchange CbCS page (see 7.7.4.3.6); and
3) a SECURITY PROTOCOL OUT command processing the Set Master Key – Change Master Key CbCS page (see 7.7.4.5.6).

NOTE 22 - The capability key used to access the two Set Master Key – Seed Exchange CbCS pages is different from the capability key used to access the Set Master Key – Change Master Key CbCS page (see 5.16.6.8.12.3).

The device server shall maintain CCS state for only one CbCS master key update CCS for all I_T nexuses. The command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to COMMAND SEQUENCE ERROR, if any of the following conditions occur:

a) a command attempts to start a new CbCS master key update CCS while state is being maintained for another; or
b) a sequence of CbCS master key update CCS commands other than the one shown in this subclause is attempted.

The device server shall discard the CbCS master key update CCS state if any of the following occur:

a) a command in the CCS does not complete with GOOD status; or
b) the entire CbCS master key update CCS command sequence shown in this subclause is not completed within ten seconds of the successful completion of processing for the SECURITY PROTOCOL OUT command for the Set Master Key – Seed Exchange CbCS page.

5.16.6.8.11.5 Changing a CbCS working keys

A working key is invalidated using the Invalidate Key CbCS page (see 7.7.4.5.3) and set to a new value using the Set Key CbCS page (see 7.7.4.5.4).

New working keys are computed based on the applicable master key and a random number seed.

Working keys are tracked using CbCS shared key identifiers (see 5.16.6.8.11.2).

5.16.6.8.12 CbCS credentials

5.16.6.8.12.1 Overview

Each CbCS credential authorizes access to:

a) a logical unit (see 3.1.84); or
b) a specific volume (see SSC-3) mounted in a specific logical unit.

The applicability of CbCS credentials to the BASIC CbCS method (see 5.16.6.8.8.2) is outside the scope of this standard.

The format of a CbCS credential is described in 6.19.2.2.

The primary components of a CbCS credential are as follows:

a) a CbCS capability whose format and preparation are described in 5.16.6.8.13; and
b) the CbCS capability key that is an integrity check value (see 3.1.77) that is computed as follows:
   A) if the credential is to be sent to a secure CDB originator (see 5.16.6.8.5), the CbCS capability key is computed based on a working key as described in 5.16.6.8.12.2; or
   B) if the credential is being prepared for use by the CbCS management application client (see 5.16.6.8.4), the CbCS capability key is computed based on a working key or the master key as described in 5.16.6.8.12.3.

Regardless of how it is computed, the CbCS capability key is used as follows:

a) by the secure CDB originator to prepare CbCS extension descriptors (see 5.16.6.8.16) sent to the secure CDB processor (see 5.16.6.8.6);
b) by the CbCS management application client to prepare CbCS extension descriptors sent to the enforcement manager (see 5.16.6.8.7); and
c) by the enforcement manager to validate the integrity of capabilities (see 5.16.6.8.13.3) in CbCS extension descriptors received by the secure CDB processor.

5.16.6.8.12.2 CbCS capability key computations for the secure CDB originator

For credentials sent to the secure CDB originator (see 5.16.6.8.5), the CbCS capability key (see 5.16.6.8.12.1) is computed without knowledge of the command for which it is being prepared using the following inputs:

a) the integrity check value algorithm specified by the INTEGRITY CHECK VALUE ALGORITHM field (see 6.19.2.3) in the CbCS capability descriptor; and
b) the following inputs to this integrity check value algorithm:
   A) all the bytes in the CbCS capability descriptor (see 6.19.2.3); and
   B) the working key specified by the KEY VERSION field (see 6.19.2.3) in the CbCS capability descriptor.

5.16.6.8.12.3 CbCS capability key computations for general use

The computation of the CbCS capability key depends on the command for which the CbCS capability key is being computed as described in this subclause. This computation is more general than the computation described in 5.16.6.8.12.2, but it produces the same results because the secure CDB originator should not be allowed to use the commands that produce the exceptional cases described in this subclause.

The CbCS capability key computations described in this subclause are used:

a) for credentials that are to be used by the CbCS management application client (see 5.16.6.8.4); and

b) by the enforcement manager (see 5.16.6.8.7) when validating a CbCS capability descriptor (see 5.16.6.8.13).

When the enforcement manager is validating a CbCS capability descriptor, the command is determined by inspecting the CDB that is being processed.

Based on the command associated with the credential, the CbCS capability key (see 5.16.6.8.12.1) is computed using the following inputs:

a) the integrity check value algorithm specified by the INTEGRITY CHECK VALUE ALGORITHM field (see 6.19.2.3) in the CbCS capability descriptor; and
b) the following inputs to this integrity check value algorithm:
   A) all the bytes in the CbCS capability descriptor (see 6.19.2.3); and
   B) the following CbCS shared key:
      a) if the command is a SECURITY PROTOCOL IN command (see 6.30) or a SECURITY PROTOCOL OUT command (see 6.31) with the SECURITY PROTOCOL field set to 07h (i.e., CbCS) and the SECURITY PROTOCOL SPECIFIC field set to a value that is greater than CFFFh, then the following shared key is used:
         A) if the command does not access the Set Master Key - Change Key CbCS page (see 7.7.4.5.6), the authentication key component of the master key (see 5.16.6.8.11) is used; or
         B) if the command accesses the Set Master Key - Change Key CbCS page, then the authentication key component of the new master key (see 7.7.4.3.6) maintained in the CbCS master key update CCS (see 5.16.6.8.11.4) is used;

      or

      b) if the command is not one of those described in step b) B) a), then the working key specified by the KEY VERSION field (see 6.19.2.3) in the CbCS capability descriptor.

5.16.6.8.13 CbCS capability descriptors

5.16.6.8.13.1 Overview

CbCS capability descriptors are components of:

a) CbCS credentials (see 5.16.6.8.12); and
b) CbCS extension descriptors (see 5.16.6.8.16).

The format of a CbCS capability descriptor is described in 6.19.2.3.

CbCS capability descriptors contain:

a) information about what commands are allowed when the CbCS capability descriptor is associated with a specific CDB via a CbCS extension descriptor;
b) information that identifies a specific logical unit (see 3.1.84) or a specific volume (see SSC-3) mounted in a specific logical unit to which the CbCS capability is bound;
c) an optional time limit on the validity of the CbCS capability descriptor;
d) information that the CbCS management application client (see 5.16.6.8.4) may use to invalidate one or more CbCS capability descriptors before the time limit expires; and
e) information that the enforcement manager uses to cryptographically validate the CbCS capability descriptor, if specified, as described in 5.16.6.8.13.

5.16.6.8.13.2 CbCS extension descriptor validation

The enforcement manager (see 5.16.6.8.7) shall validate the CbCS capability descriptor (see 6.19.2.3) included in the CbCS extension descriptor (see 5.16.6.8.16). If the validation fails, the enforcement manager shall interact with

the secure CDB processor (see 5.16.6.8.6) in a way that causes the command containing the CbCS extension descriptor to be terminated with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB.

The enforcement manager's validation of a CbCS extension descriptor shall fail if any of the following conditions occur:

1) a CbCS extension descriptor is not present on a command that table 77 (see 5.16.6.8.14) shows as requiring a CbCS capability;
2) the command is one that table 77 (see 5.16.6.8.14) shows as never being allowed when CbCS is enabled;
3) the CBCS METHOD field is set to a value that is less than the value in the minimum CbCS method CbCS parameter (see 5.16.6.8.15);
4) the CBCS METHOD field contains a value that table 222 defines as reserved (see 6.19.2.3) or a value that the enforcement manager does not support (see 7.7.4.3.3);
5) if the CBCS METHOD field is set to CAPKEY and the integrity validation described in 5.16.6.8.13.3 fails;
6) the DESIGNATION TYPE field contains a value that table 221 defines as reserved (see 6.19.2.3);
7) the DESIGNATION TYPE field value is set to 1h (i.e., logical unit designation descriptor), and the contents of the DESIGNATION DESCRIPTOR field in which a logical unit name (see SAM-4) is indicated does not match the addressed logical unit;
8) the DESIGNATION TYPE field value is set to 2h (MAM attribute descriptor) and either of the following are true:
   A) the ATTRIBUTE IDENTIFIER field in the DESIGNATION DESCRIPTOR field contains any value other than 0401h (i.e., MEDIUM SERIAL NUMBER); or
   B) the DESIGNATION DESCRIPTOR field contents do not match the MAM attribute of the volume that is accessible via the addressed logical unit;
9) the CAPABILITY EXPIRATION TIME field contains a non-zero value and the value in the CAPABILITY EXPIRATION TIME field is less than (i.e., prior to) the current time in the clock CbCS parameter (see 5.16.6.8.15);
10) the POLICY ACCESS TAG field contains a non-zero value that does not match the policy access tag CbCS parameter (see 5.16.6.8.15); or
11) the command in the CDB field of the extended CDB (see 4.3.4) that contains the CbCS extension descriptor is not permitted by the PERMISSIONS BIT MASK field (see 5.16.6.8.14).

5.16.6.8.13.3 CAPKEY CbCS method capability integrity validation

If the CbCS method is CAPKEY, the enforcement manager's validation of a CbCS capability descriptor shall fail the integrity tests if any of the described in this subclause occur.

Before attempting to cryptographically validate the integrity of the CbCS capability descriptor, the enforcement manager shall fail the validation if any of the following conditions occur in the CbCS capability descriptor (see 6.19.2.3):

a) the KEY VERSION field specifies an invalid working key as follows:
   A) the command is not a SECURITY PROTOCOL IN command (see 6.30) or a SECURITY PROTOCOL OUT command (see 6.31) with the SECURITY PROTOCOL field set to 07h (i.e., CbCS) and the SECURITY PROTOCOL SPECIFIC field set to a value that is greater than CFFFh (i.e., the KEY VERSION field is ignored for these commands);
   B) the per-logical unit working key (see 5.16.6.8.11), if any, specified by the KEY VERSION field is invalid (see 5.16.6.8.11.2); and
   C) the target-wide working key (see 5.16.6.8.11), if any, specified by the KEY VERSION field is invalid (see 5.16.6.8.11.2);

   or
b) the INTEGRITY CHECK VALUE ALGORITHM field contains a value that is:
   A) not one of those that table 92 (see 5.16.8) lists as being an integrity checking (i.e., AUTH) algorithm;
   B) is AUTH_COMBINED; or
   C) is a value that the enforcement manager does not support (see 7.7.4.3.3).

If no integrity checking configuration errors are found in the CbCS capability descriptor, the enforcement manager shall:

1) compute the CbCS capability key for the CbCS capability descriptor as described in 5.16.6.8.12.3; and
2) compute the expected contents of CbCS extension descriptor INTEGRITY CHECK VALUE field (see 5.16.6.8.16), using the following inputs:
   A) the integrity check value algorithm specified by the INTEGRITY CHECK VALUE ALGORITHM field (see 6.19.2.3) in the CbCS capability descriptor; and
   B) the following inputs to this integrity check value algorithm:
      a) all the bytes in the security token (see 5.16.6.8.10) for the I_T nexus on which the command was received as the string for which the integrity check value is to be computed; and
      b) the CbCS capability key computed in step 1) as the cryptographic key.

The enforcement manager shall fail the validation if the contents of CbCS extension descriptor INTEGRITY CHECK VALUE field do not match the computed expected contents of CbCS extension descriptor INTEGRITY CHECK VALUE field.

5.16.6.8.14 Association between commands and permission bits

The PERMISSIONS BIT MASK field in the CbCS capability (see 6.19.2.3) specifies which commands are allowed by the CbCS capability. When processing commands with the CbCS extension, the enforcement manager shall verify that the bits applicable to the encapsulated SCSI command are all set to one in the PERMISSIONS BIT MASK field before processing the command. The associations between commands and permission bits are specified in table 77 and table 78 for commands defined in this standard.

Table 77 — Associations between commands and permissions (part 1 of 3)

Command	PERMISSIONS BIT MASK bits [a]						
	DATA READ	DATA WRITE	PARM READ	PARM WRITE	SEC MGMT	RESRV	MGMT
ACCESS CONTROL IN	never allow [b]						
ACCESS CONTROL OUT	never allow [b]						
CHANGE ALIASES	always allow [c]						
EXTENDED COPY	never allow [b]						
INQUIRY	always allow [c]						
LOG SELECT				1			
LOG SENSE			1				
MANAGEMENT PROTOCOL IN							1
MANAGEMENT PROTOCOL OUT							1
MODE SELECT(6)				1			

[a] The command in the CDB field of the extended CDB (see 4.3.4) that contains the CbCS extension descriptor shall be allowed only when all of the bits marked with a 1 in the row for that command are set in the PERMISSIONS BIT MASK field of the CbCS capability in the CbCS extension descriptor. The permissions bits represented by the empty cells in a row are ignored.

[b] If the CBCS bit is set to one in the Extended INQUIRY Data VPD page (see 7.8.7), this command shall never be allowed.

[c] This command shall always be allowed regardless of whether the CbCS extension descriptor is present and if the CbCS extension descriptor is present regardless of the value in the PERMISSIONS BIT MASK field.

Table 77 — Associations between commands and permissions (part 2 of 3)

Command	PERMISSIONS BIT MASK bits [a]						
	DATA READ	DATA WRITE	PARM READ	PARM WRITE	SEC MGMT	RESRV	MGMT
MODE SELECT(10)				1			
MODE SENSE(6)			1				
MODE SENSE(10)			1				
PERSISTENT RESERVE IN			1				
PERSISTENT RESERVE OUT						1	
READ ATTRIBUTE			1				
READ BUFFER					1		
READ MEDIA SERIAL NUMBER			1				
RECEIVE COPY RESULTS	never allow [b]						
RECEIVE CREDENTIAL	always allow [c]						
RECEIVE DIAGNOSTIC RESULTS			1				
REPORT ALIASES	always allow [c]						
REPORT IDENTIFYING INFORMATION			1				
REPORT LUNS	always allow [c]						
REPORT PRIORITY			1				
REPORT SUPPORTED OPERATION CODES	always allow [c]						
REPORT SUPPORTED TASK MANAGEMENT FUNCTIONS	always allow [c]						
REPORT TARGET PORT GROUPS	always allow [c]						
REPORT TIMESTAMP			1				
REQUEST SENSE			1				
SECURITY PROTOCOL IN	see table 78						
SECURITY PROTOCOL OUT	see table 78						
SEND DIAGNOSTIC				1			
SET IDENTIFYING INFORMATION				1			
SET PRIORITY				1			
SET TARGET PORT GROUPS				1			
SET TIMESTAMP				1	1		

[a] The command in the CDB field of the extended CDB (see 4.3.4) that contains the CbCS extension descriptor shall be allowed only when all of the bits marked with a 1 in the row for that command are set in the PERMISSIONS BIT MASK field of the CbCS capability in the CbCS extension descriptor. The permissions bits represented by the empty cells in a row are ignored.

[b] If the CBCS bit is set to one in the Extended INQUIRY Data VPD page (see 7.8.7), this command shall never be allowed.

[c] This command shall always be allowed regardless of whether the CbCS extension descriptor is present and if the CbCS extension descriptor is present regardless of the value in the PERMISSIONS BIT MASK field.

Table 77 — Associations between commands and permissions (part 3 of 3)

Command	PERMISSIONS BIT MASK bits [a]						
	DATA READ	DATA WRITE	PARM READ	PARM WRITE	SEC MGMT	RESRV	MGMT
TEST UNIT READY	always allow [c]						
WRITE ATTRIBUTE				1			
WRITE BUFFER					1		

[a] The command in the CDB field of the extended CDB (see 4.3.4) that contains the CbCS extension descriptor shall be allowed only when all of the bits marked with a 1 in the row for that command are set in the PERMISSIONS BIT MASK field of the CbCS capability in the CbCS extension descriptor. The permissions bits represented by the empty cells in a row are ignored.

[b] If the CBCS bit is set to one in the Extended INQUIRY Data VPD page (see 7.8.7), this command shall never be allowed.

[c] This command shall always be allowed regardless of whether the CbCS extension descriptor is present and if the CbCS extension descriptor is present regardless of the value in the PERMISSIONS BIT MASK field.

The usage of the PERMISSIONS BIT MASK field for the SECURITY PROTOCOL IN command and the SECURITY PROTOCOL OUT command depend on the following characteristics as shown in table 78:

a) the contents of the SECURITY PROTOCOL field; and
b) the contents of the SECURITY PROTOCOL SPECIFIC field.

Table 78 — Associations between security protocol commands and permissions

Command	SECURITY PROTOCOL field	SECURITY PROTOCOL SPECIFIC field	Description
SECURITY PROTOCOL IN	00h	any	Always allowed regardless of whether the CbCS extension descriptor is present and if the CbCS extension descriptor is present regardless of the value in the PERMISSIONS BIT MASK field
SECURITY PROTOCOL IN	07h	0000h to 003Fh	
SECURITY PROTOCOL IN	07h	0040h to FFFFh	Allowed only if the CbCS extension descriptor is present and the SEC MGMT bit is set to one in the PERMISSIONS BIT MASK field
SECURITY PROTOCOL OUT	any	any	

Command standards (see 3.1.31) may describe the associations between commands and permission bits for the commands that they define. The processing requirements for those associations are the same as those described in this standard.

5.16.6.8.15 CbCS parameters

5.16.6.8.15.1 Overview

CbCS parameters:

a) provide the CbCS Management Application Client class (see 5.16.6.8.4) with a means to control the operation of the Enforcement Manager class(see 5.16.6.8.7);

b) allow the Secure CDB Processor class(see 5.16.6.8.6) to receive security tokens and other CbCS information from the Enforcement Manager class; and
c) allow any application client to receive basic operational CbCS information from the Enforcement Manager class.

CbCS parameters that are not changeable indicate which CbCS features and algorithms are supported. An application client may retrieve the unchangeable CbCS parameters by using the SECURITY PROTOCOL IN command to return the Unchangeable CbCS Parameters CbCS page (see 7.7.4.3.3).

The CbCS parameters with values that change in response to various conditions are summarized in table 79.

Table 79 — Summary of changeable CbCS parameters

Parameter	Support	Applicable CbCS page	Data direction	Capability protection	Reference
CbCS parameters that are updated automatically based on I_T nexus					
Security token	Optional [a]	Security Token	In	None	7.7.4.3.4
CbCS parameters that provide initial values for dynamically created logical units [b]					
Initial minimum CbCS method	Optional	Current CbCS Parameters	In	Working key	7.7.4.3.5
		Set Minimum CbCS Method [c]	Out	Working key	7.7.4.5.2
Initial policy access tag	Optional	Current CbCS Parameters	In	Working key	7.7.4.3.5
		Set Policy Access Tag [d]	Out	Working key	7.7.4.5.1
CbCS parameters that affect the CbCS enforcement manager processing					
Minimum CbCS method	Mandatory	Current CbCS Parameters	In	Working key	7.7.4.3.5
		Set Minimum CbCS Method [c]	Out	Working key	7.7.4.5.2
Policy Access Tag	Mandatory	Current CbCS Parameters	In	Working key	7.7.4.3.5
		Set Policy Access Tag [d]	Out	Working key	7.7.4.5.1
Clock	Mandatory	Current CbCS Parameters	In	Working key	7.7.4.3.5
CbCS Shared keys and CbCS shared key identifiers	Optional [a]	see 5.16.6.8.11			

[a] Mandatory if the CAPKEY CbCS method (see 5.16.6.8.8.3) is supported.
[b] SCSI target devices that do not dynamically create logical units may not implement these CbCS parameters. Retrieving and setting these CbCS parameters is possible only if the SECURITY PROTOCOL well known logical unit (see 8.5) is implemented. SCSI target devices that dynamically create logical units but do not implement the SECURITY PROTOCOL well known logical unit shall provide a means outside the scope of this standard for managing the values of these CbCS parameters.
[c] If a Set Minimum CbCS Method CbCS page is processed by the SECURITY PROTOOCL well known logical unit, then the initial Minimum CbCS Method CbCS parameter is changed. If a Set Minimum CbCS Method CbCS page is processed by any logical unit other than the SECURITY PROTOOCL well known logical unit, then the minimum CbCS method CbCS parameter for that logical unit is changed.
[d] If a Set Policy Access Tag CbCS page is processed by the SECURITY PROTOOCL well known logical unit, then the initial policy access tag CbCS parameter is changed. If a Set Policy Access Tag CbCS page is processed by any logical unit other than the SECURITY PROTOOCL well known logical unit, then the policy access tag CbCS parameter for that logical unit is changed.

The security token CbCS parameter is described in 5.16.6.8.10.

The minimum CbCS method parameter indicates the minimum allowable CbCS method (see 5.16.6.8.8) to be used in capabilities (see 6.19.2.3) processed by an enforcement manager (see 5.16.6.8.7). The initial minimum CbCS method CbCS parameter provides an initial value for the minimum CbCS method CbCS parameter for dynamically created logical units.

The policy access tag parameter indicates the allowable contents of the POLICY ACCESS TAG field in capabilities (see 6.19.2.3) processed by an enforcement manager (see 5.16.6.8.7). The initial policy access tag CbCS parameter provides an initial value for the policy access tag CbCS parameter for dynamically created logical units.

The clock CbCS parameter indicates the time used by the enforcement manager (see 5.16.6.8.7) when evaluating the contents of the CAPABILITY EXPIRATION TIME field in a capability (see 6.19.2.3). The clock CbCS parameter is timestamp (see 5.15) and its value is managed using the same mechanisms that are used to manage a timestamp.

The CbCS shared keys and CbCS shared key identifiers are described in 5.16.6.8.11.

5.16.6.8.16 CbCS extension descriptor format

The CbCS extension descriptor (see table 80) allows the capability-based command security technique (see 5.16.6.8) to used with a SCSI command via the parameters specified in this subclause. Support for the CbCS extension descriptor is mandatory if CBCS bit to one in the Extended INQUIRY Data VPD page (see 7.8.7). When an extended CDB (see 4.3.4) includes a CbCS extension descriptor the CDB field may contain any CDB defined in this standard or any SCSI command standard (see 3.1.31).

Table 80 — CbCS extension descriptor format

Bit / Byte	7	6	5	4	3	2	1	0
0	EXTENSION TYPE (40h)							
1	Reserved							
2								
3								
4	CbCS capability descriptor (see 6.19.2.3)							
•••								
75								
76	(MSB)	INTEGRITY CHECK VALUE						
•••								
139								(LSB)

The EXTENSION TYPE field contains 40h (i.e., CbCS extension descriptor).

The CbCS capability descriptor is defined in 6.19.2.3.

The contents of INTEGRITY CHECK VALUE field depend on the contents of the CBCS METHOD field in the CbCS capability descriptor as follows:

a) if the CBCS METHOD field does not contain CAPKEY or a vendor specific value (see table 222 in 6.19.2.3), then the INTEGRITY CHECK VALUE field is reserved;
b) if the CBCS METHOD field contains a vendor specific value, then the contents of the INTEGRITY CHECK VALUE field are vendor specific; or
c) if the CBCS METHOD field contains CAPKEY, then the INTEGRITY CHECK VALUE field contains an integrity check value (see 3.1.77) that is computed using the integrity check value algorithm specified by the

INTEGRITY CHECK VALUE ALGORITHM field in the CbCS capability descriptor (see 6.19.2.3) and the following inputs to the integrity check value algorithm:
A) all the bytes in the security token (see 5.16.6.8.10) for the I_T nexus on which the command is being sent as the string for which the integrity check value is to be computed; and
B) the CbCS capability key from the credential (see 5.16.6.8.12) in which the CbCS capability descriptor was received by the secure CDB originator as the cryptographic key.

The enforcement manager shall validate the CbCS capability descriptor and the INTEGRITY CHECK VALUE field as described in 5.16.6.8.13.2.

5.16.7 ESP-SCSI for parameter data

5.16.7.1 Overview

Subclause 5.16.7 defines a method for transferring encrypted and/or integrity checked parameter data in Data-In Buffers, Data-Out Buffers, variable length CDBs (see 4.3.3), and extended CDBs (see 4.3.4). The method is based on the Encapsulating Security Payload (see RFC 4303) standard developed by the IETF. Because of the constrained usage of ESP-SCSI parameter data in Data-In Buffers and/or Data-Out Buffers, the method defined in this standard differs from the one found in RFC 4303.

5.16.7.2 ESP-SCSI required inputs

Prior to using the ESP-SCSI descriptors defined in 5.16.7, an SA shall be created (see 5.16.2.3) with SA parameters that conform to the requirements defined in 5.16.2.2 and to the following:

a) the USAGE_TYPE SA parameter shall be set to a value for which ESP-SCSI usage is defined in table 62 (see 5.16.2.2);
b) the USAGE_DATA SA parameter shall contain at least the following:
   A) the algorithm identifier and key length for the encryption algorithm (e.g., the ALGORITHM IDENTIFIER field and KEY LENGTH field from the ENCR IKEv2-SCSI cryptographic algorithm descriptor (see 7.7.3.6.2) in the IKEv2-SCSI SAUT Cryptographic Algorithms payload (see 7.7.3.5.14) negotiated by an IKEv2-SCSI SA creation protocol (see 5.16.4)); and
   B) the algorithm identifier for the integrity algorithm (e.g., the ALGORITHM IDENTIFIER field from the INTEG IKEv2-SCSI cryptographic algorithm descriptor (see 7.7.3.6.4) in the IKEv2-SCSI SAUT Cryptographic Algorithms payload (see 7.7.3.5.14) negotiated by an IKEv2-SCSI SA creation protocol (see 5.16.4));
   and
c) the KEYMAT SA parameter shall consist of the shared keys described in 5.16.4.8.6.

ESP-SCSI depends on the following additional information derived from the contents of the USAGE_DATA SA parameter:

a) the encryption algorithm identifier shall indicate:
   A) the absence of encryption by having the value ENCR_NULL (see table 92 in 5.16.8);
   B) the size of the initialization vector, if any (e.g., as shown in table 488 (see 7.7.3.6.2));
   C) the size of the salt bytes, if any (e.g., as shown in table 488 (see 7.7.3.6.2));
   D) for combined mode encryption algorithms, the size of the integrity check value (i.e., the algorithm's MAC size as shown in table 488 (see 7.7.3.6.2));
   and
b) the integrity algorithm identifier shall indicate:
   A) the use of a combined mode encryption algorithm by having the value AUTH_COMBINED (see table 92 in 5.16.8);
   B) for non-combined mode encryption algorithms, the size of the integrity check value (e.g., as shown in table 492 (see 7.7.3.6.4)).

Each shared key in KEYMAT shall be taken from the KDF generated bits in the order shown in 5.16.4.8.6. The size of each of the shared keys in KEYMAT is determined by the negotiated encryption algorithm and integrity algorithm as described in 5.16.4.4.

5.16.7.3 ESP-SCSI data format before encryption and after decryption

Before data bytes are encrypted and after they are decrypted, they have the format shown in table 81.

Table 81 — ESP-SCSI data format before encryption and after decryption

Bit / Byte	7	6	5	4	3	2	1	0
0	UNENCRYPTED BYTES							
•••								
p-1								
p	PADDING BYTES							
•••								
j-1								
j	PAD LENGTH (j-p)							
j+1	MUST BE ZERO							

The UNENCRYPTED BYTES field contains the bytes that are to be protected via encryption or that have been decrypted.

Before encryption, the PADDING BYTES field contains zero to 255 bytes. The number of padding bytes is:

a) defined by the encryption algorithm; or
b) the number needed to cause the length of all bytes prior to encryption (i.e., j+2) to be a whole multiple of the alignment (see table 488 in 7.7.3.6.2) for the encryption algorithm being used.

The contents of the padding bytes are:

a) defined by the encryption algorithm; or
b) if the encryption algorithm does not define the padding bytes contents, a series of one byte binary values starting at one and incrementing by one in each successive byte (i.e., 01h in the first padding byte, 02h in the second padding byte, etc.).

If the encryption algorithm does not place requirements on the contents of the padding bytes (i.e., option b) is in effect), then after decryption the contents of the padding bytes shall be verified to match the series of one byte binary values described in this subclause. If this verification is not successful in a device server, the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, the additional sense code set to INVALID FIELD IN PARAMETER LIST, the SKSV bit set to one, and SENSE KEY SPECIFIC field set to indicate the last byte in the encrypted data as defined in 4.5.2.4.2. If this verification is not successful in an application client, the decrypted data should be ignored.

The PAD LENGTH field contains the number of bytes in the PADDING BYTES field.

The MUST BE ZERO field contains zero. After decryption, the contents of the MUST BE ZERO field shall be verified to be zero. If this verification is not successful in a device server, the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, the additional sense code set to INVALID FIELD IN PARAMETER LIST, the SKSV bit set to one, and SENSE KEY SPECIFIC field set to indicate the last byte in

the encrypted data as defined in 4.5.2.4.2. If this verification is not successful in an application client, the decrypted data should be ignored.

5.16.7.4 ESP-SCSI outbound data descriptors

5.16.7.4.1 Overview

When ESP-SCSI is used in a variable length CDB (see 4.3.3), an extended CDB (see 4.3.4), or parameter list data that appears in a Data-Out Buffer, the parameter list data contains one or more descriptors selected based on the criteria shown in table 82.

Table 82 — ESP-SCSI outbound data descriptors

Descriptor name	External descriptor length [a]	Initialization vector present [b]	Reference
ESP-SCSI CDB	No	No	table 83 in 5.16.7.4.2
	No	Yes	table 84 in 5.16.7.4.2
ESP-SCSI Data-Out Buffer	No	No	table 83 in 5.16.7.4.2
	No	Yes	table 84 in 5.16.7.4.2
ESP-SCSI Data-Out Buffer without length	Yes	No	table 85 in 5.16.7.4.3
	Yes	Yes	table 86 in 5.16.7.4.3

[a] This is determined by the data format defined for the Data-Out Buffer parameter data. If the format includes a length for the ESP-SCSI descriptor, then the answer to this question is yes.

[b] This is determined from the USAGE_DATA SA parameter (see 5.16.7.2).

5.16.7.4.2 ESP-SCSI CDBs or Data-Out Buffer parameter lists including a descriptor length

If the USAGE_DATA SA parameter (see 5.16.7.2) indicates an encryption algorithm whose initialization vector size is zero, then the variable length CDB (see 4.3.3), extended CDB (see 4.3.4), or Data-Out Buffer parameter list descriptor shown in table 83 contains the ESP-SCSI data.

Table 83 — ESP-SCSI CDBs or Data-Out Buffer parameter list descriptor without initialization vector

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)			DESCRIPTOR LENGTH (n-1)				
1								(LSB)
2				Reserved				
3								
4	(MSB)			DS_SAI				
•••								
7								(LSB)
8	(MSB)			DS_SQN				
•••								
15								(LSB)
16				ENCRYPTED OR AUTHENTICATED DATA				
•••								
i-1								
i	(MSB)			INTEGRITY CHECK VALUE				
•••								
n								(LSB)

The DESCRIPTOR LENGTH field, DS_SAI field, DS_SQN field, ENCRYPTED OR AUTHENTICATED DATA field, and INTEGRITY CHECK VALUE field are defined after table 84 in this subclause.

If the USAGE_DATA SA parameter indicates an indicates an encryption algorithm whose initialization vector size (i.e., s) is greater than zero, the variable length CDB (see 4.3.3), extended CDB (see 4.3.4), or Data-Out Buffer parameter data descriptor shown in table 84 contains the ESP-SCSI data.

Table 84 — ESP-SCSI CDBs or Data-Out Buffer full parameter list descriptor

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)			DESCRIPTOR LENGTH (n-1)				
1								(LSB)
2				Reserved				
3								
4	(MSB)			DS_SAI				
•••								
7								(LSB)
8	(MSB)			DS_SQN				
•••								
15								(LSB)
16	(MSB)			INITIALIZATION VECTOR				
•••								
16+s-1								(LSB)
16+s				ENCRYPTED OR AUTHENTICATED DATA				
•••								
i-1								
i	(MSB)			INTEGRITY CHECK VALUE				
•••								
n								(LSB)

The DESCRIPTOR LENGTH field specifies the number of bytes that follow in the ESP-SCSI CDB or ESP-SCSI Data-Out Buffer parameter list descriptor.

The DS_SAI field contains the value in the DS_SAI SA parameter (see 5.16.2.2) for the SA that is being used to prepare the ESP-SCSI CDB or ESP-SCSI Data-Out Buffer parameter list descriptor. If the DS_SAI value is not known to the device server, the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, the additional sense code set to INVALID FIELD IN PARAMETER LIST, the SKSV bit set to one, and SENSE KEY SPECIFIC field set as defined in 4.5.2.4.2.

The DS_SQN field should contain one plus the value in the application client's DS_SQN SA parameter (see 5.16.2.2) for the SA that is being used to prepare the ESP-SCSI CDB or ESP-SCSI Data-Out Buffer parameter list descriptor. Before sending the ESP-SCSI CDB or ESP-SCSI Data-Out Buffer parameter list, the application client should copy the contents of the DS_SQN field to its DS_SQN SA parameter.

The device server shall terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, the additional sense code set to INVALID FIELD IN PARAMETER LIST, the SKSV bit set to one, and SENSE KEY SPECIFIC field set as defined in 4.5.2.4.2 if any of the following conditions are detected:

a) the DS_SQN field is set to zero;
b) the value in the DS_SQN field is less than or equal to the value in the device server's DS_SQN SA parameter; or

c) the value in the DS_SQN field is greater than 32 plus the value in the device server's DS_SQN SA parameter.

If the DS_SQN SA parameter is equal to FFFF FFFF FFFF FFFFh, the device server shall delete the SA.

The INITIALIZATION VECTOR field, if any, contains a value that is used as an input into the encryption algorithm and/or integrity algorithm specified by the SA specified by the DS_SAI field. The INITIALIZATION VECTOR field is not encrypted. The encryption algorithm and/or integrity algorithm may define additional requirements for the INITIALIZATION VECTOR field.

The ENCRYPTED OR AUTHENTICATED DATA field contains:

a) if an encryption algorithm for the SA specified by the DS_SAI field is not ENCR_NULL, encrypted data bytes for the following:
   1) the bytes in the UNENCRYPTED BYTES field (see 5.16.7.3);
   2) the bytes in the PADDING BYTES field (see 5.16.7.3);
   3) the PAD LENGTH field byte (see 5.16.7.3); and
   4) the MUST BE ZERO field byte (see 5.16.7.3);

   or
b) otherwise, the unencrypted data bytes.

If the integrity algorithm for the SA specified by the DS_SAI field is AUTH_COMBINED (see 5.16.7.2), then the AAD input to the encryption algorithm is composed of the following bytes, in order:

1) the bytes in the DS_SAI field; and
2) the bytes in the DS_SQN field.

The INTEGRITY CHECK VALUE field contains a value that is computed as follows:

a) if the integrity algorithm is not AUTH_COMBINED, the integrity check value is computed using the specified integrity algorithm with the following bytes as inputs, in order:
   1) the bytes in the DS_SAI field;
   2) the bytes in the DS_SQN field;
   3) the bytes in the INITIALIZATION VECTOR field, if any; and
   4) the bytes in the ENCRYPTED OR AUTHENTICATED DATA field after encryption, if any, has been performed;

   or
b) if the integrity algorithm is AUTH_COMBINED, the integrity check value is computed as an additional output of the specified encryption algorithm.

Upon receipt of ESP-SCSI CDB or ESP-SCSI Data-Out Buffer parameter data, the device server shall compute an integrity check value for the ESP-SCSI CDB or ESP-SCSI Data-Out Buffer parameter data as specified by the algorithms specified by the SA specified by the DS_SAI field using the inputs shown in this subclause. If the computed integrity check value does not match the value in the INTEGRITY CHECK VALUE field, the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, the additional sense code set to INVALID FIELD IN PARAMETER LIST, the SKSV bit set to one, and SENSE KEY SPECIFIC field set as defined in 4.5.2.4.2.

If the command is not terminated due to a sequence number error or a mismatch between the computed integrity check value and the contents of the INTEGRITY CHECK VALUE field, then the device server shall copy the contents of the received DS_SQN field to its DS_SQN SA parameter.

5.16.7.4.3 ESP-SCSI Data-Out Buffer parameter lists for externally specified descriptor length

If the USAGE_DATA SA parameter (see 5.16.7.2) indicates an encryption algorithm whose initialization vector size is zero and the length of the ESP-SCSI Data-Out Buffer parameter list descriptor appears elsewhere in the parameter list, then the Data-Out Buffer parameter list descriptor shown in table 85 contains the ESP-SCSI data.

Table 85 — ESP-SCSI Data-Out Buffer parameter list descriptor without length and initialization vector

Bit / Byte	7	6	5	4	3	2	1	0
0	Reserved							
•••								
3								
4	(MSB)	DS_SAI						
•••								
7								(LSB)
8	(MSB)	DS_SQN						
•••								
15								(LSB)
16	ENCRYPTED OR AUTHENTICATED DATA							
•••								
i-1								
i	(MSB)	INTEGRITY CHECK VALUE						
•••								
n								(LSB)

The DS_SAI field, DS_SQN field, ENCRYPTED OR AUTHENTICATED DATA field, and INTEGRITY CHECK VALUE field are defined in 5.16.7.4.2.

If the USAGE_DATA SA parameter indicates an indicates an encryption algorithm whose initialization vector size (i.e., s) is greater than zero and the length of the ESP-SCSI Data-Out Buffer parameter list descriptor appears elsewhere in the parameter list, the Data-Out Buffer parameter list descriptor shown in table 86 contains the ESP-SCSI data.

Table 86 — ESP-SCSI Data-Out Buffer parameter list descriptor without length

Bit / Byte	7	6	5	4	3	2	1	0
0	Reserved							
•••								
3								
4	(MSB)	DS_SAI						
•••								
7								(LSB)
8	(MSB)	DS_SQN						
•••								
15								(LSB)
16	(MSB)	INITIALIZATION VECTOR						
•••								
16+s-1								(LSB)
16+s	ENCRYPTED OR AUTHENTICATED DATA							
•••								
i-1								
i	(MSB)	INTEGRITY CHECK VALUE						
•••								
n								(LSB)

The DS_SAI field, DS_SQN field, INITIALIZATION VECTOR field, ENCRYPTED OR AUTHENTICATED DATA field, and INTEGRITY CHECK VALUE field are defined in 5.16.7.4.2.

5.16.7.5 ESP-SCSI Data-In Buffer parameter data descriptors

5.16.7.5.1 Overview

A device server shall transfer ESP-SCSI parameter data descriptors in a Data-In Buffer only in response to a request that specifies an SA using the AC_SAI SA parameter and DS_SAI SA parameter values (see 5.16.2.2). If the specified combination of AC_SAI and DS_SAI values in a command that requests the transfer of ESP-SCSI parameter data descriptors is not known to the device server, the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, the additional sense code set to INVALID FIELD IN PARAMETER LIST or to INVALID FIELD IN CDB, the SKSV bit set to one, and SENSE KEY SPECIFIC field set as defined in 4.5.2.4.2.

When ESP-SCSI is used in parameter data which appears in a Data-In Buffer, the parameter data contains one or more descriptors selected based on the criteria shown in table 87.

Table 87 — ESP-SCSI Data-In Buffer parameter data descriptors

Descriptor name	External descriptor length [a]	Initialization vector present [b]	Reference
ESP-SCSI Data-In Buffer	No	No	table 88 in 5.16.7.5.2
	No	Yes	table 89 in 5.16.7.5.2
ESP-SCSI Data-In Buffer without length	Yes	No	table 90 in 5.16.7.5.3
	Yes	Yes	table 91 in 5.16.7.5.3

[a] This is determined by the data format defined for the Data-In Buffer parameter data. If the format includes a length for the ESP-SCSI descriptor, then the answer to this question is yes.
[b] This is determined from the USAGE_DATA SA parameter (see 5.16.7.2).

If ESP-SCSI parameter data descriptors are used in a Data-In Buffer, then the outbound data (see 5.16.7.4) should include at least one ESP-SCSI descriptor using the same SA to thwart known plaintext attacks (see 5.16.1.4).

5.16.7.5.2 ESP-SCSI Data-In Buffer parameter data including a descriptor length

If the USAGE_DATA SA parameter (see 5.16.7.2) indicates an encryption algorithm whose initialization vector size is zero, then the Data-In Buffer parameter data descriptor shown in table 88 contains the ESP-SCSI data.

Table 88 — ESP-SCSI Data-In Buffer parameter data descriptor without initialization vector

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)			DESCRIPTOR LENGTH (n-1)				
1								(LSB)
2				Reserved				
3								
4	(MSB)			AC_SAI				
•••								
7								(LSB)
8	(MSB)			AC_SQN				
•••								
15								(LSB)
16				ENCRYPTED OR AUTHENTICATED DATA				
•••								
i-1								
i	(MSB)			INTEGRITY CHECK VALUE				
•••								
n								(LSB)

The DESCRIPTOR LENGTH field, AC_SAI field, AC_SQN field, ENCRYPTED OR AUTHENTICATED DATA field, and INTEGRITY CHECK VALUE field are defined after table 89 in this subclause.

If the USAGE_DATA SA parameter indicates an indicates an encryption algorithm whose initialization vector size (i.e., s) is greater than zero, the Data-In Buffer parameter data descriptor shown in table 89 contains the ESP-SCSI data.

Table 89 — ESP-SCSI Data-In Buffer full parameter data descriptor

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)			DESCRIPTOR LENGTH (n-1)				
1								(LSB)
2				Reserved				
3								
4	(MSB)			AC_SAI				
•••								
7								(LSB)
8	(MSB)			AC_SQN				
•••								
15								(LSB)
16	(MSB)			INITIALIZATION VECTOR				
•••								
16+s-1								(LSB)
16+s				ENCRYPTED OR AUTHENTICATED DATA				
•••								
i-1								
i	(MSB)			INTEGRITY CHECK VALUE				
•••								
n								(LSB)

The DESCRIPTOR LENGTH field specifies the number of bytes that follow in the ESP-SCSI Data-In Buffer parameter data descriptor.

The AC_SAI field contains the value in the AC_SAI SA parameter (see 5.16.2.2) for the SA that is being used to prepare the ESP-SCSI Data-In Buffer parameter data descriptor. If the AC_SAI value is not known to the application client, the ESP-SCSI data-in parameter data descriptor should be ignored.

The AC_SQN field contains one plus the value in the device server's AC_SQN SA parameter (see 5.16.2.2) for the SA that is being used to prepare the ESP-SCSI data-on buffer parameter data descriptor. Before sending the ESP-SCSI Data-Out Buffer parameter list as part of a command that completes with GOOD status, the device server shall copy the contents of the AC_SQN field to its AC_SQN SA parameter. The device server shall not send two ESP-SCSI Data-Out Buffer parameter data descriptors that contain the same values in AC_SAI field and AC_SQN field.

If the AC_SQN SA parameter is equal to FFFF FFFF FFFF FFFFh, the device server shall delete the SA after the Data-In Buffer parameter data containing that value is sent.

The application client should ignore the ESP-SCSI data-in parameter data descriptor if any of the following conditions are detected:

a) the AC_SQN field is set to zero;
b) the value in the AC_SQN field is less than or equal to the value in the application client's AC_SQN SA parameter; or
c) the value in the AC_SQN field is greater than 32 plus the value in the application client's AC_SQN SA parameter.

The INITIALIZATION VECTOR field, if any, contains a value that is used as an input into the encryption algorithm and/or integrity algorithm specified by the SA specified by the AC_SAI field. The INITIALIZATION VECTOR field is not encrypted. The encryption algorithm and/or integrity algorithm may define additional requirements for the INITIALIZATION VECTOR field.

The ENCRYPTED OR AUTHENTICATED DATA field contains:

a) if an encryption algorithm specified by the SA specified by the AC_SAI field is not ENCR_NULL, encrypted data bytes for the following:
   1) the bytes in the UNENCRYPTED BYTES field (see 5.16.7.3);
   2) the bytes in the PADDING BYTES field (see 5.16.7.3);
   3) the PAD LENGTH field byte (see 5.16.7.3); and
   4) the MUST BE ZERO field byte (see 5.16.7.3);

   or
b) otherwise, the unencrypted data bytes.

If the integrity algorithm for the SA specified by the AC_SAI field is AUTH_COMBINED (see 5.16.7.2), then the AAD input to the encryption algorithm is composed of the following bytes, in order:

1) the bytes in the AC_SAI field; and
2) the bytes in the AC_SQN field;

The INTEGRITY CHECK VALUE field contains a value that is computed as follows:

a) if the integrity algorithm is not AUTH_COMBINED, the integrity check value is computed using the specified integrity algorithm with the following bytes as inputs, in order:
   1) the bytes in the AC_SAI field;
   2) the bytes in the AC_SQN field;
   3) the bytes in the INITIALIZATION VECTOR field, if any; and
   4) the bytes in the ENCRYPTED OR AUTHENTICATED DATA field after encryption, if any, has been performed;

   or
b) if the integrity algorithms is AUTH_COMBINED, the integrity check value is computed as an additional output of the specified encryption algorithm.

Upon receipt of ESP-SCSI Data-In Buffer parameter data, the application client should compute an integrity check value for the ESP-SCSI parameter data as specified by the algorithms specified by the SA specified by the AC_SAI field using the inputs shown in this subclause. If the computed integrity check value does not match the value in the INTEGRITY CHECK VALUE field, the results returned by the command should be ignored.

The application client should copy the contents of the AC_SQN field to its AC_SQN SA parameter if all of the following are true:

a) the command completed with GOOD status;
b) the ESP-SCSI data-in parameter data descriptor was not ignored due to inconsistency problems with the AC_SQN field; and

c) the computed integrity check value matched the contents of the INTEGRITY CHECK VALUE field.

5.16.7.5.3 ESP-SCSI Data-In Buffer parameter data for externally specified descriptor length

If the USAGE_DATA SA parameter (see 5.16.7.2) indicates an encryption algorithm whose initialization vector size is zero and the length of the ESP-SCSI Data-In Buffer parameter data descriptor appears elsewhere in the parameter data, then the Data-In Buffer parameter data descriptor shown in table 90 contains the ESP-SCSI data.

Table 90 — ESP-SCSI Data-In Buffer parameter data descriptor without length and initialization vector

Bit / Byte	7	6	5	4	3	2	1	0
0								
•••				Reserved				
3								
4	(MSB)							
•••				AC_SAI				
7								(LSB)
8	(MSB)							
•••				AC_SQN				
15								(LSB)
16								
•••				ENCRYPTED OR AUTHENTICATED DATA				
i-1								
i	(MSB)							
•••				INTEGRITY CHECK VALUE				
n								(LSB)

The AC_SAI field, AC_SQN field, ENCRYPTED OR AUTHENTICATED DATA field, and INTEGRITY CHECK VALUE field are defined in 5.16.7.5.2.

If the USAGE_DATA SA parameter indicates an indicates an encryption algorithm whose initialization vector size (i.e., s) is greater than zero and the length of the ESP-SCSI Data-In Buffer parameter data descriptor appears elsewhere in the parameter data, the Data-In Buffer parameter data descriptor shown in table 91 contains the ESP-SCSI data.

Table 91 — ESP-SCSI Data-In Buffer parameter data descriptor without length

Bit / Byte	7	6	5	4	3	2	1	0
0				Reserved				
•••								
3								
4	(MSB)			AC_SAI				
•••								
7								(LSB)
8	(MSB)			AC_SQN				
•••								
15								(LSB)
16	(MSB)			INITIALIZATION VECTOR				
•••								
16+s-1								(LSB)
16+s				ENCRYPTED OR AUTHENTICATED DATA				
•••								
i-1								
i	(MSB)			INTEGRITY CHECK VALUE				
•••								
n								(LSB)

The AC_SAI field, AC_SQN field, INITIALIZATION VECTOR field, ENCRYPTED OR AUTHENTICATED DATA field, and INTEGRITY CHECK VALUE field are defined in 5.16.7.5.2.

5.16.8 Security algorithm codes

Table 92 lists the security algorithm codes used in security protocol parameter data.

Table 92 — Security algorithm codes (part 1 of 2)

Code	Description	Reference
Encryption algorithms		
0001 000Ch	CBC-AES-256-HMAC-SHA-1	IEEE 1619.1
0001 0010h	CCM-128-AES-256	IEEE 1619.1
0001 0014h	GCM-128-AES-256	IEEE 1619.1
0001 0016h	XTS-AES-256-HMAC-SHA-512	IEEE 1619.1
8001 000Bh [a]	ENCR_NULL	7.7.3.6.2
8001 000Ch [a]	AES-CBC	RFC 3602
8001 0010h [a]	AES-CCM with a 16 byte MAC	RFC 4309
8001 0014h [a]	AES-GCM with a 16 byte MAC	RFC 4106
8001 0400h to 8001 FFFFh	Vendor specific	
PRF and KDF algorithms [b]		
8002 0002h [a]	IKEv2-use based on SHA-1	table 490 (see 7.7.3.6.3)
8002 0004h [a]	IKEv2-use based on AES-128 in CBC mode	
8002 0005h [a]	IKEv2-use based on SHA-256	
8002 0007h [a]	IKEv2-use based on SHA-512	
8002 0400h to 8002 FFFFh	Vendor specific	
Integrity checking (i.e., AUTH) algorithms		
8003 0002h [a]	AUTH_HMAC_SHA1_96	RFC 2404
8003 000Ch [a]	AUTH_HMAC_SHA2_256_128	RFC 4868
8003 000Eh [a]	AUTH_HMAC_SHA2_512_256	RFC 4868
F003 0000h	AUTH_COMBINED	7.7.3.6.4
8003 0400h to 8003 FFFFh	Vendor specific	

[a] The lower order 16 bits of this code value are assigned to match an IANA assigned value, if any, for an equivalent IKEv2 encryption algorithm (see 3.1.78) and values of 800xh in the high order 16 bits have x selected to match the IANA assigned IKEv2 transform type (e.g., 8001h – Encryption Algorithms, 8002h – PRFs and KDFs).

[b] PRFs are equivalent to the prf() functions defined in RFC 4306. KDFs are equivalent to the prf+() functions defined in RFC 4306.

[c] The low order 8 bits of this code value are assigned to match the auth method field in the Authentication payload (see 7.7.3.5.7).

Table 92 — Security algorithm codes (part 2 of 2)

Code	Description	Reference
Diffie-Hellman algorithms		
8004 000Eh [a]	2 048-bit MODP group (finite field D-H)	RFC 3526
8004 000Fh [a]	3 072-bit MODP group (finite field D-H)	RFC 3526
8004 0010h [a]	4 096-bit MODP group (finite field D-H)	RFC 3526
8004 0013h [a]	256-bit random ECP group	RFC 4753
8004 0015h [a]	521-bit random ECP group	RFC 4753
8004 0400h to 8004 FFFFh	Vendor specific	
SA Authentication payload authentication algorithms		
00F9 0000h [c]	SA_AUTH_NONE	5.16.4.3.4 and 7.7.3.6.6
00F9 0001h [c]	RSA Digital Signature with SHA-1	RFC 4306
00F9 0002h [c]	Shared Key Message Integrity Code	RFC 4306
00F9 0009h [c]	ECDSA with SHA-256 on the P-256 curve	RFC 4754
00F9 000Bh [c]	ECDSA with SHA-512 on the P-521 curve	RFC 4754
00F9 00C9h to 00F9 00FFh	Vendor specific	
Other algorithms		
0000 0000h to 0000 FFFFh	Restricted	IANA
All other values	Reserved	

[a] The lower order 16 bits of this code value are assigned to match an IANA assigned value, if any, for an equivalent IKEv2 encryption algorithm (see 3.1.78) and values of 800xh in the high order 16 bits have x selected to match the IANA assigned IKEv2 transform type (e.g., 8001h – Encryption Algorithms, 8002h – PRFs and KDFs).

[b] PRFs are equivalent to the prf() functions defined in RFC 4306. KDFs are equivalent to the prf+() functions defined in RFC 4306.

[c] The low order 8 bits of this code value are assigned to match the auth method field in the Authentication payload (see 7.7.3.5.7).

5.17 Identifying information

The REPORT IDENTIFYING INFORMATION command (see 6.22) and SET IDENTIFYING INFORMATION command (see 6.33) allow an application client to maintain one or more sets of identifying information associated with the peripheral device.

Identifying information shall persist through power cycles (i.e., be stored in non-volatile storage), hard resets, logical unit resets, I_T nexus losses, media format operations, and media replacement.

Table 93 defines the identifying information types.

Table 93 — Identifying information types

Code	Description	Length	Support [a]
0000000b	Peripheral device identifying information – a value describing the peripheral device (e.g., an operating system volume label).	0 to 64 bytes	Mandatory
		65 to 512 bytes	Optional
0000010b	Peripheral device text identifying information – a null-terminated (see 4.4.2) UTF-8 (see 3.1.189) format string providing an informational description of the peripheral device (e.g., a descriptive string entered by a system administrator).	0 to 256 bytes	Optional
xxxxxx1b	Restricted (see SCC-2)		
All other values	Reserved		

[a] These support requirements shall apply only if the REPORT IDENTIFYING INFORMATION command and/or SET IDENTIFYING INFORMATION command are implemented.

Identifying information is changed by:

a) the SET IDENTIFYING INFORMATION command; or
b) a mechanism outside the scope of this standard (e.g., a system administrator may be able to change identifying information through a management interface).

If a mechanism outside the scope of this standard changes the identifying information, then the device server shall establish a unit attention condition for the initiator port associated with every I_T nexus with the additional sense code set to DEVICE IDENTIFIER CHANGED.

5.18 Downloading and activating microcode

SCSI target device implementations may use microcode (e.g., firmware) that is stored in nonvolatile storage. Microcode may be changeable by an application client using the WRITE BUFFER command (see 6.39). The WRITE BUFFER command provides multiple methods for downloading microcode to the SCSI target device and activating the microcode.

Downloading and activating microcode involves the following steps:

1) **Download:** the application client transfers microcode from the Data-Out buffer to the device server in one or more WRITE BUFFER commands;

2) **Save:** after receiving the complete microcode, if defined by the download microcode mode, the device server saves the microcode to nonvolatile storage; and
3) **Activate:** after receiving the complete microcode and after saving it to nonvolatile storage if defined by the download microcode mode, the SCSI target device begins using the new microcode for the first time after an event defined by the download microcode mode.

After power on or hard reset, the SCSI target device shall use the last microcode that was saved to nonvolatile storage.

Table 94 defines the WRITE BUFFER download microcode modes with respect to the steps described in this subclause.

Table 94 — WRITE BUFFER download microcode modes

Mode	Down-load [a]	Save [b]	Activate [c]
Download microcode and activate (i.e., 04h)	yes [d]	no	yes
Download microcode, save, and activate (i.e., 05h)	yes [d]	yes	optional
Download microcode with offsets and activate (i.e., 06h)	yes [e]	no	yes
Download microcode with offsets, save, and activate (i.e., 07h)	yes [e]	yes	optional
Download microcode with offsets, save, and defer activate (i.e., 0Eh) [f]	yes [e]	yes	no
Activate deferred microcode (i.e., 0Fh)	no	no	yes

[a] Entries in the Download column are as follows. For modes labeled yes, the application client delivers microcode in the WRITE BUFFER command(s). For modes labeled no, the application client does not deliver microcode with the WRITE BUFFER command.

[b] Entries in the Save column are as follows. For modes labeled yes, the device server shall save the microcode to nonvolatile storage for use after each subsequent power on or hard reset, and shall not return GOOD status for the final command in the WRITE BUFFER sequence (i.e., the series of WRITE BUFFER commands that downloads the microcode) until the microcode has been saved. For modes labeled no, the device server shall discard the microcode on the next power on or hard reset.

[c] Entries in the Activate column are as follows. For modes labeled yes, the device server shall activate the microcode after completion of the final command in the WRITE BUFFER sequence (i.e., the series of WRITE BUFFER commands that downloads the microcode and activates it). For modes labeled optional, the device server may or may not activate the microcode image upon completion of the final command in the WRITE BUFFER sequence. For modes labeled no, the device server shall not activate the microcode upon completion of the final command in the WRITE BUFFER sequence.

[d] The application client delivers microcode in a vendor specific number of WRITE BUFFER commands (i.e., a WRITE BUFFER sequence). The device server should require that the microcode be delivered in a single command. The device server shall perform any required verification of the microcode prior to returning GOOD status for the final WRITE BUFFER command in a sequence (i.e., the WRITE BUFFER command delivering the last part of the microcode).

[e] The application client delivers microcode in one or more WRITE BUFFER commands, specifying a buffer ID and buffer offset in each command. If the device server does not receive the necessary WRITE BUFFER commands required to deliver the complete microcode before a logical unit reset occurs, an I_T nexus loss occurs, or a WRITE BUFFER command specifying a different download microcode mode is processed, the device server shall discard the new microcode. If the device server determines that it is processing the final WRITE BUFFER command (i.e., the WRITE BUFFER command delivering the last part of the microcode), it shall perform any required verification of the microcode prior to returning GOOD status for the command.

[f] Microcode downloaded with this mode is defined as deferred microcode.

When microcode is activated due to processing a WRITE BUFFER command with a mode that causes activation after processing (i.e., for modes 04h (see 6.39.5), 06h (see 6.39.7), and 0Fh (see 6.39.11)), the device server shall establish a unit attention condition (see SAM-4) for the initiator port associated with every I_T nexus except the I_T nexus on which the WRITE BUFFER command was received with the additional sense code set to MICROCODE HAS BEEN CHANGED. The application client on the I_T nexus on which the WRITE BUFFER command was transferred should respond to GOOD status for the WRITE BUFFER command the same way that it responds to a unit attention condition with an additional sense code set to MICROCODE HAS BEEN CHANGED (e.g., assume a hard reset has occurred).

When microcode is activated due to processing a WRITE BUFFER command with a mode that optionally causes activation after processing (i.e., for modes 05h (see 6.39.6) and 07h (see 6.39.8)), the device server shall establish a unit attention condition (see SAM-4) based on the setting of the ACTIVATE MICROCODE field in the Extended INQUIRY VPD page (see 7.8.7).

When microcode is activated due to power on or hard reset, the device server may establish a unit attention condition (see SAM-4) for the initiator port associated with every I_T nexus with the additional sense code set to MICROCODE HAS BEEN CHANGED in addition to the unit attention condition for the power on or hard reset.

When deferred microcode (see table 94) is activated due to a command defined by its command standard as causing deferred microcode to be activated (e.g., the FORMAT UNIT command and the START STOP UNIT command in SBC-3), the device server:

a) shall establish a unit attention condition (see SAM-4) for the initiator port associated with every I_T nexus with the additional sense code set to MICROCODE HAS BEEN CHANGED; and
b) may establish other unit attention condition(s) as defined for the command (e.g., CAPACITY DATA HAS CHANGED for the FORMAT UNIT command).

If new microcode is saved before deferred microcode is activated, the deferred microcode is not activated and the new saved microcode is considered deferred.

Table 95 summarizes how the WRITE BUFFER download microcode modes process the BUFFER ID field, the BUFFER OFFSET field, and the PARAMETER LIST LENGTH field in the WRITE BUFFER CDB.

Table 95 — WRITE BUFFER download microcode field processing

Mode	BUFFER ID field, BUFFER OFFSET field, and PARAMETER LIST LENGTH field processing
Download microcode and activate (i.e., 04h)	vendor specific
Download microcode, save, and activate (i.e., 05h)	vendor specific
Download microcode with offsets and activate (i.e., 06h)	6.39.7
Download microcode with offsets, save, and activate (i.e., 07h)	6.39.7
Download microcode with offsets, save, and defer activate (i.e., 0Eh)	6.39.7
Activate deferred microcode (i.e., 0Fh)	ignored

If the device server is unable to process a WRITE BUFFER command with a download microcode mode because of a vendor specific condition (e.g., the device server requires the microcode be delivered in order, and the BUFFER OFFSET field is not equal to the contents of the previous WRITE BUFFER command's BUFFER OFFSET field plus the contents of the previous WRITE BUFFER command's PARAMETER LIST LENGTH field), it shall terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to COMMAND SEQUENCE ERROR.

The MULTI I_T NEXUS MICROCODE DOWNLOAD field in the Extended INQUIRY Data VPD page (see 7.8.7) indicates how the device server handles concurrent attempts to download microcode using the WRITE BUFFER command download microcode modes (see table 94) from multiple I_T nexuses.

Table 96 — Multiple I_T nexus handling for WRITE BUFFER download microcode modes (part 1 of 2)

MULTI I_T NEXUS MICROCODE DOWNLOAD field [a]	Description
0h	The handling of concurrent WRITE BUFFER download microcode operations from multiple I_T nexus is vendor specific.
1h	For modes that download microcode (see table 94), the device server shall: a) if a WRITE BUFFER command with the BUFFER OFFSET field set to zero is received on any I_T nexus, the command shall be processed as described elsewhere in this subclause. This shall establish the I_T nexus for the WRITE BUFFER sequence, and cause any microcode downloaded on another I_T nexus to be discarded; b) if a WRITE BUFFER command with the BUFFER OFFSET field set to a non-zero value is received on the established I_T nexus for the WRITE BUFFER sequence, the command shall be processed as described elsewhere in this subclause; and c) if a WRITE BUFFER command with the BUFFER OFFSET field set to a non-zero value is received on an I_T nexus that is different from the established I_T nexus for the WRITE BUFFER sequence, the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to COMMAND SEQUENCE ERROR. For mode 0Fh (i.e., activate deferred microcode), the device server shall terminate any WRITE BUFFER command with mode 0Eh (i.e., download with offsets, save, and defer activate mode) received on an I_T nexus that is different from the established I_T nexus for the WRITE BUFFER sequence with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to COMMAND SEQUENCE ERROR.
2h	For modes that download microcode (see table 94), the device server shall allow concurrent sequences of WRITE BUFFER commands to be processed as described elsewhere in this subclause on more than one I_T nexus. For mode 0Fh (i.e., activate deferred microcode), the device server shall allow a WRITE BUFFER command with mode 0Eh (i.e., download with offsets, save, and defer activate mode) to be processed as described elsewhere in this subclause on an I_T nexus that is different from the established I_T nexus of the WRITE BUFFER sequence.

[a] This field is contained in the Extended INQUIRY Data VPD page (see 7.8.7).

Table 96 — Multiple I_T nexus handling for WRITE BUFFER download microcode modes (part 2 of 2)

MULTI I_T NEXUS MICROCODE DOWNLOAD field [a]	Description
3h	For modes that download microcode (see table 94), the device server shall: a) if a WRITE BUFFER command with the BUFFER OFFSET field set to zero is received on any I_T nexus, the command shall be processed as described elsewhere in this subclause. This shall establish the I_T nexus for the WRITE BUFFER sequence, and cause any microcode downloaded on another I_T nexus to be discarded; b) if a WRITE BUFFER command with the BUFFER OFFSET field set to a non-zero value is received on the established I_T nexus for the WRITE BUFFER sequence, the command shall be processed as described elsewhere in this subclause; and c) if a WRITE BUFFER command with the BUFFER OFFSET field set to a non-zero value is received on an I_T nexus that is different from the established I_T nexus for the WRITE BUFFER sequence, the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to COMMAND SEQUENCE ERROR. For mode 0Fh (i.e., activate deferred microcode), the device server shall allow a WRITE BUFFER command with mode 0Eh (i.e., download with offsets, save, and defer activate mode) to be processed as described elsewhere in this subclause on an I_T nexus that is different from the established I_T nexus for the WRITE BUFFER sequence.
4h to Fh	Reserved

[a] This field is contained in the Extended INQUIRY Data VPD page (see 7.8.7).

For all WRITE BUFFER command modes that download microcode (see table 94), the COMMAND SPECIFIC field (see 6.25.4.2) located in the command timeouts descriptor of the parameter data returned by the REPORT SUPPORTED OPERATION CODES command (see 6.25) indicates the maximum time that access to the SCSI device is limited or not possible through any SCSI ports associated with a logical unit that processes a WRITE BUFFER command that activates microcode.

6 Commands for all device types

6.1 Summary of commands for all device types

The operation codes for commands that apply to all device types are listed in table 97.

Table 97 — Commands for all device types (part 1 of 2)

Command name	Operation code	Type	Reference
ACCESS CONTROL IN	86h	O	8.3.2
ACCESS CONTROL OUT	87h	O	8.3.3
CHANGE ALIASES	A4h/0Bh [a]	O	6.2
EXTENDED COPY	83h	O	6.3
INQUIRY	12h	M	6.4
LOG SELECT	4Ch	O	6.5
LOG SENSE	4Dh	O	6.6
MANAGEMENT PROTOCOL IN	A3h/10h [a]	O	6.7
MANAGEMENT PROTOCOL OUT	A4h/10h [a]	O	6.8
MODE SELECT(6)	15h	C	6.9
MODE SELECT(10)	55h	C	6.10
MODE SENSE(6)	1Ah	C	6.11
MODE SENSE(10)	5Ah	C	6.12
PERSISTENT RESERVE IN	5Eh	C	6.13
PERSISTENT RESERVE OUT	5Fh	C	6.14
READ ATTRIBUTE	8Ch	O	6.15
READ BUFFER	3Ch	O	6.16
READ MEDIA SERIAL NUMBER	ABh/01h [a]	C	6.17
RECEIVE COPY RESULTS	84h	O	6.18
RECEIVE CREDENTIAL	7Fh/1800h [a]	O	6.19
RECEIVE DIAGNOSTIC RESULTS	1Ch	O	6.20
REPORT ALIASES	A3h/0Bh [a]	O	6.21
REPORT IDENTIFYING INFORMATION	A3h/05h [a]	O	6.22
REPORT LUNS	A0h	M	6.23
REPORT PRIORITY	A3h/0Eh [a]	O	6.24
REPORT SUPPORTED OPERATION CODES	A3h/0Ch [a]	O	6.25
REPORT SUPPORTED TASK MANAGEMENT FUNCTIONS	A3h/0Dh [a]	O	6.26
REPORT TARGET PORT GROUPS	A3h/0Ah [a]	O	6.27
REPORT TIMESTAMP	A3h/0Fh [a]	O	6.28

A numeric ordered listing of operation codes is provided in D.3.
Type Key: C = Command implementation is defined in the applicable command standard.
M = Command implementation is mandatory.
O = Command implementation is optional.
Z = Command implementation is defined in a previous standard.

[a] 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.
[b] The following operation codes are obsolete: 16h, 17h, 18h, 39h, 3Ah, 40h, 56h, and 57h.

Table 97 — Commands for all device types (part 2 of 2)

Command name	Operation code	Type	Reference
REQUEST SENSE	03h	C	6.29
SECURITY PROTOCOL IN	A2h	O	6.30
SECURITY PROTOCOL OUT	B5h	O	6.31
SEND DIAGNOSTIC	1Dh	C	6.32
SET IDENTIFYING INFORMATION	A4h/06h [a]	O	6.33
SET PRIORITY	A4h/0Eh [a]	O	6.34
SET TARGET PORT GROUPS	A4h/0Ah [a]	O	6.35
SET TIMESTAMP	A4h/0Fh [a]	O	6.36
TEST UNIT READY	00h	M	6.37
WRITE ATTRIBUTE	8Dh	O	6.38
WRITE BUFFER	3Bh	C	6.39
Obsolete [b]			

A numeric ordered listing of operation codes is provided in D.3.

Type Key: C = Command implementation is defined in the applicable command standard.
M = Command implementation is mandatory.
O = Command implementation is optional.
Z = Command implementation is defined in a previous standard.

[a] 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.

[b] The following operation codes are obsolete: 16h, 17h, 18h, 39h, 3Ah, 40h, 56h, and 57h.

6.2 CHANGE ALIASES command

6.2.1 CHANGE ALIASES command introduction

The CHANGE ALIASES command (see table 98) requests that the device server maintain and make changes to a list of associations between eight byte alias values and SCSI target device or SCSI target port designations. A designation contains a name and optional identifier information that specifies a SCSI target device or SCSI target port (see 6.2.2). The alias list may be queried by the application client via the REPORT ALIASES command (see 6.21). If the REPORT ALIASES command is supported, the CHANGE ALIASES command shall also be supported.

The CHANGE ALIASES command is a service action of the MAINTENANCE OUT command. Additional MAINTENANCE OUT service actions are defined in SCC-2 and in this standard. The MAINTENANCE OUT service actions defined in SCC-2 apply only to logical units that return a device type of 0Ch or the SCCS bit set to one in their standard INQUIRY data (see 6.4.2).

Table 98 — CHANGE ALIASES command

Bit / Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (A4h)							
1	Reserved			SERVICE ACTION (0Bh)				
2	Reserved							
•••								
5								
6	(MSB)	PARAMETER LIST LENGTH						
•••								
9								(LSB)
10	Reserved							
11	CONTROL							

The OPERATION CODE field is defined in 4.3.5.1 and shall be set as shown in table 98 for the CHANGE ALIASES command.

The SERVICE ACTION field is defined in 4.3.5.2 and shall be set as shown in table 98 for the CHANGE ALIASES command.

The PARAMETER LIST LENGTH field specifies the length in bytes of the parameter data that shall be transferred from the application client to the device server. A parameter list length value of zero specifies that no data shall be transferred and no changes shall be made in the alias list.

The CONTROL field is defined in SAM-5.

If the parameter list length results in the truncation of the header or any alias entry, then the device server shall make no changes to the alias list and terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to PARAMETER LIST LENGTH ERROR.

On successful completion of a CHANGE ALIASES command, the device server shall maintain an association of each assigned eight byte alias value to the SCSI target device or SCSI target port designation. These associations shall be cleared by a logical unit reset or I_T nexus loss. The device server shall maintain a separate alias list for each I_T nexus.

A CHANGE ALIASES command may add, change or remove entries from the alias list. Alias list entries not referenced in the CHANGE ALIASES parameter data shall not be changed.

NOTE 23 - An application client may use alias values to reference SCSI target devices or SCSI target ports in third party commands (e.g., EXTENDED COPY). The alias list provides a mechanism for eight byte third party identifier fields to reference a third party device or port whose name or addressing information is longer than eight bytes. (E.g., an application may use the CHANGE ALIASES command to establish an association between an alias value and a SCSI target device or target port designation. Then, it may send an EXTENDED COPY command containing in the parameter data an alias target descriptor (see 6.3.6.4) that includes this alias value. At the completion of the EXTENDED COPY command the application should clear this entry from the device server's alias list by sending a CHANGE ALIASES command that requests association of the alias value to a NULL DESIGNATION (see 6.2.4.2) alias format.)

If the device server has insufficient resources to make all requested changes to the alias list, then the device server shall make no changes to the alias list and shall terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INSUFFICIENT RESOURCES.

The parameter data for a CHANGE ALIASES command (see table 99) contains zero or more alias entries. If the device server processes a CHANGE ALIASES command that contains at least one alias entry while there exists any other enabled command that references an alias entry in the alias list, then the device server shall terminate the CHANGE ALIASES command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to OPERATION IN PROGRESS.

Table 99 — CHANGE ALIASES parameter list

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)							
•••				PARAMETER DATA LENGTH (n-3)				
3								(LSB)
4								
•••				Reserved				
7								
	Alias entry (or entries)							
8								
•••				Alias entry 0 (see 6.2.2)				
				⋮				
•••				Alias entry x (see 6.2.2)				
n								

The PARAMETER DATA LENGTH field should contain the number of bytes of attribute data and shall be ignored by the device server.

The format of an alias entry is described in 6.2.2.

6.2.2 Alias entry format

One alias entry (see table 100) describes one alias reported via the REPORT ALIASES command (see 6.21) or to be changed via the CHANGE ALIASES command.

Table 100 — Alias entry format

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)							
•••				ALIAS VALUE				
7								(LSB)
8				PROTOCOL IDENTIFIER				
9				Reserved				
10								
11				FORMAT CODE				
12				Reserved				
13								
14	(MSB)							
				DESIGNATION LENGTH (n-15)				
15								(LSB)
16								
•••				DESIGNATION				
n								

The ALIAS VALUE field contains the numeric alias value that the device server shall associate with the SCSI target device or target port specified by the values in the PROTOCOL IDENTIFIER, FORMAT CODE and DESIGNATION fields.

The PROTOCOL IDENTIFIER field (see table 101) specifies that the alias entry designation is independent of SCSI transport protocol or the SCSI transport protocol to which the alias entry applies.

Table 101 — Alias entry PROTOCOL IDENTIFIER field

Code	Description	Reference
00h to 0Fh	Protocol specific designation	7.6.2
10h to 7Fh	Reserved	
80h	Protocol independent designation	6.2.4
81h to FFh	Reserved	

The FORMAT CODE field contents combined with the PROTOCOL IDENTIFIER field contents defines the format of the DESIGNATION field. The subclauses that describe each PROTOCOL IDENTIFIER field usage (see table 101) define the applicable FORMAT CODE field values.

The DESIGNATION LENGTH field specifies the number of bytes of the DESIGNATION field. The DESIGNATION LENGTH value shall be a multiple of four.

The zero-padded (see 4.4.2) DESIGNATION field should designate a unique SCSI target device or target port using the following:

a) a SCSI device name or a target port name; and
b) optionally, one or more target port identifiers or SCSI transport protocol specific identifiers.

6.2.3 Alias designation validation

The device server shall not validate any designation at the time of processing either the REPORT ALIASES or CHANGE ALIASES command. Such validation shall occur only when the device server consults the alias list to resolve an alias to a designation in the context of third-party commands (e.g., EXTENDED COPY) or any other command that requires reference to the alias list.

If a designation identifies a unique SCSI target device or target port that is within a SCSI domain accessible to the device server, the designation is considered valid.

Based on the SCSI transport protocol specific requirements for a given designation format, a designation that does not identify a unique SCSI target device or target port within the SCSI domains accessible to the device server is considered invalid.

NOTE 24 - For example, a designation may be considered invalid if the device server has no ports on the SCSI domain of the designated SCSI target device or target port.

A designation having both SCSI name and SCSI port identifier information may be inconsistent if the device server is not able to access the named SCSI target device or target port through one or more of the names or identifiers in the designation. In such cases, the designation shall be treated as valid or invalid according to the SCSI transport protocol specific requirements.

Notes
25 For example, in FCP-4 both an N_Port and World Wide Name for a SCSI port may be given in a designation. The designation definition may require that the N_Port be that of the named port. In that case, the designation is invalid. Alternatively, the designation definition may view the N_Port as a hint for the named FC Port accessible to the device server through a different D_ID. In that case, the designation is valid and designate the named FC Port.
26 When only name information is provided in a designation, it is assumed that the device server has access to a mechanism for resolving names to identifiers. Access to such a service is SCSI transport protocol specific and vendor specific.

6.2.4 Alias entry protocol independent designations

6.2.4.1 Alias entry protocol independent designations overview

The protocol independent alias entry designations have a protocol identifier of 80h and one of the format codes shown in table 102.

Table 102 — Protocol independent alias entry format codes

Format Code	Name	Designation Length (bytes)	Designation Contents	Reference
00h	NULL DESIGNATION	0	none	6.2.4.2
01h to FFh	Reserved			

6.2.4.2 NULL DESIGNATION alias format

In response to an alias entry with the NULL DESIGNATION format, the device server shall remove the specified alias value from the alias list. Application clients should use the NULL DESIGNATION format in a CHANGE ALIASES command to remove an alias entry from the alias list when that alias entry is no longer needed. The NULL DESIGNATION format shall not appear in REPORT ALIASES parameter data.

6.3 EXTENDED COPY command

6.3.1 EXTENDED COPY command introduction

The EXTENDED COPY command (see table 103) provides a means to copy data from one set of logical units to another set of logical units or to the same set of logical units. The entity within a SCSI device that manages the command processing for EXTENDED COPY commands and RECEIVE COPY RESULTS commands (see 6.18) is the copy manager. The copy manager is responsible for copying data from the source devices to the destination devices. The copy source and destination devices are logical units that may reside in different SCSI devices or the same SCSI device. It is possible that the copy source device, copy destination device, and the copy manager are the same logical unit.

Table 103 — EXTENDED COPY command

Bit / Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (83h)							
1	Reserved							
•••								
9								
10	(MSB)			PARAMETER LIST LENGTH				
•••								
13								(LSB)
14	Reserved							
15	CONTROL							

The OPERATION CODE field is defined in 4.3.5.1 and shall be set as shown in table 103 for the EXTENDED COPY command.

Before the copy manager is instructed to move data, the application client controlling the data movement shall independently take any necessary actions required to prepare the copy source device and copy destination device for the EXTENDED COPY command. Such preparatory actions include but are not limited to:

a) loading tapes;
b) sending media changer commands;
c) sending MODE SELECT commands, including MODE SELECT commands that:
   A) disable reporting of recovered errors by a block copy target device by setting the PER bit to zero in the Error Recovery mode page (see SBC-3); and/or
   B) disable reporting of thin provisioning threshold events by a block copy target device by setting the TPERE bit to zero in the Error Recovery mode page;
d) sending reservation commands; and/or
e) sending tape positioning commands.

After all preparatory actions have been accomplished, the EXTENDED COPY command should be issued to the copy manager to start the data transfer.

The PARAMETER LIST LENGTH field specifies the length in bytes of the parameter data that shall be contained in the Data-Out Buffer. A parameter list length of zero specifies that copy manager shall not transfer any data or alter any internal state; this shall not be considered an error. If the parameter list length causes truncation of the parameter list in a target descriptor or segment descriptor, then no data shall be transferred and the EXTENDED COPY

command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to PARAMETER LIST LENGTH ERROR.

The CONTROL field is defined in SAM-5.

The EXTENDED COPY parameter list (see table 104) begins with a 16 byte header that contains the LIST IDENTIFIER field, the STR bit, the LIST ID USAGE field, the PRIORITY field, the length of the target descriptor list, the length of the segment descriptor list, and the length of the optional inline data. Immediately following the header is one or more target descriptors, followed by one or more segment descriptors, followed by any optional inline data.

Table 104 — EXTENDED COPY parameter list

Bit / Byte	7	6	5	4	3	2	1	0
0	LIST IDENTIFIER							
1	Reserved		STR	LIST ID USAGE		PRIORITY		
2	(MSB)	TARGET DESCRIPTOR LIST LENGTH (n-15)						
3								(LSB)
4	Reserved							
•••								
7								
8	(MSB)	SEGMENT DESCRIPTOR LIST LENGTH (m-n)						
11								(LSB)
12	(MSB)	INLINE DATA LENGTH (k-m)						
15								(LSB)
	Target descriptor(s)							
16	Target descriptor 0							
•••								
47								
	⋮							
n-31	Target descriptor x							
•••								
n								
	Segment descriptor(s)							
n+1	Segment descriptor 0 (See specific table for length.)							
•••								
n+1+l								
	⋮							
	Segment descriptor y (See specific table for length.)							
•••								
m								
m+1	Inline data							
•••								
k								

NOTE 27 - Unexpected results may occur when an application client fails to zero the reserved bytes in this parameter list. Copy managers should ensure that the reserved bytes 4 to 7 contain zeros.

The LIST IDENTIFIER field identifies the extended copy operation to the copy manager. The usage of the LIST IDENTIFIER field depends on the contents of the LIST ID USAGE field (see table 105).

A Sequential Striped (STR) bit set to one specifies to the copy manager that the majority of the disk references in the parameter list represent sequential access of several striped disks. This may be used by the copy manager to perform read operations from a copy source disk at any time and in any order during processing of an EXTENDED COPY command as described in 6.3.6.6. A STR bit set to zero specifies to the copy manager that disk references may not be sequential.

The LIST ID USAGE field (see table 105) specifies the usage of the LIST IDENTIFIER field.

Table 105 — LIST ID USAGE field (part 1 of 2)

Value	SNLID bit [a]	Meaning
00b	0 or 1	The LIST IDENTIFIER field contains a value that uniquely identifies this extended copy operation among all those being processed that were received on a specific I_T nexus. If the copy manager detects a duplicate list identifier value, 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 OPERATION IN PROGRESS. The list identifier value may be used in the RECEIVE COPY RESULTS command (see 6.18) to request status for a specific command sent on a specific I_T nexus. The copy manager shall hold data for retrieval by the application client using the RECEIVE COPY RESULTS command with the RECEIVE DATA service action (see 6.18.3) and specified by the segment descriptors.
01b		Reserved
10b	0 or 1	The LIST IDENTIFIER field contains a value that uniquely identifies this extended copy operation among all those being processed that were received on a specific I_T nexus. If the copy manager detects a duplicate list identifier value, 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 OPERATION IN PROGRESS. The list identifier value may be used in the RECEIVE COPY RESULTS command (see 6.18) to request status for a specific command sent on a specific I_T nexus. The copy manager may discard all data accessible to the application client via the RECEIVE COPY RESULTS command with the RECEIVE DATA service action. If the application client requests delivery of data that has been discarded as a result of the LIST ID USAGE field being set to 10b, then the copy manager shall respond as if the EXTENDED COPY command has not been processed.

[a] The SNLID bit in the parameter data for the OPERATING PARAMETERS service action (see 6.18.4) of the RECEIVE COPY RESULTS command.

Table 105 — LIST ID USAGE field (part 2 of 2)

Value	SNLID bit [a]	Meaning
11b	0	The device server shall: a) 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; or b) process the command as if the LIST ID USAGE field is set to 10b.
	1	If the LIST IDENTIFIER 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 PARAMETER LIST. The copy manager shall discard all data accessible to the application client via the RECEIVE COPY RESULTS command. If the application client requests delivery of data that has been discarded as a result the LIST ID USAGE field being set to 11b then the copy manager shall respond as if the EXTENDED COPY command has not been processed. If any of the segment descriptor type codes that require data to be held for the application client are present, 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 PARAMETER LIST, and no segment descriptors shall be processed (see 6.3.2).

[a] The SNLID bit in the parameter data for the OPERATING PARAMETERS service action (see 6.18.4) of the RECEIVE COPY RESULTS command.

The PRIORITY field establishes the priority of data transfer operations resulting from this EXTENDED COPY command relative to data transfer operations resulting from other commands being processed by the device server contained within the same logical unit as the copy manager. All commands other than copy commands have a priority of 1h. Priority 0h is the highest priority, with increasing priority values indicating lower priorities.

The TARGET DESCRIPTOR LIST LENGTH field contains the length in bytes of the target descriptor list that immediately follows the parameter list header. The number of target descriptors equals the length in bytes of the target descriptor list divided by 32.

An EXTENDED COPY command may reference one or more copy target devices (i.e., the name given by the EXTENDED COPY command description to copy source and/or the destination devices). Each copy target device is described by a target descriptor. All target descriptors have their formats specified by an EXTENDED COPY descriptor code. A copy manager may not support all target descriptor formats, however, the copy manager shall list all target descriptor formats supported in response to the RECEIVE COPY RESULTS command with OPERATING PARAMETERS service action (see 6.18.4). See 6.3.6 for a detailed description of the target descriptors.

Segment descriptors reference target descriptors by their position, or index, in the target descriptor list. The index for a target descriptor is computed by subtracting 16 from the starting byte number for the target descriptor in the parameter data and dividing the result by 32. The maximum number of target descriptors permitted within a parameter list is indicated by the MAXIMUM TARGET COUNT field in the copy manager's operating parameters (see 6.18.4). If the number of target descriptors exceeds the allowed number, the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to TOO MANY TARGET DESCRIPTORS.

The SEGMENT DESCRIPTOR LIST LENGTH field contains the length in bytes of the segment descriptor list that follows the target descriptors. See 6.3.7 for a detailed description of the segment descriptors. The maximum number of segment descriptors permitted within a parameter list is indicated by the MAXIMUM SEGMENT COUNT field in the copy manager's operating parameters (see 6.18.4). If the number of segment descriptors exceeds the allowed number, the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to TOO MANY SEGMENT DESCRIPTORS.

The maximum length of the target and segment descriptors permitted within a parameter list is indicated by the MAXIMUM DESCRIPTOR LIST LENGTH field in the copy manager's operating parameters (see 6.18.4). If the combined length of the target and segment descriptors exceeds the allowed value, the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to PARAMETER LIST LENGTH ERROR.

The INLINE DATA LENGTH field contains the number of bytes of inline data, after the last segment descriptor. A value of zero specifies that no inline data is present.

The copy manager shall move data from the copy source devices to the copy destination devices as specified by the segment descriptors. The specific commands issued by the copy manager to the copy source and destination devices while processing the segment descriptors is vendor specific. Upon completion of an EXTENDED COPY command with GOOD status, the copy source and destination devices, particularly stream devices, shall be positioned at deterministic locations such that the device may be repositioned to the same location by the application client with appropriate commands.

6.3.2 Errors detected before starting processing of the segment descriptors

Errors may occur during processing of an EXTENDED COPY command before the first segment descriptor is processed. These errors include CRC or parity errors while transferring the EXTENDED COPY command, invalid parameters in the CDB or parameter data, invalid segment descriptors, and inability of the copy manager to continue operating. In the event of such an exception condition, the copy manager shall:

a) terminate the EXTENDED COPY command with CHECK CONDITION status; and
b) set the VALID bit in the sense data to zero. The sense key shall contain the sense key code describing the exception condition (i.e., not COPY ABORTED).

6.3.3 Errors detected during processing of segment descriptors

Errors may occur after the copy manager has begun processing segment descriptors. These errors include invalid parameters in segment descriptors, invalid segment descriptors, unavailable targets referenced by target descriptors, inability of the copy manager to continue operating, and errors reported by source or destination copy target devices. If the copy manager receives CHECK CONDITION status from one of the copy target devices, it shall recover the sense data associated with the exception condition and clear any ACA condition associated with the CHECK CONDITION status.

If it is not possible to complete processing of a segment because the copy manager is unable to establish communications with a copy target device, because the copy target device does not respond to INQUIRY, or because the data returned in response to INQUIRY indicates an unsupported logical unit, then the EXTENDED COPY command shall be terminated with CHECK CONDITION status, with the sense key set to COPY ABORTED, and the additional sense code set to COPY TARGET DEVICE NOT REACHABLE.

If it is not possible to complete processing of a segment because the data returned in response to an INQUIRY command indicates a device type that does not match the type in the target descriptor, then the EXTENDED COPY command shall be terminated with CHECK CONDITION status, with the sense key set to COPY ABORTED, and the additional sense code set to INCORRECT COPY TARGET DEVICE TYPE.

If the copy manager has issued a command other than INQUIRY to a copy target device while processing an EXTENDED COPY command and the copy target device either fails to respond with status or responds with status other than BUSY, TASK SET FULL, ACA ACTIVE, or RESERVATION CONFLICT, then the condition shall be considered a copy target device command failure. In response to a copy target device command failure the EXTENDED COPY command shall be terminated with CHECK CONDITION status, with the sense key set to COPY ABORTED, and the additional sense code set to THIRD PARTY DEVICE FAILURE.

If a copy target device completes a command from the copy manager with a status of BUSY, TASK SET FULL, ACA ACTIVE, or RESERVATION CONFLICT, the copy manager shall either retry the command or terminate the EXTENDED COPY command as a copy target device command failure.

NOTES

28 The copy manager is assumed to employ a vendor specific retry policy that minimizes time consuming and/or fruitless repetition of retries.

29 RESERVATION CONFLICT status is listed only to give the copy manager leeway in multi-port cases. The copy manager may have multiple initiator ports that are capable of reaching a copy target device, and a persistent reservation may restrict access to a single I_T nexus. The copy manager may need to try access from multiple initiator ports to find the correct I_T nexus.

If a copy target device responds to an input or output operation with a GOOD status but less data than expected is transferred, then the EXTENDED COPY command shall be terminated with CHECK CONDITION status, with the sense key set to COPY ABORTED, and the additional sense code set to COPY TARGET DEVICE DATA UNDERRUN. If an overrun is detected, then the EXTENDED COPY command shall be terminated with CHECK CONDITION status, with the sense key set to COPY ABORTED, and the additional sense code set to COPY TARGET DEVICE DATA OVERRUN.

Following an exception condition detected during segment descriptor processing:

a) the copy manager shall terminate the EXTENDED COPY command with CHECK CONDITION status, with the sense key set to COPY ABORTED;
b) the copy manager shall indicate the segment that was being processed at the time of the exception by writing the segment number to the third and fourth bytes of the COMMAND-SPECIFIC INFORMATION field. The segment number is based on the relative position of the segment descriptor in the EXTENDED COPY parameter list (i.e., the first segment descriptor in the parameter list is assigned descriptor number zero, the second is assigned one, etc.);
c) if any data has been written to the destination for the segment being processed at the time the error occurred, the residual for the segment shall be placed in the INFORMATION field, and the VALID bit shall be set to one. The residual count shall be reported in bytes if the peripheral device type in the destination target descriptor is 03h (i.e., processor device), and in destination device blocks for all other device type codes. The residual count shall be computed by subtracting the number of bytes or blocks successfully written during the processing of the current segment from the number of bytes or blocks which would have been written if all commands had completed with GOOD status and all READ commands had returned the full data length requested. When computing the residual count, the copy manager shall include only the results of commands successfully completed by a copy destination device (i.e., commands completed by a copy destination device with GOOD status or with CHECK CONDITION status and the EOM bit set to one in the sense data). If the copy manager has used out of order transfers, the residual count shall be based only on the contiguous successfully completed transfers starting at relative byte zero of the segment (i.e., any successfully completed transfers farther from relative byte zero than the first incomplete or unsuccessful transfer shall not contribute to the computation of the residual count). If no data has been written to the destination for the segment being processed at the time the error occurred, then the VALID bit shall be set to zero and the contents of the INFORMATION field are not defined. Segment descriptors that do not specify a transfer count shall not have a valid residual count returned;

d) if the exception condition is reported by the copy source device, then:
   A) if fixed format sense data (see 4.5.3) is being returned, then the first byte of the COMMAND-SPECIFIC INFORMATION field shall be set to the starting byte number, relative to the first byte of sense data, of an area that contains the status byte and sense data delivered to the copy manager by the copy source device. The status byte and sense data shall not be modified by the copy manager or device server. A zero value indicates that no status byte and sense data is being returned for the copy source device; or
   B) if descriptor format sense data (see 4.5.2) is being returned, then the status byte and sense data delivered to the copy manager by the copy source device shall be returned in a forwarded additional sense data descriptor (see 4.5.2.7) with the SOURCE field set to 1h. The status byte shall not be modified by the copy manager or device server. The sense data may be truncated by the copy manager or device server but shall not be modified in any other way. If the sense data is truncated, then the FSDT bit in the forwarded additional sense data descriptor shall be set to one;
e) if the exception condition is reported by the copy destination device, then:
   A) if fixed format sense data is being returned, then the second byte of the COMMAND-SPECIFIC INFORMATION field shall be set to the starting byte number, relative to the first byte of sense data, of an area that contains the status byte and sense data delivered to the copy manager by the copy destination device. The status byte and sense data shall not be modified by the copy manager or device server. A zero value indicates that no status byte and sense data is being returned for the copy destination device; or
   B) if descriptor format sense data is being returned, then the status byte and sense data delivered to the copy manager by the copy destination device shall be returned in a forwarded additional sense data descriptor with the SOURCE field set to 2h. The status byte shall not be modified by the copy manager or device server. The sense data may be truncated by the copy manager or device server but shall not be modified in any other way. If the sense data is truncated, then the FSDT bit in the forwarded additional sense data descriptor shall be set to one;
f) if segment processing is terminated because a copy target device is unreachable or as the result of a failure in a command sent to a copy target device, then the SENSE-KEY SPECIFIC field shall be set as described in 4.5.2.4.5, with the FIELD POINTER field indicating the first byte of the target descriptor that identifies the copy target device;
g) if, during the processing of a segment descriptor, the copy manager detects an error in the segment descriptor, then the SENSE-KEY SPECIFIC field shall be set as described in 4.5.2.4.5, with the FIELD POINTER field indicating the byte in error. The FIELD POINTER field may be used to indicate an offset into either the parameter data or the segment descriptor. The SD bit is used to differentiate between these two cases. The SD bit shall be set to zero to indicate the FIELD POINTER field contains an offset from the start of the parameter data. The SD bit shall be set to one to indicate the FIELD POINTER field contains an offset from the start of the segment descriptor; and
h) if the LIST ID USAGE field is set to 00b or 10b, the copy manager shall preserve information for the FAILED SEGMENT DETAILS service action of the RECEIVE COPY RESULTS command (see 6.18.5). The information shall be discarded as described in 6.18.5.

6.3.4 Abort task management functions

When a task manager processes an ABORT TASK, ABORT TASK SET, or CLEAR TASK SET task management function that terminates an EXTENDED COPY command, the copy manager shall ensure that all commands and data transfers generated by the terminated EXTENDED COPY command have been terminated and are no longer transferring data before allowing the task manager to complete the task management function. This requirement shall also apply to the processing the PREEMPT AND ABORT service action of the PERSISTENT RESERVE OUT command as described in 5.9.11.5.

6.3.5 Descriptor type codes

Target descriptors, the target descriptor extension, and segment descriptors share a single set of code values (see table 106) that identify the type of descriptor.

Table 106 — EXTENDED COPY descriptor type codes

Descriptor type	Description	Reference
00h to BFh	Segment descriptors	6.3.7
C0h to DFh	Vendor specific descriptors	
E0h to FEh	Target descriptors	6.3.6
FFh [a]	Target descriptor extension	6.3.6.2

[a] Use of this descriptor type code is reserved except in byte 32 of a 64 byte target descriptor (e.g., the IPv6 target descriptor described in 7.6.3.9).

6.3.6 Target descriptors

6.3.6.1 Target descriptors introduction

The descriptor type code (see table 106) values assigned to target descriptors are shown in table 107.

Table 107 — EXTENDED COPY target descriptor type codes

Descriptor type	Description	Size (bytes)	Reference
E0h	Fibre Channel N_Port_Name target descriptor	32	7.6.3.2
E1h	Fibre Channel N_Port_ID target descriptor	32	7.6.3.3
E2h	Fibre Channel N_Port_ID with N_Port_Name checking target descriptor	32	7.6.3.4
E3h	Parallel Interface T_L target descriptor	32	7.6.3.5
E4h	Identification descriptor target descriptor	32	6.3.6.3
E5h	IPv4 target descriptor	32	7.6.3.8
E6h	Alias target descriptor	32	6.3.6.4
E7h	RDMA target descriptor	32	7.6.3.7
E8h	IEEE 1394 EUI-64 target descriptor	32	7.6.3.6
E9h	SAS Serial SCSI Protocol target descriptor	32	7.6.3.10
EAh	IPv6 target descriptor	64	7.6.3.9
EBh	IP Copy Service target descriptor	64	6.3.6.5
EBh to FEh	Reserved for target descriptors		

All target descriptors (see table 108) are 32 bytes or 64 bytes in length and begin with a four-byte header containing the DESCRIPTOR TYPE CODE field that identifies the format of the descriptor. Support for each target descriptor format is optional. If a copy manager receives an unsupported descriptor type code in a target descriptor, the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to UNSUPPORTED TARGET DESCRIPTOR TYPE CODE.

Table 108 — Target descriptor format

Bit / Byte	7	6	5	4	3	2	1	0
0	DESCRIPTOR TYPE CODE (E0h to FFh)							
1	LU ID TYPE		NUL	PERIPHERAL DEVICE TYPE				
2	(MSB)	RELATIVE INITIATOR PORT IDENTIFIER						
3								(LSB)
4	Target descriptor parameters							
•••								
27								
28	Device type specific parameters							
•••								
31								

The DESCRIPTOR TYPE CODE field is described in 6.3.5.

The LU ID TYPE field (see table 109) specifies the interpretation of the LU IDENTIFIER field in target descriptors that contain a LU IDENTIFIER field.

Table 109 — LU ID TYPE field

Code	LU IDENTIFIER field contents	Reference
00b	Logical Unit Number	SAM-4
01b	Proxy Token	8.3.1.6.2
10b to 11b	Reserved	

Support for LU ID type codes other than 00b is optional. If a copy manager receives an unsupported LU ID type code, 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.

If the LU ID TYPE field specifies that the LU IDENTIFIER field contains a logical unit number, then the LU IDENTIFIER field specifies the logical unit within the SCSI device specified by other fields in the target descriptor that shall be the source or destination for EXTENDED COPY operations.

If the LU ID TYPE field specifies that the LU IDENTIFIER field contains a proxy token (see 8.3.1.6.2), then the copy manager shall use the LU IDENTIFIER field contents to obtain proxy access rights to the logical unit associated with the proxy token. The logical unit number that represents the proxy access rights shall be the source or destination for EXTENDED COPY operations.

The copy manager should obtain a LUN value for addressing this logical unit by sending an ACCESS CONTROL OUT command with ASSIGN PROXY LUN service action (see 8.3.3.11) to the access controls coordinator of the SCSI target device that is identified by other fields in the target descriptor. The copy manager shall use a LUN assigned on the basis of a proxy token only for those commands that are necessary for the processing of the EXTENDED COPY command whose parameter data contains the proxy token. When the copy manager has

completed EXTENDED COPY commands involving a proxy token, the copy manager should release the LUN value using an ACCESS CONTROL OUT command with RELEASE PROXY LUN service action (see 8.3.3.12).

EXTENDED COPY access to proxy logical units is to be accomplished only via LU ID type 01b. If the copy manager receives a target descriptor containing LU ID type 00b and a logical unit number matching a LUN value that the copy manager has obtained using an ACCESS CONTROL OUT command with ASSIGN PROXY LUN service action, then the EXTENDED COPY command shall be terminated with CHECK CONDITION status, with the sense key set to COPY ABORTED, and the additional sense code set to COPY TARGET DEVICE NOT REACHABLE.

A null device (NUL) bit set to zero specifies that the target descriptor identifies a SCSI device that is expected to respond to an INQUIRY command and to which data movement commands may be sent. A NUL bit set to one specifies that the descriptor identifies a null device that is not expected to be the recipient of any SCSI commands. If NUL is one, bytes 4 to 27 of the target descriptor shall be ignored. If the processing required by a segment descriptor necessitates sending a command to a copy target device whose target descriptor has the NUL bit set to one, then the EXTENDED COPY command shall be terminated as if an unreachable copy target device had been encountered (see 6.3.3).

NOTE 30 - Target descriptors with the NUL bit set to one are useful for processing the residual data from previous segment descriptors without affecting any media. (E.g., a segment descriptor of type 06h (stream device to discard) with a byte count of zero, CAT equal to zero, and a null source target descriptor with PAD equal to one may be used to discard all residual data.)

The PERIPHERAL DEVICE TYPE field is described in 6.4.2. The value in the DESCRIPTOR TYPE CODE field determines the format of the target descriptor parameters that follow the four-byte header and precede the device type specific parameters. The values in the DESCRIPTOR TYPE CODE field are listed in table 106.

The value in the PERIPHERAL DEVICE TYPE field determines the format of the device type specific parameters that follow the target descriptor parameters. The device type specific parameters convey information specific to the type of device identified by the target descriptor.

Table 110 lists the peripheral device type code values having formats defined for the device type specific parameters in a target descriptor. Peripheral device types with code values not listed in table 110 are reserved in the EXTENDED COPY parameter list.

Table 110 — Device type specific parameters in target descriptors

Peripheral Device Type	Reference	Description	Shorthand
00h, 04h, 05h, 07h, and 0Eh	6.3.6.6	Block devices	Block
01h	6.3.6.7	Sequential-access devices	Stream or Tape
03h	6.3.6.8	Processor devices	Stream

The RELATIVE INITIATOR PORT IDENTIFIER field specifies the relative port identifier (see 3.1.127) of the initiator port within the SCSI device that the copy manager shall use to access the logical unit described by the target descriptor, if such access requires use of an initiator port (i.e., if the logical unit is in the same SCSI device as the copy manager, the RELATIVE INITIATOR PORT IDENTIFIER field is ignored). A RELATIVE INITIATOR PORT IDENTIFIER field set to zero specifies that the copy manager may use any initiator port or ports within the SCSI device.

The copy manager may, as part of processing a segment descriptor, verify the information in a target descriptor's device specific fields. However, when verifying the information, the copy manager shall not issue any commands that change the position of read/write media on the copy target device without returning the media to its original position. Any errors encountered while verifying the information shall be handled as described in 6.3.3.

6.3.6.2 The target descriptor extension

A 32 byte target descriptor is extended to 64 bytes by appending the target descriptor extension (see table 111) to it. The target descriptor extension is 32 bytes in length and begins with a one-byte header containing the descriptor type code value that identifies the descriptor as a target descriptor extension (i.e., FFh).

Table 111 — Target descriptor extension format

Bit / Byte	7	6	5	4	3	2	1	0
0	EXTENSION DESCRIPTOR TYPE CODE (FFh)							
1	Target descriptor specific information							
•••								
31								

The EXTENSION DESCRIPTOR TYPE CODE field is a descriptor type code value (see 6.3.5), and shall be set to the value shown in table 111 for the target descriptor extension.

The target descriptor specific information contains data that extends the target descriptor whose index is one less than that of the target descriptor extension.

If a target descriptor index in a segment descriptor references a target descriptor extension (i.e, a descriptor with the descriptor type code set to FFh), then the EXTENDED COPY command shall be terminated with CHECK CONDITION status, with the sense key set to COPY ABORTED, and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

6.3.6.3 Identification descriptor target descriptor format

The target descriptor format shown in table 112 instructs the copy manager to locate a SCSI target device and logical unit that returns a Device Identification VPD page (see 7.8.6) containing an Identification descriptor having the specified CODE SET, ASSOCIATION, DESIGNATOR TYPE, IDENTIFIER LENGTH, and IDENTIFIER field values. The copy manager may use any N_Port, target port identifier and logical unit number values that result in matching VPD field values to address the logical unit. If multiple target port identifiers and logical unit number combinations access matching VPD field values, the copy manager may use any combination to address the logical unit and shall try other combinations in the event that one combination becomes non-operational during the processing of an EXTENDED COPY command.

Table 112 — Identification descriptor target descriptor format

Bit / Byte	7	6	5	4	3	2	1	0
0	DESCRIPTOR TYPE CODE (E4h)							
1	LU ID TYPE		NUL	PERIPHERAL DEVICE TYPE				
2	(MSB)	RELATIVE INITIATOR PORT IDENTIFIER						
3								(LSB)
4	Reserved				CODE SET			
5	Reserved		ASSOCIATION		DESIGNATOR TYPE			
6	Reserved							
7	DESIGNATOR LENGTH (n-7)							
8	DESIGNATOR							
•••								
n								
n+1	Reserved							
•••								
27								
28	Device type specific parameters							
•••								
31								

The DESCRIPTOR TYPE CODE field, PERIPHERAL DEVICE TYPE field, NUL bit, RELATIVE INITIATOR PORT IDENTIFIER field, and the device type specific parameters are described in 6.3.6.1.

The LU ID TYPE field is reserved for this target descriptor.

The CODE SET field contains a code set enumeration (see 4.4.3) that indicates the format of the DESIGNATOR field.

The contents of the ASSOCIATION field, DESIGNATOR TYPE field, DESIGNATOR LENGTH field, and DESIGNATOR field are specified in 7.8.6.

The designator length shall be 20 or less. If the designator length is 20, there shall be no reserved bytes between the target descriptor parameters and the device type specific parameters.

Some combinations of code set, association, designator type, designator length and designator do not uniquely identify a logical unit to serve as a copy target device. The behavior of the copy manager when such combinations are received is unpredictable.

6.3.6.4 Alias target descriptor format

The target descriptor format shown in table 113 instructs the copy manager to locate a SCSI target port and logical unit using the alias list (see 3.1.8) designation associated with the specified alias value. The alias list is maintained using the CHANGE ALIASES command (see 6.2).

Table 113 — Alias target descriptor format

Bit / Byte	7	6	5	4	3	2	1	0
0	DESCRIPTOR TYPE CODE (E6h)							
1	LU ID TYPE		NUL	PERIPHERAL DEVICE TYPE				
2	(MSB)	RELATIVE INITIATOR PORT IDENTIFIER						
3								(LSB)
4	LU IDENTIFIER							
•••								
11								
12	ALIAS VALUE							
•••								
19								
20	Reserved							
•••								
27								
28	Device type specific parameters							
•••								
31								

The DESCRIPTOR TYPE CODE field, PERIPHERAL DEVICE TYPE field, NUL bit, RELATIVE INITIATOR PORT IDENTIFIER field, and the device type specific parameters are described in 6.3.6.1.

The ALIAS VALUE field specifies an alias value in the alias list as managed by the CHANGE ALIASES command (see 6.2) and maintained by the device server.

When the device server first processes an alias target descriptor, it shall check the value of the ALIAS VALUE field for a corresponding entry in the alias list. If the value is not in the alias list or the device server is unable to validate the designation (see 6.2.3) associated with the alias value, the command shall be terminated because the copy target device is unavailable (see 6.3.3). An application client generating EXTENDED COPY commands that include alias target descriptors in the parameter data is responsible for providing a valid entry in the alias list using the CHANGE ALIASES command (see 6.2) prior to sending the EXTENDED COPY command.

6.3.6.5 IP Copy Service target descriptor

The target descriptor format shown in table 114 instructs the copy manager to communicate with the copy service identified in the IP ADDRESS field to locate the copy target device that returns a Device Identification VPD page (see 7.8.6) containing an Identification descriptor having the specified CODE SET field, ASSOCIATION field, DESIGNATOR TYPE field, IDENTIFIER LENGTH field, and IDENTIFIER field values.

The protocol used by the copy manager to communicate with the copy service is vendor specific.

Table 114 — IP Copy Service EXTENDED COPY target descriptor format

Bit / Byte	7	6	5	4	3	2	1	0
0	DESCRIPTOR TYPE CODE (EBh)							
1	LU ID TYPE		NUL	PERIPHERAL DEVICE TYPE				
2	Reserved							IPTYPE
3 ••• 11	Reserved							
12 ••• 27	(MSB) COPY SERVICE IP ADDRESS (LSB)							
28 31	Device type specific parameters							
32	EXTENSION DESCRIPTOR TYPE CODE (FFh)							
33 35	Reserved							
36 37	(MSB) COPY SERVICE PORT NUMBER (LSB)							
38 39	(MSB) COPY SERVICE INTERNET PROTOCOL NUMBER (LSB)							
40	Reserved				CODE SET			
41	Reserved		ASSOCIATION		DESIGNATOR TYPE			
42	Reserved							
43	DESIGNATOR LENGTH (n-43)							
44 ••• n	DESIGNATOR							
n+1 ••• 63	Reserved							

The DESCRIPTOR TYPE CODE field, PERIPHERAL DEVICE TYPE field, NUL bit, RELATIVE INITIATOR PORT IDENTIFIER field, and the device type specific parameters are described in 6.3.6.1.

The LU ID TYPE field is reserved for this target descriptor.

If the IPTYPE bit is set to zero, the COPY SERVICE IP ADDRESS field shall contain a zero-padded IPv4 address (see RFC 791). If the IPTYPE bit is set to one, the COPY SERVICE IP ADDRESS field shall contain a unicast IPv6 address (see RFC 4291).

The COPY SERVICE IP ADDRESS field contains the IP address of the copy service in the format specified by the IPTYPE bit.

The EXTENSION DESCRIPTOR TYPE CODE field is described in 6.3.6.2. If the EXTENSION DESCRIPTOR TYPE CODE field does not contain the value shown in table 114, then the EXTENDED COPY command shall be terminated with CHECK CONDITION status, with the sense key set to COPY ABORTED, and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

The COPY SERVICE PORT NUMBER field shall contain the TCP port number (see 3.1.180).

The COPY SERVICE INTERNET PROTOCOL NUMBER field shall contain an Internet protocol number (see 3.1.80).

The contents of the COPY SERVICE IP ADDRESS field, the COPY SERVICE PORT NUMBER field, and the COPY SERVICE INTERNET PROTOCOL NUMBER field may be obtained using the network service descriptor with the SERVICE TYPE field set to 06h (i.e., copy service) in Management Network Addresses VPD page (see 7.8.8) returned by the logical unit that returns a Device Identification VPD page (see 7.8.6) containing an Identification descriptor having the specified CODE SET field, ASSOCIATION field, DESIGNATOR TYPE field, IDENTIFIER LENGTH field, and IDENTIFIER field values.

The CODE SET field contains a code set enumeration (see 4.4.3) that indicates the format of the DESIGNATOR field.

The contents of the ASSOCIATION field, DESIGNATOR TYPE field, DESIGNATOR LENGTH field, and DESIGNATOR field are specified in 7.8.6.

The designator length shall be 20 or less. If the designator length is 20, there shall be no reserved bytes following the DESIGNATOR field.

Some combinations of code set, association, designator type, designator length and designator do not uniquely identify a logical unit to serve as a copy target device. The behavior of the copy manager when such combinations are received is unpredictable.

6.3.6.6 Device type specific target descriptor parameters for block device types

The format for the device type specific target descriptor parameters for block device types (i.e., device type code values 00h, 04h, 05h, 07h, and 0Eh) is shown in table 115.

Table 115 — Device type specific target descriptor parameters for block device types

Bit / Byte	7	6	5	4	3	2	1	0
28	Reserved					PAD	Reserved	
29	(MSB)							
30				DISK BLOCK LENGTH				
31								(LSB)

The PAD bit is used in conjunction with the CAT bit (see 6.3.7.1) in the segment descriptor to determine what action should be taken when a segment of the copy does not fit exactly into an integer number of destination blocks (see 6.3.7.2).

The DISK BLOCK LENGTH field contains the number of bytes in a disk block, excluding any protection information (see SBC-2), for the logical device being addressed.

The copy manager may read ahead from sources of block device type (i.e., the copy manager may perform read operations from a source disk at any time and in any order during processing of an EXTENDED COPY command, provided that the relative order of writes and reads on the same blocks within the same target descriptor does not differ from their order in the segment descriptor list).

6.3.6.7 Device type specific target descriptor parameters for sequential-access device types

The format for the device type specific target descriptor parameters for the sequential-access device type (i.e., device type code value 01h) is shown in table 116.

Table 116 — Device type specific target descriptor parameters for sequential-access device types

Bit / Byte	7	6	5	4	3	2	1	0
28	Reserved					PAD	Reserved	FIXED
29	(MSB)	STREAM BLOCK LENGTH						
30								
31								(LSB)

The contents of the FIXED bit and STREAM BLOCK LENGTH field are combined with the STREAM DEVICE TRANSFER LENGTH FIELD in the segment descriptor to determine the length of the stream read or write operation as specified in table 117.

Table 117 — Stream device transfer lengths

FIXED bit	STREAM BLOCK LENGTH field	Description
0	000000h	Use variable length reads or writes. The number of bytes for each read or write is specified by the STREAM DEVICE TRANSFER LENGTH field in the segment descriptor.
0	000001h to FFFFFFh	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.
1	000000h	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
1	000001h to FFFFFFh	Use fixed record length reads or writes. The number of bytes for each read or write shall be the product of the STREAM BLOCK LENGTH field and the STREAM DEVICE TRANSFER LENGTH field in the segment descriptor.

The PAD bit is used in conjunction with the CAT bit (see 6.3.7.2) in the segment descriptor to determine what action should be taken when a segment of the copy does not fit exactly into an integer number of destination blocks (see 6.3.7.2).

All read commands issued to sequential-access type devices shall have the SILI bit equal to zero.

The copy manager shall not read ahead from sources of stream device type (i.e., the read operations required by a segment descriptor for which the source is a stream device shall not be started until all write operations for previous segment descriptors have completed).

6.3.6.8 Device type specific target descriptor parameters for processor device types

The format for the device type specific target descriptor parameters for the processor device type (i.e., device type code value 03h) is shown in table 118.

Table 118 — Device type specific target descriptor parameters for processor device types

Bit / Byte	7	6	5	4	3	2	1	0
28	Reserved					PAD	Reserved	
29	Reserved							
•••								
31								

The PAD bit is used in conjunction with the CAT bit (see 6.3.7.2) in the segment descriptor to determine what action should be taken when a segment of the copy does not fit exactly into an integer number of SEND or RECEIVE commands (see 6.3.7.2).

When the processor device is a source, the number of bytes to be transferred by a SEND command shall be specified by STREAM DEVICE TRANSFER LENGTH field in the segment descriptor. When the processor device is a destination, the number of bytes to be transferred by a RECEIVE command shall be specified by STREAM DEVICE TRANSFER LENGTH field in the segment descriptor.

6.3.7 Segment descriptors

6.3.7.1 Segment descriptors introduction

The descriptor type code (see table 106) values assigned to segment descriptors are shown in table 119.

Table 119 — EXTENDED COPY segment descriptor type codes (part 1 of 2)

Descriptor type code	Reference	Description [a]	Shorthand [a]
00h	6.3.7.3	Copy from block device to stream device	block→ stream
01h	6.3.7.4	Copy from stream device to block device	stream→ block
02h	6.3.7.5	Copy from block device to block device	block→ block
03h	6.3.7.6	Copy from stream device to stream device	stream→ stream

[a] Block devices are those with peripheral device type codes 0h (i.e, direct access block), 5h (i.e., CD/DVD), and Eh (i.e., simplified direct-access). Stream devices are those devices with peripheral device type codes 1h (i.e., sequential-access) and 3h (i.e., processor). Sequential-access stream (indicated by the term tape in the shorthand column) devices are those with peripheral device type code 1h. See 6.4.2 for peripheral device type code definitions.

[b] The application client shall use the RECEIVE COPY RESULTS with a RECEIVE DATA service action to retrieve data held for it by the copy manager (see 6.18.3).

Table 119 — EXTENDED COPY segment descriptor type codes (part 2 of 2)

Descriptor type code	Reference	Description [a]	Shorthand [a]
04h	6.3.7.7	Copy inline data to stream device	inline→ stream
05h	6.3.7.8	Copy embedded data to stream device	embedded→ stream
06h	6.3.7.9	Read from stream device and discard	stream→ discard
07h	6.3.7.10	Verify block or stream device operation	
08h	6.3.7.11	Copy block device with offset to stream device	block<o>→ stream
09h	6.3.7.12	Copy stream device to block device with offset	stream→ block<o>
0Ah	6.3.7.13	Copy block device with offset to block device with offset	block<o>→ block<o>
0Bh	6.3.7.3	Copy from block device to stream device and hold a copy of processed data for the application client [b]	block→ stream +application client
0Ch	6.3.7.4	Copy from stream device to block device and hold a copy of processed data for the application client [b]	stream→ block +application client
0Dh	6.3.7.5	Copy from block device to block device and hold a copy of processed data for the application client [b]	block→ block +application client
0Eh	6.3.7.6	Copy from stream device to stream device and hold a copy of processed data for the application client [b]	stream→ stream +application client
0Fh	6.3.7.9	Read from stream device and hold a copy of processed data for the application client [b]	stream→ discard +application client
10h	6.3.7.14	Write filemarks to sequential-access device	filemark→ tape
11h	6.3.7.15	Space records or filemarks on sequential-access device	space→ tape
12h	6.3.7.16	Locate on sequential-access device	locate→ tape
13h	6.3.7.17	Image copy from sequential-access device to sequential-access device	<i>tape→ <i>tape
14h	6.3.7.18	Register persistent reservation key	
15h	6.3.7.19	Third party persistent reservations source I_T nexus	
16h to BFh		Reserved	

[a] Block devices are those with peripheral device type codes 0h (i.e, direct access block), 5h (i.e., CD/DVD), and Eh (i.e., simplified direct-access). Stream devices are those devices with peripheral device type codes 1h (i.e., sequential-access) and 3h (i.e., processor). Sequential-access stream (indicated by the term tape in the shorthand column) devices are those with peripheral device type code 1h. See 6.4.2 for peripheral device type code definitions.

[b] The application client shall use the RECEIVE COPY RESULTS with a RECEIVE DATA service action to retrieve data held for it by the copy manager (see 6.18.3).

Segment descriptors (see table 120) begin with an eight byte header.

Table 120 — Segment descriptor header

Bit / Byte	7	6	5	4	3	2	1	0
0	DESCRIPTOR TYPE CODE (00h-3Fh)							
1	Reserved						DC	CAT
2	(MSB)	DESCRIPTOR LENGTH (n-7)						
3								(LSB)
4	(MSB)	SOURCE TARGET DESCRIPTOR INDEX						
5								(LSB)
6	(MSB)	DESTINATION TARGET DESCRIPTOR INDEX						
7								(LSB)
8	Segment descriptor parameters							
•••								
n								

The DESCRIPTOR TYPE CODE field is described in 6.3.5. Support for each segment descriptor format is optional. If a copy manager receives an unsupported descriptor type code in a segment descriptor, the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to UNSUPPORTED SEGMENT DESCRIPTOR TYPE CODE.

The destination count (DC) bit is only applicable to segment descriptors with descriptor type code values of 02h and 0Dh (see 6.3.7.5). The DC bit is reserved for all other segment descriptors.

The CAT bit is described in 6.3.7.2.

The DESCRIPTOR LENGTH field contains the length in bytes of the fields that follow the DESCRIPTOR LENGTH field in the segment descriptor. For each descriptor type code value, the length should be constant.

The SOURCE TARGET DESCRIPTOR INDEX field contains an index into the target descriptor list (see 6.3.1) identifying the source copy target device. The DESTINATION TARGET DESCRIPTOR INDEX field contains an index into the target descriptor list (see 6.3.1) identifying the destination copy target device. Some segment descriptor formats do not require a SOURCE TARGET DESCRIPTOR INDEX field or a DESTINATION TARGET DESCRIPTOR INDEX field, in which case the field is reserved.

If the copy target device identified by a SOURCE TARGET DESCRIPTOR INDEX field or a DESTINATION TARGET DESCRIPTOR INDEX field is not accessible to the copy manager, then the command shall be terminated with CHECK CONDITION status, with the sense key set to COPY ABORTED, and the additional sense code set to UNREACHABLE COPY TARGET.

6.3.7.2 Segment descriptor processing

In processing a segment descriptor, the copy manager may be required to:

a) read source data by issuing data input commands to the source device;
b) process data, an operation that generally designates data as destination data intended for transfer to the destination device; and
c) write some or all of the destination data to the destination device.

The number of blocks to read and write, the number of bytes to process, and the nature of processing are determined by the:

a) segment descriptor type code;
b) parameters of the segment descriptor; and
c) amount of residual source or destination data retained from the previous segment, if any.

Except as otherwise specified by particular segment descriptor type codes:

a) just enough whole-block read operations shall be performed to supply, together with residual source data from the previous segment or segments, the number of bytes to be processed;
b) processing consists of removing bytes from the source data and designating them as destination data, without change; and
c) as many whole-block write operations as possible shall be performed with the destination data, including any residual destination data from the previous segment or segments.

Any residual source data from the previous segment or segments shall be processed before any data read from the source device during processing of the current segment descriptor. Any residual destination data from the previous segment or segments shall be written before any data processed during processing of the current segment descriptor.

Exceptions and clarifications to these general rules are described in table 121 and the subclauses it references.

Table 121 — Descriptor Type Code Dependent Copy Manager Processing (part 1 of 2)

Segment Descriptor Type Code	Reference	Description
00h (block→ stream) or 0Bh (block→ stream+application client)	6.3.7.3	The number of bytes processed is determined by the BLOCK DEVICE NUMBER OF BLOCKS field for the source blocks (see applicable type code definition subclauses for details). [a]
02h (block→ block) or 0Dh (block→ block+application client) with DC=0	6.3.7.5	
02h (block→ block) or 0Dh (block→ block+application client) with DC=1	6.3.7.5	The number of blocks or byte range specified shall be output to the destination device. If residual destination data is sufficient to perform the output, then no data shall be processed. Otherwise, just as much data as needed shall be processed (which may involve reading data from the source device) so that the destination data (which includes any residual destination data from the previous segment) is sufficient. [a]
01h (stream→ block) or 0Ch (stream→ block+application client)	6.3.7.3	
09h (stream→ block<o>)	6.3.7.12	
03h (stream→ stream) or 0Eh (stream→ stream+application client)	6.3.7.6	The number of bytes specified in the segment descriptor shall be processed. [a]
04h (inline→ stream)	6.3.7.7	The specified number of bytes of inline or embedded data shall be appended to the destination data, and no source data shall be processed.
05h (embedded→ stream)	6.3.7.8	
06h (stream→ discard)	6.3.7.9	The specified number of bytes shall be removed from the source data and discarded.

[a] For segment descriptor type codes 0Bh, 0Ch, 0Dh and 0Eh, a copy of the processed data shall also be held for retrieval by the application client.

Table 121 — Descriptor Type Code Dependent Copy Manager Processing (part 2 of 2)

Segment Descriptor Type Code	Reference	Description
07h (verify device operation)	6.3.7.10	No data shall be processed and no read or write operations shall be performed on copy target devices. Residual source or destination data, if any, shall be retained or discarded as if the CAT bit were equal to one.
10h (filemark→ tape)	6.3.7.14	
11h (space→ tape)	6.3.7.15	
12h (locate→ tape)	6.3.7.16	
14h (register persistent reservation key)	6.3.7.18	
08h (block<o>→ stream)	6.3.7.11	The required blocks shall be read from the source device, the designated byte range shall be extracted as source data, and the designated number of bytes (starting with residual source data, if any) shall be processed.
0Ah (block<o>→ block<o>)	6.3.7.13	The source byte range specified shall be read into source data, the number of bytes specified shall be moved from source data to destination data, and the specified destination byte range shall be written using destination data.
0Fh (stream→ discard+application client)	6.3.7.9	The specified number of bytes shall be removed from the source data and held for retrieval by the application client.
13h (<i>tape→ <i>tape)	6.3.7.17	The data movement shall not involve processing as described in this subclause. Residual source or destination data, if any, shall not be used and shall be retained or discarded as if the CAT bit were equal to one.

[a] For segment descriptor type codes 0Bh, 0Ch, 0Dh and 0Eh, a copy of the processed data shall also be held for retrieval by the application client.

Reads and writes shall be performed using whole-block transfer lengths determined by the block size, transfer length, or both. Therefore some source data may remain unprocessed and some destination data may not have been transferred at the end of a segment. If so, the residue shall be handled according to the CAT bit in the segment descriptor and the PAD bits of the source and destination target descriptors, as defined in table 122.

Table 122 — PAD and CAT bit definitions

PAD bit in		CAT bit	Copy manager action
Source target descriptor	Destination target descriptor		
0 or 1	0 or 1	1	Any residual source data shall be retained as source data for a subsequent segment descriptor. Any residual destination data shall be retained as destination data for a subsequent segment descriptor. It shall not be an error if either the source or destination target index in the following segment descriptor does not match the corresponding target index with which residual data was originally associated. If the CAT bit is set to one on the last segment of an EXTENDED COPY command, any residual data shall be discarded and this shall not be considered an error.
1	1	0	Any residual source data shall be discarded. Any residual destination data shall be padded with zeroes to make a whole block transfer. [a]
0	1	0	Any residual source data shall be handled as if the CAT bit is equal to one (i.e., discarded on the last segment and retained otherwise). Any residual destination data shall be padded with zeroes to make a whole block transfer. [a]
1	0	0	Any residual source or destination data shall be discarded.
0	0	0	If there is residual source or destination data, the EXTENDED COPY command shall be terminated with CHECK CONDITION status, with the sense key set to COPY ABORTED, and the additional sense code set to UNEXPECTED INEXACT SEGMENT.

[a] When the CAT bit is set to zero and the destination target descriptor has the PAD bit set to one, the EXTENDED COPY command shall be terminated with CHECK CONDITION status, with the sense key set to COPY ABORTED, and the additional sense code set to UNEXPECTED INEXACT SEGMENT if any of the following conditions are met:

a) if any residual destination data is present after writing the designated byte range for a segment descriptor of type 09h (stream→ block <o>) or 0Ah (block<o>→ block<o>); or

b) if any residual destination data is present after the designated number of blocks have been written for a segment descriptor of type 02h (block→ block) with DC set to one, 0Dh (block→ block+application client) with DC set to one, 01h (stream→ block) or 0Ch (stream→ block+application client).

A few segment descriptors have either no source or no destination and handling of the PAD bit for those descriptors shall be as follows. For segment descriptor types 04h (inline→ stream, see 6.3.7.7) and 05h (embedded→ stream, see 6.3.7.8), the handling shall be as if the PAD is set to zero for the source target descriptor. For segment descriptor types 06h and 0Fh (stream→ discard and stream→ discard+application client, see 6.3.7.9), handling shall be as if the PAD is set to zero for the destination target descriptor.

6.3.7.3 Block device to stream device operations

The segment descriptor format shown in table 123 is used by the copy operations that move data from a block device to a stream device or vice versa.

Table 123 — Block device to or from stream device segment descriptor

Bit / Byte	7	6	5	4	3	2	1	0
0	DESCRIPTOR TYPE CODE (00h, 01h, 0Bh, or 0Ch)							
1	Reserved							CAT
2	(MSB)	DESCRIPTOR LENGTH (0014h)						
3								(LSB)
4	(MSB)	SOURCE TARGET DESCRIPTOR INDEX						
5								(LSB)
6	(MSB)	DESTINATION TARGET DESCRIPTOR INDEX						
7								(LSB)
8	Reserved							
9	(MSB)	STREAM DEVICE TRANSFER LENGTH						
10								
11								(LSB)
12	Reserved							
13	Reserved							
14	(MSB)	BLOCK DEVICE NUMBER OF BLOCKS						
15								(LSB)
16	(MSB)	BLOCK DEVICE LOGICAL BLOCK ADDRESS						
•••								
23								(LSB)

The DESCRIPTOR TYPE CODE field is described in 6.3.5 and 6.3.7.1. Two DESCRIPTOR TYPE CODE values use the segment descriptor format shown in table 123 and described in this subclause.

For descriptor type code 00h (block→ stream) or descriptor type code 0Bh (block→ stream+application client), the copy manager shall copy the data from the source block device identified by the SOURCE TARGET DESCRIPTOR INDEX field to the destination stream device identified by the DESTINATION TARGET DESCRIPTOR INDEX field using the logical blocks starting at the location identified by the BLOCK DEVICE LOGICAL BLOCK ADDRESS field. As many blocks shall be read as necessary to process (see 6.3.7.2) a number of bytes equal to the contents of the DISK BLOCK LENGTH field in the target descriptor for the source device times the contents of the BLOCK DEVICE NUMBER OF BLOCKS field. The data shall be written to the stream device starting at the current position of the media.

For descriptor type code 0Bh (block→ stream+application client), the copy manager also shall hold a copy of the processed data for delivery to the application client upon completion of the EXTENDED COPY command in response to a RECEIVE COPY RESULTS command with RECEIVE DATA service action as described in 6.18.3. The minimum amount of held data supported by the copy manager is returned in the response data for the RECEIVE COPY RESULTS command with OPERATING PARAMETERS service action (see 6.18.4). If the copy

manager supports the 0Bh descriptor type code, it also shall support the RECEIVE COPY RESULTS command with RECEIVE DATA service action.

The CAT bit is described in 6.3.7.2.

The DESCRIPTOR LENGTH field shall contain 20 (0014h). The SOURCE TARGET DESCRIPTOR INDEX and DESTINATION TARGET DESCRIPTOR INDEX fields are described in 6.3.7.1.

The STREAM DEVICE TRANSFER LENGTH field specifies the amount of data to be written on each write operation to the stream device. See 6.3.6.7 for a description of how data in the STREAM DEVICE TRANSFER LENGTH field in the segment descriptor interacts with data in the STREAM BLOCK LENGTH field in the device type specific target descriptor parameters for the sequential-access device type.

The BLOCK DEVICE NUMBER OF BLOCKS field specifies the length, in source logical blocks, of data to be processed (see 6.3.7.2) in the segment. A value of zero shall not be considered as an error. No data shall be processed, but any residual destination data retained from a previous segment shall be written if possible to the destination in whole-block transfers. A value of zero shall not modify the handling of residual data.

The BLOCK DEVICE LOGICAL BLOCK ADDRESS field specifies the starting logical block address on the block device for this segment.

6.3.7.4 Stream device to block device operations

The segment descriptor format shown in table 123 (see 6.3.7.3) also is used by the copy operations that move data from a stream device to a block device. Two DESCRIPTOR TYPE CODE values use the segment descriptor format shown in table 123 and described in this subclause.

For descriptor type code 01h (stream→ block) or descriptor type code 0Ch (stream→ block+application client), the copy manager shall copy the data from the source stream device identified by the SOURCE TARGET DESCRIPTOR INDEX field to the destination block device identified by the DESTINATION TARGET DESCRIPTOR INDEX field using the stream data starting at the current position of the stream device. The data shall be written to logical blocks starting at the location identified by the BLOCK DEVICE LOGICAL BLOCK ADDRESS field and continuing for the number of blocks specified in the BLOCK DEVICE NUMBER OF BLOCKS field.

For descriptor type code 0Ch (stream→ block+application client), the copy manager also shall hold a copy of the processed data for delivery to the application client upon completion of the EXTENDED COPY command in response to a RECEIVE COPY RESULTS command with RECEIVE DATA service action as described in 6.18.3. The minimum amount of held data supported by the copy manager is returned in the response data for the RECEIVE COPY RESULTS command with OPERATING PARAMETERS service action (see 6.18.4). If the copy manager supports the 0Ch descriptor type code, it also shall support the RECEIVE COPY RESULTS command with RECEIVE DATA service action.

The CAT bit is described in 6.3.7.2.

The DESCRIPTOR LENGTH field shall contain 20 (0014h). The SOURCE TARGET DESCRIPTOR INDEX and DESTINATION TARGET DESCRIPTOR INDEX fields are described in 6.3.7.1.

The STREAM DEVICE TRANSFER LENGTH field specifies the amount of data to be read from the source stream device on each read operation. See 6.3.6.7 for a description of how data in the STREAM DEVICE TRANSFER LENGTH field in the segment descriptor interacts with data in the STREAM BLOCK LENGTH field in the device type specific target descriptor parameters for the sequential-access device type.

The BLOCK DEVICE NUMBER OF BLOCKS field specifies the number blocks to be written by the segment. A value of zero specifies that no blocks shall be written in this segment. This shall not be considered as an error.

The BLOCK DEVICE LOGICAL BLOCK ADDRESS field specifies the starting logical block address on the block device for this segment.

6.3.7.5 Block device to block device operations

The segment descriptor format shown in table 124 is used by the copy operations that move data from a block device to a block device.

Table 124 — Block device to block device segment descriptor

Bit / Byte	7	6	5	4	3	2	1	0
0	DESCRIPTOR TYPE CODE (02h or 0Dh)							
1	Reserved						DC	CAT
2	(MSB)	DESCRIPTOR LENGTH (0018h)						
3								(LSB)
4	(MSB)	SOURCE TARGET DESCRIPTOR INDEX						
5								(LSB)
6	(MSB)	DESTINATION TARGET DESCRIPTOR INDEX						
7								(LSB)
8	Reserved							
9	Reserved							
10	(MSB)	BLOCK DEVICE NUMBER OF BLOCKS						
11								(LSB)
12	(MSB)	SOURCE BLOCK DEVICE LOGICAL BLOCK ADDRESS						
•••								
19								(LSB)
20	(MSB)	DESTINATION BLOCK DEVICE LOGICAL BLOCK ADDRESS						
•••								
27								(LSB)

The DESCRIPTOR TYPE CODE field is described in 6.3.5 and 6.3.7.1. Two DESCRIPTOR TYPE CODE values use the segment descriptor format shown in table 124 and described in this subclause.

For descriptor type code 02h (block→ block) or descriptor type code 0Dh (block→ block+application client), the copy manager shall copy the data from the source block device identified by the SOURCE TARGET DESCRIPTOR INDEX field to the destination block device identified by the DESTINATION TARGET DESCRIPTOR INDEX field using the logical blocks starting at the location identified by the SOURCE BLOCK DEVICE LOGICAL BLOCK ADDRESS field. The data shall be written to logical blocks starting at the location identified by the DESTINATION BLOCK DEVICE LOGICAL BLOCK ADDRESS field.

If the DC bit equals zero, then as many blocks shall be read as necessary to process (see 6.3.7.2) a number of bytes equal to the contents of the DISK BLOCK LENGTH field in the target descriptor for the source device times the contents of the BLOCK DEVICE NUMBER OF BLOCKS field, and as many writes as possible shall be performed using any residual destination data from the previous segment and the data processed in this segment. If the DC bit equals one, then the number of blocks specified by the BLOCK DEVICE NUMBER OF BLOCKS field shall be written to the destination block device, as many bytes shall be processed as necessary for these writes to be performed, and as many blocks shall be read as necessary to supply the data to be processed.

For descriptor type code 0Dh (block→ block+application client), the copy manager also shall hold a copy of the processed data for delivery to the application client upon completion of the EXTENDED COPY command in response to a RECEIVE COPY RESULTS command with RECEIVE DATA service action as described in 6.18.3. The minimum amount of held data supported by the copy manager is returned in the response data for the RECEIVE COPY RESULTS command with OPERATING PARAMETERS service action (see 6.18.4). If the copy manager supports the 0Dh descriptor type code, it also shall support the RECEIVE COPY RESULTS command with RECEIVE DATA service action.

The CAT bit is described in 6.3.7.2.

The destination count (DC) bit specifies whether the BLOCK DEVICE NUMBER OF BLOCKS field refers to the source or destination device. A DC bit set to zero specifies that the BLOCK DEVICE NUMBER OF BLOCKS field refers to the source device. A DC bit set to one specifies that the BLOCK DEVICE NUMBER OF BLOCKS field refers to the destination device.

The DESCRIPTOR LENGTH field shall contain 24 (0018h). The SOURCE TARGET DESCRIPTOR INDEX and DESTINATION TARGET DESCRIPTOR INDEX fields are described in 6.3.7.1.

If DC is set to zero, the BLOCK DEVICE NUMBER OF BLOCKS field specifies the number of blocks to be processed. If DC is set to one, the BLOCK DEVICE NUMBER OF BLOCKS field specifies the number of blocks to be written to the destination device. A value of zero shall not be considered as an error. If the DC bit equals one, a value of zero specifies that no destination blocks shall be written and the only processing to be performed is that any residual source or destination data from the previous segment shall be handled as residual data as described in 6.3.7.2. If the DC bit equals zero, a value of zero specifies that no source blocks shall be read and no source data shall be processed, but any residual destination data from a previous segment shall be written if possible to the destination in whole-block transfers, and any residual data shall be handled as described in 6.3.7.2.

The SOURCE BLOCK DEVICE LOGICAL BLOCK ADDRESS field specifies the logical block address from which the reading of data shall start.

The DESTINATION BLOCK DEVICE LOGICAL BLOCK ADDRESS field specifies the logical block address to which the writing of data shall begin.

6.3.7.6 Stream device to stream device operations

The segment descriptor format shown in table 125 is used by the copy operations that move data from a stream device to a stream device.

Table 125 — Stream device to stream device segment descriptor

Bit / Byte	7	6	5	4	3	2	1	0
0	DESCRIPTOR TYPE CODE (03h or 0Eh)							
1	Reserved							CAT
2	(MSB)	DESCRIPTOR LENGTH (0010h)						
3								(LSB)
4	(MSB)	SOURCE TARGET DESCRIPTOR INDEX						
5								(LSB)
6	(MSB)	DESTINATION TARGET DESCRIPTOR INDEX						
7								(LSB)
8	Reserved							
9	(MSB)	SOURCE STREAM DEVICE TRANSFER LENGTH						
10								
11								(LSB)
12	Reserved							
13	(MSB)	DESTINATION STREAM DEVICE TRANSFER LENGTH						
14								
15								(LSB)
16	(MSB)	BYTE COUNT						
•••								
19								(LSB)

The DESCRIPTOR TYPE CODE field is described in 6.3.5 and 6.3.7.1. Two DESCRIPTOR TYPE CODE values use the segment descriptor format shown in table 125 and described in this subclause.

For descriptor type code 03h (stream→ stream) or descriptor type code 0Eh (stream→ stream+application client), the copy manager shall copy the data from the source stream device identified by the SOURCE TARGET DESCRIPTOR INDEX field to the destination stream device identified by the DESTINATION TARGET DESCRIPTOR INDEX field. Data shall be read from the source stream device starting at the current position of the source stream device. Data shall be written to the destination stream device starting at the current position of the destination stream device. The BYTE COUNT field defines the number of bytes to be processed (see 6.3.7.2) by the copy manager. The copy manager shall perform read operations as necessary to supply the source data, and as many write operations as possible using the destination data.

For descriptor type code 0Eh (stream→ stream+application client), the copy manager also shall hold a copy of the processed data for delivery to the application client upon completion of the EXTENDED COPY command in response to a RECEIVE COPY RESULTS command with RECEIVE DATA service action as described in 6.18.3. The minimum amount of held data supported by the copy manager is returned in the response data for the RECEIVE COPY RESULTS command with OPERATING PARAMETERS service action (see 6.18.4). If the copy

manager supports the 0Eh descriptor type code, it also shall support the RECEIVE COPY RESULTS command with RECEIVE DATA service action.

The CAT bit is described in 6.3.7.2.

The DESCRIPTOR LENGTH field shall contain 16 (0010h). The SOURCE TARGET DESCRIPTOR INDEX and DESTINATION TARGET DESCRIPTOR INDEX fields are described in 6.3.7.1.

The SOURCE STREAM DEVICE TRANSFER LENGTH field specifies the amount of data to be read from the source stream device on each read operation. See 6.3.6.7 for a description of how data in the SOURCE STREAM DEVICE TRANSFER LENGTH field in the segment descriptor interacts with data in the STREAM BLOCK LENGTH field in the device type specific target descriptor parameters for the source sequential-access device type.

The DESTINATION STREAM DEVICE TRANSFER LENGTH field specifies the amount of data to be written to the destination stream device on each write operation. See 6.3.6.7 for a description of how data in the DESTINATION STREAM DEVICE TRANSFER LENGTH field in the segment descriptor interacts with data in the STREAM BLOCK LENGTH field in the device type specific target descriptor parameters for the destination sequential-access device type.

The BYTE COUNT field specifies the number of bytes that shall be processed for this segment descriptor. A value of zero shall not be considered as an error, and specifies that no source blocks shall be read and no source data shall be processed. However, a value of zero specifies that any residual destination data from a previous segment shall be written if possible to the destination in whole-block transfers, and any residual data shall be handled as described in 6.3.7.2.

6.3.7.7 Inline data to stream device operation

The segment descriptor format shown in table 126 instructs the copy manager to write inline data from the EXTENDED COPY parameter list to a stream device.

Table 126 — Inline data to stream device segment descriptor

Bit / Byte	7	6	5	4	3	2	1	0
0	DESCRIPTOR TYPE CODE (04h)							
1	Reserved							CAT
2	(MSB)			DESCRIPTOR LENGTH (0010h)				
3								(LSB)
4	Reserved							
5	Reserved							
6	(MSB)			DESTINATION TARGET DESCRIPTOR INDEX				
7								(LSB)
8	Reserved							
9	(MSB)			STREAM DEVICE TRANSFER LENGTH				
10								
11								(LSB)
12	(MSB)			INLINE DATA OFFSET				
•••								
15								(LSB)
16	(MSB)			INLINE DATA NUMBER OF BYTES				
•••								
19								(LSB)

The DESCRIPTOR TYPE CODE field is described in 6.3.5 and 6.3.7.1. Descriptor type code 04h (inline→ stream) instructs the copy manager to write inline data from the EXTENDED COPY parameter list to a stream device. The inline data shall be read from the optional inline data at the end of the EXTENDED COPY parameter list. The data shall be written to the destination stream device identified by the DESTINATION TARGET DESCRIPTOR INDEX field starting at the current position of the stream device. Any residual destination data from a previous segment descriptor shall be written before the data of the current segment descriptor. Any residual source data from a previous segment descriptor shall not be processed (see 6.3.7.2), and shall be handled as residual source data.

The CAT bit is described in 6.3.7.2.

The DESCRIPTOR LENGTH field shall contain 16 (0010h). The DESTINATION TARGET DESCRIPTOR INDEX field is described in 6.3.7.1.

The STREAM DEVICE TRANSFER LENGTH field specifies the amount of data to be written to the stream device on each write operation. See 6.3.6.7 for a description of how data in the STREAM DEVICE TRANSFER LENGTH field in the segment descriptor interacts with data in the STREAM BLOCK LENGTH field in the device type specific target descriptor parameters for the destination sequential-access device type.

The value in the INLINE DATA OFFSET field is added to the location of the first byte of inline data in the EXTENDED COPY parameter list (see table 104) to locate the first byte of inline data to be written to the stream device. The INLINE DATA OFFSET value shall be a multiple of 4.

The INLINE DATA NUMBER OF BYTES field specifies the number of bytes of inline data that are to be transferred to the stream device. A value of zero shall not be considered an error.

If the sum of the INLINE DATA OFFSET and the INLINE DATA NUMBER OF BYTES values exceeds the value in the INLINE DATA LENGTH field (see table 104), the copy manager shall terminate the command with CHECK CONDITION status, with the sense key set to COPY ABORTED, and the additional sense code set to INLINE DATA LENGTH EXCEEDED.

6.3.7.8 Embedded data to stream device operation

The segment descriptor format shown in table 127 instructs the copy manager to write embedded data from the segment descriptor to a stream device.

Table 127 — Embedded data to stream device segment descriptor

Bit / Byte	7	6	5	4	3	2	1	0
0	DESCRIPTOR TYPE CODE (05h)							
1	Reserved							CAT
2	(MSB)	DESCRIPTOR LENGTH (n-3)						
3								(LSB)
4	Reserved							
5	Reserved							
6	(MSB)	DESTINATION TARGET DESCRIPTOR INDEX						
7								(LSB)
8	Reserved							
9	(MSB)	STREAM DEVICE TRANSFER LENGTH						
10								
11								(LSB)
12	(MSB)	EMBEDDED DATA NUMBER OF BYTES						
13								(LSB)
14	Reserved							
15								
16	EMBEDDED DATA							
•••								
n								

The DESCRIPTOR TYPE CODE field is described in 6.3.5 and 6.3.7.1. Descriptor type code 05h (embedded→ stream) instructs the copy manager to write embedded data from the segment descriptor to a stream device. The embedded data shall be read from the segment descriptor. The data shall be written to the destination stream device identified by the DESTINATION TARGET DESCRIPTOR INDEX field starting at the current position of the stream device. Any residual destination data from a previous segment descriptor shall be written before the data of the

current segment descriptor. Any residual source data from a previous segment descriptor shall not be processed (see 6.3.7.2), and shall be handled as residual source data.

The CAT bit is described in 6.3.7.2.

The DESCRIPTOR LENGTH field shall contain the length in bytes of the fields that follow the DESCRIPTOR LENGTH field, including the embedded data. The value in the DESCRIPTOR LENGTH field shall be a multiple of 4.

The DESTINATION TARGET DESCRIPTOR INDEX field is described in 6.3.7.1.

The STREAM DEVICE TRANSFER LENGTH field specifies the amount of data to be written to the stream device on each write operation. See 6.3.6.7 for a description of how data in the STREAM DEVICE TRANSFER LENGTH field in the segment descriptor interacts with data in the STREAM BLOCK LENGTH field in the device type specific target descriptor parameters for the destination sequential-access device type.

The EMBEDDED DATA NUMBER OF BYTES field specifies the number of bytes of embedded data that are to be transferred to the stream device. A value of zero shall not be considered an error. The EMBEDDED DATA NUMBER OF BYTES value shall be less than or equal to the DESCRIPTOR LENGTH value minus 12.

6.3.7.9 Stream device to discard operation

The segment descriptor format shown in table 128 instructs the copy manager to read data from a stream device and not copy it to any destination device.

Table 128 — Stream device to discard segment descriptor

Bit / Byte	7	6	5	4	3	2	1	0
0	DESCRIPTOR TYPE CODE (06h or 0Fh)							
1	Reserved							CAT
2	(MSB)	DESCRIPTOR LENGTH (000Ch)						
3								(LSB)
4	(MSB)	SOURCE TARGET DESCRIPTOR INDEX						
5								(LSB)
6	Reserved							
7	Reserved							
8	Reserved							
9	(MSB)	STREAM DEVICE TRANSFER LENGTH						
10								
11								(LSB)
12	(MSB)	NUMBER OF BYTES						
•••								
15								(LSB)

The DESCRIPTOR TYPE CODE field is described in 6.3.5 and 6.3.7.1. Two DESCRIPTOR TYPE CODE values use the segment descriptor format shown in table 128 and described in this subclause.

For descriptor type code 06h (stream→ discard) or descriptor type code 0Fh (stream→ discard+application client), the copy manager shall read data as necessary from the source stream device identified by the SOURCE TARGET DESCRIPTOR INDEX field starting at the current position of the source stream device. The number of bytes specified by the NUMBER OF BYTES field shall be removed from the source data, starting with any residual source data from the previous segment.

For descriptor type code 06h (stream→ discard) the removed data shall be discarded and not written to any destination device. For descriptor type code 0Fh (stream→ discard+application client) the removed data shall be held for delivery to the application client upon completion of the EXTENDED COPY command in response to a RECEIVE COPY RESULTS command with RECEIVE DATA service action as described in 6.18.3. The minimum amount of held data supported by the copy manager is returned in the response data for the RECEIVE COPY RESULTS command with OPERATING PARAMETERS service action (see 6.18.4). If the copy manager supports the 0Fh (stream→ discard+application client) descriptor type code, it also shall support the RECEIVE COPY RESULTS command with RECEIVE DATA service action.

The CAT bit is described in 6.3.7.2.

The DESCRIPTOR LENGTH field shall contain 12 (000Ch). The DESTINATION TARGET DESCRIPTOR INDEX field is described in 6.3.7.1.

The SOURCE STREAM DEVICE TRANSFER LENGTH field specifies the amount of data to be read from the source stream device on each read operation. See 6.3.6.7 for a description of how data in the SOURCE STREAM DEVICE TRANSFER LENGTH field in the segment descriptor interacts with data in the STREAM BLOCK LENGTH field in the device type specific target descriptor parameters for the source sequential-access device type.

The NUMBER OF BYTES field specifies the number of bytes to be removed from the source data.

6.3.7.10 Verify device operation

The segment descriptor format shown in table 129 instructs the copy manager to verify the accessibility of a SCSI device.

Table 129 — Verify device operation segment descriptor

Bit / Byte	7	6	5	4	3	2	1	0
0	DESCRIPTOR TYPE CODE (07h)							
1	Reserved							
2	(MSB)	DESCRIPTOR LENGTH (0008h)						
3								(LSB)
4	(MSB)	SOURCE TARGET DESCRIPTOR INDEX						
5								(LSB)
6		Reserved						
7								
8	Reserved							TUR
9		Reserved						
•••								
11								

The DESCRIPTOR TYPE CODE field is described in 6.3.5 and 6.3.7.1. Descriptor type code 07h instructs the copy manager to verify the accessibility of the device identified by the SOURCE TARGET DESCRIPTOR INDEX field.

The DESCRIPTOR LENGTH field shall contain 8 (0008h). The SOURCE TARGET DESCRIPTOR INDEX field is described in 6.3.7.1.

Support for a value of one in the test unit ready (TUR) bit is optional. If setting the TUR bit to one is supported and the TUR bit is set to one, then a TEST UNIT READY command (see 6.37) shall be used to determine the readiness of the device. If setting the TUR to one is not supported and the TUR bit is set to one, then the EXTENDED COPY command shall be terminated with CHECK CONDITION status, with the sense key set to COPY ABORTED, and the additional sense code set to INVALID FIELD IN PARAMETER LIST. The SENSE-KEY SPECIFIC field shall be set as described in 6.3.3. If the TUR bit is set to zero, then the accessibility should be verified without disturbing established unit attention conditions or ACA conditions (e.g., using the INQUIRY command (see 6.4)).

6.3.7.11 Block device with offset to stream device operation

The segment descriptor format shown in table 130 is used to instruct the copy manager to move data from a block device with a byte offset to a stream device or vice versa.

Table 130 — Block device with offset to or from stream device segment descriptor

Bit / Byte	7	6	5	4	3	2	1	0
0	DESCRIPTOR TYPE CODE (08h or 09h)							
1	Reserved							CAT
2	(MSB)	DESCRIPTOR LENGTH (0018h)						
3								(LSB)
4	(MSB)	SOURCE TARGET DESCRIPTOR INDEX						
5								(LSB)
6	(MSB)	DESTINATION TARGET DESCRIPTOR INDEX						
7								(LSB)
8	Reserved							
9	(MSB)	STREAM DEVICE TRANSFER LENGTH						
10								
11								(LSB)
12	(MSB)	NUMBER OF BYTES						
•••								
15								(LSB)
16	(MSB)	BLOCK DEVICE LOGICAL BLOCK ADDRESS						
•••								
23								(LSB)
24	Reserved							
25	Reserved							
26	(MSB)	BLOCK DEVICE BYTE OFFSET						
27								(LSB)

The DESCRIPTOR TYPE CODE field is described in 6.3.5 and 6.3.7.1. Descriptor type code 08h (block<o>→ stream) instructs the copy manager to copy the data from the source block device identified by the SOURCE TARGET DESCRIPTOR INDEX field to the destination stream device identified by the DESTINATION TARGET DESCRIPTOR INDEX field using data starting at the location identified by the BLOCK DEVICE BYTE OFFSET field in the logical block identified by the BLOCK DEVICE LOGICAL BLOCK ADDRESS field and continuing for the number of bytes specified in the NUMBER OF BYTES field. The data shall be written to the stream device starting at the current position of the media.

The CAT bit is described in 6.3.7.2.

The DESCRIPTOR LENGTH field shall contain 24 (0018h). The SOURCE TARGET DESCRIPTOR INDEX and DESTINATION TARGET DESCRIPTOR INDEX fields are described in 6.3.7.1.

The STREAM DEVICE TRANSFER LENGTH field specifies the amount of data to be written on each write operation to the stream device. See 6.3.6.7 for a description of how data in the STREAM DEVICE TRANSFER LENGTH field in the segment descriptor interacts with data in the STREAM BLOCK LENGTH field in the device type specific target descriptor parameters for the sequential-access device type.

The NUMBER OF BYTES field specifies the number bytes to be read. A value of zero specifies that no bytes shall be transferred in this segment. This shall not be considered as an error.

The BLOCK DEVICE LOGICAL BLOCK ADDRESS field specifies the starting logical block address on the source block device for this segment.

The BLOCK DEVICE BYTE OFFSET field specifies the offset into the first source block at which to begin reading bytes.

6.3.7.12 Stream device to block device with offset operation

The segment descriptor format shown in table 130 (see 6.3.7.11) also is used to instruct the copy manager to move data from a stream device to a block device with a byte offset.

The DESCRIPTOR TYPE CODE field is described in 6.3.5 and 6.3.7.1. Descriptor type code 09h (stream→ block<o>) instructs the copy manager to copy the data from the source stream device identified by the SOURCE TARGET DESCRIPTOR INDEX field to the destination block device identified by the DESTINATION TARGET DESCRIPTOR INDEX field using the stream data starting at the current position of the stream device. The data shall be written starting at the location identified by the BLOCK DEVICE BYTE OFFSET field in the logical block identified by the BLOCK DEVICE LOGICAL BLOCK ADDRESS field and continuing for the number of bytes specified in the NUMBER OF BYTES field.

The content of the starting logical block on the destination device before the starting offset shall be preserved. The content on the ending logical block beyond the end of the transfer shall be preserved. The copy manager may implement this operation by reading the starting and ending logical blocks, modifying a portion of the blocks as required, and writing the full blocks to the destination device.

The CAT bit is described in 6.3.7.2.

The DESCRIPTOR LENGTH field shall contain 24 (0018h). The SOURCE TARGET DESCRIPTOR INDEX and DESTINATION TARGET DESCRIPTOR INDEX fields are described in 6.3.7.1.

The STREAM DEVICE TRANSFER LENGTH field specifies the amount of data to be written on each write operation to the stream device. See 6.3.6.7 for a description of how data in the STREAM DEVICE TRANSFER LENGTH field in the segment descriptor interacts with data in the STREAM BLOCK LENGTH field in the device type specific target descriptor parameters for the sequential-access device type.

The NUMBER OF BYTES field specifies the number bytes to be read. A value of zero specifies that no bytes shall be transferred in this segment. This shall not be considered as an error.

The BLOCK DEVICE LOGICAL BLOCK ADDRESS field specifies the starting logical block address on the destination block device for this segment.

The BLOCK DEVICE BYTE OFFSET field specifies the offset into the first destination block at which to begin writing data to the destination block device.

6.3.7.13 Block device with offset to block device with offset operation

The segment descriptor format shown in table 131 instructs the copy manager to move data from a block device with a byte offset to a block device with a byte offset.

Table 131 — Block device with offset to block device with offset segment descriptor

Bit / Byte	7	6	5	4	3	2	1	0
0	DESCRIPTOR TYPE CODE (0Ah)							
1	Reserved							CAT
2	(MSB)	DESCRIPTOR LENGTH (001Ch)						
3								(LSB)
4	(MSB)	SOURCE TARGET DESCRIPTOR INDEX						
5								(LSB)
6	(MSB)	DESTINATION TARGET DESCRIPTOR INDEX						
7								(LSB)
8	(MSB)	NUMBER OF BYTES						
•••								
11								(LSB)
12	(MSB)	SOURCE BLOCK DEVICE LOGICAL BLOCK ADDRESS						
•••								
19								(LSB)
20	(MSB)	DESTINATION BLOCK DEVICE LOGICAL BLOCK ADDRESS						
•••								
27								(LSB)
28	(MSB)	SOURCE BLOCK DEVICE BYTE OFFSET						
29								(LSB)
30	(MSB)	DESTINATION BLOCK DEVICE BYTE OFFSET						
31								(LSB)

The DESCRIPTOR TYPE CODE field is described in 6.3.5 and 6.3.7.1. Descriptor type code 0Ah (block<o>→ block<o>) instructs the copy manager to copy the data from the source block device identified by the SOURCE TARGET DESCRIPTOR INDEX field to the destination block device identified by the DESTINATION TARGET DESCRIPTOR INDEX field using data starting at the location identified by the source BLOCK DEVICE BYTE OFFSET field in the logical block identified by the SOURCE BLOCK DEVICE LOGICAL BLOCK ADDRESS field and continuing for the number of bytes specified in the NUMBER OF BYTES field. The data shall be written starting at the location identified by the DESTINATION BLOCK DEVICE BYTE OFFSET field in the logical block identified by the DESTINATION BLOCK DEVICE LOGICAL BLOCK ADDRESS field.

The content of the starting logical block on the destination device before the starting offset shall be preserved. The content on the ending logical block beyond the end of the transfer shall be preserved. The copy manager may implement this operation by reading the starting and ending logical blocks, modifying a portion of the blocks as required, and writing the full blocks to the destination device.

The CAT bit is described in 6.3.7.2.

The DESCRIPTOR LENGTH field shall contain 28 (001Ch). The SOURCE TARGET DESCRIPTOR INDEX and DESTINATION TARGET DESCRIPTOR INDEX fields are described in 6.3.7.1.

The NUMBER OF BYTES field specifies the number bytes to be read. A value of zero specifies that no bytes shall be transferred in this segment. This shall not be considered as an error.

The SOURCE BLOCK DEVICE LOGICAL BLOCK ADDRESS field specifies the starting address on the source block device for this segment.

The DESTINATION BLOCK DEVICE LOGICAL BLOCK ADDRESS field specifies the starting logical block address on the destination block device for this segment.

The SOURCE BLOCK DEVICE BYTE OFFSET field specifies the offset into the first source block at which to begin reading bytes.

The DESTINATION BLOCK DEVICE BYTE OFFSET field specifies the offset into the first destination block at which to begin writing data to the destination block device.

6.3.7.14 Write filemarks operation

The segment descriptor format shown in table 132 instructs the copy manager to write filemarks or setmarks on the destination tape device.

Table 132 — Write filemarks operation segment descriptor

Bit / Byte	7	6	5	4	3	2	1	0
0	DESCRIPTOR TYPE CODE (10h)							
1	Reserved							
2	(MSB)	DESCRIPTOR LENGTH (0008h)						
3								(LSB)
4	Reserved							
5	Reserved							
6	(MSB)	DESTINATION TARGET DESCRIPTOR INDEX						
7								(LSB)
8	Reserved						WSMK	IMMED
9	(MSB)	TRANSFER LENGTH						
10								
11								(LSB)

The DESCRIPTOR TYPE CODE field is described in 6.3.5 and 6.3.7.1. Descriptor type code 10h (filemark→ tape) instructs the copy manager to write filemarks or setmarks to the destination tape device identified by the DESTI-

NATION TARGET DESCRIPTOR INDEX field starting at the current position of the tape device. If the PERIPHERAL DEVICE TYPE field in the target descriptor identified by the DESTINATION TARGET DESCRIPTOR INDEX field does not contain 01h, the copy manager shall terminate the command with CHECK CONDITION status, with the sense key set to COPY ABORTED, and the additional sense code set to INVALID OPERATION FOR COPY SOURCE OR DESTINATION.

The DESCRIPTOR LENGTH field shall contain 8 (0008h). The DESTINATION TARGET DESCRIPTOR INDEX field is described in 6.3.7.1.

If the write setmark (WSMK) bit is set to one, the TRANSFER LENGTH field specifies the number of setmarks to be written. If the WSMK bit is set to zero, the TRANSFER LENGTH field specifies the number of filemarks to be written.

If the immediate (IMMED) bit in the segment descriptor is set to one, then the copy manager shall issue a WRITE FILEMARKS command to the destination tape device with the immediate bit is set to one. If the IMMED bit is set to zero, then the copy manager shall issue a WRITE FILEMARKS command to the destination tape device with the immediate bit is set to zero.

6.3.7.15 Space operation

The segment descriptor format shown in table 133 instructs the copy manager to send a SPACE command (see SSC-2) to the destination tape device.

Table 133 — Space operation segment descriptor

Bit / Byte	7	6	5	4	3	2	1	0
0	DESCRIPTOR TYPE CODE (11h)							
1	Reserved							
2	(MSB)	DESCRIPTOR LENGTH (0008h)						
3								(LSB)
4	Reserved							
5	Reserved							
6	(MSB)	DESTINATION TARGET DESCRIPTOR INDEX						
7								(LSB)
8	Reserved					CODE		
9	(MSB)	COUNT						
10								
11								(LSB)

The DESCRIPTOR TYPE CODE field is described in 6.3.5 and 6.3.7.1. Descriptor type code 11h (space→ tape) instructs the copy manager to send a SPACE command to the destination tape device identified by the DESTINATION TARGET DESCRIPTOR INDEX field. If the PERIPHERAL DEVICE TYPE field in the target descriptor identified by the DESTINATION TARGET DESCRIPTOR INDEX field does not contain 01h, the copy manager shall terminate the command with CHECK CONDITION status, with the sense key set to COPY ABORTED, and the additional sense code set to INVALID OPERATION FOR COPY SOURCE OR DESTINATION.

The DESCRIPTOR LENGTH field shall contain 8 (0008h). The DESTINATION TARGET DESCRIPTOR INDEX field is described in 6.3.7.1.

The CODE and COUNT fields contents in the SPACE command sent to the destination tape device shall be copied from the CODE and COUNT fields in the segment descriptor. All other fields in the SPACE command sent to the destination tape device that affect the positioning of the tape shall be set to zero.

6.3.7.16 Locate operation

The segment descriptor format shown in table 134 instructs the copy manager to send a LOCATE command (see SSC-2) to the destination tape device.

Table 134 — Locate operation segment descriptor

Bit / Byte	7	6	5	4	3	2	1	0
0	DESCRIPTOR TYPE CODE (12h)							
1	Reserved							
2	(MSB)	DESCRIPTOR LENGTH (0008h)						
3								(LSB)
4	Reserved							
5	Reserved							
6	(MSB)	DESTINATION TARGET DESCRIPTOR INDEX						
7								(LSB)
8	(MSB)	BLOCK ADDRESS						
•••								
11								(LSB)

The DESCRIPTOR TYPE CODE field is described in 6.3.5 and 6.3.7.1. Descriptor type code 12h (locate→ tape) instructs the copy manager to send a LOCATE command to the destination tape device identified by the DESTINATION TARGET DESCRIPTOR INDEX field. If the PERIPHERAL DEVICE TYPE field in the target descriptor identified by the DESTINATION TARGET DESCRIPTOR INDEX field does not contain 01h, the copy manager shall terminate the command with CHECK CONDITION status, with the sense key set to COPY ABORTED, and the additional sense code set to INVALID OPERATION FOR COPY SOURCE OR DESTINATION.

The DESCRIPTOR LENGTH field shall contain 8 (0008h). The DESTINATION TARGET DESCRIPTOR INDEX field is described in 6.3.7.1.

The BLOCK ADDRESS field contents in the LOCATE command sent to the destination tape device shall be copied from the BLOCK ADDRESS field in the segment descriptor. All other fields in the LOCATE command sent to the destination tape device that affect the positioning of the tape shall be set to zero.

NOTE 31 - The restrictions described above for the LOCATE command limit the operation to locating SCSI logical block addresses in the current tape partition.

6.3.7.17 Tape device image copy operation

The segment descriptor format shown in table 135 instructs the copy manager to perform an image copy from the source tape device to the destination tape device.

Table 135 — Tape device image copy segment descriptor

Bit / Byte	7	6	5	4	3	2	1	0
0	DESCRIPTOR TYPE CODE (13h)							
1	Reserved							
2	(MSB)	DESCRIPTOR LENGTH (0008h)						
3								(LSB)
4	(MSB)	SOURCE TARGET DESCRIPTOR INDEX						
5								(LSB)
6	(MSB)	DESTINATION TARGET DESCRIPTOR INDEX						
7								(LSB)
8	(MSB)	COUNT						
•••								
11								(LSB)

The DESCRIPTOR TYPE CODE field is described in 6.3.5 and 6.3.7.1. Descriptor type code 13h (<i>tape→ <i>tape) instructs the copy manager to create a compatible image of the source device medium identified by the SOURCE TARGET DESCRIPTOR INDEX field on the destination device medium identified by the DESTINATION TARGET DESCRIPTOR INDEX field beginning at their current positions. If the PERIPHERAL DEVICE TYPE field in the target descriptor identified by the SOURCE TARGET DESCRIPTOR INDEX field or the DESTINATION TARGET DESCRIPTOR INDEX field does not contain 01h, the copy manager shall terminate the command with CHECK CONDITION status, with the sense key set to COPY ABORTED, and the additional sense code set to INVALID OPERATION FOR COPY SOURCE OR DESTINATION.

The DESCRIPTOR LENGTH field shall contain 8 (0008h). The SOURCE TARGET DESCRIPTOR INDEX and DESTINATION TARGET DESCRIPTOR INDEX fields are described in 6.3.7.1.

The tape image copy operation terminates when:

a) the source device encounters an end-of-partition as defined by the source device;
b) the source device encounters an end-of-data as defined by the source device (i.e., BLANK CHECK sense key);
c) the copy manager has copied the number of consecutive filemarks specified in the COUNT field from the source device to the destination device; or
d) the copy manager has copied the number of consecutive filemarks and/or setmarks specified in the COUNT field from the source device to the destination device, if the RSMK bit in the Device Configuration mode page (see SSC-2) of the source device is set to one.

A COUNT field of zero specifies that the EXTENDED COPY command shall not terminate due to any number of consecutive filemarks or setmarks. Other error or exception conditions (e.g., early-warning, end-of-partition on destination device) may cause the EXTENDED COPY command to terminate prior to completion. If this occurs, the residue shall not be calculated and the INFORMATION field in the sense data shall be set to zero.

6.3.7.18 Register persistent reservation key operation

The segment descriptor format shown in table 136 instructs the copy manager to register an I_T nexus using the reservation key (see 5.9.7) specified by the RESERVATION KEY field with the SCSI target device specified by the DESTINATION TARGET DESCRIPTOR INDEX field.

Table 136 — Register persistent reservation key segment descriptor

Bit / Byte	7	6	5	4	3	2	1	0
0	DESCRIPTOR TYPE CODE (14h)							
1	Reserved							
2	(MSB)	DESCRIPTOR LENGTH (0018h)						
3								(LSB)
4	Reserved							
5	Reserved							
6	(MSB)	DESTINATION TARGET DESCRIPTOR INDEX						
7								(LSB)
8	(MSB)	RESERVATION KEY						
•••								
15								(LSB)
16	(MSB)	SERVICE ACTION RESERVATION KEY						
•••								
23								(LSB)
24	Reserved							
•••								
27								

The DESCRIPTOR TYPE CODE field is described in 6.3.5 and 6.3.7.1. Descriptor type code 14h instructs the copy manager to register an I_T nexus using the reservation key specified by the RESERVATION KEY field with the SCSI target device identified by the DESTINATION TARGET DESCRIPTOR INDEX field using a PERSISTENT RESERVE OUT command with a REGISTER service action (see 6.14.2).

The DESCRIPTOR LENGTH field shall contain 24 (0018h). The DESTINATION TARGET DESCRIPTOR INDEX field is described in 6.3.7.1.

The RESERVATION KEY and SERVICE ACTION RESERVATION KEY field contents in the PERSISTENT RESERVE OUT command sent to the destination device shall be copied from the RESERVATION KEY and SERVICE ACTION RESERVATION KEY fields in the segment descriptor.

The application client sending the EXTENDED COPY command may need to remove the reservation key held by the copy manager as described in 5.9.11 prior to sending the EXTENDED COPY command.

6.3.7.19 Third party persistent reservations source I_T nexus

The segment descriptor format shown in table 137 instructs the copy manager to send a PERSISTENT RESERVATION OUT command with REGISTER AND MOVE service action (see 5.9.8) with the specified I_T nexus after all other segment descriptors have been processed. If an error is detected any time after receiving a third party persistent source reservation I_T nexus segment descriptor, the PERSISTENT RESERVATION OUT command REGISTER AND MOVE service action shall be processed before status is returned for the EXTENDED COPY command.

This segment descriptor should be placed at or near the beginning of the list of segment descriptors to assure the copy manager processes the PERSISTENT RESERVATION OUT command with REGISTER AND MOVE service action in the event of an error that terminates the processing of segment descriptors. If an error is detected in a segment descriptor and third party persistent reservations source I_T nexus segment descriptor has not been processed, the copy manager shall not send a PERSISTENT RESERVATION OUT command with REGISTER AND MOVE service action.

Placing more than one source third party persistent reservations source I_T nexus segment descriptor in the list of descriptors is not an error. All source third party persistent reservations source I_T nexus segment descriptors known to the copy manager shall be processed after all other segment descriptors have been processed.

Table 137 — Third party persistent reservations source I_T nexus segment descriptor

Bit / Byte	7	6	5	4	3	2	1	0
0	DESCRIPTOR TYPE CODE (15h)							
1	Reserved							
2	(MSB)	DESCRIPTOR LENGTH (n-3)						
3								(LSB)
4	Reserved							
5	Reserved							
6	(MSB)	DESTINATION TARGET DESCRIPTOR INDEX						
7								(LSB)
8	(MSB)	RESERVATION KEY						
•••								
15								(LSB)
16	(MSB)	SERVICE ACTION RESERVATION KEY						
•••								
23								(LSB)
24	Reserved							
25	Reserved						UNREG	APTPL
26	(MSB)	RELATIVE TARGET PORT IDENTIFIER						
27								(LSB)
28	(MSB)	TRANSPORTID PARAMETER DATA LENGTH (n-31)						
•••								
31								(LSB)
32	TransportID							
•••								
n								

The DESCRIPTOR TYPE CODE field is described in 6.3.5 and 6.3.7.1. Descriptor type code 15h instructs the copy manager to send PERSISTENT RESERVATION OUT command with REGISTER AND MOVE service action (see 6.14) to the target port identified by the DESTINATION TARGET DESCRIPTOR INDEX field.

The DESCRIPTOR LENGTH field shall contain the length in bytes of the fields that follow the DESCRIPTOR LENGTH field. The value in the DESCRIPTOR LENGTH field shall be a multiple of 4.

If the PERIPHERAL DEVICE TYPE field in the target descriptor identified by the DESTINATION TARGET DESCRIPTOR INDEX field does not contain 01h, the copy manager shall terminate the command with CHECK CONDITION status, with the sense key set to COPY ABORTED, and the additional sense code set to INVALID OPERATION FOR COPY SOURCE OR DESTINATION.

Bytes 8 to n of the segment descriptor shall be sent as the parameter list (see 6.14.4) for the PERSISTENT RESERVE OUT command with REGISTER AND MOVE service action.

For a description of the RESERVATION KEY field, SERVICE ACTION RESERVATION KEY field, UNREG bit, APTPL bit, RELATIVE TARGET PORT IDENTIFIER field, TRANSPORTID DESCRIPTOR LENGTH field, and TransportID, see 6.14.4.

6.4 INQUIRY command

6.4.1 INQUIRY command introduction

The INQUIRY command (see table 138) requests that information regarding the logical unit and SCSI target device be sent to the application client.

Table 138 — INQUIRY command

Bit / Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (12h)							
1	Reserved						Obsolete	EVPD
2	PAGE CODE							
3	(MSB)	ALLOCATION LENGTH						
4								(LSB)
5	CONTROL							

The OPERATION CODE field is defined in 4.3.5.1 and shall be set as shown in table 138 for the INQUIRY command.

An enable vital product data (EVPD) bit set to one specifies that the device server shall return the vital product data specified by the PAGE CODE field (see 7.8). If the device server does not implement the requested vital product data page, 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.

An EVPD bit is set to zero specified that the device server shall return the standard INQUIRY data (see 6.4.2). If the PAGE CODE field is not set to zero when the EVPD bit is set to zero, 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.

When the EVPD bit is set to one, the PAGE CODE field specifies which page of vital product data information the device server shall return (see 7.8).

The ALLOCATION LENGTH field is defined in 4.3.5.6.

The CONTROL field is defined in SAM-5.

In response to an INQUIRY command received by an incorrect logical unit, the SCSI target device shall return the INQUIRY data with the peripheral qualifier set to the value defined in 6.4.2. The INQUIRY command shall return CHECK CONDITION status only when the device server is unable to return the requested INQUIRY data.

If an INQUIRY command is received from an initiator port with a pending unit attention condition (i.e., before the device server reports CHECK CONDITION status), the device server shall perform the INQUIRY command and shall not clear the unit attention condition (see SAM-4).

The device server should be able to process the INQUIRY command even when an error occurs that prohibits normal command completion.

INQUIRY data (i.e., standard INQUIRY data (see 6.4.2) and all VPD pages (see 7.8)) should be returned even though the device server is not ready for other commands. Standard INQUIRY data, the Extended INQUIRY Data VPD page (see 7.8.7), and the Device Identification VPD page (see 7.8.6) should be available without incurring any

media access delays. If the device server does store some of the standard INQUIRY data or VPD data on the media, it may return ASCII spaces (20h) in ASCII fields and zeros in other fields until the data is available from the media.

INQUIRY data may change as the SCSI target device and its logical units perform their initialization sequence. (E.g., logical units may provide a minimum command set from nonvolatile memory until they load the final firmware from the media. After the firmware has been loaded, more options may be supported and therefore different INQUIRY data may be returned.)

If INQUIRY data changes for any reason, the device server shall establish a unit attention condition for the initiator port associated with every I_T nexus (see SAM-4), with the additional sense code set to INQUIRY DATA HAS CHANGED.

NOTE 32 - The INQUIRY command may be used by an application client after a hard reset or power on condition to determine the device types for system configuration.

6.4.2 Standard INQUIRY data

The standard INQUIRY data (see table 139) shall contain at least 36 bytes.

Table 139 — Standard INQUIRY data format (part 1 of 2)

Bit / Byte	7	6	5	4	3	2	1	0
0	PERIPHERAL QUALIFIER			PERIPHERAL DEVICE TYPE				
1	RMB	Reserved						
2	VERSION							
3	Obsolete	Obsolete	NORMACA	HISUP	RESPONSE DATA FORMAT (2h)			
4	ADDITIONAL LENGTH (n-4)							
5	SCCS	ACC	TPGS		3PC	Reserved		PROTECT
6	Obsolete	ENCSERV	VS	MULTIP	Obsolete	Obsolete	Obsolete	ADDR16[a]
7	Obsolete	Obsolete	WBUS16[a]	SYNC[a]	Obsolete	Obsolete	CMDQUE	VS
8	(MSB)	T10 VENDOR IDENTIFICATION						
•••								
15								(LSB)
16	(MSB)	PRODUCT IDENTIFICATION						
•••								
31								(LSB)
32	(MSB)	PRODUCT REVISION LEVEL						
•••								
35								(LSB)
36	Vendor specific							
•••								
55								
56	Reserved				CLOCKING[a]		QAS[a]	IUS[a]
57	Reserved							

Table 139 — Standard INQUIRY data format (part 2 of 2)

Bit / Byte	7	6	5	4	3	2	1	0
58	(MSB)	VERSION DESCRIPTOR 1						
59								(LSB)
	⋮							
72	(MSB)	VERSION DESCRIPTOR 8						
73								(LSB)
74	Reserved							
•••								
95								
	Vendor specific parameters							
96	Vendor specific							
•••								
n								

[a] The meanings of these fields are specific to SPI-5 (see 6.4.3). For SCSI transport protocols other than the SCSI Parallel Interface, these fields are reserved.

The PERIPHERAL QUALIFIER field and PERIPHERAL DEVICE TYPE field identify the peripheral device connected to the logical unit. If the SCSI target device is not capable of supporting a peripheral device connected to this logical unit, the device server shall set these fields to 7Fh (i.e., PERIPHERAL QUALIFIER field set to 011b and PERIPHERAL DEVICE TYPE field set to 1Fh).

The peripheral qualifier is defined in table 140.

Table 140 — Peripheral qualifier

Qualifier	Description
000b	A peripheral device having the specified peripheral device type is connected to this logical unit. If the device server is unable to determine whether or not a peripheral device is connected, it also shall use this peripheral qualifier. This peripheral qualifier does not mean that the peripheral device connected to the logical unit is ready for access.
001b	A peripheral device having the specified peripheral device type is not connected to this logical unit. However, the device server is capable of supporting the specified peripheral device type on this logical unit.
010b	Reserved
011b	The device server is not capable of supporting a peripheral device on this logical unit. For this peripheral qualifier the peripheral device type shall be set to 1Fh. All other peripheral device type values are reserved for this peripheral qualifier.
100b to 111b	Vendor specific

The peripheral device type is defined in table 141.

Table 141 — Peripheral device type

Code	Reference [a]	Description
00h	SBC-3	Direct access block device (e.g., magnetic disk)
01h	SSC-3	Sequential-access device (e.g., magnetic tape)
02h	SSC	Printer device
03h	SPC-2	Processor device
04h	SBC	Write-once device (e.g., some optical disks)
05h	MMC-5	CD/DVD device
06h		Scanner device (obsolete)
07h	SBC	Optical memory device (e.g., some optical disks)
08h	SMC-3	Media changer device (e.g., jukeboxes)
09h		Communications device (obsolete)
0Ah to 0Bh		Obsolete
0Ch	SCC-2	Storage array controller device (e.g., RAID)
0Dh	SES	Enclosure services device
0Eh	RBC	Simplified direct-access device (e.g., magnetic disk)
0Fh	OCRW	Optical card reader/writer device
10h	BCC	Bridge Controller Commands
11h	OSD	Object-based Storage Device
12h	ADC-2	Automation/Drive Interface
13h	clause 9	Security manager device
14h to 1Dh		Reserved
1Eh	clause 8	Well known logical unit [b]
1Fh		Unknown or no device type

[a] 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 listed standards.

[b] All well known logical units use the same peripheral device type code.

A removable medium (RMB) bit set to zero indicates that the medium is not removable. A RMB bit set to one indicates that the medium is removable.

The VERSION field indicates the implemented version of this standard and is defined in table 142.

Table 142 — Version

Code	Description
00h	The device server does not claim conformance to any standard.
01h to 02h	Obsolete
03h	The device server complies to ANSI INCITS 301-1997 (a withdrawn standard).
04h	The device server complies to ANSI INCITS 351-2001 (SPC-2).
05h	The device server complies to ANSI INCITS 408-2005 (SPC-3).
06h	The device server complies to this standard.
07h	Reserved
08h to 0Ch	Obsolete
0Dh to 3Fh	Reserved
40h to 44h	Obsolete
45h to 47h	Reserved
48h to 4Ch	Obsolete
4Dh to 7Fh	Reserved
80h to 84h	Obsolete
85h to 87h	Reserved
88h to 8Ch	Obsolete
8Dh to FFh	Reserved

The Normal ACA Supported (NORMACA) bit set to one indicates that the device server supports a NACA bit set to one in the CDB CONTROL byte and supports the ACA task attribute (see SAM-4). A NORMACA bit set to zero indicates that the device server does not support a NACA bit set to one and does not support the ACA task attribute.

A hierarchical support (HISUP) bit set to zero indicates the SCSI target device does not use the hierarchical addressing model to assign LUNs to logical units. A HISUP bit set to one indicates the SCSI target device uses the hierarchical addressing model to assign LUNs to logical units.

The RESPONSE DATA FORMAT field indicates the format of the standard INQUIRY data and shall be set as shown in table 139. A RESPONSE DATA FORMAT field set to 2h indicates that the standard INQUIRY data is in the format defined in this standard. Response data format values less than 2h are obsolete. Response data format values greater than 2h are reserved.

The ADDITIONAL LENGTH field indicates the length in bytes of the remaining standard INQUIRY data. The relationship between the ADDITIONAL LENGTH field and the CDB ALLOCATION LENGTH field is defined in 4.3.5.6.

An SCC Supported (SCCS) bit set to one indicates that the SCSI target device contains an embedded storage array controller component that is addressable through this logical unit. See SCC-2 for details about storage array controller devices. An SCCS bit set to zero indicates that no embedded storage array controller component is addressable through this logical unit.

An Access Controls Coordinator (ACC) bit set to one indicates that the SCSI target device contains an access controls coordinator (see 3.1.4) that is addressable through this logical unit. An ACC bit set to zero indicates that no access controls coordinator is addressable through this logical unit. If the SCSI target device contains an access controls coordinator that is addressable through any logical unit other than the ACCESS CONTROLS well known logical unit (see 8.3), then the ACC bit shall be set to one for LUN 0.

The contents of the target port group support (TPGS) field (see table 143) indicate the support for asymmetric logical unit access (see 5.11).

Table 143 — TPGS field

Code	Description
00b	The logical unit does not support asymmetric logical unit access or supports a form of asymmetric access that is vendor specific. Neither the REPORT TARGET GROUPS nor the SET TARGET PORT GROUPS commands is supported.
01b	The logical unit supports only implicit asymmetric logical unit access (see 5.11.2.7). The logical unit is capable of changing target port asymmetric access states without a SET TARGET PORT GROUPS command. The REPORT TARGET PORT GROUPS command is supported and the SET TARGET PORT GROUPS command is not supported.
10b	The logical unit supports only explicit asymmetric logical unit access (see 5.11.2.8). The logical unit only changes target port asymmetric access states as requested with the SET TARGET PORT GROUPS command. Both the REPORT TARGET PORT GROUPS command and the SET TARGET PORT GROUPS command are supported.
11b	The logical unit supports both explicit and implicit asymmetric logical unit access. Both the REPORT TARGET PORT GROUPS command and the SET TARGET PORT GROUPS commands are supported.

A Third-Party Copy (3PC) bit set to one indicates that the SCSI target device contains a copy manager that is addressable through this logical unit. A 3PC bit set to zero indicates that no copy manager is addressable through this logical unit.

A PROTECT bit set to zero indicates that the logical unit does not support protection information. A PROTECT bit set to one indicates that the logical unit supports:

a) type 1 protection, type 2 protection, or type 3 protection (see SBC-3); or
b) logical block protection (see SSC-4).

More information about the type of protection the logical unit supports is available in the SPT field (see 7.8.7).

An Enclosure Services (ENCSERV) bit set to one indicates that the SCSI target device contains an embedded enclosure services component that is addressable through this logical unit. See SES-3 for details about enclosure services. An ENCSERV bit set to zero indicates that no embedded enclosure services component is addressable through this logical unit.

A Multi Port (MULTIP) bit set to one indicates that this is a multi-port (two or more ports) SCSI target device and conforms to the SCSI multi-port device requirements found in the applicable standards (e.g., SAM-4, a SCSI transport protocol standard and possibly provisions of a command standard). A MULTIP bit set to zero indicates that this SCSI target device has a single port and does not implement the multi-port requirements.

The CMDQUE bit shall be set to one indicating that the logical unit supports the command management model defined in SAM-4.

The T10 VENDOR IDENTIFICATION field contains eight bytes of left-aligned ASCII data (see 4.4.1) identifying the vendor of the logical unit. The T10 vendor identification shall be one assigned by INCITS. A list of assigned T10 vendor identifications is in Annex E and on the T10 web site (http://www.t10.org).

NOTE 33 - The T10 web site (http://www.t10.org) provides a convenient means to request an identification code.

The PRODUCT IDENTIFICATION field contains sixteen bytes of left-aligned ASCII data (see 4.4.1) defined by the vendor.

The PRODUCT REVISION LEVEL field contains four bytes of left-aligned ASCII data defined by the vendor.

The VERSION DESCRIPTOR fields provide for identifying up to eight standards to which the SCSI target device and/or logical unit claim conformance. The value in each VERSION DESCRIPTOR field shall be selected from table 144. All version descriptor values not listed in table 144 are reserved. Technical Committee T10 of INCITS maintains an electronic copy of the information in table 144 on its world wide web site (http://www.t10.org/). In the event that the T10 world wide web site is no longer active, access may be possible via the INCITS world wide web site (http://www.incits.org), the ANSI world wide web site (http://www.ansi.org), the IEC site (http://www.iec.ch/), the ISO site (http://www.iso.ch/), or the ISO/IEC JTC 1 web site (http://www.jtc1.org/). It is recommended that the first version descriptor be used for the SCSI architecture standard, followed by the physical transport standard if any, followed by the SCSI transport protocol standard, followed by the appropriate SPC-x version, followed by the device type command set, followed by a secondary command set if any.

Table 144 — Version descriptor values (part 1 of 11)

Standard	Version Descriptor Value
ACS-2 (no version claimed)	1761h
ADC (no version claimed)	03C0h
ADC ANSI INCITS 403-2005	03D7h
ADC T10/1558-D revision 7	03D6h
ADC T10/1558-D revision 6	03D5h
ADC-2 (no version claimed)	04A0h
ADC-2 ANSI INCITS 441-2008	04ACh
ADC-2 T10/1741-D revision 7	04A7h
ADC-2 T10/1741-D revision 8	04AAh
ADC-3 (no version claimed)	0500h
ADC-3 T10/1895-D revision 04	0502h
ADP (no version claimed)	09C0h
ADT (no version claimed)	09E0h
ADT ANSI INCITS 406-2005	09FDh
ADT T10/1557-D revision 14	09FAh
ADT T10/1557-D revision 11	09F9h
ADT-2 (no version claimed)	0A20h
ADT-2 T10/1742-D revision 06	0A22h
ADT-2 T10/1742-D revision 08	0A27h
ADT-2 T10/1742-D revision 09	0A28h
ATA/ATAPI-6 (no version claimed)	15E0h
ATA/ATAPI-6 ANSI INCITS 361-2002	15FDh
ATA/ATAPI-7 (no version claimed)	1600h
ATA/ATAPI-7 ANSI INCITS 397-2005	161Ch
ATA/ATAPI-7 T13/1532-D revision 3	1602h
A numeric ordered listing of the version descriptor value assignments is provided in D.8.	

Table 144 — Version descriptor values (part 2 of 11)

Standard	Version Descriptor Value
ATA/ATAPI-8 ATA8-AAM (no version claimed)	1620h
ATA/ATAPI-8 ATA8-AAM ANSI INCITS 451-2008	1628h
ATA/ATAPI-8 ATA8-APT Parallel Transport (no version claimed)	1621h
ATA/ATAPI-8 ATA8-AST Serial Transport (no version claimed)	1622h
ATA/ATAPI-8 ATA8-ACS ATA/ATAPI Command Set (no version claimed)	1623h
ATA/ATAPI-8 ATA8-ACS ANSI INCITS 452-2009 w/ Amendment 1	162Ah
BCC (no version claimed)	0380h
EPI (no version claimed)	0B20h
EPI ANSI INCITS TR-23 1999	0B3Ch
EPI T10/1134 revision 16	0B3Bh
Fast-20 (no version claimed)	0AC0h
Fast-20 ANSI INCITS 277-1996	0ADCh
Fast-20 T10/1071 revision 06	0ADBh
FC 10GFC (no version claimed)	0EA0h
FC 10GFC ISO/IEC 14165-116	0EA3h
FC 10GFC ANSI INCITS 364-2003	0EA2h
FC 10GFC ISO/IEC 14165-116 with AM1	0EA5h
FC 10GFC ANSI INCITS 364-2003 with AM1 ANSI INCITS 364/AM1-2007	0EA6h
FC-AL (no version claimed)	0D40h
FC-AL ANSI INCITS 272-1996	0D5Ch
FC-AL-2 (no version claimed)	0D60h
FC-AL-2 ISO/IEC 14165-122 with AM1 & AM2	0D65h
FC-AL-2 ANSI INCITS 332-1999 with AM1-2003 & AM2-2006	0D63h
FC-AL-2 ANSI INCITS 332-1999 with Amnd 2 AM2-2006	0D64h
FC-AL-2 ANSI INCITS 332-1999 with Amnd 1 AM1-2003	0D7Dh
FC-AL-2 ANSI INCITS 332-1999	0D7Ch
FC-AL-2 T11/1133-D revision 7.0	0D61h
FC-DA (no version claimed)	12E0h
FC-DA ISO/IEC 14165-341	12E9h
FC-DA ANSI INCITS TR-36 2004	12E8h
FC-DA T11/1513-DT revision 3.1	12E2h
FC-DA-2 (no version claimed)	12C0h
FC-DA-2 T11/1870DT revision 1.04	12C3h
FC-DA-2 T11/1870DT revision 1.06	12C5h
FC-FLA (no version claimed)	1320h
FC-FLA ANSI INCITS TR-20 1998	133Ch
FC-FLA T11/1235 revision 7	133Bh
FC-FS (no version claimed)	0DA0h
A numeric ordered listing of the version descriptor value assignments is provided in D.8.	

Table 144 — Version descriptor values (part 3 of 11)

Standard	Version Descriptor Value
FC-FS ISO/IEC 14165-251	0DBDh
FC-FS ANSI INCITS 373-2003	0DBCh
FC-FS T11/1331-D revision 1.2	0DB7h
FC-FS T11/1331-D revision 1.7	0DB8h
FC-FS-2 (no version claimed)	0E00h
FC-FS-2 ANSI INCITS 242-2007 with AM1 ANSI INCITS 242/AM1-2007	0E03h
FC-FS-2 ANSI INCITS 242-2007	0E02h
FC-FS-3 (no version claimed)	0EE0h
FC-FS-3 T11/1861-D revision 0.9	0EE2h
FC-FS-3 T11/1861-D revision 1.0	0EE7h
FC-FS-3 T11/1861-D revision 1.10	0EE9h
FC-FS-4 (no version claimed)	0F60h
FC-LS (no version claimed)	0E20h
FC-LS ANSI INCITS 433-2007	0E29h
FC-LS T11/1620-D revision 1.62	0E21h
FC-LS-2 (no version claimed)	0F00h
FC-LS-2 T11/2103-D revision 2.11	0F03h
FC-LS-2 T11/2103-D revision 2.21	0F05h
FC-LS-3 (no version claimed)	0F80h
FCP (no version claimed)	08C0h
FCP ANSI INCITS 269-1996	08DCh
FCP T10/0993-D revision 12	08DBh
FC-PH (no version claimed)	0D20h
FC-PH ANSI INCITS 230-1994	0D3Bh
FC-PH ANSI INCITS 230-1994 with Amnd 1 ANSI INCITS 230/AM1-1996	0D3Ch
FC-PH-3 (no version claimed)	0D80h
FC-PH-3 ANSI INCITS 303-1998	0D9Ch
FC-PI (no version claimed)	0DC0h
FC-PI ANSI INCITS 352-2002	0DDCh
FC-PI-2 (no version claimed)	0DE0h
FC-PI-2 ANSI INCITS 404-2006	0DE4h
FC-PI-2 T11/1506-D revision 5.0	0DE2h
FC-PI-3 (no version claimed)	0E60h
FC-PI-3 T11/1625-D revision 2.0	0E62h
FC-PI-3 T11/1625-D revision 2.1	0E68h
FC-PI-3 T11/1625-D revision 4.0	0E6Ah
FC-PI-4 (no version claimed)	0E80h
FC-PI-4 ANSI INCITS 450-2009	0E88h
A numeric ordered listing of the version descriptor value assignments is provided in D.8.	

Table 144 — Version descriptor values (part 4 of 11)

Standard	Version Descriptor Value
FC-PI-4 T11/1647-D revision 8.0	0E82h
FC-PI-5 (no version claimed)	0F20h
FC-PI-5 T11/2118-D revision 2.00	0F27h
FC-PI-5 T11/2118-D revision 3.00	0F28h
FC-PI-5 T11/2118-D revision 6.00	0F2Ah
FC-PI-5 T11/2118-D revision 6.10	0F2Bh
FC-PI-6 (no version claimed)	0F40h
FC-PLDA (no version claimed)	1340h
FC-PLDA ANSI INCITS TR-19 1998	135Ch
FC-PLDA T11/1162 revision 2.1	135Bh
FCP-2 (no version claimed)	0900h
FCP-2 ANSI INCITS 350-2003	0917h
FCP-2 T10/1144-D revision 8	0918h
FCP-2 T10/1144-D revision 4	0901h
FCP-2 T10/1144-D revision 7	0915h
FCP-2 T10/1144-D revision 7a	0916h
FCP-3 (no version claimed)	0A00h
FCP-3 ISO/IEC 14776-223	0A1Ch
FCP-3 ANSI INCITS 416-2006	0A11h
FCP-3 T10/1560-D revision 4	0A0Fh
FCP-3 T10/1560-D revision 3f	0A07h
FCP-4 (no version claimed)	0A40h
FCP-4 T10/1828-D revision 01	0A42h
FCP-4 T10/1828-D revision 02	0A44h
FCP-4 T10/1828-D revision 02b	0A45h
FC-SCM (no version claimed)	12A0h
FC-SCM T11/1824DT revision 1.0	12A3h
FC-SCM T11/1824DT revision 1.1	12A5h
FC-SCM T11/1824DT revision 1.4	12A7h
FC-SP (no version claimed)	0E40h
FC-SP ANSI INCITS 426-2007	0E45h
FC-SP T11/1570-D revision 1.6	0E42h
FC-SP-2 (no version claimed)	0EC0h
FC-Tape (no version claimed)	1300h
FC-Tape ANSI INCITS TR-24 1999	131Ch
FC-Tape T11/1315 revision 1.17	131Bh
FC-Tape T11/1315 revision 1.16	1301h
IEEE 1394 (no version claimed)	14A0h
A numeric ordered listing of the version descriptor value assignments is provided in D.8.	

Table 144 — Version descriptor values (part 5 of 11)

Standard	Version Descriptor Value
ANSI IEEE 1394-1995	14BDh
IEEE 1394a (no version claimed)	14C0h
IEEE 1394b (no version claimed)	14E0h
IEEE 1667 (no version claimed)	FFC0h
IEEE 1667-2006	FFC1h
IEEE 1667-2009	FFC2h
iSCSI (no version claimed)	0960h
MMC (no version claimed)	0140h
MMC ANSI INCITS 304-1997	015Ch
MMC T10/1048-D revision 10a	015Bh
MMC-2 (no version claimed)	0240h
MMC-2 ANSI INCITS 333-2000	025Ch
MMC-2 T10/1228-D revision 11a	025Bh
MMC-2 T10/1228-D revision 11	0255h
MMC-3 (no version claimed)	02A0h
MMC-3 ANSI INCITS 360-2002	02B8h
MMC-3 T10/1363-D revision 10g	02B6h
MMC-3 T10/1363-D revision 9	02B5h
MMC-4 (no version claimed)	03A0h
MMC-4 ANSI INCITS 401-2005	03BFh
MMC-4 T10/1545-D revision 3	03BDh
MMC-4 T10/1545-D revision 3d	03BEh
MMC-4 T10/1545-D revision 5	03B0h
MMC-4 T10/1545-D revision 5a	03B1h
MMC-5 (no version claimed)	0420h
MMC-5 T10/1675-D revision 04	0432h
MMC-5 ANSI INCITS 430-2007	0434h
MMC-5 T10/1675-D revision 03	042Fh
MMC-5 T10/1675-D revision 03b	0431h
MMC-6 (no version claimed)	04E0h
MMC-6 T10/1836-D revision 02b	04E3h
MMC-6 T10/1836-D revision 02g	04E5h
MMC-6 ANSI INCITS 468-2010	04E6h
OCRW (no version claimed)	0280h
OCRW ISO/IEC 14776-381	029Eh
OSD (no version claimed)	0340h
OSD ANSI INCITS 400-2004	0356h
OSD T10/1355-D revision 10	0355h
A numeric ordered listing of the version descriptor value assignments is provided in D.8.	

Table 144 — Version descriptor values (part 6 of 11)

Standard	Version Descriptor Value
OSD T10/1355-D revision 0	0341h
OSD T10/1355-D revision 7a	0342h
OSD T10/1355-D revision 8	0343h
OSD T10/1355-D revision 9	0344h
OSD-2 (no version claimed)	0440h
OSD-2 ANSI INCITS 458-2011	0448h
OSD-2 T10/1729-D revision 4	0444h
OSD-2 T10/1729-D revision 5	0446h
OSD-3 (no version claimed)	0560h
PQI (no version claimed)	2200h
RBC (no version claimed)	0220h
RBC ANSI INCITS 330-2000	023Ch
RBC T10/1240-D revision 10a	0238h
SAM (no version claimed)	0020h
SAM ANSI INCITS 270-1996	003Ch
SAM T10/0994-D revision 18	003Bh
SAM-2 (no version claimed)	0040h
SAM-2 ISO/IEC 14776-412	005Eh
SAM-2 ANSI INCITS 366-2003	005Ch
SAM-2 T10/1157-D revision 24	0055h
SAM-2 T10/1157-D revision 23	0054h
SAM-3 (no version claimed)	0060h
SAM-3 ANSI INCITS 402-2005	0077h
SAM-3 T10/1561-D revision 14	0076h
SAM-3 T10/1561-D revision 7	0062h
SAM-3 T10/1561-D revision 13	0075h
SAM-4 (no version claimed)	0080h
SAM-4 ISO/IEC 14776-414	0092h
SAM-4 ANSI INCITS 447-2008	0090h
SAM-4 T10/1683-D revision 13	0087h
SAM-4 T10/1683-D revision 14	008Bh
SAM-5 (no version claimed)	00A0h
SAM-5 T10/2104-D revision 4	00A2h
SAS (no version claimed)	0BE0h
SAS ANSI INCITS 376-2003	0BFDh
SAS T10/1562-D revision 05	0BFCh
SAS T10/1562-D revision 01	0BE1h
SAS T10/1562-D revision 03	0BF5h
A numeric ordered listing of the version descriptor value assignments is provided in D.8.	

Table 144 — Version descriptor values (part 7 of 11)

Standard	Version Descriptor Value
SAS T10/1562-D revision 04	0BFAh
SAS T10/1562-D revision 04	0BFBh
SAS-1.1 (no version claimed)	0C00h
SAS-1.1 ANSI INCITS 417-2006	0C11h
SAS-1.1 T10/1601-D revision 10	0C0Fh
SAS-1.1 T10/1601-D revision 9	0C07h
SAS-2 (no version claimed)	0C20h
SAS-2 ANSI INCITS 457-2010	0C2Ah
SAS-2 T10/1760-D revision 14	0C23h
SAS-2 T10/1760-D revision 15	0C27h
SAS-2 T10/1760-D revision 16	0C28h
SAS-2.1 (no version claimed)	0C40h
SAS-2.1 T10/2125-D revision 04	0C48h
SAS-2.1 T10/2125-D revision 06	0C4Ah
SAS-2.1 T10/2125-D revision 07	0C4Bh
SAS-3 (no version claimed)	0C60h
SAT (no version claimed)	1EA0h
SAT ANSI INCITS 431-2007	1EADh
SAT T10/1711-D revision 9	1EABh
SAT T10/1711-D revision 8	1EA7h
SAT-2 (no version claimed)	1EC0h
SAT-2 T10/1826-D revision 06	1EC4h
SAT-2 T10/1826-D revision 09	1EC8h
SAT-2 ANSI INCITS 465-2010	1ECAh
SAT-3 (no version claimed)	1EE0h
SBC (no version claimed)	0180h
SBC ANSI INCITS 306-1998	019Ch
SBC T10/0996-D revision 08c	019Bh
SBC-2 (no version claimed)	0320h
SBC-2 ISO/IEC 14776-322	033Eh
SBC-2 ANSI INCITS 405-2005	033Dh
SBC-2 T10/1417-D revision 16	033Bh
SBC-2 T10/1417-D revision 5a	0322h
SBC-2 T10/1417-D revision 15	0324h
SBC-3 (no version claimed)	04C0h
SBP-2 (no version claimed)	08E0h
SBP-2 ANSI INCITS 325-1998	08FCh
SBP-2 T10/1155-D revision 04	08FBh
A numeric ordered listing of the version descriptor value assignments is provided in D.8.	

Table 144 — Version descriptor values (part 8 of 11)

Standard	Version Descriptor Value
SBP-3 (no version claimed)	0980h
SBP-3 ANSI INCITS 375-2004	099Ch
SBP-3 T10/1467-D revision 5	099Bh
SBP-3 T10/1467-D revision 1f	0982h
SBP-3 T10/1467-D revision 3	0994h
SBP-3 T10/1467-D revision 4	099Ah
SCC (no version claimed)	0160h
SCC ANSI INCITS 276-1997	017Ch
SCC T10/1047-D revision 06c	017Bh
SCC-2 (no version claimed)	01E0h
SCC-2 ANSI INCITS 318-1998	01FCh
SCC-2 T10/1125-D revision 04	01FBh
SES (no version claimed)	01C0h
SES ANSI INCITS 305-1998	01DCh
SES T10/1212-D revision 08b	01DBh
SES ANSI INCITS 305-1998 w/ Amendment ANSI INCITS.305/AM1-2000	01DEh
SES T10/1212 revision 08b w/ Amendment ANSI INCITS.305/AM1-2000	01DDh
SES-2 (no version claimed)	03E0h
SES-2 ANSI INCITS 448-2008	03F0h
SES-2 T10/1559-D revision 16	03E1h
SES-2 T10/1559-D revision 19	03E7h
SES-2 T10/1559-D revision 20	03EBh
SES-3 (no version claimed)	0580h
SIP (no version claimed)	08A0h
SIP ANSI INCITS 292-1997	08BCh
SIP T10/0856-D revision 10	08BBh
SMC (no version claimed)	01A0h
SMC ISO/IEC 14776-351	01BEh
SMC ANSI INCITS 314-1998	01BCh
SMC T10/0999-D revision 10a	01BBh
SMC-2 (no version claimed)	02E0h
SMC-2 ANSI INCITS 382-2004	02FEh
SMC-2 T10/1383-D revision 7	02FDh
SMC-2 T10/1383-D revision 5	02F5h
SMC-2 T10/1383-D revision 6	02FCh
SMC-3 (no version claimed)	0480h
SMC-3 T10/1730-D revision 15	0482h
SOP (no version claimed)	21E0h
A numeric ordered listing of the version descriptor value assignments is provided in D.8.	

Table 144 — Version descriptor values (part 9 of 11)

Standard	Version Descriptor Value
SPC (no version claimed)	0120h
SPC ANSI INCITS 301-1997	013Ch
SPC T10/0995-D revision 11a	013Bh
SPC-2 (no version claimed)	0260h
SPC-2 ISO/IEC 14776-452	0278h
SPC-2 ANSI INCITS 351-2001	0277h
SPC-2 T10/1236-D revision 20	0276h
SPC-2 T10/1236-D revision 12	0267h
SPC-2 T10/1236-D revision 18	0269h
SPC-2 T10/1236-D revision 19	0275h
SPC-3 (no version claimed)	0300h
SPC-3 ISO/IEC 14776-453	0316h
SPC-3 ANSI INCITS 408-2005	0314h
SPC-3 T10/1416-D revision 7	0301h
SPC-3 T10/1416-D revision 21	0307h
SPC-3 T10/1416-D revision 22	030Fh
SPC-3 T10/1416-D revision 23	0312h
SPC-4 (no version claimed)	0460h
SPC-4 T10/1731-D revision 16	0461h
SPC-4 T10/1731-D revision 18	0462h
SPC-4 T10/1731-D revision 23	0463h
SPI (no version claimed)	0AA0h
SPI ANSI INCITS 253-1995	0ABAh
SPI T10/0855-D revision 15a	0AB9h
SPI ANSI INCITS 253-1995 with SPI Amnd ANSI INCITS 253/AM1-1998	0ABCh
SPI T10/0855-D revision 15a with SPI Amnd revision 3a	0ABBh
SPI-2 (no version claimed)	0AE0h
SPI-2 ANSI INCITS 302-1999	0AFCh
SPI-2 T10/1142-D revision 20b	0AFBh
SPI-3 (no version claimed)	0B00h
SPI-3 ANSI INCITS 336-2000	0B1Ch
SPI-3 T10/1302-D revision 14	0B1Ah
SPI-3 T10/1302-D revision 10	0B18h
SPI-3 T10/1302-D revision 13a	0B19h
SPI-4 (no version claimed)	0B40h
SPI-4 ANSI INCITS 362-2002	0B56h
SPI-4 T10/1365-D revision 7	0B54h
SPI-4 T10/1365-D revision 9	0B55h
A numeric ordered listing of the version descriptor value assignments is provided in D.8.	

Table 144 — Version descriptor values (part 10 of 11)

Standard	Version Descriptor Value
SPI-4 T10/1365-D revision 10	0B59h
SPI-5 (no version claimed)	0B60h
SPI-5 ANSI INCITS 367-2003	0B7Ch
SPI-5 T10/1525-D revision 6	0B7Bh
SPI-5 T10/1525-D revision 3	0B79h
SPI-5 T10/1525-D revision 5	0B7Ah
SPL (no version claimed)	20A0h
SPL T10/2124-D revision 6a	20A3h
SPL T10/2124-D revision 7	20A5h
SPL-2 (no version claimed)	20C0h
SRP (no version claimed)	0940h
SRP ANSI INCITS 365-2002	095Ch
SRP T10/1415-D revision 16a	0955h
SRP T10/1415-D revision 10	0954h
SSA-PH2 (no version claimed)	1360h
SSA-PH2 ANSI INCITS 293-1996	137Ch
SSA-PH2 T10.1/1145-D revision 09c	137Bh
SSA-PH3 (no version claimed)	1380h
SSA-PH3 ANSI INCITS 307-1998	139Ch
SSA-PH3 T10.1/1146-D revision 05b	139Bh
SSA-S2P (no version claimed)	0880h
SSA-S2P ANSI INCITS 294-1996	089Ch
SSA-S2P T10.1/1121-D revision 07b	089Bh
SSA-S3P (no version claimed)	0860h
SSA-S3P ANSI INCITS 309-1998	087Ch
SSA-S3P T10.1/1051-D revision 05b	087Bh
SSA-TL1 (no version claimed)	0840h
SSA-TL1 ANSI INCITS 295-1996	085Ch
SSA-TL1 T10.1/0989-D revision 10b	085Bh
SSA-TL2 (no version claimed)	0820h
SSA-TL2 ANSI INCITS 308-1998	083Ch
SSA-TL2 T10.1/1147-D revision 05b	083Bh
SSC (no version claimed)	0200h
SSC ANSI INCITS 335-2000	021Ch
SSC T10/0997-D revision 22	0207h
SSC T10/0997-D revision 17	0201h
SSC-2 (no version claimed)	0360h
SSC-2 ANSI INCITS 380-2003	037Dh
A numeric ordered listing of the version descriptor value assignments is provided in D.8.	

Table 144 — Version descriptor values (part 11 of 11)

Standard	Version Descriptor Value
SSC-2 T10/1434-D revision 9	0375h
SSC-2 T10/1434-D revision 7	0374h
SSC-3 (no version claimed)	0400h
SSC-3 T10/1611-D revision 04a	0403h
SSC-3 T10/1611-D revision 05	0407h
SSC-4 (no version claimed)	0520h
SST (no version claimed)	0920h
SST T10/1380-D revision 8b	0935h
UAS (no version claimed)	1740h
UAS T10/2095-D revision 02	1743h
UAS T10/2095-D revision 04	1747h
UAS ANSI INCITS 471-2010	1748h
UAS-2 (no version claimed)	1780h
Universal Serial Bus Specification, Revision 1.1	1728h
Universal Serial Bus Specification, Revision 2.0	1729h
USB Mass Storage Class Bulk-Only Transport, Revision 1.0	1730h
Version Descriptor Not Supported or No Standard Identified	0000h
Reserved	all others
A numeric ordered listing of the version descriptor value assignments is provided in D.8.	

6.4.3 SCSI Parallel Interface specific INQUIRY data

Portions of bytes 6 and 7 and all of byte 56 of the standard INQUIRY data shall be used only by SCSI target devices that implement the SCSI Parallel Interface. These fields are noted in table 139. For details on how the SPI-specific fields relate to the SCSI Parallel Interface see SPI-n (where n is 2 or greater). Table 145 shows just the SPI-specific standard INQUIRY fields. The definitions of the SCSI Parallel Interface specific fields shall be as follows.

Table 145 — SPI-specific standard INQUIRY bits

Bit / Byte	7	6	5	4	3	2	1	0
6	see table 139							ADDR16
7	see table 139		WBUS16	SYNC	see table 139	Obsolete	see table 139	
	⋮							
56	Reserved				CLOCKING		QAS	IUS

A wide SCSI address 16 (ADDR16) bit of one indicates that the SCSI target device supports 16-bit wide SCSI addresses. A value of zero indicates that the SCSI target device does not support 16-bit wide SCSI addresses.

A wide bus 16 (WBUS16) bit of one indicates that the SCSI target device supports 16-bit wide data transfers. A value of zero indicates that the SCSI target device does not support 16-bit wide data transfers.

A synchronous transfer (SYNC) bit of one indicates that the SCSI target device supports synchronous data transfer. A value of zero indicates the SCSI target device does not support synchronous data transfer.

The obsolete bit 2 in byte 7 indicates whether the SCSI target device supports an obsolete data transfers management mechanism defined in SPI-2.

Table 146 defines the relationships between the ADDR16 and WBUS16 bits.

Table 146 — Maximum logical device configuration table

ADDR16	WBUS16	Description
0	0	8 bit wide data path on a single cable with 8 SCSI IDs supported
0	1	16 bit wide data path on a single cable with 8 SCSI IDs supported
1	1	16 bit wide data path on a single cable with 16 SCSI IDs supported

The CLOCKING field shall not apply to asynchronous transfers and is defined in table 147.

Table 147 — CLOCKING field

Code	Description
00b	Indicates the target port supports only ST
01b	Indicates the target port supports only DT
10b	Reserved
11b	Indicates the target port supports ST and DT

A quick arbitration and selection supported (QAS) bit of one indicates that the target port supports quick arbitration and selection. A value of zero indicates that the target port does not support quick arbitration and selection.

An information units supported (IUS) bit of one indicates that the SCSI target device supports information unit transfers. A value of zero indicates that the SCSI target device does not support information unit transfers.

NOTE 34 - The acronyms ST and DT and the terms 'quick arbitration and selection' and 'information units' are defined in SPI-5.

6.5 LOG SELECT command

6.5.1 Introduction

The LOG SELECT command (see table 148) provides a means for an application client to manage statistical information maintained by the SCSI target device about the SCSI target device or its logical units. Device servers that implement the LOG SELECT command shall also implement the LOG SENSE command. Structures in the form of log parameters within log pages are defined as a way to manage the log data. The LOG SELECT command provides a method for sending zero or more log pages via the Data-Out Buffer. This standard defines the format of the log pages (see 7.3) and defines some of the conditions and events that are logged.

Table 148 — LOG SELECT command

Bit / Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (4Ch)							
1	Reserved						PCR	SP
2	PC		PAGE CODE					
3	SUBPAGE CODE							
4	Reserved							
•••								
6								
7	(MSB)	PARAMETER LIST LENGTH						
8								(LSB)
9	CONTROL							

The OPERATION CODE field is defined in 4.3.5.1 and shall be set as shown in table 148 for the LOG SELECT command.

The parameter code reset (PCR) bit instructs a device server whether or not to set parameters to their vendor specific default values (e.g., zero) as described in table 151.

The save parameters (SP) bit instructs a device server whether or not to save parameters to non-volatile memory as described in table 151.

The page control (PC) field specifies which values the device server shall process for bounded data counter log parameters (see 7.3.2.2.2.2) and unbounded data counter log parameters (see 7.3.2.2.2.3) in response to a LOG SELECT command as described in table 149. The PC field shall be ignored for ASCII format list log parameters (see 7.3.2.2.2.4) and binary format list log parameters (see 7.3.2.2.2.5).

Table 149 — Page control (PC) field

Value	Description
00b	Threshold values [a]
01b	Cumulative values [a]
10b	Default threshold values
11b	Default cumulative values

[a] The threshold values and cumulative values for data counter parameters are:
1) the current values if there has been an update to a cumulative parameter value (e.g., by a LOG SELECT command or by a device specific event) in the specified page or pages since the last logical unit reset occurred;
2) the saved values, if saved parameters are implemented, current values have been saved, and an update has not occurred since the last logical unit reset; or
3) the vendor specific default values, if saved values are not available or not implemented.

When evaluated together, the combination of the values in the PCR bit, the SP bit, and the PC field specify the actions that a SCSI target device performs while processing a LOG SELECT command.

If the PARAMETER LIST LENGTH field is set to zero, the PAGE CODE field and SUBPAGE CODE field specify the log page or log pages to which the other CDB fields apply (see 6.5.2).

Since each log page in the parameter list contains a PAGE CODE field and SUBPAGE CODE field (see 7.3.2.1), 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:

a) the PARAMETER LIST LENGTH field contains a value other than zero, and:
   A) the PAGE CODE field contains a value other than zero; or
   B) the SUBPAGE CODE field contains a value other than zero.

The PARAMETER LIST LENGTH field specifies the length in bytes of the parameter list that shall be located in the Data-Out Buffer.

If the PARAMETER LIST LENGTH field contains a value other than zero, the actions that a SCSI target device performs after receiving a LOG SELECT command are determined by the values in the PCR bit, the SP bit, and the PC field as described in table 298 (see 7.3.2.1).

If the PARAMETER LIST LENGTH field contains zero, no log pages shall be transferred. This condition shall not be considered an error. The LOG SELECT command shall be processed as described in 6.5.2.

The CONTROL field is defined in SAM-5.

6.5.2 Processing LOG SELECT when the parameter list length is zero

If the PARAMETER LIST LENGTH field is set to zero (i.e., when there is no parameter data being sent with a LOG SELECT command), the SCSI target device responds by processing the log parameter values as described in this subclause.

The PAGE CODE field and SUBPAGE CODE field (see table 150) specify the log page or log pages to which the other CDB fields apply (see table 151).

Table 150 — PAGE CODE field and SUBPAGE CODE field

PAGE CODE field	SUBPAGE CODE field	Description
00h	00h	All log parameters in all log pages [a]
00h to 3Fh	01h to FEh	All log parameters in the log page specified by the page code and subpage code
00h to 3Fh	FFh	All log parameters in the log pages specified by page code and all subpage codes
01h to 3Fh	00h	All log parameters in the log page specified by the page code

[a] This is equivalent to the LOG SELECT command operation specified by SPC-3.

Table 151 defines the meaning of the combinations of values for the PCR bit, the SP bit, and the PC field.

Table 151 — PCR bit, SP bit, and PC field meanings when parameter list length is zero (part 1 of 3)

PCR bit	SP bit	PC field	Description
0b	0b	0xb	This is not an error. The device server shall make no changes to any log parameter values and shall not save any values to non-volatile media.
0b	1b	00b	The device server shall make no changes to any log parameter values and shall process the optional saving of current parameter values as follows: a) if the values are current threshold data counter parameters, then: A) if the device server implements saving of the current threshold values, the device server shall save all current threshold values to non-volatile media; or B) if the device server does not implement saving of the current threshold values, the device server shall terminate the command[a]. or b) if the values are current list parameters, then: A) if the device server implements saving of the current list parameters, the device server shall save all current list parameters to non-volatile media; or B) if the device server does not implement saving of the current list parameters, the device server shall terminate the command[a].

[a] 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.

[b] Vendor specific default threshold values and vendor specific default cumulative values may be zero.

Table 151 — PCR bit, SP bit, and PC field meanings when parameter list length is zero (part 2 of 3)

PCR bit	SP bit	PC field	Description
0b	1b	01b	The device server shall make no change to any log parameter values and shall process the optional saving of current parameter values as follows: a) if the values are current cumulative data counter parameters, then: A) if the device server implements saving of the current cumulative values, the device server shall save all current cumulative values to non-volatile media; or B) if the device server does not implement saving of the current cumulative values, the device server shall terminate the command[a]. or b) if the values are current list parameters, then: A) if the device server implements saving of the current list parameters, the device server shall save all current list parameters to non-volatile media; or B) if the device server does not implement saving of the current list parameters, the device server shall terminate the command[a].
0b	xb	10b	The device server shall set all current threshold values to the vendor specific default threshold values[b] and shall not save any values to non-volatile media.
0b	xb	11b	The device server shall set all current cumulative values to the vendor specific default cumulative values[b] and shall not save any values to non-volatile media.
1b	0b	xxb	The device server shall: 1) set all current threshold values to the vendor specific default threshold values[b]; 2) set all current cumulative values to the vendor specific default cumulative values[b]; 3) set all list parameters to their vendor specific default values; and 4) not save any values to non-volatile media.
1b	1b	00b	The device server shall process the optional saving of current threshold values as follows: a) if the device server implements saving of the current threshold values, the device server shall: 1) save all current threshold values to non-volatile media; Note: Device servers compliant with SPC-3 may save the log parameter values to non-volatile media after setting the log parameter values instead of before setting the log parameter values. 2) set all current threshold values to the vendor specific default threshold values[b]; 3) set all current cumulative values to the vendor specific default cumulative values[b], and 4) set all list parameters to their vendor specific default values. or b) if the device server does not implement saving of the current threshold values, the device server shall terminate the command[a].

[a] 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.

[b] Vendor specific default threshold values and vendor specific default cumulative values may be zero.

Table 151 — PCR bit, SP bit, and PC field meanings when parameter list length is zero (part 3 of 3)

PCR bit	SP bit	PC field	Description
1b	1b	01b	The device server shall process the optional saving of current cumulative values as follows: a) if the device server implements saving of the current cumulative values, the device server shall: 1) save all current cumulative values to non-volatile media; Note: Device servers compliant with SPC-3 may save the log parameter values to non-volatile media after setting the log parameter values instead of before setting the log parameter values. 2) set all current threshold values to the vendor specific default threshold values[b]; 3) set all current cumulative values to the vendor specific default cumulative values[b], and 4) set all list parameters to their vendor specific default values. or b) if the device server does not implement saving of the current cumulative values, the device server shall terminate the command[a].
1b	1b	1xb	The device server shall: 1) set all current threshold values to the vendor specific default threshold values[b]; 2) set all current cumulative values to the vendor specific default cumulative values[b]; 3) set all list parameters to their vendor specific default values; and 4) not save any values to non-volatile media.

[a] 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.

[b] Vendor specific default threshold values and vendor specific default cumulative values may be zero.

The current cumulative values may be updated by the device server as defined for the specific log page or by the application client using the LOG SELECT command. The current threshold values may only be modified by the application client via the LOG SELECT command.

NOTE 35 - Log pages or log parameters that are not available may become available at some later time (e.g., after the logical unit has become ready).

6.6 LOG SENSE command

The LOG SENSE command (see table 152) provides a means for the application client to retrieve statistical or other operational information maintained by the SCSI target device about the SCSI target device or its logical units. It is a complementary command to the LOG SELECT command.

Table 152 — LOG SENSE command

Bit / Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (4Dh)							
1	Reserved						Obsolete	SP
2	PC		PAGE CODE					
3	SUBPAGE CODE							
4	Reserved							
5	(MSB)	PARAMETER POINTER						
6								(LSB)
7	(MSB)	ALLOCATION LENGTH						
8								(LSB)
9	CONTROL							

The OPERATION CODE field is defined in 4.3.5.1 and shall be set as shown in table 152 for the LOG SENSE command.

Saving parameters is an optional function of the LOG SENSE command. If the logical unit does not implement saving log parameters and if the save parameters (SP) bit is set to one, 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.

An SP bit set to zero specifies the device server shall perform the specified LOG SENSE command and shall not save any log parameters. If saving log parameters is implemented, an SP bit set to one specifies that the device server shall perform the specified LOG SENSE command and shall save all log parameters identified as saveable by the DS bit (see 7.3) to a nonvolatile, vendor specific location.

The PC field shall be ignored for ASCII format list log parameters (see 7.3.2.2.2.4) and binary format list log parameters (see 7.3.2.2.2.5).

For bounded data counter log parameters (see 7.3.2.2.2.2) and unbounded data counter log parameters (see 7.3.2.2.2.3), the page control (PC) field (see table 149 in 6.5.1) specifies which log parameter values are to be returned by a device server in response to a LOG SENSE command.

For ASCII format list log parameters (see 7.3.2.2.2.4) and binary format list log parameters (see 7.3.2.2.2.5), the PC field shall be ignored. If the parameters specified by the PAGE CODE field and SUBPAGE CODE field in the CDB are list parameters, then the parameter values returned by a device server in response to a LOG SENSE command are determined as follows:

1) the current list log parameter values, if there has been an update to a list log parameter value (e.g., by a LOG SELECT command or by a device specific event) in the specified page or pages since the last logical unit reset occurred;

2) the saved list log parameter values, if saved log parameters are implemented and an update has not occurred since the last logical unit reset; or
3) the vendor specific default list log parameter values, if saved values are not available or not implemented and an update has not occurred since the last logical unit reset.

The PAGE CODE field and SUBPAGE CODE field specify which log page of data is being requested (see 7.3). If the log page specified by the page code and subpage code combination is reserved or not implemented, 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 PARAMETER POINTER field allows the application client to request parameter data beginning from a specific parameter code to the maximum allocation length or the maximum parameter code supported by the logical unit, whichever is less. If the PARAMETER POINTER field is set to:

a) 0000h, then the device server shall begin the transfer of the parameter data with the log parameter having the lowest parameter code that is implemented by the device server for the specified log page;
b) a value specifying the parameter code of a log parameter implemented by the device server for the specified log page, then the device server shall begin the transfer of the parameter data with the log parameter having the specified code;
c) a value that does not specify a parameter code implemented by the device server for the specified log page but is less than the largest parameter code implemented by the device server, then the device server shall begin the transfer of the parameter data with the log parameter implemented by the device server that has the next largest parameter code after the specified value; or
d) a value greater than the largest parameter code implemented by the device server for the specified log page, then the device server shall terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB.

Log parameters within the specified log page shall be transferred in ascending order according to parameter code.

The ALLOCATION LENGTH field is defined in 4.3.5.6.

The CONTROL field is defined in SAM-5.

6.7 MANAGEMENT PROTOCOL IN command

6.7.1 MANAGEMENT PROTOCOL IN command description

The MANAGEMENT PROTOCOL IN command (see table 153) is used to retrieve management protocol information (see 6.7.2) or the results of one or more MANAGEMENT PROTOCOL OUT commands (see 6.8).

Table 153 — MANAGEMENT PROTOCOL IN command

Bit / Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (A3h)							
1	Reserved			SERVICE ACTION (10h)				
2	MANAGEMENT PROTOCOL							
3	MANAGEMENT PROTOCOL SPECIFIC1							
•••								
5								
6	(MSB)	ALLOCATION LENGTH						
•••								
9								(LSB)
10	MANAGEMENT PROTOCOL SPECIFIC2							
11	CONTROL							

The OPERATION CODE field is defined in 4.3.5.1 and shall be set as shown in table 153 for the MANAGEMENT PROTOCOL IN command.

The MANAGEMENT PROTOCOL field (see table 154) specifies which management protocol is being used.

Table 154 — MANAGEMENT PROTOCOL field in MANAGEMENT PROTOCOL IN command

Code	Description	Reference
00h	Management protocol information	6.7.2
01h to 2Fh	Reserved	
30h to 35h	Defined by the SNIA	3.1.172
36h to EFh	Reserved	
F0h to FFh	Vendor Specific	

The contents of the MANAGEMENT PROTOCOL SPECIFIC1 field and MANAGEMENT PROTOCOL SPECIFIC2 field depend on the protocol specified by the MANAGEMENT PROTOCOL field (see table 154).

The ALLOCATION LENGTH field is defined in 4.3.5.6.

The CONTROL field is defined in SAM-5.

Indications of data overrun or underrun and the mechanism, if any, for processing retries depend on the protocol specified by the MANAGEMENT PROTOCOL field (see table 154).

Any association between a previous MANAGEMENT PROTOCOL OUT command and the data transferred by a MANAGEMENT PROTOCOL IN command depends on the protocol specified by the MANAGEMENT PROTOCOL field

(see table 154). If the device server has no data to transfer (e.g., the results for any previous MANAGEMENT PROTOCOL OUT commands are not yet available), the device server may transfer data indicating it has no other data to transfer.

The format of the data transferred depends on the protocol specified by the MANAGEMENT PROTOCOL field (see table 154).

The device server shall retain data resulting from a MANAGEMENT PROTOCOL OUT command, if any, until one of the following events is processed:

a) transfer of the data via a MANAGEMENT PROTOCOL IN command from the same I_T_L nexus as defined by the protocol specified by the MANAGEMENT PROTOCOL field (see table 154);
b) logical unit reset (See SAM-4); or
c) I_T nexus loss (See SAM-4) associated with the I_T nexus that sent the MANAGEMENT PROTOCOL OUT command.

If the data is lost due to one of these events the application client may send a new MANAGEMENT PROTOCOL OUT command to retry the operation.

6.7.2 Management protocol information description

6.7.2.1 Overview

The purpose of the management protocol information management protocol (i.e., the MANAGEMENT PROTOCOL field set to 00h in a MANAGEMENT PROTOCOL IN command) is to transfer management protocol related information from the logical unit. A MANAGEMENT PROTOCOL IN command in which the MANAGEMENT PROTOCOL field is set to 00h is not associated with any previous MANAGEMENT PROTOCOL OUT command and shall be processed without regard for whether a MANAGEMENT PROTOCOL OUT command has been processed.

If the MANAGEMENT PROTOCOL IN command is supported, the MANAGEMENT PROTOCOL value of 00h shall be supported as defined in this standard.

6.7.2.2 CDB description

When the MANAGEMENT PROTOCOL field is set to 00h in a MANAGEMENT PROTOCOL IN command, the MANAGEMENT PROTOCOL SPECIFIC1 field is reserved and the MANAGEMENT PROTOCOL SPECIFIC2 field (see table 155) contains a single numeric value as defined in 3.6.

Table 155 — MANAGEMENT PROTOCOL SPECIFIC2 field for MANAGEMENT PROTOCOL IN protocol 00h

Code	Description	Support	Reference
00h	Supported management protocol list	Mandatory	6.7.2.3
01h to FFh	Reserved		

All other CDB fields for MANAGEMENT PROTOCOL IN command shall meet the requirements stated in 6.7.1.

Each time a MANAGEMENT PROTOCOL IN command with the MANAGEMENT PROTOCOL field set to 00h is received, the device server shall transfer the data defined 6.7.2.3 starting with byte 0.

6.7.2.3 Supported management protocols list description

If the MANAGEMENT PROTOCOL field is set to 00h and the MANAGEMENT PROTOCOL SPECIFIC2 field is set to 00h in a MANAGEMENT PROTOCOL IN command, then the parameter data shall have the format shown in table 156.

Table 156 — Supported management protocols MANAGEMENT PROTOCOL IN parameter data

Bit / Byte	7	6	5	4	3	2	1	0
0	Reserved							
•••								
5								
6	(MSB)	SUPPORTED MANAGEMENT PROTOCOL LIST LENGTH (n-7)						
7								(LSB)
Supported management protocol list								
8	SUPPORTED MANAGEMENT PROTOCOL [first] (00h)							
	⋮							
n	SUPPORTED MANAGEMENT PROTOCOL [last]							

The SUPPORTED MANAGEMENT PROTOCOL LIST LENGTH field indicates the total length, in bytes, of the supported management protocol list that follows.

Each SUPPORTED MANAGEMENT PROTOCOL field in the supported management protocols list shall contain one of the management protocol values supported by the logical unit. The values shall be listed in ascending order starting with 00h.

The total data length shall conform to the ALLOCATION LENGTH field requirements (see 4.3.5.6).

6.8 MANAGEMENT PROTOCOL OUT command

The MANAGEMENT PROTOCOL OUT command (see table 157) is used to send data to the logical unit. The data sent specifies one or more operations to be performed by the logical unit. The format and function of the operations depends on the contents of the MANAGEMENT PROTOCOL field (see table 157). Depending on the protocol specified by the MANAGEMENT PROTOCOL field, the application client may use the MANAGEMENT PROTOCOL IN command (see 6.7) to retrieve data derived from these operations.

Table 157 — MANAGEMENT PROTOCOL OUT command

Bit / Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (A4h)							
1	Reserved			SERVICE ACTION (10h)				
2	MANAGEMENT PROTOCOL							
3	MANAGEMENT PROTOCOL SPECIFIC1							
•••								
5								
6	(MSB)	PARAMETER LIST LENGTH						
•••								
9								(LSB)
10	MANAGEMENT PROTOCOL SPECIFIC2							
11	CONTROL							

The OPERATION CODE field is defined in 4.3.5.1 and shall be set as shown in table 157 for the MANAGEMENT PROTOCOL OUT command.

The MANAGEMENT PROTOCOL field (see table 158) specifies which management protocol is being used.

Table 158 — MANAGEMENT PROTOCOL field in MANAGEMENT PROTOCOL OUT command

Code	Description	Reference
00h to 2Fh	Reserved	3.1.172
30h to 35h	Defined by the SNIA	
36h to EFh	Reserved	
F0h to FFh	Vendor Specific	

The contents of the MANAGEMENT PROTOCOL SPECIFIC1 field and MANAGEMENT PROTOCOL SPECIFIC2 field depend on the protocol specified by the MANAGEMENT PROTOCOL field (see table 158).

The PARAMETER LIST LENGTH field is defined in 4.3.5.5.

The CONTROL field is defined in SAM-5.

Any association between a MANAGEMENT PROTOCOL OUT command and a subsequent MANAGEMENT PROTOCOL IN command depends on the protocol specified by the MANAGEMENT PROTOCOL field (see table 158). Each management protocol shall define whether:

a) the device server shall complete the command with GOOD status as soon as it determines the data has been correctly received. An indication that the data has been processed is obtained by sending a MANAGEMENT PROTOCOL IN command and receiving the results in the associated data transfer; or
b) the device server shall complete the command with GOOD status only after the data has been successfully processed and an associated MANAGEMENT PROTOCOL IN command is not required.

The format of the data transferred depends on the protocol specified by the MANAGEMENT PROTOCOL field (see table 158).

6.9 MODE SELECT(6) command

The MODE SELECT(6) command (see table 159) provides a means for the application client to specify medium, logical unit, or peripheral device parameters to the device server. Device servers that implement the MODE SELECT(6) command shall also implement the MODE SENSE(6) command. Application clients should issue MODE SENSE(6) prior to each MODE SELECT(6) to determine supported mode pages, page lengths, and other parameters.

Table 159 — MODE SELECT(6) command

Bit / Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (15h)							
1	Reserved			PF	Reserved			SP
2	Reserved							
3								
4	PARAMETER LIST LENGTH							
5	CONTROL							

The OPERATION CODE field is defined in 4.3.5.1 and shall be set as shown in table 159 for the MODE SELECT(6) command.

Logical units shall share mode parameter header and block descriptor values across all I_T nexuses. I_T nexus loss shall not affect mode parameter header, block descriptor, and mode page values.

Logical units shall maintain current and saved values of each mode page based on any of the policies listed in table 160. The mode page policy used for each mode page may be reported in the Mode Page Policy VPD page (see 7.8.9).

Table 160 — Mode page policies

Mode page policy	Number of mode page copies
Shared	One copy of the mode page that is shared by all I_T nexuses.
Per target port	A separate copy of the mode page for each target port with each copy shared by all initiator ports.
Per I_T nexus	A separate copy of the mode page for each I_T nexus

After a logical unit reset, each mode parameter header, block descriptor, and mode page shall revert to saved values if supported or default values if saved values are not supported.

If an application client sends a MODE SELECT command that changes any parameters applying to other I_T nexuses, the device server shall establish a unit attention condition (see SAM-4) 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 MODE PARAMETERS CHANGED.

A page format (PF) bit set to zero specifies that all parameters after the block descriptors are vendor specific. A PF bit set to one specifies that the MODE SELECT parameters following the header and block descriptor(s) are structured as pages of related parameters and are as defined in this standard.

A save pages (SP) bit set to zero specifies that the device server shall perform the specified MODE SELECT operation, and shall not save any mode pages. If the logical unit implements no distinction between current and saved mode pages and the SP bit is set to zero, the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB. An SP bit set to one specifies that the device server shall perform the specified MODE SELECT operation, and shall save to a nonvolatile vendor specific location all the saveable mode pages including any sent in the Data-Out Buffer. Mode pages that are saved are specified by the parameter saveable (PS) bit that is returned in the first byte of each mode page by the MODE SENSE command (see 7.5). If the PS bit is set to one in the MODE SENSE data, then the mode page shall be saveable by issuing a MODE SELECT command with the SP bit set to one. If the logical unit does not implement saved mode pages and the SP bit is set to one, 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 PARAMETER LIST LENGTH field specifies the length in bytes of the mode parameter list that shall be contained in the Data-Out Buffer. A parameter list length of zero specifies that the Data-Out Buffer shall be empty. This condition shall not be considered as an error.

If the parameter list length results in the truncation of any mode parameter header, mode parameter block descriptor(s), or mode page, 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 PARAMETER LIST LENGTH ERROR.

The CONTROL field is defined in SAM-5.

The mode parameter list for the MODE SELECT and MODE SENSE commands is defined in 7.5. Parts of each mode parameter list are defined in a device-type dependent manner. Definitions for the parts of each mode parameter list that are unique for each device-type may be found in the applicable command standards (see 3.1.31).

The device server shall terminate the MODE SELECT command with CHECK CONDITION status, set the sense key to ILLEGAL REQUEST, set the additional sense code to INVALID FIELD IN PARAMETER LIST, and shall not change any mode parameters in response to any of the following conditions:

a) if the application client sets any field that is reported as not changeable by the device server to a value other than its current value;
b) if the application client sets any field in the mode parameter header or block descriptor(s) to an unsupported value;
c) if an application client sends a mode page with a page length not equal to the page length returned by the MODE SENSE command for that mode page;
d) if the application client sends an unsupported value for a mode parameter and rounding is not implemented for that mode parameter; or
e) if the application client sets any reserved field in the mode parameter list to a non-zero value and the device server checks reserved fields.

If the application client sends a value for a mode parameter that is outside the range supported by the device server and rounding is implemented for that mode parameter, the device server handles the condition by either:

a) rounding the parameter to an acceptable value and terminating the command as described in 5.4; or
b) terminating 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 device server may alter any mode parameter in any mode page, even those reported as non-changeable, as a result of changes to other mode parameters.

The device server validates the non-changeable mode parameters against the current values that existed for those mode parameters prior to the MODE SELECT command.

NOTE 36 - The current values calculated by the device server may affect the application client's operation. The application client may issue a MODE SENSE command after each MODE SELECT command, to determine the current values.

6.10 MODE SELECT(10) command

The MODE SELECT(10) command (see table 161) provides a means for the application client to specify medium, logical unit, or peripheral device parameters to the device server. See the MODE SELECT(6) command (6.9) for a description of the fields and operation of this command. Application clients should issue MODE SENSE(10) prior to each MODE SELECT(10) to determine supported mode pages, page lengths, and other parameters. Device servers that implement the MODE SELECT(10) command shall also implement the MODE SENSE(10) command.

The OPERATION CODE field is defined in 4.3.5.1 and shall be set as shown in table 161 for the MODE SELECT(10) command.

The CONTROL field is defined in SAM-5.

See the MODE SELECT(6) command (6.9) for a description of the other fields and operation of this command.

Table 161 — MODE SELECT(10) command

Bit / Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (55h)							
1	Reserved			PF	Reserved			SP
2	Reserved							
•••								
6								
7	(MSB)	PARAMETER LIST LENGTH						
8								(LSB)
9	CONTROL							

6.11 MODE SENSE(6) command

6.11.1 MODE SENSE(6) command introduction

The MODE SENSE(6) command (see table 162) provides a means for a device server to report parameters to an application client. It is a complementary command to the MODE SELECT(6) command. Device servers that implement the MODE SENSE(6) command shall also implement the MODE SELECT(6) command.

Table 162 — MODE SENSE(6) command

Bit / Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (1Ah)							
1	Reserved				DBD	Reserved		
2	PC		PAGE CODE					
3	SUBPAGE CODE							
4	ALLOCATION LENGTH							
5	CONTROL							

The OPERATION CODE field is defined in 4.3.5.1 and shall be set as shown in table 162 for the MODE SENSE(6) command.

A disable block descriptors (DBD) bit set to zero specifies that the device server may return zero or more block descriptors in the returned MODE SENSE data (see 7.5). A DBD bit set to one specifies that the device server shall not return any block descriptors in the returned MODE SENSE data.

The page control (PC) field specifies the type of mode parameter values to be returned in the mode pages. The PC field is defined in table 163.

Table 163 — Page control (PC) field

Code	Type of parameter	Reference
00b	Current values	6.11.2
01b	Changeable values	6.11.3
10b	Default values	6.11.4
11b	Saved values	6.11.5

The PC field only affects the mode parameters within the mode pages, however the PS bit, SPF bit, PAGE CODE field, SUBPAGE CODE field, and PAGE LENGTH field should return current values (i.e., as if PC is set to 00b). The mode parameter header and mode parameter block descriptor should return current values.

Some SCSI target devices may not distinguish between current and saved mode parameters and report identical values in response to a PC field of either 00b or 11b. See also the description of the save pages (SP) bit in the MODE SELECT command.

The PAGE CODE and SUBPAGE CODE fields (see table 164) specify which mode pages and subpages to return.

Table 164 — Mode page code usage for all devices

Page Code	Subpage Code	Description
00h	vendor specific	Vendor specific (does not require page format)
01h to 1Fh	00h	See specific device types (page_0 format)
	01h to DFh	See specific device types (sub_page format)
	E0h to FEh	Vendor specific (sub_page format)
	FFh	Return all subpages for the specified device specific mode page in the page_0 format for subpage 00h and in the sub_page format for subpages 01h to FEh
20h to 3Eh	00h	Vendor specific (page_0 format required)
	01h to FEh	Vendor specific (sub_page format required)
	FFh	Return all subpages for the specified vendor specific mode page in the page_0 format for subpage 00h and in the sub_page format for subpages 01h to FEh
3Fh	00h	Return all subpage 00h mode pages in page_0 format
	01h to FEh	Reserved
	FFh	Return all subpages for all mode pages in the page_0 format for subpage 00h and in the sub_page format for subpages 01h to FEh

The ALLOCATION LENGTH field is defined in 4.3.5.6.

The CONTROL field is defined in SAM-5.

An application client may request any one or all of the supported mode pages from the device server. If an application client issues a MODE SENSE command with a page code or subpage code value not implemented by the

logical unit, 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 an application client requests all supported mode pages, the device server shall return the supported pages in ascending page code order beginning with mode page 01h. If mode page 00h is implemented, the device server shall return mode page 00h after all other mode pages have been returned.

If the PC field and the PAGE CODE field are both set to zero, the device server should return a mode parameter header and block descriptor, if applicable.

The mode parameter list for all device types for MODE SELECT and MODE SENSE is defined in 7.5. Parts of the mode parameter list are specifically defined for each device type. Definitions for the parts of each mode parameter list that are unique for each device-type may be found in the applicable command standards (see 3.1.31).

6.11.2 Current values

A PC field value of 00b requests that the device server return the current values of the mode parameters. The current values returned are:

a) the current values of the mode parameters established by the last successful MODE SELECT command;
b) the saved values of the mode parameters if a MODE SELECT command has not successfully completed since the mode parameters were restored to their saved values (see 6.9); or
c) the default values of the mode parameters if a MODE SELECT command has not successfully completed since the mode parameters were restored to their default values (see 6.9).

6.11.3 Changeable values

A PC field value of 01b requests that the device server return a mask denoting those mode parameters that are changeable. In the mask, the bits in the fields of the mode parameters that are changeable all shall be set to one and the bits in the fields of the mode parameters that are non-changeable (i.e., defined by the logical unit) all shall be set to zero.

If the logical unit does not implement changeable parameters mode pages and the device server receives a MODE SENSE command with 01b in the PC 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.

An attempt to change a non-changeable mode parameter using the MODE SELECT command shall result in an error condition (see 6.9).

The application client should issue a MODE SENSE command with the PC field set to 01b and the PAGE CODE field set to 3Fh to determine which mode pages are supported, which mode parameters within the mode pages are changeable, and the supported length of each mode page prior to issuing any MODE SELECT commands.

6.11.4 Default values

A PC field value of 10b requests that the device server return the default values of the mode parameters. Unsupported parameters shall be set to zero. Default values should be accessible even if the logical unit is not ready.

6.11.5 Saved values

A PC field value of 11b requests that the device server return the saved values of the mode parameters. Mode parameters not supported by the logical unit shall be set to zero. If saved values are not implemented, the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to SAVING PARAMETERS NOT SUPPORTED.

The method of saving parameters is vendor specific. The parameters are preserved in such a manner that they are retained when the device is powered down. All saveable mode pages should be considered saved when a MODE SELECT command issued with the SP bit set to one has completed with a GOOD status or after the successful completion of a FORMAT UNIT command.

6.11.6 Initial responses

After a logical unit reset, the device server shall respond in the following manner:

a) if default values are requested, report the default values;
b) if saved values are requested, report valid restored mode parameters, or restore the mode parameters and report them. If the saved values of the mode parameters are not able to be accessed from the nonvolatile vendor specific location, the command shall be terminated with CHECK CONDITION status, with the sense key set to NOT READY. If saved parameters are not implemented, respond as defined in 6.11.5; or
c) if current values are requested and the current values have been sent by the application client via a MODE SELECT command, the current values shall be returned. If the current values have not been sent, the device server shall return:
   A) the saved values, if saving is implemented and saved values are available; or
   B) the default values.

6.12 MODE SENSE(10) command

The MODE SENSE(10) command (see table 165) provides a means for a device server to report parameters to an application client. It is a complementary command to the MODE SELECT(10) command. Device servers that implement the MODE SENSE(10) command shall also implement the MODE SELECT(10) command.

Table 165 — MODE SENSE(10) command

Bit / Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (5Ah)							
1	Reserved			LLBAA	DBD	Reserved		
2	PC		PAGE CODE					
3	SUBPAGE CODE							
4	Reserved							
•••								
6								
7	(MSB)	ALLOCATION LENGTH						
8								(LSB)
9	CONTROL							

The OPERATION CODE field is defined in 4.3.5.1 and shall be set as shown in table 165 for the MODE SENSE(10) command.

If the Long LBA Accepted (LLBAA) bit is set to one, the device server is allowed to return parameter data with the LONGLBA bit equal to one (see 7.5.4). If LLBAA bit is set to zero, the LONGLBA bit shall be zero in the parameter data returned by the device server.

The CONTROL field is defined in SAM-5.

See the MODE SENSE(6) command (6.11) for a description of the other fields and operation of this command.

6.13 PERSISTENT RESERVE IN command

6.13.1 PERSISTENT RESERVE IN command introduction

The PERSISTENT RESERVE IN command (see table 166) is used to obtain information about persistent reservations and reservation keys (i.e., registrations) that are active within a device server. This command is used in conjunction with the PERSISTENT RESERVE OUT command (see 6.14).

Table 166 — PERSISTENT RESERVE IN command

Bit / Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (5Eh)							
1	Reserved			SERVICE ACTION				
2	Reserved							
•••								
6								
7	(MSB)	ALLOCATION LENGTH						
8								(LSB)
9	CONTROL							

The OPERATION CODE field is defined in 4.3.5.1 and shall be set as shown in table 166 for the PERSISTENT RESERVE IN command.

The SERVICE ACTION field is defined in 4.3.5.2. The service action codes for the PERSISTENT RESERVE IN command are defined in table 167.

Table 167 — PERSISTENT RESERVE IN service action codes

Code	Name	Description	Reference
00h	READ KEYS	Reads all registered reservation keys (i.e., registrations) as described in 5.9.6.2	6.13.2
01h	READ RESERVATION	Reads the current persistent reservations as described in 5.9.6.3	6.13.3
02h	REPORT CAPABILITIES	Returns capability information	6.13.4
03h	READ FULL STATUS	Reads complete information about all registrations and the persistent reservations, if any	6.13.5
04h to 1Fh	Reserved	Reserved	

The ALLOCATION LENGTH field is defined in 4.3.5.6. The PERSISTENT RESERVE IN parameter data includes a length field that indicates the number of parameter data bytes available to be returned. The allocation length should be set to a value large enough to return the length field for the specified service action.

The CONTROL field is defined in SAM-5.

6.13.2 READ KEYS service action

The READ KEYS service action requests that the device server return a parameter list containing a header and a list of each currently registered I_T nexus' reservation key. If multiple I_T nexuses have registered with the same key, then that key value shall be listed multiple times, once for each such registration.

For more information on READ KEYS see 5.9.6.2.

The format for the parameter data provided in response to a PERSISTENT RESERVE IN command with the READ KEYS service action is shown in table 168.

Table 168 — PERSISTENT RESERVE IN parameter data for READ KEYS

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)							
•••				PRGENERATION				
3								(LSB)
4	(MSB)							
•••				ADDITIONAL LENGTH (n-7)				
7								(LSB)
	Reservation key list							
8	(MSB)							
•••				Reservation key [first]				
15								(LSB)
	⋮							
n-7	(MSB)							
•••				Reservation key [last]				
n								(LSB)

The Persistent Reservations Generation (PRGENERATION) field shall contain the value of a 32-bit wrapping counter maintained by the device server that shall be incremented every time a PERSISTENT RESERVE OUT command requests a REGISTER service action, a REGISTER AND IGNORE EXISTING KEY service action, a REGISTER AND MOVE service action, a CLEAR service action, a PREEMPT service action, or a PREEMPT AND ABORT service action. The counter shall not be incremented by a PERSISTENT RESERVE IN command, by a PERSISTENT RESERVE OUT command that performs a RESERVE or RELEASE service action, or by a PERSISTENT RESERVE OUT command that is terminated due to an error or reservation conflict. Regardless of the APTPL bit value the PRgeneration value shall be set to zero by a power on.

The ADDITIONAL LENGTH field contains a count of the number of bytes in the Reservation key list. The relationship between the ADDITIONAL LENGTH field and the CDB ALLOCATION LENGTH field is defined in 4.3.5.6.

The reservation key list contains the 8-byte reservation keys for all I_T nexuses that have been registered (see 5.9.7).

6.13.3 READ RESERVATION service action

6.13.3.1 READ RESERVATION service action introduction

The READ RESERVATION service action requests that the device server return a parameter list containing a header and the persistent reservation, if any, that is present in the device server.

For more information on READ RESERVATION see 5.9.6.3.

6.13.3.2 Format of PERSISTENT RESERVE IN parameter data for READ RESERVATION

When no persistent reservation is held, the format for the parameter data provided in response to a PERSISTENT RESERVE IN command with the READ RESERVATION service action is shown in table 169.

Table 169 — PERSISTENT RESERVE IN parameter data for READ RESERVATION with no reservation held

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)							
•••				PRGENERATION				
3								(LSB)
4	(MSB)							
•••				ADDITIONAL LENGTH (0)				
7								(LSB)

The PRGENERATION field shall be as defined for the PERSISTENT RESERVE IN command with READ KEYS service action parameter data (see 6.13.2).

The ADDITIONAL LENGTH field shall be set to zero, indicating that no persistent reservation is held.

When a persistent reservation is held, the format for the parameter data provided in response to a PERSISTENT RESERVE IN command with the READ RESERVATION service action is shown in table 170.

Table 170 — PERSISTENT RESERVE IN parameter data for READ RESERVATION with reservation

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)							
•••				PRGENERATION				
3								(LSB)
4	(MSB)							
•••				ADDITIONAL LENGTH (10h)				
7								(LSB)
8	(MSB)							
•••				RESERVATION KEY				
15								(LSB)
16								
•••				Obsolete				
19								
20				Reserved				
21	SCOPE				TYPE			
22				Obsolete				
23								

The PRGENERATION field shall be as defined for the PERSISTENT RESERVE IN command with READ KEYS service action parameter data.

The ADDITIONAL LENGTH field contains a count of the number of bytes to follow and shall be set to 16. The relationship between the ADDITIONAL LENGTH field and the CDB ALLOCATION LENGTH field is defined in 4.3.5.6.

The RESERVATION KEY field shall contain the reservation key under which the persistent reservation is held (see 5.9.10).

The SCOPE field shall be set to LU_SCOPE (see 6.13.3.3).

The TYPE field shall contain the persistent reservation type (see 6.13.3.4) specified in the PERSISTENT RESERVE OUT command that created the persistent reservation.

The obsolete fields in bytes 16 to 19, byte 22, and byte 23 were defined in a previous standard.

6.13.3.3 Persistent reservations scope

The SCOPE field (see table 171) shall be set to LU_SCOPE, specifying that the persistent reservation applies to the entire logical unit.

Table 171 — Persistent reservation SCOPE field

Code	Name	Description
0h	LU_SCOPE	Persistent reservation applies to the full logical unit
1h to 2h		Obsolete
3h to Fh		Reserved

The LU_SCOPE scope shall be implemented by all device servers that implement PERSISTENT RESERVE OUT.

6.13.3.4 Persistent reservations type

The TYPE field (see table 172) specifies the characteristics of the persistent reservation being established for all logical blocks within the logical unit. Table 47 (see 5.9.1) defines the persistent reservation types under which each command defined in this standard is allowed to be processed. Each other command standard (see 3.1.31) defines the persistent reservation types under which each command defined in that command standard is allowed to be processed.

Table 172 — Persistent reservation TYPE field (part 1 of 2)

Code	Name	Description
0h		Obsolete
1h	Write Exclusive	**Access Restrictions**: Some commands (e.g., media-access write commands) are only allowed for the persistent reservation holder (see 5.9.10). **Persistent Reservation Holder:** There is only one persistent reservation holder.
2h		Obsolete
3h	Exclusive Access	**Access Restrictions**: Some commands (e.g., media-access commands) are only allowed for the persistent reservation holder (see 5.9.10). **Persistent Reservation Holder:** There is only one persistent reservation holder.
4h		Obsolete
5h	Write Exclusive – Registrants Only	**Access Restrictions**: Some commands (e.g., media-access write commands) are only allowed for registered I_T nexuses. **Persistent Reservation Holder:** There is only one persistent reservation holder (see 5.9.10).
6h	Exclusive Access – Registrants Only	**Access Restrictions**: Some commands (e.g., media-access commands) are only allowed for registered I_T nexuses. **Persistent Reservation Holder:** There is only one persistent reservation holder (see 5.9.10).

Table 172 — Persistent reservation TYPE field (part 2 of 2)

Code	Name	Description
7h	Write Exclusive – All Registrants	**Access Restrictions**: Some commands (e.g., media-access write commands) are only allowed for registered I_T nexuses. **Persistent Reservation Holder:** Each registered I_T nexus is a persistent reservation holder (see 5.9.10).
8h	Exclusive Access – All Registrants	**Access Restrictions**: Some commands (e.g., media-access commands) are only allowed for registered I_T nexuses. **Persistent Reservation Holder:** Each registered I_T nexus is a persistent reservation holder (see 5.9.10).
9h to Fh	Reserved	

6.13.4 REPORT CAPABILITIES service action

The REPORT CAPABILITIES service action requests that the device server return information on persistent reservation features.

The format for the parameter data provided in response to a PERSISTENT RESERVE IN command with the REPORT CAPABILITIES service action is shown in table 173.

Table 173 — PERSISTENT RESERVE IN parameter data for REPORT CAPABILITIES

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)	LENGTH (0008h)						
1								(LSB)
2	Reserved			CRH	SIP_C	ATP_C	Reserved	PTPL_C
3	TMV	ALLOW COMMANDS			Reserved			PTPL_A
4	PERSISTENT RESERVATION TYPE MASK							
5								
6	Reserved							
7								

The LENGTH field indicates the length in bytes of the parameter data. The relationship between the LENGTH field and the CDB ALLOCATION LENGTH field is defined in 4.3.5.6.

A Compatible Reservation Handling (CRH) bit set to one indicates that the device server supports the exceptions to the SPC-2 RESERVE and RELEASE commands described in 5.9.3. A CRH bit set to zero indicates that RESERVE(6) command, RESERVE(10) command, RELEASE(6) command, and RELEASE(10) command are processed as defined in SPC-2.

A Specify Initiator Ports Capable (SIP_C) bit set to one indicates that the device server supports the SPEC_I_PT bit in the PERSISTENT RESERVE OUT command parameter data (see 6.14.3). An SIP_C bit set to zero indicates that the device server does not support the SPEC_I_PT bit in the PERSISTENT RESERVE OUT command parameter data.

An All Target Ports Capable (ATP_C) bit set to one indicates that the device server supports the ALL_TG_PT bit in the PERSISTENT RESERVE OUT command parameter data. An ATP_C bit set to zero indicates that the device server does not support the ALL_TG_PT bit in the PERSISTENT RESERVE OUT command parameter data.

A Persist Through Power Loss Capable (PTPL_C) bit set to one indicates that the device server supports the persist through power loss capability (see 5.9.5) for persistent reservations and the APTPL bit in the PERSISTENT RESERVE OUT command parameter data. An PTPL_C bit set to zero indicates that the device server does not support the persist through power loss capability.

A Type Mask Valid (TMV) bit set to one indicates that the PERSISTENT RESERVATION TYPE MASK field contains a bit map indicating which persistent reservation types are supported by the device server. A TMV bit set to zero indicates that the PERSISTENT RESERVATION TYPE MASK field shall be ignored.

The ALLOW COMMANDS field (see table 174) indicates whether certain commands are allowed through certain types of persistent reservations.

Table 174 — ALLOW COMMANDS field

Code	Description
000b	No information is provided about whether certain commands are allowed through certain types of persistent reservations.
001b	The device server allows the TEST UNIT READY command (see table 47 in 5.9.1) through Write Exclusive and Exclusive Access persistent reservations and does not provide information about whether the following commands are allowed through Write Exclusive persistent reservations: a) the MODE SENSE command, READ ATTRIBUTE command, READ BUFFER command, RECEIVE COPY RESULTS command, RECEIVE DIAGNOSTIC RESULTS command, REPORT SUPPORTED OPERATION CODES command, and REPORT SUPPORTED TASK MANAGEMENT FUNCTION command (see table 47 in 5.9.1); and b) the READ DEFECT DATA command (see SBC-3).
010b	The device server allows the TEST UNIT READY command through Write Exclusive and Exclusive Access persistent reservations and does not allow the following commands through Write Exclusive persistent reservations: a) the MODE SENSE command, READ ATTRIBUTE command, READ BUFFER command, RECEIVE DIAGNOSTIC RESULTS command, REPORT SUPPORTED OPERATION CODES command, and REPORT SUPPORTED TASK MANAGEMENT FUNCTION command; and b) the READ DEFECT DATA command. The device server does not allow the RECEIVE COPY RESULTS command through Write Exclusive or Exclusive Access persistent reservations.
011b	The device server allows the TEST UNIT READY command through Write Exclusive and Exclusive Access persistent reservations and allows the following commands through Write Exclusive persistent reservations: a) the MODE SENSE command, READ ATTRIBUTE command, READ BUFFER command, RECEIVE DIAGNOSTIC RESULTS command, REPORT SUPPORTED OPERATION CODES command, and REPORT SUPPORTED TASK MANAGEMENT FUNCTION command; and b) the READ DEFECT DATA command. The device server does not allow the RECEIVE COPY RESULTS command through Write Exclusive or Exclusive Access persistent reservations.
100b	The device server allows the TEST UNIT READY command and the RECEIVE COPY RESULTS command through Write Exclusive and Exclusive Access persistent reservations and allows the following commands through Write Exclusive persistent reservations: a) the MODE SENSE command, READ ATTRIBUTE command, READ BUFFER command, RECEIVE DIAGNOSTIC RESULTS command, REPORT SUPPORTED OPERATION CODES command, and REPORT SUPPORTED TASK MANAGEMENT FUNCTION command; and b) the READ DEFECT DATA command.
101b to 111b	Reserved

A Persist Through Power Loss Activated (PTPL_A) bit set to one indicates that the persist through power loss capability is activated (see 5.9.5). A PTPL_A bit set to zero indicates that the persist through power loss capability is not activated.

The PERSISTENT RESERVATION TYPE MASK field (see table 175) contains a bit map that indicates the persistent reservation types that are supported by the device server.

Table 175 — Persistent Reservation Type Mask format

Bit / Byte	7	6	5	4	3	2	1	0
4	WR_EX_AR	EX_AC_RO	WR_EX_RO	Reserved	EX_AC	Reserved	WR_EX	Reserved
5	Reserved							EX_AC_AR

A Write Exclusive – All Registrants (WR_EX_AR) bit set to one indicates that the device server supports the Write Exclusive – All Registrants persistent reservation type. An WR_EX_AR bit set to zero indicates that the device server does not support the Write Exclusive – All Registrants persistent reservation type.

An Exclusive Access – Registrants Only (EX_AC_RO) bit set to one indicates that the device server supports the Exclusive Access – Registrants Only persistent reservation type. An EX_AC_RO bit set to zero indicates that the device server does not support the Exclusive Access – Registrants Only persistent reservation type.

A Write Exclusive – Registrants Only (WR_EX_RO) bit set to one indicates that the device server supports the Write Exclusive – Registrants Only persistent reservation type. An WR_EX_RO bit set to zero indicates that the device server does not support the Write Exclusive – Registrants Only persistent reservation type.

An Exclusive Access (EX_AC) bit set to one indicates that the device server supports the Exclusive Access persistent reservation type. An EX_AC bit set to zero indicates that the device server does not support the Exclusive Access persistent reservation type.

A Write Exclusive (WR_EX) bit set to one indicates that the device server supports the Write Exclusive persistent reservation type. An WR_EX bit set to zero indicates that the device server does not support the Write Exclusive persistent reservation type.

An Exclusive Access – All Registrants (EX_AC_AR) bit set to one indicates that the device server supports the Exclusive Access – All Registrants persistent reservation type. An EX_AC_AR bit set to zero indicates that the device server does not support the Exclusive Access – All Registrants persistent reservation type.

6.13.5 READ FULL STATUS service action

The READ FULL STATUS service action requests that the device server return a parameter list describing the registration and persistent reservation status of each currently registered I_T nexus for the logical unit.

For more information on READ FULL STATUS see 5.9.6.4.

The format for the parameter data provided in response to a PERSISTENT RESERVE IN command with the READ FULL STATUS service action is shown in table 176.

Table 176 — PERSISTENT RESERVE IN parameter data for READ FULL STATUS

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)							
•••				PRGENERATION				
3								(LSB)
4	(MSB)							
•••				ADDITIONAL LENGTH (n-7)				
7								(LSB)
	Full status descriptors							
8								
•••				Full status descriptor [first] (see table 177)				
				⋮				
•••				Full status descriptor [last] (see table 177)				
n								

The PRGENERATION field shall be as defined for the PERSISTENT RESERVE IN command with READ KEYS service action parameter data (see 6.13.2).

The ADDITIONAL LENGTH field contains a count of the number of bytes to follow in the full status descriptors. The relationship between the ADDITIONAL LENGTH field and the CDB ALLOCATION LENGTH field is defined in 4.3.5.6.

The format of the full status descriptors is shown in table 177. Each full status descriptor describes one or more registered I_T nexuses. The device server shall return persistent reservations status information for every registered I_T nexus.

Table 177 — PERSISTENT RESERVE IN full status descriptor format

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)							
•••				RESERVATION KEY				
7								(LSB)
8								
•••				Reserved				
11								
12				Reserved			ALL_TG_PT	R_HOLDER
13			SCOPE			TYPE		
14								
•••				Reserved				
17								
18	(MSB)							
19				RELATIVE TARGET PORT IDENTIFIER				(LSB)
20	(MSB)							
•••				ADDITIONAL DESCRIPTOR LENGTH (n-23)				
23								(LSB)
24								
•••				TRANSPORTID				
n								

The RESERVATION KEY field contains the reservation key.

A Reservation Holder (R_HOLDER) bit set to one indicates that all I_T nexuses described by this full status descriptor are registered and are persistent reservation holders (see 5.9.10). A R_HOLDER bit set to zero indicates that all I_T nexuses described by this full status descriptor are registered but are not persistent reservation holders.

An All Target Ports (ALL_TG_PT) bit set to zero indicates that this full status descriptor represents a single I_T nexus. An ALL_TG_PT bit set to one indicates that:

a) this full status descriptor represents all the I_T nexuses that are associated with both:
   A) the initiator port specified by the TRANSPORTID field; and
   B) every target port in the SCSI target device;
b) all the I_T nexuses are registered with the same reservation key; and
c) all the I_T nexuses are either reservation holders or not reservation holders as indicated by the R_HOLDER bit.

The device server is not required to return an ALL_TG_PT bit set to one. Instead, it may return separate full status descriptors for each I_T nexus.

If the R_HOLDER bit is set to one (i.e., if the I_T nexus described by this full status descriptor is a reservation holder), the SCOPE field and the TYPE field are as defined in the READ RESERVATION service action parameter data (see

6.13.3). If the R_HOLDER bit is set to zero, the contents of the SCOPE field and the TYPE field are not defined by this standard.

If the ALL_TG_PT bit set to zero, the RELATIVE TARGET PORT IDENTIFIER field contains the relative port identifier (see 3.1.127) of the target port that is part of the I_T nexus described by this full status descriptor. If the ALL_TG_PT bit is set to one, the contents of the RELATIVE TARGET PORT IDENTIFIER field are not defined by this standard.

The ADDITIONAL DESCRIPTOR LENGTH field contains a count of the number of bytes that follow in the descriptor (i.e., the size of the TransportID).

The TRANSPORTID field contains a TransportID (see 7.6.4) identifying the initiator port that is part of the I_T nexus or I_T nexuses described by this full status descriptor.

6.14 PERSISTENT RESERVE OUT command

6.14.1 PERSISTENT RESERVE OUT command introduction

The PERSISTENT RESERVE OUT command (see table 178) is used to request service actions that reserve a logical unit for the exclusive or shared use of a particular I_T nexus. The command uses other service actions to manage and remove such persistent reservations.

I_T nexuses performing PERSISTENT RESERVE OUT service actions are identified by a registered reservation key provided by the application client. An application client may use the PERSISTENT RESERVE IN command to obtain the reservation key, if any, for the I_T nexus holding a persistent reservation and may use the PERSISTENT RESERVE OUT command to preempt that persistent reservation.

Table 178 — PERSISTENT RESERVE OUT command

Bit / Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (5Fh)							
1	Reserved			SERVICE ACTION				
2	SCOPE				TYPE			
3	Reserved							
4								
5	(MSB)	PARAMETER LIST LENGTH						
•••								
8								(LSB)
9	CONTROL							

The OPERATION CODE field is defined in 4.3.5.1 and shall be set as shown in table 178 for the PERSISTENT RESERVE OUT command.

The SERVICE ACTION field is defined in 4.3.5.2 and 6.14.2.

If a PERSISTENT RESERVE OUT command is attempted, but there are insufficient device server resources to complete the operation, the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INSUFFICIENT REGISTRATION RESOURCES.

The PERSISTENT RESERVE OUT command contains fields that specify a persistent reservation service action, the intended scope of the persistent reservation, and the restrictions caused by the persistent reservation. The SCOPE field and TYPE field are defined in 6.13.3.3 and 6.13.3.4. If a SCOPE field specifies a scope that is not implemented, 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.

Fields contained in the PERSISTENT RESERVE OUT parameter list specify the information required to perform a particular persistent reservation service action.

The PARAMETER LIST LENGTH field specifies the number of bytes of parameter data for the PERSISTENT RESERVE OUT command.

The parameter list shall be 24 bytes in length and the PARAMETER LIST LENGTH field shall contain 24 (18h), if the following conditions are true:

a) the SPEC_I_PT bit (see 6.14.3) is set to zero; and
b) the service action is not REGISTER AND MOVE.

If the SPEC_I_PT bit is set to zero, the service action is not REGISTER AND MOVE, and the parameter list length is not 24, 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 PARAMETER LIST LENGTH ERROR.

If the parameter list length is larger than the device server is able to process, the command should be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to PARAMETER LIST LENGTH ERROR.

The CONTROL field is defined in SAM-5.

6.14.2 PERSISTENT RESERVE OUT service actions

When processing the PERSISTENT RESERVE OUT service actions, the device server shall increment the PRgeneration value as specified in 6.13.2.

The PERSISTENT RESERVE OUT command service actions are defined in table 179.

Table 179 — PERSISTENT RESERVE OUT service action codes

Code	Name	Description	PRGENERATION field incremented (see 6.13.2)	Parameter list format
00h	REGISTER	Register a reservation key with the device server (see 5.9.7) or unregister a reservation key (see 5.9.11.3).	Yes	Basic (see 6.14.3)
01h	RESERVE	Creates a persistent reservation having a specified SCOPE and TYPE (see 5.9.9). The SCOPE and TYPE of a persistent reservation are defined in 6.13.3.3 and 6.13.3.4.	No	Basic (see 6.14.3)
02h	RELEASE	Releases the selected persistent reservation (see 5.9.11.2).	No	Basic (see 6.14.3)
03h	CLEAR	Clears all reservation keys (i.e., registrations) and all persistent reservations (see 5.9.11.6).	Yes	Basic (see 6.14.3)
04h	PREEMPT	Preempts persistent reservations and/or removes registrations (see 5.9.11.4).	Yes	Basic (see 6.14.3)
05h	PREEMPT AND ABORT	Preempts persistent reservations and/or removes registrations and aborts all commands for all preempted I_T nexuses (see 5.9.11.4 and 5.9.11.5).	Yes	Basic (see 6.14.3)
06h	REGISTER AND IGNORE EXISTING KEY	Register a reservation key with the device server (see 5.9.7) or unregister a reservation key (see 5.9.11.3).	Yes	Basic (see 6.14.3)
07h	REGISTER AND MOVE	Register a reservation key for another I_T nexus with the device server and move a persistent reservation to that I_T nexus (see 5.9.8)	Yes	Register and move (see 6.14.4)
08h to 1Fh	Reserved			

6.14.3 Basic PERSISTENT RESERVE OUT parameter list

The parameter list format shown in table 180 shall be used by the PERSISTENT RESERVE OUT command with any service action except the REGISTER AND MOVE service action. All fields shall be sent, even if the field is not required for the specified service action and scope values.

Table 180 — PERSISTENT RESERVE OUT parameter list

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)							
•••				RESERVATION KEY				
7								(LSB)
8	(MSB)							
•••				SERVICE ACTION RESERVATION KEY				
15								(LSB)
16								
•••				Obsolete				
19								
20	Reserved				SPEC_I_PT	ALL_TG_PT	Reserved	APTPL
21	Reserved							
22				Obsolete				
23								
24								
•••				Additional parameter data				
n								

The obsolete fields in bytes 16 to 19, byte 22 and byte 23 were defined in a previous standard.

The RESERVATION KEY field contains an 8-byte value provided by the application client to the device server to identify the I_T nexus that is the source of the PERSISTENT RESERVE OUT command. The device server shall verify that the contents of the RESERVATION KEY field in a PERSISTENT RESERVE OUT command parameter data matches the registered reservation key for the I_T nexus from which the command was received, except for:

a) the REGISTER AND IGNORE EXISTING KEY service action where the RESERVATION KEY field shall be ignored; and
b) the REGISTER service action for an unregistered I_T nexus where the RESERVATION KEY field shall contain zero.

Except as noted above, when a PERSISTENT RESERVE OUT command specifies a RESERVATION KEY field other than the reservation key registered for the I_T nexus the device server shall complete the command with RESERVATION CONFLICT status. Except as noted above, the reservation key of the I_T nexus shall be verified to be correct regardless of the SERVICE ACTION and SCOPE field values.

The SERVICE ACTION RESERVATION KEY field contains information needed for the following service actions: REGISTER, REGISTER AND IGNORE EXISTING KEY, PREEMPT, and PREEMPT AND ABORT. The SERVICE ACTION RESERVATION KEY field is ignored for the following service actions: RESERVE, RELEASE, and CLEAR.

For the REGISTER service action and REGISTER AND IGNORE EXISTING KEY service action, the SERVICE ACTION RESERVATION KEY field contains:

a) the new reservation key to be registered in place of the registered reservation key; or
b) zero to unregister the registered reservation key.

For the PREEMPT service action and PREEMPT AND ABORT service action, the SERVICE ACTION RESERVATION KEY field contains the reservation key of:

a) the registrations to be removed; and
b) if the SERVICE ACTION RESERVATION KEY field identifies a persistent reservation holder (see 5.9.10), persistent reservations that are to be preempted.

If the Specify Initiator Ports (SPEC_I_PT) bit is set to zero, the device server shall apply the registration only to the I_T nexus that sent the PERSISTENT RESERVE OUT command. If the SPEC_I_PT bit is set to one for any service action except the REGISTER service action, 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 PARAMETER LIST. If the SPEC_I_PT bit is set to one for the REGISTER service action, the additional parameter data (see table 181) shall include a list of transport IDs and the device server shall also apply the registration to the I_T nexus for each initiator port specified by a TransportID. If a registration fails for any initiator port (e.g., if the logical unit does not have enough resources available to hold the registration information), no registrations shall be made, and the command shall be terminated with CHECK CONDITION status.

Table 181 — PERSISTENT RESERVE OUT specify initiator ports additional parameter data

Bit / Byte	7	6	5	4	3	2	1	0
24	TRANSPORTID PARAMETER DATA LENGTH (n-27)							
•••								
27								
	TransportIDs list							
28	TransportID [first]							
•••								
	⋮							
	TransportID [last]							
•••								
n								

The TRANSPORTID PARAMETER DATA LENGTH field specifies the number of bytes of TransportIDs that follow.

The command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST:

a) if the value in the PARAMETER LIST LENGTH field in the CDB does not include all of the additional parameter list bytes specified by the TRANSPORTID PARAMETER DATA LENGTH field; or
b) if the value in the TRANSPORTID PARAMETER DATA LENGTH field results in the truncation of a TransportID.

The format of a TransportID is specified in 7.6.4.

The All Target Ports (ALL_TG_PT) bit is valid only for the REGISTER service action and the REGISTER AND IGNORE EXISTING KEY service action, and shall be ignored for all other service actions. Support for the ALL_TG_PT bit is optional. If the device server receives a REGISTER service action or a REGISTER AND IGNORE EXISTING KEY service action with the ALL_TG_PT bit set to one, it shall create the specified registration on all target ports in the SCSI target device known to the device server (i.e., as if the same registration request had been received individually through each target port). If the device server receives a REGISTER service action or a REGISTER AND IGNORE EXISTING KEY service action with the ALL_TG_PT bit set to zero, it shall apply the registration only to the target port through which the PERSISTENT RESERVE OUT command was received. If a device server that does not support an ALL_TG_PT bit set to one receives that value in a REGISTER service action or a REGISTER AND IGNORE EXISTING KEY service action, 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.

The Activate Persist Through Power Loss (APTPL) bit is valid only for the REGISTER service action and the REGISTER AND IGNORE EXISTING KEY service action, and shall be ignored for all other service actions. Support for an APTPL bit equal to one is optional. If a device server that does not support an APTPL bit set to one receives that value in a REGISTER service action or a REGISTER AND IGNORE EXISTING KEY service action, 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.

If the last valid APTPL bit value received by the device server is zero, the loss of power in the SCSI target device shall release the persistent reservation for the logical unit and remove all registered reservation keys (see 5.9.7). If the last valid APTPL bit value received by the device server is one, the logical unit shall retain any persistent reservation(s) that may be present and all reservation keys (i.e., registrations) for all I_T nexuses even if power is lost and later returned (see 5.9.5).

Table 182 summarizes which fields are set by the application client and interpreted by the device server for each service action and scope value.

Table 182 — PERSISTENT RESERVE OUT service actions and valid parameters (part 1 of 2)

Service action	Allowed SCOPE	**Parameters** (part 1 of 2)			
		TYPE	RESERVATION KEY	SERVICE ACTION RESERVATION KEY	APTPL
REGISTER	ignored	ignored	valid	valid	valid
REGISTER AND IGNORE EXISTING KEY	ignored	ignored	ignored	valid	valid
RESERVE	LU_SCOPE	valid	valid	ignored	ignored
RELEASE	LU_SCOPE	valid	valid	ignored	ignored
CLEAR	ignored	ignored	valid	ignored	ignored
PREEMPT	LU_SCOPE	valid	valid	valid	ignored
PREEMPT AND ABORT	LU_SCOPE	valid	valid	valid	ignored
REGISTER AND MOVE	LU_SCOPE	valid	valid	valid	not applicable [a]

[a] The parameter list format for the REGISTER AND MOVE service action is described in 6.14.4.

Table 182 — PERSISTENT RESERVE OUT service actions and valid parameters (part 2 of 2)

Service action	Allowed SCOPE	Parameters (part 2 of 2)	
		ALL_TG_PT	SPEC_I_PT
REGISTER	ignored	valid	valid
REGISTER AND IGNORE EXISTING KEY	ignored	valid	invalid
RESERVE	LU_SCOPE	ignored	invalid
RELEASE	LU_SCOPE	ignored	invalid
CLEAR	ignored	ignored	invalid
PREEMPT	LU_SCOPE	ignored	invalid
PREEMPT AND ABORT	LU_SCOPE	ignored	invalid
REGISTER AND MOVE	LU_SCOPE	not applicable [a]	invalid

[a] The parameter list format for the REGISTER AND MOVE service action is described in 6.14.4.

6.14.4 PERSISTENT RESERVE OUT command with REGISTER AND MOVE service action parameters

The parameter list format shown in table 183 shall be used by the PERSISTENT RESERVE OUT command with REGISTER AND MOVE service action.

Table 183 — PERSISTENT RESERVE OUT with REGISTER AND MOVE service action parameter list

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)							
•••				RESERVATION KEY				
7								(LSB)
8	(MSB)							
•••				SERVICE ACTION RESERVATION KEY				
15								(LSB)
16				Reserved				
17				Reserved			UNREG	APTPL
18	(MSB)							
19				RELATIVE TARGET PORT IDENTIFIER				(LSB)
20	(MSB)							
•••				TRANSPORTID PARAMETER DATA LENGTH (n-23)				
23								(LSB)
24								
•••				TransportID				
n								

The RESERVATION KEY field contains an 8-byte value provided by the application client to the device server to identify the I_T nexus that is the source of the PERSISTENT RESERVE OUT command. The device server shall verify that the contents of the RESERVATION KEY field in a PERSISTENT RESERVE OUT command parameter data matches the registered reservation key for the I_T nexus from which the command was received. If a PERSISTENT RESERVE OUT command specifies a RESERVATION KEY field other than the reservation key registered for the I_T nexus, the device server shall complete the command with RESERVATION CONFLICT status.

The SERVICE ACTION RESERVATION KEY field contains the reservation key to be registered to the specified I_T nexus.

The Activate Persist Through Power Loss (APTPL) bit set to one is optional. If a device server that does not support an APTPL bit set to one receives that value, it shall return CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

If the last valid APTPL bit value received by the device server is zero, the loss of power in the SCSI target device shall release the persistent reservation for the logical unit and remove all registered reservation keys (see 5.6.5). If the last valid APTPL bit value received by the device server is one, the logical unit shall retain any persistent reservation(s) that may be present and all reservation keys (i.e., registrations) for all I_T nexuses even if power is lost and later returned (see 5.9.5).

The unregister (UNREG) bit set to zero specifies that the device server shall not unregister the I_T nexus on which the PERSISTENT RESERVE OUT command REGISTER AND MOVE service action was received. An UNREG bit set to one specifies that the device server shall unregister the I_T nexus on which the PERSISTENT RESERVE OUT command REGISTER AND MOVE service action was received.

The RELATIVE TARGET PORT IDENTIFIER field specifies the relative port identifier (see 3.1.127) of the target port in the I_T nexus to which the persistent reservation is to be moved.

The TRANSPORTID DESCRIPTOR LENGTH field specifies the number of bytes of the TransportID that follows, shall be a minimum of 24 bytes, and shall be a multiple of 4.

The TransportID specifies the initiator port in the I_T nexus to which the persistent reservation is to be moved. The format of the TransportID is defined in 7.6.4.

The command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST:

a) if the value in the PARAMETER LIST LENGTH field in the CDB does not include all of the parameter list bytes specified by the TRANSPORTID PARAMETER DATA LENGTH field; or
b) if the value in the TRANSPORTID PARAMETER DATA LENGTH field results in the truncation of a TransportID.

6.15 READ ATTRIBUTE command

6.15.1 READ ATTRIBUTE command introduction

The READ ATTRIBUTE command (see table 184) allows an application client to read attribute values from medium auxiliary memory.

Table 184 — READ ATTRIBUTE command

Bit / Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (8Ch)							
1	Reserved			SERVICE ACTION				
2 ••• 4	Restricted (see SMC-2)							
5	LOGICAL VOLUME NUMBER							
6	Reserved							
7	PARTITION NUMBER							
8	(MSB)	FIRST ATTRIBUTE IDENTIFIER						
9								(LSB)
10 ••• 13	(MSB)	ALLOCATION LENGTH						(LSB)
14	Reserved							
15	CONTROL							

The OPERATION CODE field is defined in 4.3.5.1 and shall be set as shown in table 184 for the READ ATTRIBUTE command.

If the medium auxiliary memory is not accessible because there is no medium present, the READ ATTRIBUTE command shall be terminated with CHECK CONDITION status, with the sense key set to NOT READY, and the additional sense code set to MEDIUM NOT PRESENT.

If the medium is present but the medium auxiliary memory is not accessible, the READ ATTRIBUTE command shall be terminated with CHECK CONDITION status, with the sense key set to MEDIUM ERROR, and the additional sense code set to LOGICAL UNIT NOT READY, AUXILIARY MEMORY NOT ACCESSIBLE.

If the medium auxiliary memory is not operational, the READ ATTRIBUTE command shall be terminated with CHECK CONDITION status, with the sense key set to MEDIUM ERROR, and the additional sense code set to AUXILIARY MEMORY READ ERROR.

The SERVICE ACTION field is defined in 4.3.5.2. The service actions defined for the READ ATTRIBUTE command are shown in table 185.

Table 185 — READ ATTRIBUTE service action codes

Code	Name	Description	Reference
00h	ATTRIBUTE VALUES	Return attribute values.	6.15.2
01h	ATTRIBUTE LIST	Return a list of available attribute identifiers, identifiers that are in the read only state or in the read/write state (see 5.13).	6.15.3
02h	LOGICAL VOLUME LIST	Return a list of known logical volume numbers.	6.15.4
03h	PARTITION LIST	Return a list of known partition numbers.	6.15.5
04h	Restricted		SMC-3
05h	SUPPORTED ATTRIBUTES	Return a list of supported attribute identifiers, identifiers that are in the read only state, in the read/write state, or in the nonexistent state (see 5.13).	6.15.6
06h to 1Fh	Reserved		

The LOGICAL VOLUME NUMBER field specifies a logical volume (e.g., the medium auxiliary memory storage for one side of a double sided medium) within the medium auxiliary memory. The number of logical volumes of the medium auxiliary memory shall equal that of the attached medium. If the medium only has a single logical volume, then its logical volume number shall be zero.

The PARTITION NUMBER field specifies a partition (see SSC-2) within a logical volume. The number of partitions of the medium auxiliary memory shall equal that of the attached medium. If the medium only has a single partition, then its partition number shall be zero.

If the combination of logical volume number and partition number is not valid, 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 FIRST ATTRIBUTE IDENTIFIER field specifies the attribute identifier of the first attribute to be returned. If the specified attribute is in the unsupported state or nonexistent state (see 5.13), the READ 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 CDB.

The ALLOCATION LENGTH field is defined in 4.3.5.6.

The CONTROL field is defined in SAM-5.

The format of parameter data returned by the READ ATTRIBUTE command depends on the service action specified.

6.15.2 ATTRIBUTE VALUES service action

The READ ATTRIBUTE command with ATTRIBUTE VALUES service action returns parameter data containing the attributes that are in the read state or read/write state (see 5.13) specified by the PARTITION NUMBER, LOGICAL VOLUME NUMBER, and FIRST ATTRIBUTE IDENTIFIER fields in the CDB. The returned parameter data shall contain the requested attributes in ascending numerical order by attribute identifier value and in the format shown in table 186.

Table 186 — READ ATTRIBUTE with ATTRIBUTE VALUES service action parameter list format

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)							
•••				AVAILABLE DATA (n-3)				
3								(LSB)
				Attribute(s)				
4								
•••				Attribute 0 (see 7.4.1)				
				⋮				
•••				Attribute x (see 7.4.1)				
n								

The AVAILABLE DATA field shall contain the number of bytes of attribute information in the parameter list. The relationship between the AVAILABLE DATA field and the CDB ALLOCATION LENGTH field is defined in 4.3.5.6.

The format of the attributes is described in 7.4.1.

6.15.3 ATTRIBUTE LIST service action

The READ ATTRIBUTE command with ATTRIBUTE LIST service action returns parameter data containing the attribute identifiers for the attributes that are in the read only state or in the read/write state (see 5.13) in the specified partition and volume number. The contents of FIRST ATTRIBUTE IDENTIFIER field in the CDB shall be ignored. The returned parameter data shall contain the requested attribute identifiers in ascending numerical order by attribute identifier value and in the format shown in table 187.

Table 187 — READ ATTRIBUTE with ATTRIBUTE LIST service action parameter list format

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)							
•••				AVAILABLE DATA (n-3)				
3								(LSB)
	Attribute identifiers							
4				ATTRIBUTE IDENTIFIER 0				
5								
	⋮							
n-1				ATTRIBUTE IDENTIFIER x				
n								

The AVAILABLE DATA field shall contain the number of bytes of attribute identifiers in the parameter list. The relationship between the AVAILABLE DATA field and the CDB ALLOCATION LENGTH field is defined in 4.3.5.6.

An ATTRIBUTE IDENTIFIER field is returned for each attribute that is in the read only state or in the read/write state (see 5.13) in the specified partition and volume number. See 7.4.2 for a description of the attribute identifier values.

6.15.4 LOGICAL VOLUME LIST service action

The READ ATTRIBUTE command with LOGICAL VOLUME LIST service action returns parameter data (see table 188) identifying the supported number of logical volumes. The contents of LOGICAL VOLUME NUMBER, PARTITION NUMBER, and FIRST ATTRIBUTE IDENTIFIER fields in the CDB shall be ignored.

Table 188 — READ ATTRIBUTE with LOGICAL VOLUME LIST service action parameter list format

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)							
1				AVAILABLE DATA (0002h)				(LSB)
2	FIRST LOGICAL VOLUME NUMBER							
3	NUMBER OF LOGICAL VOLUMES AVAILABLE							

The AVAILABLE DATA field shall contain two. The relationship between the AVAILABLE DATA field and the CDB ALLOCATION LENGTH field is defined in 4.3.5.6.

The FIRST LOGICAL VOLUME NUMBER field indicates the first volume available. Logical volume numbering should start at zero.

The NUMBER OF LOGICAL VOLUMES AVAILABLE field indicates the number of volumes available.

6.15.5 PARTITION LIST service action

The READ ATTRIBUTE command with PARTITION LIST service action returns parameter data (see table 189) identifying the number of partitions supported in the specified logical volume number. The contents of PARTITION NUMBER and FIRST ATTRIBUTE IDENTIFIER fields in the CDB shall be ignored.

Table 189 — READ ATTRIBUTE with PARTITION LIST service action parameter list format

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)	AVAILABLE DATA (0002h)						
1								(LSB)
2	FIRST PARTITION NUMBER							
3	NUMBER OF PARTITIONS AVAILABLE							

The AVAILABLE DATA field shall contain two. The relationship between the AVAILABLE DATA field and the CDB ALLOCATION LENGTH field is defined in 4.3.5.6.

The FIRST PARTITION NUMBER field indicates the first partition available on the specified logical volume number. Partition numbering should start at zero.

The NUMBER OF PARTITIONS AVAILABLE field indicates the number of partitions available on the specified logical volume number.

6.15.6 SUPPORTED ATTRIBUTES service action

The READ ATTRIBUTE command with SUPPORTED ATTRIBUTES service action returns parameter data containing the attribute identifiers for the attributes that are in the read only state, in the read/write state, or in the nonexistent state (see 5.13) in the specified partition and volume number. The contents of FIRST ATTRIBUTE IDENTIFIER field in the CDB shall be ignored. The returned parameter data shall contain the requested attribute identifiers in ascending numerical order by attribute identifier value and in the format shown in table 190.

Table 190 — READ ATTRIBUTE with SUPPORTED ATTRIBUTES service action parameter list format

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)	AVAILABLE DATA (n-3)						
•••								
3								(LSB)
	Attribute identifiers							
4	ATTRIBUTE IDENTIFIER 0							
5								
	⋮							
n-1	ATTRIBUTE IDENTIFIER x							
n								

The AVAILABLE DATA field shall contain the number of bytes of attribute identifiers in the parameter list. The relationship between the AVAILABLE DATA field and the CDB ALLOCATION LENGTH field is defined in 4.3.5.6.

An ATTRIBUTE IDENTIFIER field is returned for each attribute that is in the read only state, in the read/write state, or in the nonexistent state (see 5.13) in the specified partition and volume number. See 7.4.2 for a description of the attribute identifier values.

6.16 READ BUFFER command

6.16.1 READ BUFFER command introduction

The READ BUFFER command (see table 191) is used in conjunction with the WRITE BUFFER command for:

a) testing logical unit buffer memory;
b) testing the integrity of the service delivery subsystem;
c) downloading microcode (see 5.18); and
d) retrieving error history.

Table 191 — READ BUFFER command

Bit / Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (3Ch)							
1	Reserved			MODE				
2	BUFFER ID							
3	(MSB)	BUFFER OFFSET						
4								
5								(LSB)
6	(MSB)	ALLOCATION LENGTH						
7								
8								(LSB)
9	CONTROL							

The OPERATION CODE field is defined in 4.3.5.1 and shall be set as shown in table 191 for the READ BUFFER command.

The function of this command and the meaning of fields within the CDB depend on the contents of the MODE field. The MODE field is defined in table 192.

Table 192 — READ BUFFER MODE field

Code	Description	References
00h	Combined header and data [a]	6.16.2
01h	Vendor specific [a]	6.16.3
02h	Data	6.16.4
03h	Descriptor	6.16.5
04h to 09h	Reserved	
0Ah	Read data from echo buffer	6.16.6
0Bh	Echo buffer descriptor	6.16.7
0Ch to 19h	Reserved	
1Ah	Enable expander communications protocol and Echo buffer	6.16.8
1Bh	Reserved	
1Ch	Error history	5.14 and 6.16.9
1Dh to 1Fh	Reserved	
[a] Modes 00h and 01h are not recommended.		

If the mode is not set to one, the ALLOCATION LENGTH field is defined in 4.3.5.6.

The CONTROL field is defined in SAM-5.

6.16.2 Combined header and data mode (00h)

In this mode, a four-byte header followed by data bytes is returned to the application client in the Data-In Buffer. The allocation length should be set to four or greater. The BUFFER ID and the BUFFER OFFSET fields are reserved.

The four-byte READ BUFFER header (see table 193) is followed by data bytes from the buffer.

Table 193 — READ BUFFER header

Bit / Byte	7	6	5	4	3	2	1	0
0	Reserved							
1	(MSB)			BUFFER CAPACITY				
2								
3								(LSB)
4				Data				
•••								
n								

The BUFFER CAPACITY field specifies the total number of data bytes available in the buffer. The buffer capacity is not reduced to reflect the actual number of bytes written using the WRITE BUFFER command with combined header and data mode (see 6.39.2). The relationship between the BUFFER CAPACITY field and the CDB ALLOCATION LENGTH field is defined in 4.3.5.6. Following the READ BUFFER header, the device server shall transfer data from the buffer.

NOTE 37 - The buffer is shared by all application clients and I_T nexuses. If one application client writes the buffer, a second application client writes the buffer, and then the first application client reads the buffer, the read may or may not retrieve the data from the second application client.

6.16.3 Vendor specific mode (01h)

In this mode, the meanings of the BUFFER ID, BUFFER OFFSET, and ALLOCATION LENGTH fields are not specified by this standard.

6.16.4 Data mode (02h)

In this mode, the Data-In Buffer is filled only with logical unit buffer data. The BUFFER ID field specifies a buffer within the logical unit from which data shall be transferred. The vendor assigns buffer ID codes to buffers within the logical unit. Buffer ID zero shall be supported. If more than one buffer is supported, then additional buffer ID codes shall be assigned contiguously, beginning with one. Buffer ID code assignments for the READ BUFFER command with data mode shall be the same as for the WRITE BUFFER command with data mode (see 6.39.4). If an unsupported buffer ID code is selected, 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 BUFFER OFFSET field contains the byte offset within the specified buffer from which data shall be transferred. The application client should conform to the offset boundary requirements returned in the READ BUFFER descriptor (see 6.16.5). If the device server is unable to accept the specified buffer offset, 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.

NOTE 38 - The buffer is shared by all application clients and I_T nexuses. If one application client writes the buffer, a second application client writes the buffer, and then the first application client reads the buffer, the read may or may not retrieve the data from the second application client.

6.16.5 Descriptor mode (03h)

In this mode, a maximum of four bytes of READ BUFFER descriptor information is returned. The BUFFER OFFSET field is reserved in this mode. The allocation length should be set to four or greater. The READ BUFFER descriptor is defined as shown in table 194.

Table 194 — READ BUFFER descriptor

Bit / Byte	7	6	5	4	3	2	1	0
0	OFFSET BOUNDARY							
1	(MSB)							
2	BUFFER CAPACITY							
3								(LSB)

For READ BUFFER commands, the OFFSET BOUNDARY field (see table 195) applies to the following modes:

a) data (i.e., 02h) (see 6.16.4); and
b) error history (i.e., 1Ch) (see 6.16.9).

For WRITE BUFFER commands, the OFFSET BOUNDARY field (see table 195) applies to the following modes:

a) data (i.e., 02h) (see 6.39.4);

b) download microcode with offsets and activate (i.e., 06h) (see 6.39.7);
c) download microcode with offsets, save, and activate (i.e., 07h) (see 6.39.8); and
d) download microcode with offsets, save, and defer active (i.e., 0Eh) (see 6.39.10).

For data mode (i.e., 02h), the boundary alignment indicated by the OFFSET BOUNDARY field applies only to the buffer specified by the BUFFER ID field. For modes other than data to which the OFFSET BOUNDARY field applies, the boundary alignment applies regardless of the buffer specified by the BUFFER ID field.

Table 195 — OFFSET BOUNDARY field

Code	Description
00h to FEh	Multiples of 2^{code} (e.g., 00h means multiples of 1 byte or no offset restrictions, 01h means multiples of 2 bytes or even offsets, 02h means multiples of 4 bytes)
FFh	000000h is the only supported buffer offset

The BUFFER CAPACITY field indicates the maximum size in bytes of the buffer specified by the BUFFER ID field for the:

a) READ BUFFER command with data mode (i.e., 02h); and
b) WRITE BUFFER command with:
   A) data mode (i.e., 02h);
   B) download microcode with offsets and activate mode (i.e., 06h);
   C) download microcode with offsets, save, and activate mode (i.e., 07h); and
   D) download microcode with offsets, save, and defer activate mode (i.e., 0Eh).

6.16.6 Read data from echo buffer mode (0Ah)

In this mode the device server transfers data to the application client from the echo buffer that was written by the most recent WRITE BUFFER command with the MODE field set to write data to echo buffer (see 6.39.9) received on the same I_T nexus. The READ BUFFER command shall return the same number of bytes of data as received in the prior WRITE BUFFER command with the mode field set to write data to echo buffer, limited by the allocation length as described in 4.3.5.6.

The BUFFER ID and BUFFER OFFSET fields are ignored in this mode.

If no WRITE BUFFER command with the mode set to write data to echo buffer received on this I_T nexus has completed without an error, then the READ BUFFER command shall terminate with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to COMMAND SEQUENCE ERROR. If the data in the echo buffer has been overwritten by another I_T nexus and the device server supports error reporting on echo buffer overwrites (i.e., the EBOS bit is set to one in the echo buffer descriptor (see 6.16.7)), then the READ BUFFER command shall be terminated with CHECK CONDITION status, with the sense key set to ABORTED COMMAND, and the additional sense code set to ECHO BUFFER OVERWRITTEN.

After a WRITE BUFFER command with the mode set to write data to echo buffer has completed without an error, the application client may send multiple READ BUFFER commands with the mode set to read data from echo buffer in order to read the echo buffer data multiple times.

6.16.7 Echo buffer descriptor mode (0Bh)

In this mode, a maximum of four bytes of READ BUFFER descriptor information is returned. The device server shall return the descriptor information for the echo buffer. If there is no echo buffer implemented, the device server shall return all zeros in the READ BUFFER descriptor. The BUFFER ID field and BUFFER OFFSET field are reserved in

this mode. The allocation length should be set to four or greater. The READ BUFFER descriptor is defined as shown in table 196.

Table 196 — Echo buffer descriptor

Bit / Byte	7	6	5	4	3	2	1	0
0	Reserved							EBOS
1	Reserved							
2	Reserved			(MSB)				
3	BUFFER CAPACITY							(LSB)

The BUFFER CAPACITY field shall return the size of the echo buffer in bytes aligned to a four-byte boundary. The maximum echo buffer size is 4 096 bytes.

If the echo buffer is implemented, the echo buffer descriptor shall be implemented.

An echo buffer overwritten supported (EBOS) bit set to one indicates either:

a) the device server returns the ECHO BUFFER OVERWRITTEN additional sense code if the data being read from the echo buffer is not the data previously written by the same I_T nexus, or
b) the device server ensures echo buffer data returned to each I_T nexus is the same as that previously written by that I_T nexus.

An EBOS bit set to zero specifies that the echo buffer may be overwritten by any intervening command received on any I_T nexus.

A READ BUFFER command with the mode set to echo buffer descriptor may be used to determine the echo buffer capacity and supported features before a WRITE BUFFER command with the mode set to write data to echo buffer (see 6.39.9) is sent.

6.16.8 Enable expander communications protocol and Echo buffer (1Ah)

Receipt of a READ BUFFER command with this mode (1Ah) causes a communicative expander (see SPI-5) to enter the expanded communications protocol mode. Device servers in SCSI target devices that receive a READ BUFFER command with this mode shall process it as if it were a READ BUFFER command with mode 0Ah (see 6.16.6).

6.16.9 Error history mode (1Ch)

6.16.9.1 Error history overview

This mode is used to manage and retrieve error history (see 5.14).

If the device server is unable to process a READ BUFFER command with the MODE field set to 1Ch because of a vendor specific condition, then the device server shall terminate the READ BUFFER command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to COMMAND SEQUENCE ERROR.

The BUFFER ID field (see table 197) specifies the action that the device server shall perform, and the parameter data, if any, that the device server shall return.

Table 197 — Error history BUFFER ID field

Code	Description	Buffer offset	Error history I_T nexus constrained	Reference
00h	Return error history directory	0000h	Yes	6.16.9.2
01h	Return error history directory and create new error history snapshot (see 3.1.54)	0000h	Yes	6.16.9.2
02h	Return error history directory and establish new error history I_T nexus (see 3.1.53)	0000h	No	6.16.9.2
03h	Return error history directory, establish new error history I_T nexus, and create new error history snapshot	0000h	No	6.16.9.2
04h to 0Fh	Reserved		Yes	
10h to EFh	Return error history	0000h to FFFFh	Yes	6.16.9.3
F0h to FDh	Reserved		Yes	
FEh	Clear error history I_T nexus	Ignored	Yes	6.16.9.4
FFh	Clear error history I_T nexus and release error history snapshot	Ignored	Yes	6.16.9.5

The command shall be terminated with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to OPERATION IN PROGRESS if the device server receives a READ BUFFER command:

a) with the MODE field set to 1Ch;
b) with the BUFFER ID field set to a value that table 197 shows as constrained by error history I_T nexus;
c) if an error history I_T nexus exists and the command is received from an I_T nexus that is different that I_T nexus; and
d) an error history snapshot exists.

The BUFFER OFFSET field specifies the byte offset from the start of the buffer specified by the BUFFER ID field from which the device server shall return data. The application client should conform to the offset boundary requirements indicated in the READ BUFFER descriptor (see 6.16.5). If the buffer offset is not one of those shown in table 197 or the device server is unable to accept the specified buffer offset, then the device server shall terminate the READ BUFFER command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB.

6.16.9.2 Error history directory

Whenever allowed by established error history I_T nexus constraints (see 6.16.9.1), if any, all error history device server actions return an error history directory (see table 199). Some error history device server actions also discard the existing error history snapshot (see 3.1.54) and create a new error history snapshot (see table 198).

Table 198 — Summary of error history directory device server actions

BUFFER ID field	Establish new error history I_T nexus (see 3.1.53)	Error history snapshot (see 3.1.54)	
		Preserved (if exists)	Created
00h	No [a]	Yes	No [b]
01h	No [a]	No	Yes
02h	Yes	Yes	No [b]
03h	Yes	No	Yes

[a] If no error history I_T nexus is established, a new one is established.
[b] If no error history snapshot exists, a new one is created.

The error history directory is defined in table 199.

Table 199 — Error history directory

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)							
•••				T10 VENDOR IDENTIFICATION				
7								(LSB)
8				VERSION				
9		Reserved			EHS_RETRIEVED	EHS_SOURCE		CLR_SUP
10								
•••				Reserved				
29								
30	(MSB)							
31				DIRECTORY LENGTH (n-31)				(LSB)
				Error history directory list				
32								
•••				Error history directory entry [first] (see table 202)				
39								
				⋮				
n-7								
•••				Error history directory entry [last] (see table 202)				
n								

The T10 VENDOR IDENTIFICATION field contains eight bytes of left-aligned ASCII data (see 4.4.1) identifying the manufacturer of the logical unit. The T10 vendor identification shall be one assigned by INCITS. A list of assigned T10 vendor identifications is in Annex E and on the T10 web site (http://www.t10.org).

NOTE 39 - The T10 VENDOR IDENTIFICATION field may contain a different value than the VENDOR IDENTIFICATION field in the standard INQUIRY data (see 6.4.2) (e.g., this field may indicate a disk drive component vendor while the standard INQUIRY data indicates the original equipment manufacturer).

The VERSION field indicates the version and format of the vendor specific error history. The VERSION field is assigned by the vendor indicated in the T10 VENDOR IDENTIFICATION field.

The error history retrieved (EHS_RETRIEVED) field (see table 200) indicates whether a clear error history device server action has been requested for the error history snapshot. EHS_RETRIEVED field shall be set to 00b or 10b when the error history snapshot is created.

Table 200 — EHS_RETRIEVED field

Code	Description
00b	No information
01b	The error history I_T nexus has requested buffer ID FEh (i.e., clear error history I_T nexus) or buffer ID FFh (i.e., clear error history I_T nexus and release snapshot) for the current error history snapshot.
10b	An error history I_T nexus has not requested buffer ID FEh (i.e., clear error history I_T nexus) or buffer ID FFh (i.e., clear error history I_T nexus and release snapshot) for the current error history snapshot.
11b	Reserved

The error history source (EHS_SOURCE) field (see table 201) indicates the source of the error history snapshot.

Table 201 — EHS_SOURCE field

Code	Description
00b	The error history snapshot was created by the device server and was not created due to processing a READ BUFFER command.
01b	Error history snapshot was created due to processing of the current READ BUFFER command
10b	Error history snapshot was created due to processing of a previous READ BUFFER command
11b	Reserved

A clear support (CLR_SUP) bit set to one indicates that the CLR bit is supported in the WRITE BUFFER command download error history mode (see 6.39.14). A CLR_SUP bit set to zero indicates that the CLR bit is not supported.

The DIRECTORY LENGTH field indicates the number of error history directory list bytes available to be transferred. This value shall not be altered even if the allocation length is not sufficient to transfer the entire error history directory list.

The error history directory list contains an error history directory entry (see table 202) for each supported buffer ID in the range of 00h to EFh. The first entry shall be for buffer ID 00h and the entries shall be in order of ascending buffer IDs. The supported buffer IDs are not required to be contiguous. There shall not be any entries for buffer IDs greater than or equal to F0h.

Table 202 — Error history directory entry

Bit / Byte	7	6	5	4	3	2	1	0
0	SUPPORTED BUFFER ID							
1	Reserved							
•••								
3								
4	(MSB)	MAXIMUM AVAILABLE LENGTH						
•••								
7								(LSB)

The SUPPORTED BUFFER ID field indicates the error history buffer ID associated with this entry.

The MAXIMUM AVAILABLE LENGTH field indicates the maximum number of data bytes contained in the buffer indicated by the SUPPORTED BUFFER ID field. The actual number of bytes available for transfer may be smaller.

6.16.9.3 Error history data buffer

Unless an error is encountered, the device server shall return parameter data that contains error history in a vendor specific format from the error history snapshot from the specified buffer at the specified buffer offset.

If the device server receives a READ BUFFER command with the MODE field set to 1Ch from the established error history I_T nexus and the BUFFER ID field is set to a value that the error history directory (see 6.16.9.2) shows as not supported, then the command shall be terminated with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.

If the value in the BUFFER OFFSET field is not supported, 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 amount of error history in the specified buffer shall be less than or equal to the number of bytes indicated by the MAXIMUM AVAILABLE LENGTH field in the error history directory (see 6.16.9.2).

6.16.9.4 Clear error history I_T nexus

If the BUFFER ID field is set to FEh, the device server shall:

a) clear the error history I_T nexus, if any; and
b) not transfer any data.

6.16.9.5 Clear error history I_T nexus and release snapshot

If the BUFFER ID field is set to FFh, the device server shall:

a) clear the error history I_T nexus, if any,
b) release the error history snapshot, if any; and

c) not transfer any data.

6.17 READ MEDIA SERIAL NUMBER command

The READ MEDIA SERIAL NUMBER command (see table 203) reports the media serial number reported by the device and the currently mounted media.

Table 203 — READ MEDIA SERIAL NUMBER command

Bit / Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (ABh)							
1	Reserved			SERVICE ACTION (01h)				
2	Reserved							
•••								
5								
6	(MSB)	ALLOCATION LENGTH						
•••								
9								(LSB)
10	Reserved							
11	CONTROL							

The OPERATION CODE field is defined in 4.3.5.1 and shall be set as shown in table 203 for the READ MEDIA SERIAL NUMBER command.

The SERVICE ACTION field is defined in 4.3.5.2 and shall be set as shown in table 203 for the READ MEDIA SERIAL NUMBER command.

The ALLOCATION LENGTH field is defined in 4.3.5.6.

The CONTROL field is defined in SAM-5.

The READ MEDIA SERIAL NUMBER parameter data format is shown in table 204.

Table 204 — READ MEDIA SERIAL NUMBER parameter data format

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)	MEDIA SERIAL NUMBER LENGTH (4n-4)						
•••								
3								(LSB)
4	MEDIA SERIAL NUMBER							
•••								
4n-1								

The MEDIA SERIAL NUMBER LENGTH field shall contain the number of bytes in the MEDIA SERIAL NUMBER field. The media serial number length shall be a multiple of four. The relationship between the MEDIA SERIAL NUMBER LENGTH field and the CDB ALLOCATION LENGTH field is defined in 4.3.5.6.

The MEDIA SERIAL NUMBER field shall contain the vendor specific serial number of the media currently installed. If the number of bytes in the vendor specific serial number is not a multiple of four, then up to three bytes containing zero shall be appended to the highest numbered bytes of the MEDIA SERIAL NUMBER field.

If the media serial number is not available (e.g., the currently installed media has no valid media serial number), zero shall be returned in the MEDIA SERIAL NUMBER LENGTH field.

If the media serial number is not accessible because there is no media present, the command shall be terminated with CHECK CONDITION status, with the sense key set to NOT READY, and the additional sense code set to MEDIUM NOT PRESENT.

6.18 RECEIVE COPY RESULTS command

6.18.1 RECEIVE COPY RESULTS command introduction

The RECEIVE COPY RESULTS command (see table 205) provides a means for the application client to receive information about the copy manager or the results of a previous or current EXTENDED COPY command (see 6.3).

Table 205 — RECEIVE COPY RESULTS command

Bit / Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (84h)							
1	Reserved			SERVICE ACTION				
2	LIST IDENTIFIER							
3	Reserved							
•••								
9								
10	(MSB)	ALLOCATION LENGTH						
•••								
13								(LSB)
14	Reserved							
15	CONTROL							

The OPERATION CODE field is defined in 4.3.5.1 and shall be set as shown in table 205 for the RECEIVE COPY RESULTS command.

The SERVICE ACTION field is defined in 4.3.5.2. The service actions defined for the RECEIVE COPY RESULTS command are shown in table 206.

Table 206 — RECEIVE COPY RESULTS service action codes

Code	Name	Description	Returns Data While EXTENDED COPY Is In Progress	Reference
00h	COPY STATUS	Return the current copy status of the EXTENDED COPY command identified by the LIST IDENTIFIER field.	Yes	6.18.2
01h	RECEIVE DATA	Return the held data read by EXTENDED COPY command identified by the LIST IDENTIFIER field.	No	6.18.3
03h	OPERATING PARAME-TERS	Return copy manager operating parameters.	Yes	6.18.4
04h	FAILED SEGMENT DETAILS	Return copy target device sense data and other information about the progress of processing a segment descriptor whose processing was not completed during processing of the EXTENDED COPY command identified by the LIST IDENTIFIER field.	No	6.18.5
05h to 1Eh	Reserved			
1Fh	Vendor Specific			

The LIST IDENTIFIER field specifies the EXTENDED COPY command (see 6.3) about which information is to be transferred. The RECEIVE COPY RESULTS command shall return information from the EXTENDED COPY command received on the same I_T nexus with a list identifier that matches the list identifier specified in the RECEIVE COPY RESULTS CDB.

If the LIST IDENTIFIER field specifies an EXTENDED COPY command that had the LIST ID USAGE field set to 10b or 11b in the parameter data (see 6.3.1), the copy manager may respond to a RECEIVE COPY RESULTS command as if the EXTENDED COPY command had never been received.

The actual length of the RECEIVE COPY RESULTS parameter data is available in the AVAILABLE DATA parameter data field. The ALLOCATION LENGTH field is defined in 4.3.5.6. See the RECEIVE COPY RESULTS service action definitions for additional requirements.

The CONTROL field is defined in SAM-5.

6.18.2 COPY STATUS service action

In response to the COPY STATUS service action, the copy manager shall return the current status of the EXTENDED COPY command (see 6.3) specified by the LIST IDENTIFIER field in the CDB. If no EXTENDED COPY command known to the copy manager has a matching list identifier, 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.

Table 207 shows the format of the information returned by the copy manager in response to the COPY STATUS service action. If a device server supports the EXTENDED COPY command with the LIST ID USAGE field is set to 00b or 10b, then it shall also support the RECEIVE COPY RESULTS command with COPY STATUS service action.

Table 207 — Parameter data for the COPY STATUS service action

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)							
•••				AVAILABLE DATA (00000008h)				
3								(LSB)
4	HDD			COPY MANAGER STATUS				
5	(MSB)							
6				SEGMENTS PROCESSED				(LSB)
7				TRANSFER COUNT UNITS				
8	(MSB)							
•••				TRANSFER COUNT				
11								(LSB)

After completion of an EXTENDED COPY command, the copy manager shall preserve all data returned by a COPY STATUS service action for a vendor specific period of time. The copy manager shall discard the COPY STATUS data when:

a) a RECEIVE COPY RESULTS command with COPY STATUS service action is received on the same I_T nexus with a matching list identifier;
b) when another EXTENDED COPY command is received on the same I_T nexus and the list identifier matches the list identifier associated with the data preserved for the COPY STATUS service action;
c) when the copy manager detects a logical unit reset or I_T nexus loss; or
d) when the copy manager requires the resources used to preserve the data.

The AVAILABLE DATA field shall contain the number of bytes present in the parameter data that follows. The relationship between the AVAILABLE DATA field and the CDB ALLOCATION LENGTH field is defined in 4.3.5.6.

The held data discarded (HDD) bit indicates whether held data has been discarded. If HDD bit is set to one, held data has been discarded as described in 6.18.4. If HDD bit is set to zero, held data has not been discarded.

The COPY MANAGER STATUS field contains the current status of the EXTENDED COPY command specified by the LIST IDENTIFIER field in the CDB as defined in table 208.

Table 208 — COPY MANAGER STATUS field

Code	Meaning
00h	Operation in progress
01h	Operation completed without errors
02h	Operation completed with errors
03h to 7Fh	Reserved

The SEGMENTS PROCESSED field contains the number of segments the copy manager has processed for the EXTENDED COPY command specified by the LIST IDENTIFIER field in the CDB including the segment currently being processed. This field shall be zero if the copy manager has not yet begun processing segment descriptors.

The TRANSFER COUNT UNITS field specifies the units for the TRANSFER COUNT field as defined in table 209.

Table 209 — COPY STATUS TRANSFER COUNT UNITS field

Code	Meaning [a]	Multiplier to convert TRANSFER COUNT field to bytes
00h	Bytes	1
01h	Kibibytes	2^{10} or 1024
02h	Mebibytes	2^{20}
03h	Gibibytes	2^{30}
04h	Tebibytes	2^{40}
05h	Pebibytes	2^{50}
06h	Exbibytes	2^{60}
07h to FFh	Reserved	

[a] See 3.5.2.

The TRANSFER COUNT field specifies the amount of data written to a destination device for the EXTENDED COPY command specified by the LIST IDENTIFIER field in the CDB prior to receiving the RECEIVE COPY RESULTS command with COPY STATUS service action.

6.18.3 RECEIVE DATA service action

If the copy manager supports those segment descriptors that require data to be held for transfer to the application client, then the RECEIVE DATA service action causes the copy manager to return the held data using the format shown in table 210. If a copy manager supports any of the segment descriptor type codes that require data to be held for the application client (see 6.3.5), then it shall also support the RECEIVE COPY RESULTS command with RECEIVE DATA service action.

The EXTENDED COPY 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 any of the following conditions exist:

a) no EXTENDED COPY command known to the copy manager has a list identifier that matches the LIST IDENTIFIER field in the CDB; or
b) if the LIST IDENTIFIER field in the CDB identifies an EXTENDED COPY command that still is being processed by the copy manager.

Table 210 — Parameter data for the RECEIVE DATA service action

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)							
•••				AVAILABLE DATA (n-3)				
3								(LSB)
4								
•••				HELD DATA				
n								

Following completion of an EXTENDED COPY command, the copy manager shall preserve all data returned by a RECEIVE DATA service action for a vendor specific period of time. The application client should issue a RECEIVE COPY RESULTS command with RECEIVE DATA service action as soon as practical following completion of the EXTENDED COPY command to ensure that the data is not discarded by the copy manager. The copy manager shall discard the buffered inline data:

a) after all data held for a specific EXTENDED COPY command has been successfully transferred to the application client;
b) when a RECEIVE COPY RESULTS command with RECEIVE DATA service action has been received on the same I_T nexus with a matching list identifier, with the ALLOCATION LENGTH field set to zero;
c) when another EXTENDED COPY command is received on the same I_T nexus and the list identifier matches the list identifier associated with the data preserved for RECEIVE DATA service action;
d) when the copy manager detects a logical unit reset or I_T nexus loss; or
e) when the copy manager requires the resources used to preserve the data.

The AVAILABLE DATA field shall contain the number of bytes of held data available for delivery to the application client. The relationship between the AVAILABLE DATA field and the CDB ALLOCATION LENGTH field is defined in 4.3.5.6.

The HELD DATA field contains the data held by the copy manager for delivery to the application client as prescribed by several segment descriptor type codes. Unless the copy manager's held data limit (see 6.18.4) is exceeded, the first byte held in response to the first segment descriptor in the EXTENDED COPY parameter list prescribing the holding of data (i.e., the oldest byte held) is returned in byte 4. The last byte held in response to the last segment descriptor in the EXTENDED COPY parameter list prescribing the holding of data (i.e., the newest byte held) is returned in byte n.

6.18.4 OPERATING PARAMETERS service action

In response to the OPERATING PARAMETERS service action, the copy manager shall return its operating parameter information in the format shown in table 211. If a device server supports the EXTENDED COPY command (see 6.3), then it shall also support the RECEIVE COPY RESULTS command with OPERATING PARAMETERS service action.

Table 211 — Parameter data for the OPERATING PARAMETERS service action (part 1 of 2)

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)							
•••				AVAILABLE DATA (n-3)				
3								(LSB)
4				Reserved				SNLID
5								
6				Reserved				
7								
8	(MSB)							
9				MAXIMUM TARGET DESCRIPTOR COUNT				(LSB)
10	(MSB)							
11				MAXIMUM SEGMENT DESCRIPTOR COUNT				(LSB)
12	(MSB)							
•••				MAXIMUM DESCRIPTOR LIST LENGTH				
15								(LSB)
16	(MSB)							
•••				MAXIMUM SEGMENT LENGTH				
19								(LSB)
20	(MSB)							
•••				MAXIMUM INLINE DATA LENGTH				
23								(LSB)
24	(MSB)							
•••				HELD DATA LIMIT				
27								(LSB)
28	(MSB)							
•••				MAXIMUM STREAM DEVICE TRANSFER SIZE				
31								(LSB)
32				Reserved				
33								
34	(MSB)							
35				TOTAL CONCURRENT COPIES				(LSB)

Table 211 — Parameter data for the OPERATING PARAMETERS service action (part 2 of 2)

Bit / Byte	7	6	5	4	3	2	1	0
36	MAXIMUM CONCURRENT COPIES							
37	DATA SEGMENT GRANULARITY (log 2)							
38	INLINE DATA GRANULARITY (log 2)							
39	HELD DATA GRANULARITY (log 2)							
40	Reserved							
41								
42								
43	IMPLEMENTED DESCRIPTOR LIST LENGTH (n-43)							
44	List of implemented descriptor type codes (ordered)							
•••								
n								

The AVAILABLE DATA field shall contain the number of bytes following the AVAILABLE DATA field in the parameter data (i.e., the total number of parameter data bytes minus 4). The relationship between the AVAILABLE DATA field and the CDB ALLOCATION LENGTH field is defined in 4.3.5.6.

A supports no list identifier (SNLID) bit set to one indicates the copy manager supports an EXTENDED COPY (see 6.3) command parameter list in which the LIST ID USAGE field is set to 11b and the LIST IDENTIFIER field is set to zero as described in table 105 (see 6.3.1). A SNLID bit set to zero indicates the copy manager does not support an EXTENDED COPY command parameter list in which the LIST ID USAGE field is set to 11b.

The MAXIMUM TARGET COUNT field contains the maximum number of target descriptors that the copy manager allows in a single EXTENDED COPY target descriptor list.

The MAXIMUM SEGMENT COUNT field contains the maximum number of segment descriptors that the copy manager allows in a single EXTENDED COPY segment descriptor list.

The MAXIMUM DESCRIPTOR LIST LENGTH field contains the maximum length, in bytes, of the target descriptor list and segment descriptor list. This length includes the embedded data but excludes inline data that follows the descriptors.

The MAXIMUM SEGMENT LENGTH field indicates the length, in bytes, of the largest amount of data that the copy manager supports writing via a single segment. Bytes introduced as a result of the PAD bit being set to one (see 6.3.7) are not counted towards this limit. A value of zero indicates that the copy manager places no limits on the amount of data written by a single segment.

The MAXIMUM INLINE DATA LENGTH field indicates the length, in bytes, of the largest amount of inline data that the copy manager supports in the EXTENDED COPY parameter list. This does not include data included as embedded data within the segment descriptors. The MAXIMUM INLINE DATA LENGTH field applies only to segment descriptors containing the 04h descriptor type code (see 6.3.7.7). The field shall be set to zero when the 04h descriptor type code is not supported by the copy manager.

The HELD DATA LIMIT field indicates the length, in bytes, of the minimum amount of data the copy manager guarantees to hold for return to the application client via the RECEIVE COPY RESULTS command with RECEIVE DATA service action (see 6.18.3). If the processing of segment descriptors requires more data to be held, the copy manager may discard some of the held data in a vendor specific manner that retains the held bytes from the most

recently processed segment descriptors. The discarding of held data bytes shall not be considered an error. If held data is discarded, the HDD bit shall be set as described in 6.18.2.

If the SNLID bit is set to one, the TOTAL CONCURRENT COPIES field contains the maximum number of EXTENDED COPY commands with the LIST ID USAGE field set to 00b, 10b, or 11b that are supported for concurrent processing by the copy manager. If the SNLID bit is set to zero, then the TOTAL CONCURRENT COPIES field shall be set to zero.

The MAXIMUM CONCURRENT COPIES field contains the maximum number of EXTENDED COPY commands with the LIST ID USAGE field set to 00b or 10b that are supported for concurrent processing by the copy manager. If the SNLID bit is set to one, then the contents of the TOTAL CONCURRENT COPIES field shall be greater than or equal to the contents of the MAXIMUM CONCURRENT COPIES field.

Each EXTENDED COPY command with the LIST ID USAGE field set to 00b or 10b reduces the number of EXTENDED COPY commands with the LIST ID USAGE field set to 11b that may be processed concurrently. However, the converse may not be true (e.g. when the number of outstanding EXTENDED COPY commands with the LIST ID USAGE field set to 11b exceeds the difference between the total concurrent copies and maximum concurrent copies).

The DATA SEGMENT GRANULARITY field indicates the length of the smallest data block that copy manager permits in a non-inline segment descriptor (i.e., segment descriptors with type codes other than 04h). The amount of data transferred by a single segment descriptor shall be a multiple of the granularity. The DATA SEGMENT GRANULARITY value is expressed as a power of two. Bytes introduced as a result of the PAD bit being set to one (see 6.3.7) are not counted towards the data length granularity.

The INLINE DATA GRANULARITY field indicates the length of the of the smallest block of inline data that the copy manager permits being written by a segment descriptor containing the 04h descriptor type code (see 6.3.7.7). The amount of inline data written by a single segment descriptor shall be a multiple of the granularity. The INLINE DATA GRANULARITY value is expressed as a power of two. Bytes introduced as a result of the PAD bit being set to one (see 6.3.7) are not counted towards the length granularity.

If the copy manager encounters a data or inline segment descriptor that violates either the data segment granularity or the inline data granularity, the EXTENDED COPY command shall be terminated with CHECK CONDITION status, with the sense key set to COPY ABORTED, and the additional sense code set to COPY SEGMENT GRANULARITY VIOLATION.

The HELD DATA GRANULARITY field indicates the length of the smallest block of held data that the copy manager shall transfer to the application client in response to a RECEIVE COPY RESULTS command with RECEIVE DATA service action (see 6.18.3). The amount of data held by the copy manager in response to any one segment descriptor shall be a multiple of this granularity. The HELD DATA GRANULARITY value is expressed as a power of two.

The MAXIMUM STREAM DEVICE TRANSFER SIZE field indicates the maximum transfer size, in bytes, supported for stream devices.

The IMPLEMENTED DESCRIPTOR LIST LENGTH field contains the length, in bytes, of the list of implemented descriptor type codes.

The list of implemented descriptor type codes contains one byte for each segment or target DESCRIPTOR TYPE CODE value (see 6.3.5) supported by the copy manager, with a unique supported DESCRIPTOR TYPE CODE value in each byte. The DESCRIPTOR TYPE CODE values shall appear in the list in ascending numerical order.

6.18.5 FAILED SEGMENT DETAILS service action

In response to the FAILED SEGMENT DETAILS service action, the copy manager shall return details of the segment processing failure that caused termination of the EXTENDED COPY command (see 6.3) specified by the LIST IDENTIFIER field in the CDB. Table 212 shows the format of the information returned by the copy manager in response to a FAILED SEGMENT DETAILS service action. If a device server supports the EXTENDED COPY command (see 7.4), then it shall also support the RECEIVE COPY RESULTS command with FAILED SEGMENT DETAILS service action.

When processing of an EXTENDED COPY command is aborted and processing of a segment descriptor is incomplete, the copy manager shall preserve details about the progress in processing of that descriptor. These details enable the application client to obtain information it needs to determine the state in which copy target devices, in particular stream devices, have been left by incomplete processing.

The RECEIVE COPY RESULTS command with FAILED SEGMENT DETAILS service action 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 any of the following conditions exist:

a) no EXTENDED COPY command known to the copy manager has a list identifier that matches the LIST IDENTIFIER field in the CDB; or
b) if the LIST IDENTIFIER field in the CDB identifies an EXTENDED COPY command that still is being processed by the copy manager.

Table 212 — Parameter data for the FAILED SEGMENT DETAILS service action

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)							
•••				AVAILABLE DATA (n-3)				
3								(LSB)
4								
•••				Reserved				
55								
56				EXTENDED COPY COMMAND STATUS				
57				Reserved				
58	(MSB)							
59				SENSE DATA LENGTH (n-59)				(LSB)
60								
•••				SENSE DATA				
n								

The application client should issue a RECEIVE COPY RESULTS command with FAILED SEGMENT DETAILS service action immediately following failure of the EXTENDED COPY command to ensure that the information is not discarded by the copy manager. The copy manager shall discard the failed segment details:

a) after all failed segment details held for a specific EXTENDED COPY command have been successfully transferred to the application client;
b) when a RECEIVE COPY RESULTS command with FAILED SEGMENT DETAILS service action has been received on the same I_T nexus with a matching list identifier, with the ALLOCATION LENGTH field set to zero;

c) when another EXTENDED COPY command is received on the same I_T nexus using the same list identifier;
d) when the copy manager detects a logical unit reset or I_T nexus loss; or
e) when the copy manager requires the resources used to preserve the data.

The AVAILABLE DATA field shall contain the number of bytes of failed segment details available for delivery to the application client. If no failed segment details data is available for the specified list identifier then the AVAILABLE DATA field shall be set to zero and no data beyond the AVAILABLE DATA field shall be returned. The relationship between the AVAILABLE DATA field and the CDB ALLOCATION LENGTH field is defined in 4.3.5.6.

The COPY COMMAND STATUS field contains the SCSI status value that was returned for the EXTENDED COPY command identified by the LIST IDENTIFIER field in the CDB.

The SENSE DATA LENGTH field indicates how many bytes of sense data are present in the SENSE DATA field.

The SENSE DATA field contains a copy of the sense data that the copy manager prepared as part of terminating the EXTENDED COPY command identified by the list identifier with CHECK CONDITION status.

6.19 RECEIVE CREDENTIAL command

6.19.1 RECEIVE CREDENTIAL command description

6.19.1.1 Overview

The RECEIVE CREDENTIAL command (see table 213) allows a secure CDB originator (see 5.16.6.2) to receive a credential from a security manager device server (e.g., a CbCS management device server (see 5.16.6.8.3)) for use in a CDB (e.g., use in the CbCS extension descriptor (see 5.16.6.8.16)).

Table 213 — RECEIVE CREDENTIAL command

Bit / Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (7Fh)							
1	CONTROL							
2	Reserved							
•••								
6								
7	ADDITIONAL CDB LENGTH (n-7)							
8	(MSB)	SERVICE ACTION (1800h)						
9								(LSB)
10	(MSB)	ALLOCATION LENGTH						
11								(LSB)
12	(MSB)	AC_SAI						
•••								
15								(LSB)
16	ENCRYPTED REQUEST DESCRIPTOR							
•••								
n								

The OPERATION CODE field is defined in 4.3.5.1 and shall be set as shown in table 213 for the RECEIVE CREDENTIAL command.

The ADDITIONAL CDB LENGTH field is defined in 4.3.3.

The SERVICE ACTION field is defined in 4.3.5.2 and shall be set as shown in table 213 for the RECEIVE CREDENTIAL command.

The ALLOCATION LENGTH field is defined in 4.3.5.6.

The AC_SAI field contains the value of the AC_SAI SA parameter (see 5.16.2.2) for the SA to be used to encrypt the parameter data as described in 6.19.2.1.

The ENCRYPTED REQUEST DESCRIPTOR field shall contain an ESP-SCSI CDB descriptor (see 5.16.7.4). Before encryption and after decryption, the UNENCRYPTED BYTES field (see 5.16.7.3) that are used to compute the ENCRYPTED OR AUTHENTICATED DATA field (see 5.16.7.4) contents shall have the format shown in table 214.

Table 214 — RECEIVE CREDENTIAL command unencrypted bytes format

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)	CREDENTIAL REQUEST TYPE						
1								(LSB)
2	CREDENTIAL REQUEST DESCRIPTOR							
•••								
n								

The CREDENTIAL REQUEST TYPE field (see table 215) specifies type of credential being requested and the format of the CREDENTIAL REQUEST DESCRIPTOR field.

Table 215 — CREDENTIAL REQUEST TYPE field

Code	Description	Reference
0001h	CbCS logical unit	6.19.1.2
0002h	CbCS logical unit and volume	6.19.1.3
all other codes	Reserved	

The CREDENTIAL REQUEST DESCRIPTOR field specifies the information needed to request the credential as described in table 215.

The CONTROL field is defined in SAM-5.

If return of the requested credential is not permitted, the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to ACCESS DENIED - NO ACCESS RIGHTS.

The DS_SAI field in the ENCRYPTED REQUEST DESCRIPTOR field contains the value of the DS_SAI SA parameter (see 5.16.2.2) for the SA to be used to encrypt the unencrypted bytes and the parameter data as described.

If the device server is not maintaining an SA with an AC_SAI SA parameter that matches the AC_SAI field contents and a DS_SAI SA parameter that matches the DS_SAI field contents, 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 device server is maintaining the SA specified by the AC_SAI field and the DS_SAI field, then the SA shall be verified for use by this RECEIVE CREDENTIAL command as follows:

a) the USAGE_TYPE SA parameter (see 5.16.2.2) shall be verified to be equal to 82h (i.e., CbCS authentication and credential encryption; and
b) the USAGE_DATA SA parameter (see 5.16.2.2) shall be verified not to contain an ALGORITHM IDENTIFIER field (see 7.7.3.6) that is set to ENCR_NULL based on the contents the IKEv2-SCSI SAUT Cryptographic Algorithm payload (see 7.7.3.5.14) for the ENCR algorithm type (see 7.7.3.6.2) during creation of the SA (see 5.16.2.3).

If any of these SA verifications fails, 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.

6.19.1.2 CbCS logical unit credential request descriptor

If the credential request type field is set to 0001h (i.e., CbCS logical unit), then the format of the CREDENTIAL REQUEST DESCRIPTOR field is as shown in table 216.

Table 216 — CbCS logical unit credential request descriptor format

Bit / Byte	7	6	5	4	3	2	1	0
0	DESIGNATION DESCRIPTOR							
•••								
19								

The format of the DESIGNATION DESCRIPTOR field is defined in table 531 (see 7.8.6.1). 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 any of the fields in the DESIGNATION DESCRIPTOR field are set as follows:

a) the DESIGNATOR TYPE field contains any value other than 3h (i.e., NAA);
b) the ASSOCIATION field contains any value other than 00b (i.e., logical unit) or 10b (i.e., SCSI target device); or
c) the DESIGNATOR LENGTH field is set to a value that is larger than 16.

6.19.1.3 CbCS logical unit and volume credential request descriptor

If the credential request type field is set to 0002h (i.e., CbCS logical unit and volume), then the format of the CREDENTIAL REQUEST DESCRIPTOR field is as shown in table 217.

Table 217 — CbCS logical unit and volume credential request descriptor format

Bit / Byte	7	6	5	4	3	2	1	0
0	DESIGNATION DESCRIPTOR							
•••								
19								
20	MAM ATTRIBUTE							
•••								
56								

The format of the DESIGNATION DESCRIPTOR field is defined in table 531 (see 7.8.6.1). 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 any of the fields in the DESIGNATION DESCRIPTOR field are set as follows:

a) the DESIGNATOR TYPE field contains any value other than 3h (i.e., NAA);
b) the ASSOCIATION field contains any value other than 00b (i.e., logical unit) or 10b (i.e., SCSI target device); or
c) the DESIGNATOR LENGTH field is set to a value that is larger than 16.

The format of the MAM ATTRIBUTE field is defined in table 387 (see 7.4.1). If the ATTRIBUTE IDENTIFIER field in the MAM ATTRIBUTE field contains any value other than 0401h (i.e., MEDIUM SERIAL NUMBER), 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.

6.19.2 RECEIVE CREDENTIAL parameter data

6.19.2.1 RECEIVE CREDENTIAL parameter data encryption

The RECEIVE CREDENTIAL parameter data shall be one of the ESP-SCSI Data-In Buffer descriptors shown in table 87 (see 5.16.7.5.1). The SA specified by the AC_SAI field and the DS_SAI field in the CDB shall be used to construct the ESP-SCSI Data-In Buffer descriptor as described in 5.16.7.5.

Before processing the parameter data, the application client should validate and decrypt the ESP-SCSI Data-In Buffer descriptor as described in 5.16.7.5. If any errors are detected by the validation and decryption processing, the parameter data should be ignored.

6.19.2.2 RECEIVE CREDENTIAL decrypted parameter data

Before encryption and after decryption, the UNENCRYPTED BYTES field (see 5.16.7.3) that are used to compute the ENCRYPTED OR AUTHENTICATED DATA field (see 5.16.7.5) contents shall contain a CbCS credential (see table 218).

Table 218 — CbCS credential format

Bit / Byte	7	6	5	4	3	2	1	0
0	Reserved				CREDENTIAL FORMAT (1h)			
1	Reserved							
2	(MSB)	CREDENTIAL LENGTH (n-3)						
3								(LSB)
4	(MSB)	CAPABILITY LENGTH (k-5)						
5								(LSB)
6		CbCS capability descriptor (see 6.19.2.3)						
•••								
k								
k+1	(MSB)	CAPABILITY KEY LENGTH (n-k-4))						
•••								
k+4								(LSB)
k+5		CAPABILITY KEY						
•••								
n								

The CREDENTIAL FORMAT field (see table 219) indicates the format of the credential.

Table 219 — Credential format values

Value	Description
0h	Reserved
1h	The format defined by this standard
2h to Fh	Reserved

The CREDENTIAL LENGTH field indicates the number of bytes that follow in the credential including the capability length, the CbCS capability descriptor, the capability key length, and the capability key.

The CAPABILITY LENGTH field indicates the number of bytes that follow in the capability.

The contents of the CbCS capability descriptor are defined in 6.19.2.3.

The CAPABILITY KEY LENGTH field indicates the number of bytes that follow in the capability key.

The CAPABILITY KEY field contains an integrity check value (see 3.1.77) that is computed and used as described in 5.16.6.8.12.

6.19.2.3 CbCS capability descriptor

6.19.2.3.1 Overview

A CbCS capability descriptor (see table 220) specifies the commands that are allowed by the CbCS extension descriptor in which it appears.

Table 220 — CbCS capability descriptor format

Bit / Byte	7	6	5	4	3	2	1	0
0	DESIGNATION TYPE				KEY VERSION			
1	CBCS METHOD							
2	(MSB)	CAPABILITY EXPIRATION TIME						
•••								
7								(LSB)
8	(MSB)	INTEGRITY CHECK VALUE ALGORITHM						
•••								
11								(LSB)
12		PERMISSIONS BIT MASK						
•••								
15								
16	(MSB)	POLICY ACCESS TAG						
•••								
19								(LSB)
20	DESIGNATION DESCRIPTOR							
•••								
57								
58	DISCRIMINATOR							
•••								
71								

The DESIGNATION TYPE field (see table 221) specifies the format of the Designation descriptor.

Table 221 — DESIGNATION TYPE field

Code	Description	DESIGNATION DESCRIPTOR field format reference
0h	Reserved	
1h	Logical unit designation descriptor	6.19.2.3.2
2h	MAM attribute designation descriptor	6.19.2.3.3
3h to Fh	Reserved	

The KEY VERSION field specifies which working key (see 5.16.6.8.11), is being used to compute the capability key (see 5.16.6.8.12) for this CbCS capability.

The CBCS METHOD field (see table 222) specifies the CbCS method used by this CbCS capability.

Table 222 — CBCS METHOD field

Code	CbCS method	Reference
00h	BASIC	5.16.6.8.8.2
01h	CAPKEY	5.16.6.8.8.3
02h to EFh	Reserved	
F0h to FEh	Vendor specific	
FFh	Reserved	

The CAPABILITY EXPIRATION TIME field specifies expiration time of this CbCS capability as the number of milliseconds that have elapsed since midnight, 1 January 1970 UT. If the CAPABILITY EXPIRATION TIME field is set to zero, this CbCS capability does not have an expiration time.

If the CAPABILITY EXPIRATION TIME field is not set to zero, then:

a) the clock maintained by the CbCS management device server (see 5.16.6.8.3) should be synchronized with the clock maintained by the enforcement manager (see 5.16.6.8.7). The method for synchronizing the clocks is outside the scope of this standard, however, the protocol should be implemented in a secure manner (e.g., it should not be possible for an adversary to set the clock in the SCSI device or in the secure CDB processor backwards to enable the reuse of expired CbCS credentials). The value in the enforcement manager's clock is available in the Current CbCS Parameters CbCS page (see 7.7.4.3.5) to assist in this synchronization;
b) the CbCS management device server should set the CAPABILITY EXPIRATION TIME field to a value that is at least an order of magnitude larger than the allowed deviation between the clocks.

The INTEGRITY CHECK VALUE ALGORITHM field specifies the algorithm used to compute the capability key and other integrity check values for this CbCS capability. The value in the INTEGRITY CHECK VALUE ALGORITHM field is selected from the codes that the Unchangeable CbCS Parameters CbCS page (see 7.7.4.3.3) lists as supported integrity check value algorithms.

The PERMISSIONS BIT MASK field (see table 223) specifies the permissions allowed by this CbCS capability. More than one permissions bit may be set. The relationship between commands and bits in the PERMISSIONS BIT MASK

field is defined in for the commands defined by this standard and in the command standard (see 3.1.31) that defines commands for a specific device type.

Table 223 — PERMISSIONS BIT MASK field format

Bit / Byte	7	6	5	4	3	2	1	0
0	DATA READ	DATA WRITE	PARM READ	PARM WRITE	SEC MGMT	RESRV	MGMT	PHY ACC
1	Reserved							
2								
3	Restricted (see applicable command standard)							

A DATA READ bit set to zero indicates a command has no read permission for user data and protection information. A DATA READ bit set to one indicates a command has permission to read user data and protection information.

A DATA WRITE bit set to zero indicates a command has no write permission for user data and protection information. A DATA WRITE bit set to one indicates a command has permission to write user data and protection information.

A parameter data read (PARM READ) bit set to zero indicates a command has no parameter data read permission. A PARM READ bit set to one indicates a command has permission to read parameter data.

A parameter data write (PARM WRITE) bit set to zero indicates a command has no parameter data write permission. A PARM WRITE bit set to one indicates a command has permission to write parameter data.

A security management (SEC MGMT) bit set to zero indicates a command has no security management permission. A SEC MGMT bit set to one indicates a command has security management permission.

A reservation (RESRV) bit set to zero indicates a command has no persistent reservation permission. A RESRV bit set to one indicates a has permission to make or modify persistent reservations.

A management (MGMT) bit set to zero indicates a command has no storage management permission. A MGMT bit set to one indicates a command has storage management permission. Storage management is outside the scope of this standard.

A physical access (PHY ACC) bit set to zero indicates a command has no permission to affect physical access to the logical unit or volume. A PHY ACC bit set to one indicates a command has permission to affect physical access to the logical unit or volume (see SSC-3).

If the POLICY ACCESS TAG field contains a value other than zero, the policy access tag attribute of the logical unit (see 5.16.6.8.15) is compared to the POLICY ACCESS TAG field contents as part of validating the CbCS capability (see 5.16.6.8.13.2). If the POLICY ACCESS TAG field contains zero, then no comparison is made.

The DESIGNATION DESCRIPTOR field is used during the validation of the CbCS capability (see 5.16.6.8.13.2) to ensure that the command is being addressed to the correct logical unit or volume (see SSC-3). The format of the DESIGNATION DESCRIPTOR field is defined by the value in the DESIGNATION TYPE field as described in table 221.

If the CREDENTIAL REQUEST TYPE field in a RECEIVE CREDENTIAL command is set to 0001h (i.e., CbCS logical unit), then the DESIGNATION DESCRIPTOR field shall contain a logical unit designation descriptor that matches the DESIGNATION DESCRIPTOR field (see 6.19.1.2) in the CREDENTIAL REQUEST DESCRIPTOR field in the CDB. If the CREDENTIAL REQUEST TYPE field in a RECEIVE CREDENTIAL command is set to 0002h (i.e., CbCS logical unit and volume), then the DESIGNATION DESCRIPTOR field shall contain a MAM attribute designation descriptor that matches the MAM ATTRIBUTE field (see 6.19.1.3) in the CREDENTIAL REQUEST DESCRIPTOR field in the CDB.

The DISCRIMINATOR field provides uniqueness to the CbCS capability descriptor and may be used to limit the delegation or prevent leakage of the CbCS capability to other application clients. The CbCS management device server (see 5.16.6.8.3) shall not return the same CbCS capability descriptor to two secure CDB originators.

The enforcement manager (see 5.16.6.8.7) shall validate each CbCS capability descriptor it receives as described in 5.16.6.8.13.2.

6.19.2.3.2 Logical unit designation descriptor format

If the DESIGNATION TYPE field is set to 0001h (i.e., logical unit designation descriptor), then the format of the DESIGNATION DESCRIPTOR field is as shown in table 224.

Table 224 — Logical unit designation descriptor format

Bit / Byte	7	6	5	4	3	2	1	0
0	DESIGNATION DESCRIPTOR							
•••								
19								
20	Reserved							
•••								
37								

The format of the DESIGNATION DESCRIPTOR field is defined in table 531 (see 7.8.6.1) with the following additional requirements:

a) the DESIGNATOR TYPE field shall contain 3h (i.e., NAA);
b) the ASSOCIATION field shall 00b (i.e., logical unit); and
c) the DESIGNATOR LENGTH field shall set to a value that is smaller than 17.

6.19.2.3.3 Volume designation descriptors

If the DESIGNATION TYPE field is set to 0002h (i.e., volume unit designation descriptor), then the format of the DESIGNATION DESCRIPTOR field is as shown in table 225.

Table 225 — Volume designation descriptor format

Bit / Byte	7	6	5	4	3	2	1	0
0	MAM ATTRIBUTE							
•••								
36								
37	Reserved							

The format of the MAM ATTRIBUTE field is defined in table 387 (see 7.4.1) with the following additional requirements:

a) the MAM ATTRIBUTE field shall contain 0401h (i.e., MEDIUM SERIAL NUMBER); and
b) the ATTRIBUTE LENGTH field shall contain 0020h.

6.20 RECEIVE DIAGNOSTIC RESULTS command

The RECEIVE DIAGNOSTIC RESULTS command (see table 226) requests that data be sent to the application client Data-In Buffer. The data is either data based on the most recent SEND DIAGNOSTIC command (see 6.32) or is a diagnostic page specified by the PAGE CODE field.

Table 226 — RECEIVE DIAGNOSTIC RESULTS command

Bit / Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (1Ch)							
1	Reserved							PCV
2	PAGE CODE							
3	(MSB)	ALLOCATION LENGTH						
4								(LSB)
5	CONTROL							

The OPERATION CODE field is defined in 4.3.5.1 and shall be set as shown in table 226 for the RECEIVE DIAGNOSTIC RESULTS command.

A page code valid (PCV) bit set to zero specifies that the device server return parameter data based on the most recent SEND DIAGNOSTIC command (e.g., the diagnostic page with the same page code as that specified in the most recent SEND DIAGNOSTIC command). The response to a RECEIVE DIAGNOSTIC RESULTS command with the PCV bit set to zero is vendor specific if:

a) the most recent SEND DIAGNOSTIC command was not a SEND DIAGNOSTIC command defining parameter data to return;
b) a RECEIVE DIAGNOSTIC RESULTS command with a PCV bit set to one has been processed since the last SEND DIAGNOSTIC command was processed; or
c) no SEND DIAGNOSTIC command defining parameter data to return has been processed since power on, hard reset, or logical unit reset.

A page code valid (PCV) bit set to one specifies that the device server return the diagnostic page specified in the PAGE CODE field. Page code values are defined in 7.2 or in another command standard (see 3.1.31).

NOTES
40 Logical units compliant with previous versions of this standard (e.g., SPC-2) may transfer more than one diagnostic page in the parameter data if the PCV bit is set to zero and the previous SEND DIAGNOSTIC command sent more than one diagnostic page in the parameter list.
41 To ensure that the diagnostic command information is not destroyed by a command sent from another I_T nexus, the logical unit should be reserved.
42 Although diagnostic software is generally device-specific, this command and the SEND DIAGNOSTIC command provide a means to isolate the operating system software from the device-specific diagnostic software. The operating system may remain device-independent.

The ALLOCATION LENGTH field is defined in 4.3.5.6.

The CONTROL field is defined in SAM-5.

See 7.2 for RECEIVE DIAGNOSTIC RESULTS diagnostic page format definitions.

6.21 REPORT ALIASES command

The REPORT ALIASES command (see table 227) requests that the device server return the alias list. The alias list is managed using the CHANGE ALIASES command (see 6.2). If the CHANGE ALIASES command is supported, the REPORT ALIASES command shall also be supported.

The REPORT ALIASES command is a service action of the MAINTENANCE IN command. Additional MAINTENANCE IN service actions are defined in SCC-2 and in this standard. The MAINTENANCE IN service actions defined in SCC-2 apply only to logical units that return a device type of 0Ch or the SCCS bit set to one in their standard INQUIRY data (see 6.4.2).

Table 227 — REPORT ALIASES command

Bit / Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (A3h)							
1	Reserved			SERVICE ACTION (0Bh)				
2	Reserved							
•••								
5								
6	(MSB)	ALLOCATION LENGTH						
•••								
9								(LSB)
10	Reserved							
11	CONTROL							

The OPERATION CODE field is defined in 4.3.5.1 and shall be set as shown in table 227 for the REPORT ALIASES command.

The SERVICE ACTION field is defined in 4.3.5.2 and shall be set as shown in table 227 for the REPORT ALIASES command.

The ALLOCATION LENGTH field is defined in 4.3.5.6.

The CONTROL field is defined in SAM-5.

The parameter data returned by a REPORT ALIASES command (see table 228) contains zero or more alias entries.

Table 228 — REPORT ALIASES parameter data

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)							
•••				ADDITIONAL LENGTH (n-3)				
3								(LSB)
4				Reserved				
5				Reserved				
6				NUMBER OF ALIASES (x)				
7								
				Alias entry (or entries)				
8								
•••				Alias entry 0 (see 6.2.2)				
				⋮				
•••				Alias entry x (see 6.2.2)				
n								

The ADDITIONAL LENGTH field indicates the number of bytes in the remaining parameter data. The relationship between the ADDITIONAL LENGTH field and the CDB ALLOCATION LENGTH field is defined in 4.3.5.6.

The NUMBER OF ALIASES field indicates the number of alias entries in the alias list and shall not be changed if the CDB contains an insufficient allocation length.

The parameter data shall include one alias entry for each alias in the alias list. The format of an alias entry is described in 6.2.2.

6.22 REPORT IDENTIFYING INFORMATION command

6.22.1 REPORT IDENTIFYING INFORMATION command overview

The REPORT IDENTIFYING INFORMATION command (see table 229) requests that the device server send identifying information (see 5.17) to the application client. The REPORT IDENTIFYING INFORMATION command is an extension to the REPORT PERIPHERAL DEVICE/COMPONENT DEVICE IDENTIFIER service action of the MAINTENANCE IN command defined in SCC-2. Additional MAINTENANCE IN and MAINTENANCE OUT service actions are defined in SCC-2 and in this standard.

The MAINTENANCE IN service actions defined only in SCC-2 shall apply only to logical units that return a device type of 0Ch (i.e., storage array controller device) or the SCCS bit set to one in their standard INQUIRY data (see 6.4.2). When a logical unit returns a device type of 0Ch or the SCCS bit set to one in its standard INQUIRY data, the implementation requirements for the SCC-2 MAINTENANCE IN service actions shall be as specified in SCC-2.

Otherwise the MAINTENANCE IN service action definitions and implementation requirements stated in this standard shall apply.

The device server shall return the same identifying information regardless of the I_T nexus being used to retrieve the identifying information.

Processing a REPORT IDENTIFYING INFORMATION command may require the enabling of a nonvolatile memory within the logical unit. If the nonvolatile memory is not ready, the command shall be terminated with CHECK CONDITION status, and not wait for the nonvolatile memory to become ready. The sense key shall be set to NOT READY and the additional sense code shall be set as described in table 284 (see 6.37). This information should allow the application client to determine the action required to cause the device server to become ready.

Table 229 — REPORT IDENTIFYING INFORMATION command

Bit / Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (A3h)							
1	Reserved			SERVICE ACTION (05h)				
2	Reserved							
3								
4	Restricted (see SCC-2)							
5								
6	(MSB)	ALLOCATION LENGTH						
•••								
9								(LSB)
10	INFORMATION TYPE							Reserved
11	CONTROL							

The OPERATION CODE field is defined in 4.3.5.1 and shall be set as shown in table 229 for the REPORT IDENTIFYING INFORMATION command.

The SERVICE ACTION field is defined in 4.3.5.2 and shall be set as shown in table 229 for the REPORT IDENTIFYING INFORMATION command.

The ALLOCATION LENGTH field is defined in 4.3.5.6.

The INFORMATION TYPE field (see table 230) specifies the type of information to be reported. If the specified information type is not implemented by the device server, 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.

Table 230 — INFORMATION TYPE field

Code	Description	Reference
0000000b	Peripheral device identifying information (see 5.17).	6.22.2
0000010b	Peripheral device text identifying information (see 5.17).	6.22.2
1111111b	Identifying information supported – The parameter data contains a list of supported identifying information types and the maximum length of each.	6.22.3
xxxxxx1b	Restricted	SCC-2
All other	Reserved	

The CONTROL field is defined in SAM-5.

6.22.2 IDENTIFYING INFORMATION parameter data

The REPORT IDENTIFYING INFORMATION parameter data format used when the INFORMATION TYPE field is set to 0000000b or 0000010b is shown in table 231.

Table 231 — REPORT IDENTIFYING INFORMATION parameter data

Bit / Byte	7	6	5	4	3	2	1	0
0	Reserved							
1								
2	(MSB)	INFORMATION LENGTH (n-3)						
3								(LSB)
4	INFORMATION							
•••								
n								

The INFORMATION LENGTH field indicates the length in bytes of the INFORMATION field. The relationship between the INFORMATION LENGTH field and the CDB ALLOCATION LENGTH field is defined in 4.3.5.6.

The INFORMATION field contains the identifying information that has the specified information type (see 6.22.1).

6.22.3 IDENTIFYING INFORMATION SUPPORTED parameter data

The REPORT IDENTIFYING INFORMATION parameter data format used when the INFORMATION TYPE field is set to 1111111b is shown in table 232.

Table 232 — REPORT IDENTIFYING INFORMATION SUPPORTED parameter data

Bit / Byte	7	6	5	4	3	2	1	0
0	Reserved							
1								
2	(MSB)	IDENTIFYING INFORMATION LENGTH (n-3)						
3								(LSB)
	Identifying information descriptor list							
4	Identifying information descriptor [first] (see table 233)							
•••								
7								
	⋮							
n-3	Identifying information descriptor [last] (see table 233)							
•••								
n								

The IDENTIFYING INFORMATION LENGTH field indicates the length in bytes of the identifying information descriptor list. The relationship between the IDENTIFYING INFORMATION LENGTH field and the CDB ALLOCATION LENGTH field is defined in 4.3.5.6.

The identifying information descriptor list contains an identifying information descriptor (see table 233) for each identifying information type supported by the device server. The identifying information descriptors shall be sorted in increasing order by information type.

Table 233 — Identifying information descriptor

Bit / Byte	7	6	5	4	3	2	1	0
0	INFORMATION TYPE							Reserved
1	Reserved							
2	(MSB)	MAXIMUM INFORMATION LENGTH						
3								(LSB)

The INFORMATION TYPE field indicates the information type (see 5.17).

The MAXIMUM INFORMATION LENGTH field indicates the maximum number of bytes supported for identifying information that has the indicated information type (see 5.17).

6.23 REPORT LUNS command

The REPORT LUNS command (see table 234) requests that the peripheral device logical unit inventory accessible to the I_T nexus be sent to the application client. The logical unit inventory is a list that shall include the logical unit numbers of all logical units having a PERIPHERAL QUALIFIER value of 000b (see 6.4.2). Logical unit numbers for logical units with PERIPHERAL QUALIFIER values other than 000b and 011b may be included in the logical unit inventory. Logical unit numbers for logical units with a PERIPHERAL QUALIFIER value of 011b shall not be included in the logical unit inventory.

Table 234 — REPORT LUNS command

Bit / Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (A0h)							
1	Reserved							
2	SELECT REPORT							
3	Reserved							
•••								
5								
6	(MSB)	ALLOCATION LENGTH						
•••								
9								(LSB)
10	Reserved							
11	CONTROL							

The OPERATION CODE field is defined in 4.3.5.1 and shall be set as shown in table 234 for the REPORT LUNS command.

The SELECT REPORT field (see table 235) specifies the types of logical unit addresses that shall be reported.

Table 235 — SELECT REPORT field

Code	Description
00h	The list shall contain the logical units accessible to the I_T nexus with the following addressing methods (see SAM-4): a) logical unit addressing method, b) peripheral device addressing method; c) flat space addressing method; and d) extended logical unit addressing method. If there are no logical units, the LUN LIST LENGTH field shall be zero.
01h	The list shall contain only well known logical units, if any. If there are no well known logical units, the LUN LIST LENGTH field shall be zero.
02h	The list shall contain all logical units accessible to the I_T nexus.
03h to FFh	Reserved

The ALLOCATION LENGTH field is defined in 4.3.5.6.

The CONTROL field is defined in SAM-5.

The REPORT LUNS command shall return CHECK CONDITION status only when the device server is unable to return the requested report of the logical unit inventory.

If a REPORT LUNS command is received from an I_T nexus with a pending unit attention condition (i.e., before the device server reports CHECK CONDITION status), the device server shall perform the REPORT LUNS command (see SAM-4).

The REPORT LUNS parameter data should be returned even though the device server is not ready for other commands. The report of the logical unit inventory should be available without incurring any media access delays. If the device server is not ready with the logical unit inventory or if the inventory list is null for the requesting I_T nexus and the SELECT REPORT field set to 02h, then the device server shall provide a default logical unit inventory that contains at least LUN 0 or the REPORT LUNS well known logical unit (see 8.2). A non-empty peripheral device logical unit inventory that does not contain either LUN 0 or the REPORT LUNS well known logical unit is valid.

If a REPORT LUNS command is received for a logical unit that the SCSI target device does not support and the device server is not capable of returning the logical unit inventory, 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 LOGICAL UNIT NOT SUPPORTED.

The device server shall report those devices in the logical unit inventory using the format shown in table 236.

Table 236 — REPORT LUNS parameter data format

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)							
•••				LUN LIST LENGTH (n-7)				
3								(LSB)
4								
•••				Reserved				
7								
	LUN list							
8								
•••				LUN [first]				
15								
	⋮							
n-7								
•••				LUN [last]				
n								

The LUN LIST LENGTH field shall contain the length in bytes of the LUN list that is available to be transferred. The LUN list length is the number of logical unit numbers in the logical unit inventory multiplied by eight. The relationship between the LUN LIST LENGTH field and the CDB ALLOCATION LENGTH field is defined in 4.3.5.6.

6.24 REPORT PRIORITY command

The REPORT PRIORITY command (see table 237) requests the priority that has been assigned to one or more I_T nexuses associated with the logical unit (i.e., I_T_L nexuses).

The REPORT PRIORTY command is a service action of the MAINTENANCE IN command. Additional MAINTENANCE IN service actions are defined in SCC-2 and in this standard. The MAINTENANCE IN service actions defined in SCC-2 apply only to logical units that return a device type of 0Ch or the SCCS bit set to one in their standard INQUIRY data (see 6.4.2).

Table 237 — REPORT PRIORITY command

Bit / Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (A3h)							
1	Reserved			SERVICE ACTION (0Eh)				
2	PRIORITY REPORTED		Reserved					
3	Reserved							
•••								
5								
6	(MSB)		ALLOCATION LENGTH					
•••								
9								(LSB)
10	Reserved							
11	CONTROL							

The OPERATION CODE field is defined in 4.3.5.1 and shall be set as shown in table 237 for the REPORT PRIORTY command.

The SERVICE ACTION field is defined in 4.3.5.2 and shall be set as shown in table 237 for the REPORT PRIORTY command.

The PRIORITY REPORTED field (see table 238) specifies the information to be returned in the parameter data.

Table 238 — PRIORITY REPORTED field

Code	Description
00b	Only the priority for the I_T nexus on which the command was received shall be reported in the REPORT PRIORITY parameter data.
01b	The priority for each I_T nexus that is not set to the initial command priority shall be reported in the REPORT PRIORITY parameter data.
10b to 11b	Reserved

The ALLOCATION LENGTH field is defined in 4.3.5.6. The allocation length should be at least four.

The CONTROL field is defined in SAM-5.

The format of the parameter data returned by the REPORT PRIORITY command is shown in table 239.

Table 239 — REPORT PRIORITY parameter data format

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)							
•••				PRIORITY PARAMETER DATA LENGTH (n-3)				
3								(LSB)
	Priority descriptors							
4								
•••				Priority descriptor [first] (see table 240)				
				⋮				
•••				Priority descriptor [last] (see table 240)				
n								

The PRIORITY PARAMETER DATA LENGTH field indicates the number of bytes of parameter data that follow.

Each priority descriptor (see table 240) contains priority information for a single I_T_L nexus.

Table 240 — Priority descriptor format

Bit / Byte	7	6	5	4	3	2	1	0
0				Reserved	CURRENT PRIORITY			
1				Reserved				
2	(MSB)							
3				RELATIVE TARGET PORT IDENTIFIER				(LSB)
4				Reserved				
5				Reserved				
6	(MSB)							
7				ADDITIONAL DESCRIPTOR LENGTH (n-7)				(LSB)
8								
•••				TRANSPORTID				
n								

The CURRENT PRIORITY field contains the priority assigned to the I_T_L nexus represented by this descriptor. If the PRIORITY REPORTED field in this command is set to 00b and the priority for the I_T_L nexus associated with this command is set to the initial command priority, then the CURRENT PRIORITY field shall be set to zero. The priority assigned to an I_T_L nexus may be used as a command priority for commands received via that I_T_L nexus (see SAM-4).

The RELATIVE TARGET PORT IDENTIFIER field contains the relative port identifier (see 3.1.127) of the target port that is part of the I_T_L nexus to which the current priority applies.

The ADDITIONAL DESCRIPTOR LENGTH field indicates the number of bytes that follow in the descriptor (i.e., the size of the TransportID).

The TRANSPORTID field contains a TransportID (see 7.6.4) identifying the initiator port that is part of the I_T_L nexus to which the current priority applies.

6.25 REPORT SUPPORTED OPERATION CODES command

6.25.1 REPORT SUPPORTED OPERATION CODES command introduction

The REPORT SUPPORTED OPERATION CODES command (see table 241) requests information on commands the addressed logical unit supports. An application client may request a list of all operation codes and service actions supported by the logical unit or the command support data for a specific command.

The REPORT SUPPORTED OPERATION CODES command is a service action of the MAINTENANCE IN command. Additional MAINTENANCE IN service actions are defined in SCC-2 and in this standard. The MAINTENANCE IN service actions defined in SCC-2 apply only to logical units that return a device type of 0Ch or the SCCS bit set to one in their standard INQUIRY data (see 6.4.2).

Table 241 — REPORT SUPPORTED OPERATION CODES command

Bit / Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (A3h)							
1	Reserved			SERVICE ACTION (0Ch)				
2	RCTD	Reserved				REPORTING OPTIONS		
3	REQUESTED OPERATION CODE							
4	(MSB)	REQUESTED SERVICE ACTION						
5								(LSB)
6	(MSB)	ALLOCATION LENGTH						
•••								
9								(LSB)
10	Reserved							
11	CONTROL							

The OPERATION CODE field is defined in 4.3.5.1 and shall be set as shown in table 241 for the REPORT SUPPORTED OPERATION CODES command.

The SERVICE ACTION field is defined in 4.3.5.2 and shall be set as shown in table 241 for the REPORT SUPPORTED OPERATION CODES command.

A return command timeouts descriptor (RCTD) bit set to one specifies that the command timeouts descriptor (see 6.25.4) shall be included in each command descriptor (see 6.25.2) that is returned or in the one_command parameter data (see 6.25.3) that is returned. A RCTD bit set to zero specifies that the command timeouts descriptor shall not be returned.

The REPORTING OPTIONS field (see table 242) specifies the information to be returned in the parameter data.

Table 242 — REPORT SUPPORTED OPERATION CODES REPORTING OPTIONS field

Code	Description	Parameter Data Reference
000b	A list of all operation codes and service actions supported by the logical unit shall be returned in the all_commands parameter data format. The REQUESTED OPERATION CODE CDB field and REQUESTED SERVICE ACTION CDB field shall be ignored.	6.25.2
001b	The command support data for the operation code specified in the REQUESTED OPERATION CODE field shall be returned in the one_command parameter data format. The REQUESTED SERVICE ACTION CDB field shall be ignored. If the REQUESTED OPERATION CODE field specifies an operation code that has service actions, 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.	6.25.3
010b	The command support data for the operation code and service action specified in the REQUESTED OPERATION CODE CDB field and REQUESTED SERVICE ACTION CDB field shall be returned in the one_command parameter data format. If the REQUESTED OPERATION CODE CDB field specifies an operation code that does not have service actions, 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.	6.25.3
011b to 111b	Reserved	

The REQUESTED OPERATION CODE field specifies the operation code of the command to be returned in the one_command parameter data format (see 6.25.3).

The REQUESTED SERVICE ACTION field specifies the service action of the command to be returned in the one_command parameter data format.

The ALLOCATION LENGTH field is defined in 4.3.5.6.

The CONTROL field is defined in SAM-5.

6.25.2 All_commands parameter data format

The REPORT SUPPORTED OPERATION CODES all_commands parameter data format (see table 243) begins with a four-byte header that contains the length in bytes of the parameter data followed by a list of supported commands. Each command descriptor contains information about a single supported command CDB (i.e., one operation code and service action combination, or one non-service-action operation code). The list of command descriptors shall contain all commands supported by the logical unit.

Table 243 — All_commands parameter data

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)							
•••				COMMAND DATA LENGTH (n-3)				
3								(LSB)
	Command descriptors							
4								
•••				Command descriptor 0 (see table 244)				
				⋮				
•••				Command descriptor x (see table 244)				
n								

The COMMAND DATA LENGTH field indicates the length in bytes of the command descriptor list.

Each command descriptor (see table 244) contains information about a single supported command CDB.

Table 244 — Command descriptor format

Bit / Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE							
1	Reserved							
2	(MSB)			SERVICE ACTION				
3								(LSB)
4	Reserved							
5	Reserved						CTDP	SERVACTV
6	(MSB)			CDB LENGTH				
7								(LSB)
8								
•••				Command timeouts descriptor, if any (see 6.25.4)				
19								

The OPERATION CODE field contains the operation code of a command supported by the logical unit.

The SERVICE ACTION field contains a supported service action of the supported operation code indicated by the OPERATION CODE field. If the operation code indicated in the OPERATION CODE field does not have any service actions, the SERVICE ACTION field shall be set to 00h.

A command timeouts descriptor present (CTDP) bit set to one indicates that the command timeouts descriptor (see 6.25.4) is included in this command descriptor. A CTDP bit set to zero indicates that the command timeouts descriptor is not included in this command descriptor.

A service action valid (SERVACTV) bit set to zero indicates the operation code indicated by the OPERATION CODE field does not have service actions and the SERVICE ACTION field contents are reserved. A SERVACTV bit set to one indicates the operation code indicated by the OPERATION CODE field has service actions and the contents of the SERVICE ACTION field are valid.

The CDB LENGTH field contains the length of the command CDB in bytes for the operation code indicated in the OPERATION CODE field, and if the SERVACTV bit is set to the service action indicated by the SERVICE ACTION field.

If the RCTD bit is set to one in the REPORT SUPPORTED OPERATION CODES CDB (see 6.25.1), the command timeouts descriptor (see table 247 in 6.25.4) shall be included. If the RCTD bit is set to zero, the command timeouts descriptor shall not be included.

6.25.3 One_command parameter data format

The REPORT SUPPORTED OPERATION CODES one_command parameter data format (see table 245) contains information about the CDB and a usage map for bits in the CDB for the command specified by the REPORTING OPTIONS, REQUESTED OPERATION CODE, and REQUESTED SERVICE ACTION fields in the REPORT SUPPORTED OPERATION CODES CDB.

Table 245 — One_command parameter data

Bit / Byte	7	6	5	4	3	2	1	0
0	Reserved							
1	CTDP	Reserved				SUPPORT		
2	(MSB)	CDB SIZE (n-3)						
3								(LSB)
4		CDB USAGE DATA						
•••								
n								
n+1		Command timeouts descriptor, if any (see 6.25.4)						
•••								
n+12								

A command timeouts descriptor present (CTDP) bit set to one indicates that the command timeouts descriptor (see 6.25.4) is included in the parameter data. A CTDP bit set to zero indicates that the command timeouts descriptor is not included in the parameter data.

The SUPPORT field is defined in table 246.

Table 246 — SUPPORT values

Support	Description
000b	Data about the requested SCSI command is not currently available. All data after byte 1 is not valid. A subsequent request for command support data may be successful.
001b	The device server does not support the requested command. All data after byte 1 is undefined.
010b	Reserved
011b	The device server supports the requested command in conformance with a SCSI standard. The parameter data format conforms to the definition in table 245.
100b	Reserved
101b	The device server supports the requested command in a vendor specific manner. The parameter data format conforms to the definition in table 245.
110b to 111b	Reserved

The CDB SIZE field contains the size of the CDB USAGE DATA field in the parameter data, and the number of bytes in the CDB for command being queried (i.e., the command specified by the REPORTING OPTIONS, REQUESTED OPERATION CODE, and REQUESTED SERVICE ACTION fields in the REPORT SUPPORTED OPERATION CODES CDB).

The CDB USAGE DATA field contains information about the CDB for the command being queried. The first byte of the CDB USAGE DATA field shall contain the operation code for the command being queried. If the command being queried contains a service action, then that service action code shall be placed in the CDB USAGE DATA field in the same location as the SERVICE ACTION field of the command CDB. All other bytes of the CDB USAGE DATA field shall contain a usage map for bits in the CDB for the command being queried.

The bits in the usage map shall have a one-for-one correspondence to the CDB for the command being queried. If the device server evaluates a bit in the CDB for the command being queried, the usage map shall contain a one in the corresponding bit position. If any bit representing part of a field is returned as one, all bits for the field shall be returned as one. If the device server ignores or treats as reserved a bit in the CDB for the command being queried, the usage map shall contain a zero in the corresponding bit position. The usage map bits for a given CDB field all shall have the same value.

For example, the CDB usage bit map for the REPORT SUPPORTED OPERATION CODES command is: A3h, 0Ch, 87h, FFh, FFh, FFh, FFh, FFh, FFh, FFh, 00h, 07h. This example assumes that the logical unit only supports the low-order three bits of the CDB CONTROL byte. The first byte contains the operation code, and the second byte contains three reserved bits and the service action. The remaining bytes contain the usage map.

If the RCTD bit is set to one in the REPORT SUPPORTED OPERATION CODES CDB (see 6.25.1), the command timeouts descriptor (see table 247 in 6.25.4) shall be included. If the RCTD bit is set to zero, the command timeouts descriptor shall not be included.

6.25.4 Command timeouts descriptor

6.25.4.1 Overview

The command timeouts descriptor (see table 247) returns timeout information for commands supported by the logical unit based on the time from the start of processing for the command to its reported completion.

Values returned in the command timeouts descriptor do not include times that are outside the control of the device server (e.g., prior commands with the IMMED bit set to one in the CDB, concurrent commands from the same or different I_T nexuses, manual unloads, power on self tests, prior aborted commands, commands that force cache synchronization, delays in the service delivery subsystem).

For commands that cause a change in power condition (see 5.12), values returned in the command timeouts descriptor do not include the power condition transition time (e.g., the time to spinup rotating media).

Values returned in the command timeouts descriptor should not be used to compare products.

Table 247 — Command timeouts descriptor format

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)			DESCRIPTOR LENGTH (0Ah)				
1								(LSB)
2	Reserved							
3	COMMAND SPECIFIC							
4	(MSB)			NOMINAL COMMAND PROCESSING TIMEOUT				
•••								
7								(LSB)
8	(MSB)			RECOMMENDED COMMAND TIMEOUT				
•••								
11								(LSB)

The DESCRIPTOR LENGTH field indicates the number of bytes that follow in the command timeouts descriptor.

The COMMAND SPECIFIC field contains timeout information (see table 248) that is specific to one or more commands. If no command specific timeout information is defined by this or the applicable command standard (see 3.1.31) the COMMAND SPECIFIC field is reserved.

Table 248 — Command timeouts descriptor COMMAND SPECIFIC field usage in this standard

Command	Reference
WRITE BUFFER	6.25.4.2

A non-zero value in the NOMINAL COMMAND PROCESSING TIMEOUT field indicates the minimum amount of time in seconds the application client should wait prior to querying for the progress of the command identified by the parameter data that contains this command timeouts descriptor. A value of zero in the NOMINAL COMMAND PROCESSING TIMEOUT field indicates that no timeout is indicated.

NOTE 43 - The value contained in the NOMINAL COMMAND PROCESSING TIMEOUT field may include time required for typical device error recovery procedures expected to occur on a regular basis.

A non-zero value in the RECOMMENDED COMMAND TIMEOUT field specifies the recommended time in seconds the application client should wait prior to timing out the command identified by the parameter data that contains this command timeouts descriptor. A value of zero in the RECOMMENDED COMMAND TIMEOUT field indicates that no time is indicated.

The device server should set the recommended command timeout to a value greater than or equal to the nominal command processing timeout.

6.25.4.2 WRITE BUFFER command timeouts descriptor COMMAND SPECIFIC field usage

For the WRITE BUFFER command (see 6.39), the COMMAND SPECIFIC field usage is reserved for all modes except the following:

a) download microcode mode (04h);
b) download microcode and save mode (05h);
c) download microcode with offsets mode (06h);
d) download microcode with offsets and save mode (07h);
e) download microcode with offsets and defer activation mode (0Eh) only if the microcode is activated by an event other than an activate deferred microcode mode; and
f) activate deferred microcode mode (0Fh).

If the command timeouts descriptor describes one of the WRITE BUFFER modes listed in this subclause, then the COMMAND SPECIFIC field indicates the maximum time, in one second increments, that access to the SCSI device is limited or not possible through any SCSI ports associated with a logical unit that processes a WRITE BUFFER command that specifies one of the named modes. A value of zero in the COMMAND SPECIFIC field indicates that the no maximum time is indicated.

6.26 REPORT SUPPORTED TASK MANAGEMENT FUNCTIONS command

The REPORT SUPPORTED TASK MANAGEMENT FUNCTIONS command (see table 249) requests information on task management functions (see SAM-4) the addressed logical unit supports.

The REPORT SUPPORTED TASK MANAGEMENT FUNCTIONS command is a service action of the MAINTENANCE IN command. Additional MAINTENANCE IN service actions are defined in SCC-2 and in this standard. The MAINTENANCE IN service actions defined in SCC-2 apply only to logical units that return a device type of 0Ch or the SCCS bit set to one in their standard INQUIRY data (see 6.4.2).

Table 249 — REPORT SUPPORTED TASK MANAGEMENT FUNCTIONS command

Bit / Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (A3h)							
1	Reserved			SERVICE ACTION (0Dh)				
2	REPD	Reserved						
3	Reserved							
•••								
5								
6	(MSB)	ALLOCATION LENGTH						
•••								
9								(LSB)
10	Reserved							
11	CONTROL							

The OPERATION CODE field is defined in 4.3.5.1 and shall be set as shown in table 249 for the REPORT SUPPORTED TASK MANAGEMENT FUNCTIONS command.

The SERVICE ACTION field is defined in 4.3.5.2 and shall be set as shown in table 249 for the REPORT SUPPORTED TASK MANAGEMENT FUNCTIONS command.

A return extended parameter data (REPD) bit set to one specifies that the task management timeout information shall be included in the parameter data that is returned. A REPD bit set to zero specifies that the task management timeout information shall not be returned.

The ALLOCATION LENGTH field specifies the number of bytes that have been allocated for the returned parameter data. The allocation length should be at least four. If the allocation length is less than 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.

The CONTROL field is defined in SAM-5.

The format of the parameter data returned by the REPORT SUPPORTED TASK MANAGEMENT FUNCTIONS command depends on the value of the REPD bit as follows:

a) if the REPD bit is set to zero, the REPORT SUPPORTED TASK MANAGEMENT FUNCTION parameter data returned is shown in table 250; and
b) if the REPD bit is set to one, the REPORT SUPPORTED TASK MANAGEMENT FUNCTION parameter data returned is shown in table 251.

The REPORT SUPPORTED TASK MANAGEMENT FUNCTIONS basic parameter data format is shown in table 250.

Table 250 — REPORT SUPPORTED TASK MANAGEMENT FUNCTIONS basic parameter data

Bit / Byte	7	6	5	4	3	2	1	0
0	ATS	ATSS	CACAS	CTSS	LURS	QTS	Obsolete	Obsolete
1	Reserved					QAES	QTSS	ITNRS
2	Reserved							
3	REPORT SUPPORTED TASK MANAGEMENT FUNCTIONS ADDITIONAL DATA LENGTH (00h)							

The REPORT SUPPORTED TASK MANAGEMENT FUNCTIONS extended parameter data format is shown in table 251.

Table 251 — REPORT SUPPORTED TASK MANAGEMENT FUNCTIONS extended parameter data

Bit / Byte	7	6	5	4	3	2	1	0
0	ATS	ATSS	CACAS	CTSS	LURS	QTS	Obsolete	Obsolete
1	Reserved					QAES	QTSS	ITNRS
2	Reserved							
3	REPORT SUPPORTED TASK MANAGEMENT FUNCTIONS ADDITIONAL DATA LENGTH (0Ch)							
4	Reserved							TMFTMOV
5	Reserved							
6	ATTS	ATSTS	CACATS	CTSTS	LURTS	QTTS	Reserved	Reserved
7	Reserved					QAETS	QTSTS	ITNRTS
8	(MSB)	TASK MANAGEMENT FUNCTIONS LONG TIMEOUT						
•••								
11								(LSB)
12	(MSB)	TASK MANAGEMENT FUNCTIONS SHORT TIMEOUT						
•••								
15								(LSB)

An ABORT TASK supported (ATS) bit set to one indicates the ABORT TASK task management function (see SAM-4) is supported by the logical unit. An ATS bit set to zero indicates the ABORT TASK task management function is not supported.

An ABORT TASK SET supported (ATSS) bit set to one indicates the ABORT TASK SET task management function (see SAM-4) is supported by the logical unit. An ATSS bit set to zero indicates the ABORT TASK SET task management function is not supported.

A CLEAR ACA supported (CACAS) bit set to one indicates the CLEAR ACA task management function (see SAM-4) is supported by the logical unit. A CACAS bit set to zero indicates the CLEAR ACA task management function is not supported.

A CLEAR TASK SET supported (CTSS) bit set to one indicates the CLEAR TASK SET task management function (see SAM-4) is supported by the logical unit. A CTSS bit set to zero indicates the CLEAR TASK SET task management function is not supported.

A LOGICAL UNIT RESET supported (LURS) bit set to one indicates the LOGICAL UNIT RESET task management function (see SAM-4) is supported by the logical unit. An LURS bit set to zero indicates the LOGICAL UNIT RESET task management function is not supported.

A QUERY TASK supported (QTS) bit set to one indicates the QUERY TASK task management function (see SAM-4) is supported by the logical unit. A QTS bit set to zero indicates the QUERY TASK task management function is not supported.

A QUERY ASYNCHRONOUS EVENT supported (QAES) bit set to one indicates the QUERY ASYNCHRONOUS EVENT task management function (see SAM-4) is supported by the logical unit. A QAES bit set to zero indicates the QUERY ASYNCHRONOUS EVENT task management function is not supported.

A QUERY TASK SET supported (QTSS) bit set to one indicates the QUERY TASK SET task management function (see SAM-4) is supported by the logical unit. A QTSS bit set to zero indicates the QUERY TASK SET task management function is not supported.

An I_T NEXUS RESET supported (ITNRS) bit set to one indicates the I_T NEXUS RESET task management function (see SAM-4) is supported by the logical unit. An ITNRS bit set to zero indicates the I_T NEXUS RESET task management function is not supported.

The REPORT SUPPORTED TASK MANAGEMENT FUNCTIONS ADDITIONAL DATA LENGTH field indicates the number of bytes that follow in the parameter data. The relationship between the REPORT SUPPORTED TASK MANAGEMENT FUNCTIONS ADDITIONAL DATA LENGTH field and the CDB allocation length field is defined in 4.3.5.6.

A task management function timeouts valid (TMFTMOV) bit set to one indicates the contents of the TASK MANAGEMENT FUNCTIONS SHORT TIMEOUT field and TASK MANAGEMENT FUNCTIONS LONG TIMEOUT field are valid. A TMFTMOV bit set to zero indicates the contents of the TASK MANAGEMENT FUNCTIONS SHORT TIMEOUT field and TASK MANAGEMENT FUNCTIONS LONG TIMEOUT field are not valid and should be ignored.

An ABORT TASK timeout selector (ATTS) bit set to one indicates that the value in the TASK MANAGEMENT FUNCTIONS SHORT TIMEOUT field applies to the ABORT TASK task management function. An ATTS bit set to zero indicates that the value in the TASK MANAGEMENT FUNCTIONS LONG TIMEOUT field applies to the ABORT TASK task management function.

An ABORT TASK SET timeout selector (ATSTS) bit set to one indicates that the value in the TASK MANAGEMENT FUNCTIONS SHORT TIMEOUT field applies to the ABORT TASK SET task management function. An ATSTS bit set to zero indicates that the value in the TASK MANAGEMENT FUNCTIONS LONG TIMEOUT field applies to the ABORT TASK SET task management function.

A CLEAR ACA timeout selector (CACATS) bit set to one indicates that the value in the TASK MANAGEMENT FUNCTIONS SHORT TIMEOUT field applies to the CLEAR ACA task management function. A CACATS bit set to zero indicates that the value in the TASK MANAGEMENT FUNCTIONS LONG TIMEOUT field applies to the CLEAR ACA task management function.

A CLEAR TASK SET timeout selector (CTSTS) bit set to one indicates that the value in the TASK MANAGEMENT FUNCTIONS SHORT TIMEOUT field applies to the CLEAR TASK SET task management function. A CTSTS bit set to zero indicates that the value in the TASK MANAGEMENT FUNCTIONS LONG TIMEOUT field applies to the CLEAR TASK SET task management function.

A LOGICAL UNIT RESET timeout selector (LURTS) bit set to one indicates that the value in the TASK MANAGEMENT FUNCTIONS SHORT TIMEOUT field applies to the LOGICAL UNIT RESET task management function. A LURTS bit set to zero indicates that the value in the TASK MANAGEMENT FUNCTIONS LONG TIMEOUT field applies to the LOGICAL UNIT RESET task management function.

A QUERY TASK timeout selector (QTTS) bit set to one indicates that the value in the TASK MANAGEMENT FUNCTIONS SHORT TIMEOUT field applies to the QUERY TASK task management function. A QTTS bit set to zero indicates that the value in the TASK MANAGEMENT FUNCTIONS LONG TIMEOUT field applies to the QUERY TASK task management function.

A QUERY ASYNCHRONOUS EVENT timeout selector (QAETS) bit set to one indicates that the value in the TASK MANAGEMENT FUNCTIONS SHORT TIMEOUT field applies to the QUERY ASYNCHRONOUS EVENT task management function. A QAETS bit set to zero indicates that the value in the TASK MANAGEMENT FUNCTIONS LONG TIMEOUT field applies to the QUERY ASYNCHRONOUS EVENT task management function.

A QUERY TASK SET timeout selector (QTSTS) bit set to one indicates that the value in the TASK MANAGEMENT FUNCTIONS SHORT TIMEOUT field applies to the QUERY TASK SET task management function. A QTSTS bit set to

zero indicates that the value in the TASK MANAGEMENT FUNCTIONS LONG TIMEOUT field applies to the QUERY TASK SET task management function.

An I_T NEXUS RESET timeout selector (ITNRTS) bit set to one indicates that the value in the TASK MANAGEMENT FUNCTIONS SHORT TIMEOUT field applies to the I_T NEXUS RESET task management function. An ITNRTS bit set to zero indicates that the value in the TASK MANAGEMENT FUNCTIONS LONG TIMEOUT field applies to the I_T NEXUS RESET task management function.

If the TMFTMOV bit is set to one and the TASK MANAGEMENT FUNCTIONS LONG TIMEOUT field is not set to zero, then the contents of the TASK MANAGEMENT FUNCTIONS LONG TIMEOUT field indicate the recommended time in 100 millisecond increments that the application client should wait prior to timing out a task management function for which the applicable selector bit is set to zero. If the TMFTMOV bit is set to zero or the TASK MANAGEMENT FUNCTIONS LONG TIMEOUT field is set to zero, then the recommended timeout is unspecified for any task management function for which the applicable selector bit is set to zero.

If the TMFTMOV bit is set to one and the TASK MANAGEMENT FUNCTIONS SHORT TIMEOUT field is not set to zero, then the contents of the TASK MANAGEMENT FUNCTIONS SHORT TIMEOUT field indicate the recommended time in 100 millisecond increments that the application client should wait prior to timing out a task management function for which the applicable selector bit is set to one. If the TMFTMOV bit is set to zero or the TASK MANAGEMENT FUNCTIONS SHORT TIMEOUT field is set to zero, then the recommended timeout is unspecified for any task management function for which the applicable selector bit is set to one.

6.27 REPORT TARGET PORT GROUPS command

The REPORT TARGET PORT GROUPS command (see table 252) requests that the device server send target port group information to the application client. This command shall be supported by logical units that report in the standard INQUIRY data (see 6.4.2) that they support asymmetric logical unit access (i.e., return a non-zero value in the TPGS field).

The REPORT TARGET PORT GROUPS command is a service action of the MAINTENANCE IN command. Additional MAINTENANCE IN service actions are defined in SCC-2 and in this standard. The MAINTENANCE IN service actions defined only in SCC-2 apply only to logical units that return a device type of 0Ch or the SCCS bit set to one in their standard INQUIRY data.

Table 252 — REPORT TARGET PORT GROUPS command

Bit / Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (A3h)							
1	PARAMETER DATA FORMAT			SERVICE ACTION (0Ah)				
2	Reserved							
•••								
5								
6	(MSB)	ALLOCATION LENGTH						
•••								
9								(LSB)
10	Reserved							
11	CONTROL							

The OPERATION CODE field is defined in 4.3.5.1 and shall be set as shown in table 252 for the REPORT TARGET PORT GROUPS command.

The SERVICE ACTION field is defined in 4.3.5.2 and shall be set as shown in table 252 for the REPORT TARGET PORT GROUPS command.

The PARAMETER DATA FORMAT field (see table 253) specifies the requested format for the parameter data returned by the REPORT TARGET PORT GROUPS command. Device servers that do not support the PARAMETER DATA FORMAT field shall return the length only header parameter data format (see table 254).

Table 253 — PARAMETER DATA FORMAT field

Code	Description	Reference
000b	Length only header parameter data format	table 254
001b	Extended header parameter data format	table 255
010b to 111b	Reserved	

The ALLOCATION LENGTH field is defined in 4.3.5.6.

The CONTROL field is defined in SAM-5.

Returning REPORT TARGET PORT GROUPS parameter data may require the enabling of a nonvolatile memory. If the nonvolatile memory is not ready, the command shall be terminated with CHECK CONDITION status, rather than wait for the nonvolatile memory to become ready. The sense key shall be set to NOT READY and the additional sense code shall be set as described in table 284 (see 6.37).

The length only header format for the parameter data returned by the REPORT TARGET PORT GROUPS command is shown in table 254.

Table 254 — Length only header parameter data format

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)							
•••				RETURN DATA LENGTH (n-3)				
3								(LSB)
	Target port group descriptor(s)							
4								
•••				Target port group descriptor [first] (see table 256)				
				⋮				
•••				Target port group descriptor [last] (see table 256)				
n								

The RETURN DATA LENGTH field indicates the length in bytes of the list of target port groups. The relationship between the RETURN DATA LENGTH field and the CDB ALLOCATION LENGTH field is defined in 4.3.5.6.

There shall be one target port group descriptor (see table 256) for each target port group.

The extended header format for the parameter data returned by the REPORT TARGET PORT GROUPS command is shown in table 255.

Table 255 — Extended header parameter data format

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)							
•••		RETURN DATA LENGTH (n-3)						
3								(LSB)
4	Reserved	FORMAT TYPE (001b)			Reserved			
5	IMPLICIT TRANSITION TIME							
6		Reserved						
7								
	Target port group descriptor(s)							
8		Target port group descriptor [first] (see table 256)						
•••								
	⋮							
		Target port group descriptor [last] (see table 256)						
•••								
n								

The RETURN DATA LENGTH field indicates the length in bytes of the list of target port groups. The relationship between the RETURN DATA LENGTH field and the CDB ALLOCATION LENGTH field is defined in 4.3.5.6.

The FORMAT TYPE field indicates the returned parameter data format and shall be set to 001b to indicate the extended header parameter data format.

The IMPLICIT TRANSITION TIME field indicates the minimum amount of time in seconds the application client should wait prior to timing out an implicit state transition (see 5.11.2.2). A value of zero indicates that the implicit transition time is not specified.

There shall be one target port group descriptor (see table 256) for each target port group.

Table 256 — Target port group descriptor format

Bit / Byte	7	6	5	4	3	2	1	0
0	PREF	0b	0b	0b	ASYMMETRIC ACCESS STATE			
1	T_SUP	O_SUP	Reserved	LBD_SUP	U_SUP	S_SUP	AN_SUP	AO_SUP
2	(MSB)	TARGET PORT GROUP						
3								(LSB)
4	Reserved							
5	STATUS CODE							
6	Vendor specific							
7	TARGET PORT COUNT							
	Target port descriptor(s)							
8 ••• 11	Target port descriptor [first] (see table 259)							
	⋮							
n-3 ••• n	Target port descriptor [last] (see table 259)							

A preferred target port (PREF) bit set to one indicates that the primary target port group is a preferred primary target port group for accessing the addressed logical unit (see 5.11.2.6). A PREF bit set to zero indicates the primary target port group is not a preferred primary target port group.

The ASYMMETRIC ACCESS STATE field (see table 257) contains the target port group's current target port asymmetric access state.

Table 257 — ASYMMETRIC ACCESS STATE field

Code	State (see 5.11.2.4)	Type (see 5.11.2.1)
0h	Active/optimized	Primary
1h	Active/non-optimized	Primary
2h	Standby	Primary
3h	Unavailable	Primary
4h	Logical block dependent	Primary
5h to Dh	Reserved	
Eh	Offline	Secondary
Fh	Transitioning between states	Primary

If any of the T_SUP bit, O_SUP bit, LBD_SUP bit, U_SUP bit, S_SUP bit, AN_SUP bit, or AO_SUP bit are set to one, then the T_SUP bit, O_SUP bit, LBD_SUP bit, U_SUP bit, S_SUP bit, AN_SUP bit, and AO_SUP bit are as defined in this

standard. If the T_SUP bit, O_SUP bit, U_SUP bit, S_SUP bit, AN_SUP bit, and AO_SUP bit are all set to zero, then which target port asymmetric access states are supported is vendor specific.

A transitioning supported (T_SUP) bit set to one indicates that the device server supports returning the ASYMMETRIC ACCESS STATE field set to Fh (i.e., transitioning between states). A T_SUP bit set to zero indicates that the device server does not return an ASYMMETRIC ACCESS STATE field set to Fh.

An offline supported (O_SUP) bit set to one indicates that the offline secondary target port asymmetric access state is supported. A O_SUP bit set to zero indicates that the offline secondary target port asymmetric access state is not supported.

A logical block dependent (LBD_SUP) bit set to one indicates that the logical block dependent primary target port asymmetric access state is supported. An LBD_SUP bit set to zero indicates that the logical block dependent primary target port asymmetric access state is not supported.

An unavailable supported (U_SUP) bit set to one indicates that the unavailable primary target port asymmetric access state is supported. A U_SUP bit set to zero indicates that the unavailable primary target port asymmetric access state is not supported.

A standby supported (S_SUP) bit set to one indicates that the standby primary target port asymmetric access state is supported. An S_SUP bit set to zero indicates that the standby primary target port asymmetric access state is not supported.

An active/non-optimized supported (AN_SUP) bit set to one indicates that the active/non-optimized primary target port asymmetric access state is supported. An AN_SUP bit set to zero indicates that the active/non-optimized primary target port asymmetric access state is not supported.

An active/optimized supported (AO_SUP) bit set to one indicates that the active/optimized primary target port asymmetric access state is supported. An AO_SUP bit set to zero indicates that the active/optimized primary target port asymmetric access state is not supported.

The TARGET PORT GROUP field contains an identification of the target port group described by this target port group descriptor. Target port group information is also returned in the Device Identification VPD page (see 7.8.6).

The STATUS CODE field (see table 258) indicates why a target port group may be in a specific target port asymmetric access state. It provides a mechanism to indicate error conditions.

Table 258 — STATUS CODE field

Code	Description
00h	No status available.
01h	The target port asymmetric access state altered by SET TARGET PORT GROUPS command.
02h	The target port asymmetric access state altered by implicit asymmetrical logical unit access behavior.
03h to FFh	Reserved

The TARGET PORT COUNT field indicates the number of target ports that are in that target port group and the number of target port descriptors in the target port group descriptor. Every target port group shall contain at least one target port. The target port group descriptor shall include one target port descriptor for each target port in the target port group.

Table 259 — Target port descriptor format

Bit / Byte	7	6	5	4	3	2	1	0
0	Reserved							
1								
2	(MSB)	RELATIVE TARGET PORT IDENTIFIER						
3								(LSB)

The RELATIVE TARGET PORT IDENTIFIER field contains a relative port identifier (see 3.1.127) of a target port in the target port group.

6.28 REPORT TIMESTAMP command

The REPORT TIMESTAMP command (see table 260) requests that the device server return the value of the logical unit's timestamp.

The REPORT TIMESTAMP command is a service action of the MAINTENANCE IN command. Additional MAINTENANCE IN service actions are defined in SCC-2 and in this standard. The MAINTENANCE IN service actions defined only in SCC-2 apply only to logical units that return a device type of 0Ch or the SCCS bit set to one in their standard INQUIRY data (see 6.4.2).

Table 260 — REPORT TIMESTAMP command

Bit / Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (A3h)							
1	Reserved			SERVICE ACTION (0Fh)				
2	Reserved							
•••								
5								
6	(MSB)	ALLOCATION LENGTH						
•••								
9								(LSB)
10	Reserved							
11	CONTROL							

The OPERATION CODE field is defined in 4.3.5.1 and shall be set as shown in table 260 for the REPORT TIMESTAMP command.

The SERVICE ACTION field is defined in 4.3.5.2 and shall be set as shown in table 260 for the REPORT TIMESTAMP command.

The ALLOCATION LENGTH field is defined in 4.3.5.6.

The CONTROL field is defined in SAM-5.

The format for the parameter data returned by the REPORT TIMESTAMP command is shown in table 261.

Table 261 — REPORT TIMESTAMP parameter data format

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)			TIMESTAMP PARAMETER DATA LENGTH (000Ah)				
1								(LSB)
2				Reserved		TIMESTAMP ORIGIN		
3				Reserved				
4	(MSB)			TIMESTAMP				
•••								
9								(LSB)
10				Reserved				
11				Reserved				

The TIMESTAMP PARAMETER DATA LENGTH field indicates the number of bytes of parameter data that follow. The relationship between the TIMESTAMP PARAMETER DATA LENGTH field and the CDB ALLOCATION LENGTH field is defined in 4.3.5.6.

The TIMESTAMP ORIGIN field indicates the origin of the timestamp (see 5.15).

The TIMESTAMP field contains the current value of the timestamp (see 5.15).

6.29 REQUEST SENSE command

The REQUEST SENSE command (see table 262) requests that the device server transfer parameter data containing sense data (see 4.5) to the application client.

Table 262 — REQUEST SENSE command

Bit / Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (03h)							
1	Reserved							DESC
2	Reserved							
3								
4	ALLOCATION LENGTH							
5	CONTROL							

The OPERATION CODE field is defined in 4.3.5.1 and shall be set as shown in table 262 for the REQUEST SENSE command.

The descriptor format (DESC) bit (see table 263) specifies which sense data format the device server shall return in the parameter data.

Table 263 — DESC bit

Code	Descriptor format sense data supported?	Description
0b	yes or no	The device server shall return fixed format sense data (see 4.5.3) in the parameter data.
1b	yes	The device server shall return descriptor format sense data (see 4.5.2) in the parameter data.
	no	The device server shall return no parameter data and terminate the REQUEST SENSE command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.
Note: Device servers that are compliant with SPC-3 are capable of ignoring a DESC bit that is set to one.		

The ALLOCATION LENGTH field is defined in 4.3.5.6. Application clients should request 252 bytes of sense data to ensure they retrieve all the sense data. If fewer than 252 bytes are requested, sense data may be lost since the REQUEST SENSE command with any allocation length clears the sense data.

The CONTROL field is defined in SAM-5.

Sense data shall be available and cleared under the conditions defined in SAM-5.

Upon completion of the REQUEST SENSE command, the logical unit shall return to the same power condition (see 5.12) that was active before the REQUEST SENSE command was received. A REQUEST SENSE command shall not reset any power condition timers.

The device server shall return CHECK CONDITION status for a REQUEST SENSE command only to report exception conditions specific to the REQUEST SENSE command itself. Examples of conditions that cause a REQUEST SENSE command to return CHECK CONDITION status are:

a) an invalid field value is detected in the CDB;
b) the device server does not support the REQUEST SENSE command (see 4.3.1);
c) the initiator port that sent the REQUEST SENSE command is pending enrollment for access controls (see 8.3.1.7) and no other sense data is available to return;
d) an unrecovered error is detected by a SCSI target port; or
e) a malfunction prevents return of the sense data.

If a REQUEST SENSE command is terminated with CHECK CONDITION status, then:

a) any parameter data that is transferred is invalid (i.e., the parameter data does not contain sense data); and
b) if the REQUEST SENSE command was received on an I_T nexus with a pending unit attention condition (i.e., before the device server reports CHECK CONDITION status), then the device server shall not clear the pending unit attention condition (see SAM-5).

The REQUEST SENSE command shall be capable of being processed to completion with GOOD status in any power condition (see 5.12). However, some implementations may require a level of power consumption that is higher than associated with the current power condition in any other circumstance. Details of how much power processing a REQUEST SENSE command requires is outside the scope of this standard.

Except as described elsewhere in this subclause, the device server shall process a REQUEST SENSE command as follows:

1) return applicable sense data in the parameter data as follows:
   1) if the logical unit that reports a peripheral qualifier of 011b or 001b in its standard INQUIRY data (see 6.4.2), then return parameter data containing sense data with the sense key set to ILLEGAL REQUEST and the additional sense code shall be set to LOGICAL UNIT NOT SUPPORTED;
   2) if the logical unit that reports a peripheral qualifier of 000b in its standard INQUIRY data because it has a peripheral device connected but is not ready for access, then return parameter data containing sense data appropriate to the condition that is making the logical unit not operational;
   3) if the REQUEST SENSE command was received on an I_T nexus with a pending unit attention condition, then return parameter data containing sense data for the unit attention and clear the unit attention condition as described in SAM-4;
   4) if a recovered error occurs during the processing of the REQUEST SENSE command, then return parameter data containing sense data with the sense key set to RECOVERED ERROR;
   5) if deferred error sense data (see 4.5.5) is available, then return parameter data containing sense data for the deferred error;
   6) return pollable sense data (see 5.6.2), if any; or
   7) return parameter data containing sense data with the sense key set to NO SENSE and the additional sense code set to NO ADDITIONAL SENSE INFORMATION;

   and
2) complete the REQUEST SENSE command with GOOD status.

Device servers shall return at least 18 bytes of parameter data in response to a REQUEST SENSE command if the allocation length is 18 or greater and the DESC bit is set to zero. Application clients may determine how much sense data has been returned by examining the ALLOCATION LENGTH field in the CDB and the ADDITIONAL SENSE LENGTH field in the sense data. Device servers shall not adjust the additional sense length to reflect truncation if the allocation length is less than the sense data available.

6.30 SECURITY PROTOCOL IN command

The SECURITY PROTOCOL IN command (see table 264) is used to retrieve security protocol information (see 7.7.1) or the results of one or more SECURITY PROTOCOL OUT commands (see 6.31).

Table 264 — SECURITY PROTOCOL IN command

Bit / Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (A2h)							
1	SECURITY PROTOCOL							
2	SECURITY PROTOCOL SPECIFIC							
3								
4	INC_512	Reserved						
5	Reserved							
6	(MSB)	ALLOCATION LENGTH						
•••								
9								(LSB)
10	Reserved							
11	CONTROL							

The OPERATION CODE field is defined in 4.3.5.1 and shall be set as shown in table 264 for the SECURITY PROTOCOL IN command.

The SECURITY PROTOCOL field (see table 265) specifies which security protocol is being used.

Table 265 — SECURITY PROTOCOL field in SECURITY PROTOCOL IN command

Code	Description	Reference
00h	Security protocol information	7.7.1
01h to 06h	Defined by the TCG	3.1.182
07h	CbCS	7.7.4
08h to 1Fh	Reserved	
20h	Tape Data Encryption	SSC-3
21h	Data Encryption Configuration	ADC-3
22h to 3Fh	Reserved	
40h	SA Creation Capabilities	7.7.2
41h	IKEv2-SCSI	7.7.3
42h to EBh	Reserved	
ECh	JEDEC Universal Flash Storage	UFS
EDh	SDcard TrustedFlash Security Systems Specification 1.1.3	3.1.152
EEh	Authentication in Host Attachments of Transient Storage Devices	IEEE 1667
EFh	ATA Device Server Password Security	SAT-2
F0h to FFh	Vendor Specific	

Editors Note 1 - ROW: SECURITY PROTOCOL field code values 22h to 2Fh are tentatively reserved for SSC-x uses.

The contents of the SECURITY PROTOCOL SPECIFIC field depend on the protocol specified by the SECURITY PROTOCOL field (see table 265).

A 512 increment (INC_512) bit set to one specifies that the ALLOCATION LENGTH field (see 4.3.5.6) expresses the maximum number of bytes available to receive data in increments of 512 bytes (e.g., a value of one means 512 bytes, two means 1 024 bytes, etc.). Pad bytes may or may not be appended to meet this length. Pad bytes shall have a value of 00h. An INC_512 bit set to zero specifies that the ALLOCATION LENGTH field expresses the number of bytes to be transferred.

Indications of data overrun or underrun and the mechanism, if any, for processing retries depend on the protocol specified by the SECURITY PROTOCOL field (see table 265).

The CONTROL field is defined in SAM-5.

Any association between a previous SECURITY PROTOCOL OUT command and the data transferred by a SECURITY PROTOCOL IN command depends on the protocol specified by the SECURITY PROTOCOL field (see table 265). If the device server has no data to transfer (e.g., the results for any previous SECURITY PROTOCOL OUT commands are not yet available), the device server may transfer data indicating it has no other data to transfer.

The format of the data transferred depends on the protocol specified by the SECURITY PROTOCOL field (see table 265).

The device server shall retain data resulting from a SECURITY PROTOCOL OUT command, if any, until one of the following events is processed:

a) transfer of the data via a SECURITY PROTOCOL IN command from the same I_T_L nexus as defined by the protocol specified by the SECURITY PROTOCOL field (see table 265);
b) logical unit reset (See SAM-4); or
c) I_T nexus loss (See SAM-4) associated with the I_T nexus that sent the SECURITY PROTOCOL OUT command.

If the data is lost due to one of these events the application client may send a new SECURITY PROTOCOL OUT command to retry the operation.

6.31 SECURITY PROTOCOL OUT command

The SECURITY PROTOCOL OUT command (see table 266) is used to send data to the logical unit. The data sent specifies one or more operations to be performed by the logical unit. The format and function of the operations depends on the contents of the SECURITY PROTOCOL field (see table 267). Depending on the protocol specified by the SECURITY PROTOCOL field, the application client may use the SECURITY PROTOCOL IN command (see 6.30) to retrieve data derived from these operations.

Table 266 — SECURITY PROTOCOL OUT command

Bit / Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (B5h)							
1	SECURITY PROTOCOL							
2	SECURITY PROTOCOL SPECIFIC							
3								
4	INC_512	Reserved						
5	Reserved							
6	(MSB)	TRANSFER LENGTH						
•••								
9								(LSB)
10	Reserved							
11	CONTROL							

The OPERATION CODE field is defined in 4.3.5.1 and shall be set as shown in table 266 for the SECURITY PROTOCOL OUT command.

The SECURITY PROTOCOL field (see table 267) specifies which security protocol is being used.

Table 267 — SECURITY PROTOCOL field in SECURITY PROTOCOL OUT command

Code	Description	Reference
00h	Reserved	
01h to 06h	Defined by the TCG	3.1.182
07h	CbCS	7.7.4
08h to 1Fh	Reserved	
20h	Tape Data Encryption	SSC-3
21h	Data Encryption Configuration	ADC-3
22h to 40h	Reserved	
41h	IKEv2-SCSI	7.7.3
42h to EBh	Reserved	
ECh	JEDEC Universal Flash Storage	UFS
EDh	SDcard TrustedFlash Security Systems Specification 1.1.3	3.1.152
EEh	Authentication in Host Attachments of Transient Storage Devices	IEEE 1667
EFh	ATA Device Server Password Security	SAT-2
F0h to FFh	Vendor Specific	

Editors Note 2 - ROW: SECURITY PROTOCOL field code values 22h to 2Fh are tentatively reserved for SSC-x uses.

The contents of the SECURITY PROTOCOL SPECIFIC field depend on the protocol specified by the SECURITY PROTOCOL field (see table 267).

A 512 increment (INC_512) bit set to one specifies that the TRANSFER LENGTH field (see 4.3.5.4) expresses the number of bytes to be transferred in increments of 512 bytes (e.g., a value of one means 512 bytes, two means 1 024 bytes, etc.). Pad bytes shall be appended as needed to meet this requirement. Pad bytes shall have a value of 00h. A INC_512 bit set to zero specifies that the TRANSFER LENGTH field indicates the number of bytes to be transferred.

The CONTROL field is defined in SAM-5.

Any association between a SECURITY PROTOCOL OUT command and a subsequent SECURITY PROTOCOL IN command depends on the protocol specified by the SECURITY PROTOCOL field (see table 267). Each protocol shall define whether:

a) the device server shall complete the command with GOOD status as soon as it determines the data has been correctly received. An indication that the data has been processed is obtained by sending a SECURITY PROTOCOL IN command and receiving the results in the associated data transfer; or
b) the device server shall complete the command with GOOD status only after the data has been successfully processed and an associated SECURITY PROTOCOL IN command is not required.

The format of the data transferred depends on the protocol specified by the SECURITY PROTOCOL field (see table 267).

6.32 SEND DIAGNOSTIC command

The SEND DIAGNOSTIC command (see table 268) requests that the device server perform diagnostic operations on the SCSI target device, on the logical unit, or on both. Logical units that support this command shall implement, at a minimum, the default self-test feature (i.e., the SELFTEST bit equal to one and a parameter list length of zero).

Table 268 — SEND DIAGNOSTIC command

Bit / Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (1Dh)							
1	SELF-TEST CODE			PF	Reserved	SELFTEST	DEVOFFL	UNITOFFL
2	Reserved							
3	(MSB)	PARAMETER LIST LENGTH						
4								(LSB)
5	CONTROL							

The OPERATION CODE field is defined in 4.3.5.1 and shall be set as shown in table 268 for the SEND DIAGNOSTIC command.

The SELF-TEST CODE field defined in table 269 and further described in table 270.

Table 269 — SELF-TEST CODE field

Code	Name	Description
000b		This value is used when an operation not described in this table is being requested (see table 270).
001b	Background short self-test	The device server shall start its short self-test (see 5.8.3) in the background mode (see 5.8.4.3).
010b	Background extended self-test	The device server shall start its extended self-test (see 5.8.3) in the background mode (see 5.8.4.3).
011b	Reserved	
100b	Abort background self-test	The device server shall abort the current self-test running in background mode (see 5.8.4.3). This value is only valid if a previous SEND DIAGNOSTIC command specified a background self-test, and that operation has not completed. If either of these conditions is not met, 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.
101b	Foreground short self-test	The device server shall start its short self-test (see 5.8.3) in the foreground mode (see 5.8.4.2).
110b	Foreground extended self-test	The device server shall start its extended self-test (see 5.8.3) in the foreground mode (see 5.8.4.2).
111b	Reserved	

The page format (PF) bit (see table 270) specifies the format of any parameter list sent by the application client for a SEND DIAGNOSTIC command and may specify the format of the parameter data, if any, returned by the device server in response to a subsequent RECEIVE DIAGNOSTIC RESULTS command (see 6.20). The implementation of the PF bit is optional.

NOTE 44 - Logical units compliant with SPC-2 may transfer more than one diagnostic page in the SEND DIAGNOSTIC command's parameter list and by doing so may request that more than one diagnostic page be transmitted in the RECEIVE DIAGNOSTIC RESULTS command's parameter data.

The self-test (SELFTEST) bit (see table 270) specifies whether the device server shall perform the default self-test (see 5.8.2).

A SCSI target device offline (DEVOFFL) bit set to one specifies that the device server may perform a default self-test (see 5.8.2) that affects any logical unit in the SCSI target device (e.g., by alteration of reservations, log parameters, or sense data). A DEVOFFL bit set to zero specifies that, after the device server has completed a default self-test specified in the SEND DIAGNOSTIC command, no logical unit shall exhibit any effects resulting from the device server's processing the SEND DIAGNOSTIC command that are detectable by any application client. If the SELFTEST bit is set to zero, the device server shall ignore the DEVOFFL bit.

A unit offline (UNITOFFL) bit set to one specifies that the device server may perform a default self-test (see 5.8.2) that affects the user accessible medium on the logical unit (e.g., write operations to the user accessible medium or repositioning of the medium on sequential access devices). A UNITOFFL bit set to zero specifies that, after the device server has completed a default self-test specified in the SEND DIAGNOSTIC command, the user accessible medium shall exhibit no effects resulting from the device server's processing the SEND DIAGNOSTIC command that are detectable by any application client. When the SELFTEST bit is set to zero, the device server shall ignore the UNITOFFL bit.

The PARAMETER LIST LENGTH field (see table 270) specifies the length in bytes of the parameter list that shall be transferred from the application client Data-Out Buffer to the device server. A parameter list length of zero specifies that no data shall be transferred. This condition shall not be considered an error.

The CONTROL field is defined in SAM-5.

Table 270 defines:

a) the meaning of the combinations of values for the SELF-TEST CODE field, the PF bit, the SELFTEST bit, and the PARAMETER LIST LENGTH field; and
b) under which conditions the DEVOFFL bit and the UNITOFFL bit, if implemented, may be used and when these bits are ignored.

Table 270 — SELF-TEST CODE field, the PF bit, the SELFTEST bit, and the PARAMETER LIST LENGTH field meanings
(part 1 of 4)

SELF-TEST CODE field	PF bit	SELF TEST bit	PARA-METER LIST LENGTH field	Description [a, b]
000b	0	0	0000h	The device server shall complete the command with GOOD status.
000b	0	0	>0000h	The device server shall receive the amount of vendor specific parameter list specified by the PARAMETER LIST LENGTH field. If the parameter list is valid and requests that the device server perform a diagnostic operation [c], then: a) if the operation succeeds, the device server shall complete the command with GOOD status; b) if the operation fails, then the device server shall terminate the command with CHECK CONDITION status with the sense key set to HARDWARE ERROR and the additional sense code set to the appropriate value to describe the failure; and c) if parameter data is required to be returned by the device server for the operation, then a subsequent RECEIVE DIAGNOSTIC RESULTS command (see 6.20) determines the device server's response. If the parameter list is valid and does not request the device server to perform a diagnostic operation, then the device server processes the vendor specific data. A subsequent RECEIVE DIAGNOSTIC RESULTS command may be required for an application client to determine the device server's response. If the parameter list is not valid, 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.

[a] Each description assumes that all other bits and fields in the CDB that are not defined for a particular case are valid. If, in any case, some other bit or field in the CDB is not valid, then the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.

[b] The device server shall ignore the DEVOFFL bit and the UNITOFFL bit in all cases where their use is not included in the description.

[c] Before beginning any self-test, the device server shall:
a) stop all running power condition timers;
b) not stop any process that causes a background function to occur (e.g., not stop any timers or counters associated with background functions); and
c) after completing the command, the device server shall reinitialize and restart all enabled power condition timers.

Table 270 — SELF-TEST CODE field, the PF bit, the SELFTEST bit, and the PARAMETER LIST LENGTH field meanings
(part 2 of 4)

SELF-TEST CODE field	PF bit	SELF TEST bit	PARA-METER LIST LENGTH field	Description [a, b]
000b	0	1	0000h	The device server shall perform its default self-test, and if implemented, use the values in the DEVOFFL bit and the UNITOFFL bit for processing the self-test If the self-test succeeds, the device server shall return GOOD status for the command. If the self-test fails, then the device server shall terminate the command with CHECK CONDITION status with the sense key set to HARDWARE ERROR and the additional sense code set to the appropriate value to describe the failure.
000b	0	1	>0000h	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.
000b	1	0	0000h	The device server shall: a) complete the command with GOOD status; and b) if requested by a subsequent RECEIVE DIAGNOSTIC RESULTS command, then: A) if the PF bit is supported by the device server, then return a single diagnostic page (see 7.2) as specified by the command; or B) if the PF bit is not supported by the device server, then return vendor specific parameter data as specified by the command.

[a] Each description assumes that all other bits and fields in the CDB that are not defined for a particular case are valid. If, in any case, some other bit or field in the CDB is not valid, then the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.

[b] The device server shall ignore the DEVOFFL bit and the UNITOFFL bit in all cases where their use is not included in the description.

[c] Before beginning any self-test, the device server shall:
a) stop all running power condition timers;
b) not stop any process that causes a background function to occur (e.g., not stop any timers or counters associated with background functions); and
c) after completing the command, the device server shall reinitialize and restart all enabled power condition timers.

Table 270 — SELF-TEST CODE field, the PF bit, the SELFTEST bit, and the PARAMETER LIST LENGTH field meanings (part 3 of 4)

SELF-TEST CODE field	PF bit	SELF TEST bit	PARA-METER LIST LENGTH field	Description [a, b]
000b	1	0	>0000h	The device server shall receive the amount of parameter list specified by the PARAMETER LIST LENGTH field. If the PF bit is supported by the device server, then: a) the parameter list shall be a single diagnostic page; and b) if the specified parameter list length results in the truncation of the diagnostic page (e.g., the parameter list length is less than the page length specified in the diagnostic page), then the device server shall terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB. If the PF bit is not supported by the device server, then the parameter list is vendor specific. If the parameter list is valid and requests that the device server perform a diagnostic operation [c], then: a) if the operation succeeds, the device server shall complete the command with GOOD status; b) if the operation fails, then the device server shall terminate the command with CHECK CONDITION status with the sense key set to HARDWARE ERROR and the additional sense code set to the appropriate value to describe the failure; and c) if parameter data is required to be returned by the device server for the operation, then a subsequent RECEIVE DIAGNOSTIC RESULTS command (see 6.20) determines the device server's response. If the parameter list is valid and does not request the device server to perform a diagnostic operation, then the device server processes the data as defined in 7.2. A subsequent RECEIVE DIAGNOSTIC RESULTS command may be required for an application client to determine the device server's response. If the parameter list is not valid, 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.

[a] Each description assumes that all other bits and fields in the CDB that are not defined for a particular case are valid. If, in any case, some other bit or field in the CDB is not valid, then the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.

[b] The device server shall ignore the DEVOFFL bit and the UNITOFFL bit in all cases where their use is not included in the description.

[c] Before beginning any self-test, the device server shall:
a) stop all running power condition timers;
b) not stop any process that causes a background function to occur (e.g., not stop any timers or counters associated with background functions); and
c) after completing the command, the device server shall reinitialize and restart all enabled power condition timers.

Table 270 — SELF-TEST CODE field, the PF bit, the SELFTEST bit, and the PARAMETER LIST LENGTH field meanings
(part 4 of 4)

SELF-TEST CODE field	PF bit	SELF TEST bit	PARA-METER LIST LENGTH field	Description [a, b]
000b	1	1	n/a	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.
>000b	0	0	0000h	The device server shall perform the operation defined in table 269. [c]
>000b	0	0	>0000h	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.
>000b	0	1	n/a	
>000b	1	0	0000h	The device server shall perform the operation defined in table 269. [c] If the device server supports the PF bit, then, for a subsequent RECEIVE DIAGNOSTIC RESULTS command, the device server shall return a single diagnostic page as specified by the command. If the device server does not support the PF bit, then, the parameter data for a subsequent RECEIVE DIAGNOSTIC RESULTS command is vendor specific.
>000b	1	0	>0000h	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.
>000b	1	1	n/a	

[a] Each description assumes that all other bits and fields in the CDB that are not defined for a particular case are valid. If, in any case, some other bit or field in the CDB is not valid, then the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.

[b] The device server shall ignore the DEVOFFL bit and the UNITOFFL bit in all cases where their use is not included in the description.

[c] Before beginning any self-test, the device server shall:
a) stop all running power condition timers;
b) not stop any process that causes a background function to occur (e.g., not stop any timers or counters associated with background functions); and
c) after completing the command, the device server shall reinitialize and restart all enabled power condition timers.

6.33 SET IDENTIFYING INFORMATION command

The SET IDENTIFYING INFORMATION command (see table 271) requests that the device server set identifying information (see 5.17) in the logical unit to the value received in the SET IDENTIFYING INFORMATION parameter list. The SET IDENTIFYING INFORMATION command is an extension to the SET PERIPHERAL DEVICE/ COMPONENT DEVICE IDENTIFIER service action of the MAINTENANCE OUT command defined in SCC-2. Additional MAINTENANCE IN and MAINTENANCE OUT service actions are defined in SCC-2 and in this standard.

The MAINTENANCE OUT service actions defined only in SCC-2 shall apply only to logical units that return a device type of 0Ch (i.e., storage array controller device) or the SCCS bit set to one in their standard INQUIRY data (see 6.4.2). When a logical unit returns a device type of 0Ch or the SCCS bit set to one in its standard INQUIRY data, the implementation requirements for the SCC-2 MAINTENANCE OUT service actions shall be as specified in SCC-2. Otherwise the MAINTENANCE OUT service action definitions and implementation requirements stated in this standard shall apply.

Processing a SET IDENTIFYING INFORMATION command may require the enabling of a nonvolatile memory within the logical unit. If the nonvolatile memory is not ready, the command shall be terminated with CHECK CONDITION status, and not wait for the nonvolatile memory to become ready. The sense key shall be set to NOT READY and the additional sense code shall be set as described in table 284 (see 6.37). This information should allow the application client to determine the action required to cause the device server to become ready.

On successful completion of a SET IDENTIFYING INFORMATION command that changes identifying information saved by the logical unit, the device server shall establish a unit attention condition (see SAM-4) for the initiator port associated with every I_T nexus except the I_T nexus on which the SET IDENTIFYING INFORMATION command was received, with the additional sense code set to DEVICE IDENTIFIER CHANGED.

Table 271 — SET IDENTIFYING INFORMATION command

Bit / Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (A4h)							
1	Reserved			SERVICE ACTION (06h)				
2	Reserved							
3								
4	Restricted (see SCC-2)							
5								
6	(MSB)	PARAMETER LIST LENGTH						
•••								
9								(LSB)
10	INFORMATION TYPE							Reserved
11	CONTROL							

The OPERATION CODE field is defined in 4.3.5.1 and shall be set as shown in table 271 for the SET IDENTIFYING INFORMATION command.

The SERVICE ACTION field is defined in 4.3.5.2 and shall be set as shown in table 271 for the SET IDENTIFYING INFORMATION command.

The PARAMETER LIST LENGTH field specifies the length in bytes of the identifying information that shall be transferred from the application client to the device server. A parameter list length of zero specifies that no data shall be transferred, and that subsequent REPORT IDENTIFYING INFORMATION commands shall return the INFORMATION LENGTH field set to zero for the specified information type.

The INFORMATION TYPE field (see table 272) specifies the identifying information type to be set. If the specified information type is not implemented by the device server, 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.

Table 272 — INFORMATION TYPE field

Code	Description
0000000b	Peripheral device identifying information (see 5.17). If the PARAMETER LIST LENGTH field is set to greater than the maximum length of the peripheral device identifying information (see 5.17) supported by the device server, then the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.
0000010b	Peripheral device text identifying information (see 5.17). If the PARAMETER LIST LENGTH field is set to a value greater than the maximum length of the peripheral device text identifying information (see 5.17) supported by the device server, then the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB. If the format of the INFORMATION field is incorrect, 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.
xxxxxx1b	Restricted (see SCC-2)
All other	Reserved

The CONTROL field is defined in SAM-5.

The SET IDENTIFYING INFORMATION parameter list (see table 273) contains the identifying information to be set by the device server.

Table 273 — SET IDENTIFYING INFORMATION parameter list

Bit Byte	7	6	5	4	3	2	1	0
0	INFORMATION							
•••								
n								

The INFORMATION field specifies the identifying information to be set for the specified information type (see 5.17).

Upon successful completion of a SET IDENTIFYING INFORMATION command, the identifying information that is saved by the logical unit shall persist through logical unit resets, hard resets, power loss, I_T nexus losses, media format operations, and media replacement.

6.34 SET PRIORITY command

The SET PRIORITY command (see table 274) requests that a priority be set to the specified value. The priority set by this command shall remain in effect until one of the following occurs:

a) another SET PRIORITY command is received;
b) hard reset;
c) logical unit reset; or
d) power off.

The priority set by this command shall not be affected by an I_T nexus loss.

The priority set by a SET PRIORITY command may be used as a command priority (see SAM-4) for commands received by the logical unit via an I_T nexus (i.e., an I_T_L nexus).

The SET PRIORITY command is a service action of the MAINTENANCE OUT command. Additional MAINTENANCE OUT service actions are defined in SCC-2 and in this standard. The MAINTENANCE OUT service actions defined only in SCC-2 apply only to logical units that return a device type of 0Ch or the SCCS bit set to one in their standard INQUIRY data.

Table 274 — SET PRIORITY command

Bit / Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (A4h)							
1	Reserved			SERVICE ACTION (0Eh)				
2	I_T_L NEXUS TO SET		Reserved					
3	Reserved							
•••								
5								
6	(MSB)	PARAMETER LIST LENGTH						
•••								
9								(LSB)
10	Reserved							
11	CONTROL							

The OPERATION CODE field is defined in 4.3.5.1 and shall be set as shown in table 274 for the SET PRIORITY command.

The SERVICE ACTION field is defined in 4.3.5.2 and shall be set as shown in table 274 for the SET PRIORITY command.

The I_T_L NEXUS TO SET field (see table 275) specifies the I_T_L nexus and the location of the priority value to be assigned to that I_T_L nexus.

Table 275 — I_T_L NEXUS TO SET field

Code	Description
00b	The priority for the I_T_L nexus associated with this command shall be set to the value contained in the PRIORITY TO SET field in the SET PRIORITY parameter list (see table 276). All fields in the SET PRIORITY parameter list except the PRIORITY TO SET field shall be ignored. If the parameter list length is zero, the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to PARAMETER LIST LENGTH ERROR.
01b	The priority for the I_T_L nexus specified by the logical unit that is processing this command, the RELATIVE TARGET PORT IDENTIFIER field, and the TRANSPORTID field in the SET PRIORITY parameter list (see table 276) shall be set to the value specified by the PRIORITY TO SET field in the SET PRIORITY parameter list. If the parameter list length results in the truncation of the RELATIVE TARGET PORT IDENTIFIER field, the ADDITIONAL DESCRIPTOR LENGTH field, or the TRANSPORTID 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 PARAMETER LIST LENGTH ERROR. On successful completion of a SET PRIORITY command the device server shall establish a unit attention condition (see SAM-4) for the initiator port associated with the I_T nexus specified by the TRANSPORTID field and the RELATIVE TARGET PORT IDENTIFIER field, with the additional sense code set to PRIORITY CHANGED.
10b	The priority value specified in the INITIAL COMMAND PRIORITY field of the Control Extension mode page (see 7.5.8) shall be used for all I_T_L nexuses associated with the logical unit that is processing this command regardless of any prior priority. The contents of the SET PRIORITY parameter list shall be ignored. On successful completion of a SET PRIORITY command the device server shall establish a unit attention condition (see SAM-4) for the initiator port associated with every other I_T_L nexus, with the additional sense code set to PRIORITY CHANGED.
11b	Reserved

The PARAMETER LIST LENGTH field specifies the length in bytes of the SET PRIORITY parameter list (see table 276) that shall be contained in the Data-Out Buffer. A parameter list length of zero specifies that the Data-Out Buffer shall be empty. This condition shall not be considered as an error.

The CONTROL field is defined in SAM-5.

The format for the parameter data returned by the SET PRIORITY command is shown in table 276.

Table 276 — SET PRIORITY parameter list format

Bit / Byte	7	6	5	4	3	2	1	0
0	Reserved				PRIORITY TO SET			
1	Reserved							
2	(MSB)	RELATIVE TARGET PORT IDENTIFIER						
3								(LSB)
4	Reserved							
5	Reserved							
6	(MSB)	ADDITIONAL LENGTH (n-7)						
7								(LSB)
8	TRANSPORTID							
•••								
n								

The PRIORITY TO SET field specifies the priority to be assigned to the I_T_L nexus specified by the I_T_L NEXUS TO SET field in the CDB. The value in the PRIORITY TO SET field shall be returned in subsequent REPORT PRIORITY commands (see 6.24) until one of the conditions described in this subclause occurs. A priority to set value of zero specifies the I_T_L nexus specified by the I_T_L NEXUS TO SET field shall be set to the value specified in the INITIAL COMMAND PRIORITY field of the Control Extension mode page (see 7.5.8). The contents of the I_T_L NEXUS TO SET field may specify that the PRIORITY TO SET field shall be ignored.

The RELATIVE TARGET PORT IDENTIFIER field contains the relative port identifier (see 3.1.127) of the target port that is part of the I_T_L nexus for which the priority is to be set. The contents of the I_T_L NEXUS TO SET field may specify that the RELATIVE TARGET PORT IDENTIFIER field shall be ignored.

The ADDITIONAL LENGTH field specifies the number of bytes that follow in the SET PRIORITY parameter list (i.e., the size of the TransportID).

The TRANSPORTID field contains a TransportID (see 7.6.4) identifying the initiator port that is part of the I_T_L nexus for which the priority is to be set. The contents of the I_T_L NEXUS TO SET field may specify that the TRANSPORTID field shall be ignored.

6.35 SET TARGET PORT GROUPS command

The SET TARGET PORT GROUPS command (see table 277) requests that the device server set the primary target port asymmetric access state of all of the target ports in the specified primary target port groups and/or the secondary target port asymmetric access state of the specified target ports. See 5.11 for details regarding the transition between target port asymmetric access states. This command is mandatory for all logical units that report in the standard INQUIRY data (see 6.4.2) that they support explicit asymmetric logical units access (i.e., the TPGS field contains either 10b or 11b).

The SET TARGET PORT GROUPS command is a service action of the MAINTENANCE OUT command. Additional MAINTENANCE OUT service actions are defined in SCC-2 and in this standard. The MAINTENANCE OUT service actions defined only in SCC-2 apply only to logical units that return a device type of 0Ch or the SCCS bit set to one in their standard INQUIRY data.

Table 277 — SET TARGET PORT GROUPS command

Bit / Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (A4h)							
1	Reserved			SERVICE ACTION (0Ah)				
2	Reserved							
•••								
5								
6	(MSB)	PARAMETER LIST LENGTH						
•••								
9								(LSB)
10	Reserved							
11	CONTROL							

The OPERATION CODE field is defined in 4.3.5.1 and shall be set as shown in table 277 for the SET TARGET PORT GROUPS command.

The SERVICE ACTION field is defined in 4.3.5.2 and shall be set as shown in table 277 for the SET TARGET PORT GROUPS command.

The PARAMETER LIST LENGTH field specifies the length in bytes of the target port group management parameters that shall be transferred from the application client to the device server. A parameter list length of zero specifies that no data shall be transferred, and that no change shall be made in the target port asymmetric access state of any target port groups or target ports. If the parameter list length violates the vendor specific length requirements, 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 CONTROL field is defined in SAM-5.

The allowable values to which target port asymmetric access states may be set is vendor specific and should be reported in the REPORT TARGET PORT GROUP parameter data (see 6.27).

Primary target port groups that are not specified in a parameter list may change primary target port asymmetric access states as a result of the SET TARGET PORT GROUPS command. This shall not be considered an implicit target port asymmetric access state change.

If the SET TARGET PORT GROUPS attempts to establish an invalid combination of target port asymmetric access states or attempts to establish an unsupported target port asymmetric access state, 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 PARAMETER LIST.

If the SET TARGET PORT GROUPS command has been performed, the completion of the command depends upon which of the following conditions apply:

a) if the transition is treated as a single indivisible event (see 5.11.2.5), then the SET TARGET PORT GROUPS command shall not complete until the transition to the requested state has completed; or
b) if the transition is not treated as a single indivisible event (i.e., the device server supports other commands (see 5.11.2.5) when those commands are routed though a target port that is transitioning between target port asymmetric access states), then the SET TARGET PORT GROUPS command may complete before the transition into the requested state has completed.

If the SET TARGET PORT GROUPS command is not performed successfully, the completion of the command depends upon which of the following conditions apply:

a) if the processing of a SET TARGET PORT GROUPS command requires the enabling of a nonvolatile memory and the nonvolatile memory is not ready, then the command shall be terminated with CHECK CONDITION status, rather than wait for the logical unit to become ready. The sense key shall be set to NOT READY and the additional sense code shall be set as described in table 284 (see 6.37); or
b) if a failure occurred before the transition was completed, the command shall be terminated with CHECK CONDITION status, with the sense key set to HARDWARE ERROR, and the additional sense code set to SET TARGET PORT GROUPS COMMAND FAILED.

If two SET TARGET PORT GROUPS commands are performed concurrently, the target port asymmetric access state change behavior is vendor specific. A SCSI target device should not process multiple SET TARGET PORT GROUPS concurrently.

The SET TARGET PORT GROUPS parameter data format is shown in table 278.

Table 278 — SET TARGET PORT GROUPS parameter list format

Bit / Byte	7	6	5	4	3	2	1	0
0 ••• 3	Reserved							
	Set target port group descriptor(s)							
4 ••• 7	Set target port group descriptor 0 (see table 279)							
	⋮							
n-3 ••• n	Set target port group descriptor x (see table 279)							

The format of the set target port group descriptor is defined in table 279.

Table 279 — Set target port group descriptor parameter list

Bit / Byte	7	6	5	4	3	2	1	0
0	Reserved				ASYMMETRIC ACCESS STATE			
1	Reserved							
2	(MSB)	TARGET PORT GROUP OR TARGET PORT						
3								(LSB)

If the ASYMMETRIC ACCESS STATE field (see table 280) specifies a primary target port asymmetric access state, then all the target ports in the specified target port group shall transition to the specified state (see 5.11.2.5). If the ASYMMETRIC ACCESS STATE field specifies a secondary target port asymmetric access state, then the specified target port shall transition to the specified state.

Table 280 — ASYMMETRIC ACCESS STATE field

Value	State (see 5.11.2.4)	Type (see 5.11.2.1)
0h	Active/optimized	Primary
1h	Active/non-optimized	Primary
2h	Standby	Primary
3h	Unavailable	Primary
4h	Illegal Request [a]	
5h to Dh	Reserved	
Eh	Offline	Secondary
Fh	Illegal Request [a]	

[a] If the ASYMMETRIC ACCESS STATE field in any target port group descriptor contains Fh, 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.

If the ASYMMETRIC ACCESS STATE field (see table 280) specifies a primary target port asymmetric access state, then the TARGET PORT GROUP OR TARGET PORT field specifies a primary target port group for which the primary target port asymmetric access state shall be changed. If the ASYMMETRIC ACCESS STATE field specifies a secondary target port asymmetric access state, then the TARGET PORT GROUP OR TARGET PORT field specifies the relative target port identifier (see 3.1.127) of the target port for which the secondary target port asymmetric access state shall be changed.

6.36 SET TIMESTAMP command

The SET TIMESTAMP command (see table 281) requests that the device server initialize the timestamp (see 5.15), if the SCSIP bit is set to one or the TCMOS bit is set to one in the Control Extension mode page (see 7.5.8). If the SCSIP bit is set to zero, the SET TIMESTAMP 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 SET TIMESTAMP command is a service action of the MAINTENANCE OUT command. Additional MAINTENANCE OUT service actions are defined in SCC-2 and in this standard. The MAINTENANCE OUT service actions defined only in SCC-2 apply only to logical units that return a device type of 0Ch or the SCCS bit set to one in their standard INQUIRY data (see 6.4.2).

Table 281 — SET TIMESTAMP command

Bit / Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (A4h)							
1	Reserved			SERVICE ACTION (0Fh)				
2	Reserved							
•••								
5								
6	(MSB)	PARAMETER LIST LENGTH						
•••								
9								(LSB)
10	Reserved							
11	CONTROL							

The OPERATION CODE field is defined in 4.3.5.1 and shall be set as shown in table 281 for the SET TIMESTAMP command.

The SERVICE ACTION field is defined in 4.3.5.2 and shall be set as shown in table 281 for the SET TIMESTAMP command.

The PARAMETER LIST LENGTH field specifies the length in bytes of the SET TIMESTAMP parameters that shall be transferred from the application client to the device server. A parameter list length of zero specifies that no data shall be transferred, and that no change shall be made to the timestamp.

The CONTROL field is defined in SAM-5.

The format for the parameter list for the SET TIMESTAMP command is shown in table 282.

Table 282 — SET TIMESTAMP parameter list format

Bit / Byte	7	6	5	4	3	2	1	0
0	Reserved							
•••								
3								
4	(MSB)	TIMESTAMP						
•••								
9								(LSB)
10	Reserved							
11	Reserved							

The TIMESTAMP field shall contain the initial value of the timestamp in the format defined in 5.15. The timestamp should be the number of milliseconds that have elapsed since midnight, 1 January 1970 UT. If the high order byte in the TIMESTAMP field is greater than F0h, 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.

On successful completion of a SET TIMESTAMP command the device server shall establish a unit attention condition for the initiator port associated with every I_T nexus except the I_T nexus on which the SET TIMESTAMP command was received (see SAM-4), with the additional sense code set to TIMESTAMP CHANGED.

6.37 TEST UNIT READY command

The TEST UNIT READY command (see table 283) provides a means to check if the logical unit is ready. This is not a request for a self-test. If the logical unit is able to accept an appropriate medium-access command without returning CHECK CONDITION status, this command shall return a GOOD status. If the logical unit is unable to become operational or is in a state such that an application client action (e.g., START UNIT command) is required to make the logical unit ready, the command shall be terminated with CHECK CONDITION status, with the sense key set to NOT READY.

Table 283 — TEST UNIT READY command

Bit / Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (00h)							
1	Reserved							
•••								
4								
5	CONTROL							

The OPERATION CODE field is defined in 4.3.5.1 and shall be set as shown in table 283 for the TEST UNIT READY command.

The CONTROL field is defined in SAM-5.

Table 284 defines the suggested CHECK CONDITION status responses to the TEST UNIT READY command. Other conditions (e.g., deferred errors, reservations, or target port asymmetric access state changes) may result in other responses (e.g., GOOD status, CHECK CONDITION status, BUSY status, or RESERVATION CONFLICT status, each with or without other sense key and additional sense code values).

Table 284 — Preferred TEST UNIT READY responses

Status	Sense Key	Additional Sense Code
CHECK CONDITION	ILLEGAL REQUEST	LOGICAL UNIT NOT SUPPORTED
CHECK CONDITION	NOT READY	LOGICAL UNIT DOES NOT RESPOND TO SELECTION
CHECK CONDITION	NOT READY	MEDIUM NOT PRESENT
CHECK CONDITION	NOT READY	LOGICAL UNIT NOT READY, CAUSE NOT REPORTABLE
CHECK CONDITION	NOT READY	LOGICAL UNIT IS IN PROCESS OF BECOMING READY
CHECK CONDITION	NOT READY	LOGICAL UNIT NOT READY, INITIALIZING COMMAND REQUIRED
CHECK CONDITION	NOT READY	LOGICAL UNIT NOT READY, MANUAL INTERVENTION REQUIRED
CHECK CONDITION	NOT READY	LOGICAL UNIT NOT READY, FORMAT IN PROGRESS
CHECK CONDITION	NOT READY	LOGICAL UNIT NOT READY, SANITIZE IN PROGRESS

6.38 WRITE ATTRIBUTE command

The WRITE ATTRIBUTE command (see table 285) allows an application client to write attributes to medium auxiliary memory. Device servers that implement the WRITE ATTRIBUTE command shall also implement the READ ATTRIBUTE command (see 6.15). Application clients should issue READ ATTRIBUTE commands prior to using this command to discover device server support for medium auxiliary memory.

Table 285 — WRITE ATTRIBUTE command

Bit / Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (8Dh)							
1	Reserved							WTC
2	Restricted (see SMC-2)							
3								
4								
5	LOGICAL VOLUME NUMBER							
6	Reserved							
7	PARTITION NUMBER							
8	Reserved							
9								
10	(MSB)	PARAMETER LIST LENGTH						
•••								
13								(LSB)
14	Reserved							
15	CONTROL							

The OPERATION CODE field is defined in 4.3.5.1 and shall be set as shown in table 285 for the WRITE ATTRIBUTE command.

The write-through cache (WTC) bit set to one specifies the attributes in the parameter list shall be synchronized with the medium auxiliary memory during the processing of the WRITE ATTRIBUTE command and GOOD status shall not be returned until the attributes have been synchronized with the medium auxiliary memory. The WTC bit is set to zero specifies no requirement related to the attributes in the parameter list being synchronized with the medium auxiliary memory during the processing of the WRITE ATTRIBUTE command.

The LOGICAL VOLUME NUMBER field specifies a logical volume (e.g., the medium auxiliary memory storage for one side of a double sided medium) within the medium auxiliary memory. The number of logical volumes of the medium auxiliary memory shall equal that of the attached medium. If the medium only has a single logical volume, then its logical volume number shall be zero.

The PARTITION NUMBER field specifies a partition (see SSC-2) within a logical volume. The number of partitions of the medium auxiliary memory shall equal that of the attached medium. If the medium only has a single partition, then its partition number shall be zero.

If the combination of logical volume number and partition number is not valid, 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 PARAMETER LIST LENGTH field specifies the length in bytes of the parameter list contained in the Data-Out Buffer. A parameter list length of zero specifies that no parameter data is present; this shall not be considered an error. If the parameter list length results in the truncation of an attribute, the WRITE ATTRIBUTE command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to PARAMETER LIST LENGTH ERROR.

The CONTROL field is defined in SAM-5.

The parameter list shall have the format shown in table 286. Attributes should be sent in ascending numerical order. If the attributes are not in order, then no attributes shall be changed and the WRITE 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.

Table 286 — WRITE ATTRIBUTE parameter list format

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)							
•••				PARAMETER DATA LENGTH (n-3)				
3								(LSB)
	Attribute(s)							
4								
•••				Attribute 0 (see 7.4.1)				
				⋮				
•••				Attribute x (see 7.4.1)				
n								

The PARAMETER DATA LENGTH field should contain the number of bytes of attribute data and shall be ignored by the device server.

The format of the attributes is described in 7.4.1.

If there is not enough space to write the attributes to the medium auxiliary memory, then no attributes shall be changed and the WRITE ATTRIBUTE command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to AUXILIARY MEMORY OUT OF SPACE.

If the medium auxiliary memory is not accessible because there is no medium present, then no attributes shall be changed and the WRITE ATTRIBUTE command shall be terminated with CHECK CONDITION status, with the sense key set to NOT READY, and the additional sense code set to MEDIUM NOT PRESENT.

If the medium is present but the medium auxiliary memory is not accessible, then no attributes shall be changed and the WRITE ATTRIBUTE command shall be terminated with CHECK CONDITION status, with the sense key set to MEDIUM ERROR, and the additional sense code set to LOGICAL UNIT NOT READY, AUXILIARY MEMORY NOT ACCESSIBLE.

If the medium auxiliary memory is not operational (e.g., bad checksum), the WRITE ATTRIBUTE command shall be terminated with CHECK CONDITION status, with the sense key set to MEDIUM ERROR, and the additional sense code set to AUXILIARY MEMORY WRITE ERROR.

If the WRITE ATTRIBUTE command parameter data contains an attribute with an ATTRIBUTE LENGTH field (see 7.4.1) set to zero, then one of the following actions shall occur:

a) if the attribute state is unsupported or read only (see 5.13), then no attributes shall be changed and the WRITE 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;
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 ATTRIBUTE command and not be reported by the READ ATTRIBUTE command with ATTRIBUTE LIST service action; or
c) if the attribute state is nonexistent, the attribute in the WRITE ATTRIBUTE command parameter list shall be ignored; this shall not be considered an error.

No attributes shall be changed, the WRITE 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 if the parameter data contains any of the following:

a) an attempt to change an attribute in the read only state (see 5.13);
b) an attribute with incorrect ATTRIBUTE LENGTH field (see 7.4.1) contents; or
c) an attribute with unsupported ATTRIBUTE VALUE field (see 7.4.1) contents.

6.39 WRITE BUFFER command

6.39.1 WRITE BUFFER command introduction

The WRITE BUFFER command (see table 287) is used in conjunction with the READ BUFFER command for:

a) testing logical unit buffer memory;
b) testing the integrity of the service delivery subsystem;
c) downloading microcode (see 5.18); and
d) downloading application client error history (see 5.14).

Table 287 — WRITE BUFFER command

Bit / Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (3Bh)							
1	Reserved			MODE				
2	BUFFER ID							
3	(MSB)			BUFFER OFFSET				
4								
5								(LSB)
6	(MSB)			PARAMETER LIST LENGTH				
7								
8								(LSB)
9	CONTROL							

This command shall not alter any medium of the logical unit when the data mode or the combined header and data mode is specified.

The OPERATION CODE field is defined in 4.3.5.1 and shall be set as shown in table 287 for the WRITE BUFFER command.

The function of this command and the meaning of fields within the CDB depend on the contents of the MODE field. The MODE field is defined in table 288.

Table 288 — WRITE BUFFER MODE field

Code	Description	Reference
00h	Combined header and data [a]	6.39.2
01h	Vendor specific [a]	6.39.3
02h	Data	6.39.4
03h	Reserved	
04h	Download microcode and activate	5.18 and 6.39.5
05h	Download microcode, save, and activate	5.18 and 6.39.6
06h	Download microcode with offsets and activate	5.18 and 6.39.7
07h	Download microcode with offsets, save, and activate	5.18 and 6.39.8
08h to 09h	Reserved	
0Ah	Write data to echo buffer	6.39.9
0Bh to 0Dh	Reserved	
0Eh	Download microcode with offsets, save, and defer active	5.18 and 6.39.10
0Fh	Activate deferred microcode	5.18 and 6.39.11
10h to 19h	Reserved	
1Ah	Enable expander communications protocol and Echo buffer	6.39.12
1Bh	Disable expander communications protocol	6.39.13
1Ch	Download application client error history	5.14 and 6.39.14
1Dh to 1Fh	Reserved	
[a] Modes 00h and 01h are not recommended.		

The CONTROL field is defined in SAM-5.

6.39.2 Combined header and data mode (00h)

In this mode, data to be transferred is preceded by a four-byte header. The four-byte header consists of all reserved bytes.

The BUFFER ID and the BUFFER OFFSET fields shall be set to zero.

The PARAMETER LIST LENGTH field specifies the maximum number of bytes that shall be transferred from the Data-Out Buffer. This number includes four bytes of header, so the data length to be stored in the device server's buffer is parameter list length minus four. The application client should attempt to ensure that the parameter list length is not greater than four plus the BUFFER CAPACITY field value (see 6.16.2) that is returned in the header of the READ BUFFER command combined header and data mode (i.e., 0h). If the parameter list length exceeds the buffer capacity, 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.

6.39.3 Vendor specific mode (01h)

In this mode, the meaning of the BUFFER ID, BUFFER OFFSET, and PARAMETER LIST LENGTH fields are not specified by this standard.

6.39.4 Data mode (02h)

In this mode, the Data-Out Buffer contains buffer data destined for the logical unit. The BUFFER ID field identifies a specific buffer within the logical unit. The vendor assigns buffer ID codes to buffers within the logical unit. Buffer ID zero shall be supported. If more than one buffer is supported, then additional buffer ID codes shall be assigned contiguously, beginning with one. If an unsupported buffer ID code is selected, 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 BUFFER OFFSET field specifies the location in the buffer to which the data is written. The application client should conform to the offset boundary requirements returned in the READ BUFFER descriptor (see 6.16.5). If the device server is unable to process the specified buffer offset, 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 PARAMETER LIST LENGTH field specifies the maximum number of bytes that shall be transferred from the Data-Out Buffer to be stored in the specified buffer beginning at the buffer offset. The application client should attempt to ensure that the parameter list length plus the buffer offset does not exceed the capacity of the specified buffer. The capacity of the buffer is indicated by the BUFFER CAPACITY field in the READ BUFFER descriptor (see 6.16.5). If the BUFFER OFFSET and PARAMETER LIST LENGTH fields specify a transfer in excess of the buffer capacity, 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.

6.39.5 Download microcode and activate mode (04h)

In this mode, microcode shall be transferred to the device server and activated (see 5.18).

The BUFFER ID field, BUFFER OFFSET field, and PARAMETER LIST LENGTH field are vendor specific.

6.39.6 Download microcode, save, and activate mode (05h)

In this mode, microcode shall be transferred to the device server, saved to nonvolatile storage, and activated (see 5.18) based on the setting of the ACTIVATE MICROCODE field in the Extended INQUIRY VPD page (see 7.8.7).

The BUFFER ID field, BUFFER OFFSET field, and PARAMETER LIST LENGTH field are vendor specific.

6.39.7 Download microcode with offsets and activate mode (06h)

In this mode, microcode shall be transferred to the device server using one or more WRITE BUFFER commands and activated (see 5.18).

The BUFFER ID field specifies a buffer within the logical unit. The vendor assigns buffer ID codes to buffers within the logical unit. A buffer ID value of zero shall be supported. If more than one buffer is supported, then additional buffer ID codes shall be assigned contiguously, beginning with one. If an unsupported buffer ID code is specified, 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 BUFFER OFFSET field specifies the location in the buffer to which the microcode is written. The application client shall send commands that conform to the offset boundary requirements returned in the READ BUFFER descriptor

(see 6.16.5). If the device server is unable to process the specified buffer offset, 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 PARAMETER LIST LENGTH field specifies the maximum number of bytes that shall be present in the Data-Out Buffer to be stored in the specified buffer beginning at the buffer offset. The application client should ensure that the parameter list length plus the buffer offset does not exceed the capacity of the specified buffer. The capacity of the buffer is indicated by the BUFFER CAPACITY field in the READ BUFFER descriptor (see 6.16.5). If the BUFFER OFFSET and PARAMETER LIST LENGTH fields specify a transfer in excess of the buffer capacity, 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.

6.39.8 Download microcode with offsets, save, and activate mode (07h)

In this mode, microcode shall be transferred to the device server using one or more WRITE BUFFER commands, saved to nonvolatile storage, and activated (see 5.18) based on the setting of the ACTIVATE MICROCODE field in the Extended INQUIRY VPD page (see 7.8.7).

The BUFFER ID field, BUFFER OFFSET field, and PARAMETER LIST LENGTH field are defined in the download microcode with offsets mode (see 6.39.7).

6.39.9 Write data to echo buffer mode (0Ah)

In this mode the device server transfers data from the application client and stores it in an echo buffer. An echo buffer is assigned in the same manner by the device server as it would for a write operation. Data shall be sent aligned on four-byte boundaries.

The BUFFER ID and BUFFER OFFSET fields shall be ignored in this mode.

> NOTE 45 - It is recommended that the logical unit assign echo buffers on a per I_T nexus basis to limit the number of exception conditions that may occur when I_T nexuses are present.

Upon successful completion of a WRITE BUFFER command the data shall be preserved in the echo buffer unless there is an intervening command to any logical unit in which case the data may be changed.

The PARAMETER LIST LENGTH field specifies the maximum number of bytes that shall be transferred from the Data-Out Buffer to be stored in the echo buffer. The application client should ensure that the parameter list length does not exceed the capacity of the echo buffer. The capacity of the echo buffer is indicated by the BUFFER CAPACITY field in the READ BUFFER echo buffer descriptor (see 6.16.7). If the PARAMETER LIST LENGTH field specifies a transfer in excess of the buffer capacity, 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.

6.39.10 Download microcode with offsets, save, and defer activate mode (0Eh)

In this mode, microcode shall be transferred to the device server using one or more WRITE BUFFER commands, saved to nonvolatile storage, and considered deferred (see 5.18).

The deferred microcode shall be activated and no longer considered deferred when one of the following occurs:

a) a power on;
b) a hard reset;
c) a START STOP UNIT command is processed (see SBC-3);
d) a FORMAT UNIT command is processed (see SBC-3); or

e) a WRITE BUFFER command with the activate deferred microcode mode (0Fh) is processed (see 6.39.11).

The BUFFER ID field, BUFFER OFFSET field, and PARAMETER LIST LENGTH field are defined in the download microcode with offsets mode (see 6.39.7).

6.39.11 Activate deferred microcode mode (0Fh)

In this mode, deferred microcode that has been saved using the download microcode with offsets, save, and defer activate mode (see 6.39.10), if any, shall be activated and no longer considered deferred (see 5.18).

The the BUFFER ID field, the BUFFER OFFSET field, and PARAMETER LIST LENGTH field shall be ignored in this mode.

If there is no deferred microcode that has been saved using the download microcode with offsets, save, and defer activate mode, then the WRITE BUFFER command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to COMMAND SEQUENCE ERROR.

6.39.12 Enable expander communications protocol and Echo buffer mode (1Ah)

Receipt of a WRITE BUFFER command with this mode causes a communicative expander (see SPI-5) to enter the expanded communications protocol mode.

Device servers in SCSI target devices that receive a WRITE BUFFER command with this mode shall process it as if it were a WRITE BUFFER command with the write data to Echo buffer mode (see 6.39.9).

6.39.13 Disable expander communications protocol mode (1Bh)

Receipt of a WRITE BUFFER command with this mode causes a communicative expander (see SPI-5) to exit the expanded communications protocol mode and return to simple expander operation.

Device servers in SCSI target devices that receive a WRITE BUFFER command with this mode 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.

6.39.14 Download application client error history mode (1Ch)

In this mode the device server transfers application client error history from the application client and stores it in the error history (see 5.14). The format of the application client error history parameter list is defined in table 289.

The BUFFER ID field and BUFFER OFFSET field shall be ignored in this mode.

Upon successful completion of a WRITE BUFFER command, the information contained in the application client error history parameter list shall be appended to the application client error history in a format determined by the logical unit.

The PARAMETER LIST LENGTH field specifies the length in bytes of the application client error history parameter list that shall be transferred from the application client to the device server. If the PARAMETER LIST LENGTH field specifies a transfer that exceeds the error history capacity, 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 device server shall not return an error based on the contents of any of the field values defined in table 289 except:

a) the CLR bit;

b) the ERROR LOCATION LENGTH field; and
c) the APPLICATION CLIENT ERROR HISTORY LENGTH field.

Table 289 — Application client error history parameter list format

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)							
•••				T10 VENDOR IDENTIFICATION				
7								(LSB)
8	(MSB)							
9				ERROR TYPE				(LSB)
10				Reserved				CLR
11				Reserved				
12	(MSB)							
•••				TIMESTAMP				
17								(LSB)
18				Reserved				
19								
20		Reserved			CODE SET			
21				ERROR LOCATION FORMAT				
22	(MSB)							
23				ERROR LOCATION LENGTH (m-25)				(LSB)
24	(MSB)							
25				APPLICATION CLIENT ERROR HISTORY LENGTH (n-m)				(LSB)
26	(MSB)							
•••				ERROR LOCATION				
m								(LSB)
m+1								
•••				APPLICATION CLIENT ERROR HISTORY				
n								

The T10 VENDOR IDENTIFICATION field contains eight bytes of left-aligned ASCII data (see 4.4.1) identifying the vendor providing the application client error history. The T10 vendor identification shall be one assigned by INCITS. A list of assigned T10 vendor identifications is in Annex E and on the T10 web site (http://www.t10.org).

The ERROR TYPE field (see table 290) specifies the error detected by the application client.

Table 290 — ERROR TYPE field

Code	Description
0000h	No error specified by the application client
0001h	An unknown error was detected by the application client
0002h	The application client detected corrupted data
0003h	The application client detected a permanent error
0004h	The application client detected a service response of SERVICE DELIVERY OR TARGET FAILURE (see SAM-4).
0005h to 7FFFh	Reserved
8000h to FFFFh	Vendor specific

If the CLR_SUP bit is set to one in the error history directory parameter data (see 6.16.9.2), a CLR bit set to one specifies that the device server shall:

a) clear the portions of the error history that the device server allows to be cleared; and
b) ignore any application client error history specified in the parameter list.

If the CLR_SUP bit is set to one in the error history directory parameter data, a CLR bit set to zero specifies that the device server shall not ignore the CLR bit.

If the CLR_SUP bit is set to one in the error history directory parameter data, a CLR bit set to zero specifies that the device server shall:

a) not clear the error history; and
b) process all application client error history specified in the parameter list.

If the CLR_SUP bit is set to zero in the error history directory parameter data, the device server shall ignore the CLR bit.

The TIMESTAMP field shall contain:

a) a time based on the timestamp reported by the REPORT TIMESTAMP command, if the device server supports a device clock (see 5.15);
b) the number of milliseconds that have elapsed since midnight, 1 January 1970 UT (see 3.1.185); or
c) zero, if the application client is not able to determine the UT of the log entry.

The CODE SET field contains a code set enumeration (see 4.4.3) that indicates the format of the APPLICATION CLIENT ERROR HISTORY field.

The ERROR LOCATION FORMAT field (see table 291) specifies the format of the ERROR LOCATION field.

Table 291 — ERROR LOCATION FORMAT field

Code	Description
00h	No error history location specified by the application client
01h	For direct-access block devices (see SBC-3 and RBC), the ERROR LOCATION field specifies the logical block address associated the specified application client error history. For other peripheral device types, this code is reserved.
02h to 7Fh	Reserved
80h to FFh	Vendor specific

The ERROR LOCATION LENGTH field specifies the length of the ERROR LOCATION field. The ERROR LOCATION LENGTH field value shall be a multiple of four. An ERROR LOCATION LENGTH field set to zero specifies that there is no error location information.

The APPLICATION CLIENT ERROR HISTORY LENGTH field specifies the length of the APPLICATION CLIENT ERROR HISTORY field. The APPLICATION CLIENT ERROR HISTORY LENGTH field value shall be a multiple of four. An APPLICATION CLIENT ERROR HISTORY field set to zero specifies that there is no vendor specific information.

The ERROR LOCATION field specifies the location at which the application client detected the error, in the format specified by the ERROR LOCATION FORMAT field.

The APPLICATION CLIENT ERROR HISTORY field contains vendor specific application client error history (see 5.14.1).

7 Parameters for all device types

7.1 Overview

Parameters for all device types are defined in this clause as follows:

a) diagnostic parameters are defined in 7.2;
b) log parameters are defined in 7.3;
c) medium auxiliary attributes are defined in 7.4;
d) mode parameters are defined in 7.5;
e) protocol specific parameters are defined in 7.6;
f) security parameters are defined in 7.7; and
g) vital product data are defined in 7.8.

7.2 Diagnostic parameters

7.2.1 Summary of diagnostic page codes

The page code assignments for diagnostic pages are summarized in table 292.

Table 292 — Summary of diagnostic page codes

Diagnostic page description	Page code	Reference
Defined by SES-3 for: a) standalone enclosure services devices (i.e., logical units with the PERIPHERAL DEVICE TYPE field set to 0Dh in standard INQUIRY data (see 6.4.2)); and b) attached enclosure services devices (i.e., logical units with the ENCSERV bit set to one in standard INQUIRY data).	01h to 2Fh	SES-3
Protocol Specific	3Fh	7.2.3
Supported Diagnostic Pages	00h	7.2.4
Restricted (see applicable command standard)	40h to 7Fh	
Vendor specific [a]		
Reserved	All other codes	
A numeric ordered listing of diagnostic page codes is provided in D.4.		
[a] The following page codes are vendor specific: 80h to FFh.		

7.2.2 Diagnostic page format and page codes for all device types

This subclause describes the diagnostic page structure and the diagnostic pages that are applicable to all SCSI devices. Diagnostic pages specific to each device type are described in the command standard (see 3.1.31) that applies to that device type.

A SEND DIAGNOSTIC command with a PF bit set to one specifies that the SEND DIAGNOSTIC parameter list consists of a single diagnostic page and that the data returned by the subsequent RECEIVE DIAGNOSTIC RESULTS command that has the PCV bit set to zero shall use the diagnostic page format defined in table 293. A

RECEIVE DIAGNOSTIC RESULTS command with a PCV bit set to one specifies that the device server return a diagnostic page using the format defined in table 293.

Table 293 — Diagnostic page format

Bit / Byte	7	6	5	4	3	2	1	0
0	PAGE CODE							
1	Page code specific							
2	(MSB)	PAGE LENGTH (n-3)						
3								(LSB)
4	Diagnostic parameters							
•••								
n								

Each diagnostic page defines a function or operation that the device server shall perform as a result of a SEND DIAGNOSTIC command or the information being returned as a result of a RECEIVE DIAGNOSTIC RESULTS command with the PCV bit equal to one. The diagnostic parameters contain data that is formatted according to the page code specified.

The PAGE CODE field (see 7.2.1) identifies the diagnostic page.

The PAGE LENGTH field contains the length in bytes of the diagnostic parameters that follow this field. If the application client sends a SEND DIAGNOSTIC command with a parameter list containing a PAGE LENGTH field that results in the truncation of any parameter, 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 PARAMETER LIST.

The diagnostic parameters are defined for each diagnostic page code. The diagnostic parameters within a diagnostic page may be defined differently in a SEND DIAGNOSTIC command than in a RECEIVE DIAGNOSTIC RESULTS command.

7.2.3 Protocol Specific

The Protocol Specific diagnostic page (see table 294) provides access to SCSI transport protocol specific diagnostic parameters.

Table 294 — Protocol Specific diagnostic page

Bit / Byte	7	6	5	4	3	2	1	0
0	PAGE CODE (3Fh)							
1	Reserved				PROTOCOL IDENTIFIER			
2	(MSB)	PAGE LENGTH (n-3)						
3								(LSB)
4	SCSI transport protocol specific diagnostic parameters							
•••								
n								

The PAGE CODE field is described in 7.2.2, and shall be set to 3Fh to indicate a Protocol Specific diagnostic page follows.

The PROTOCOL IDENTIFIER field contains one of the values shown in table 424 (see 7.6.1) to identify the SCSI transport protocol standard that defines the SCSI transport protocol specific diagnostic parameters. The SCSI transport protocol specific data is defined by the corresponding SCSI transport protocol standard.

The PAGE LENGTH field specifies the length in bytes of the following supported page list.

The SCSI transport protocol specific diagnostic parameters are defined by the SCSI transport protocol standard that corresponds to the value in the PROTOCOL IDENTIFIER field.

7.2.4 Supported Diagnostic Pages

The Supported Diagnostic Pages diagnostic page (see table 295) returns the list of diagnostic pages implemented by the device server. This diagnostic page shall be implemented if the device server implements the diagnostic page format option of the SEND DIAGNOSTIC and RECEIVE DIAGNOSTIC RESULTS commands.

Table 295 — Supported Diagnostic Pages diagnostic page

Bit / Byte	7	6	5	4	3	2	1	0
0	PAGE CODE (00h)							
1	Reserved							
2	(MSB)	PAGE LENGTH (n-3)						
3								(LSB)
4		SUPPORTED PAGE LIST						
•••								
n								

The definition of this diagnostic page for the SEND DIAGNOSTIC command includes only the first four bytes. If the PAGE LENGTH field is not zero, the device server shall terminate the SEND DIAGNOSTIC 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. This diagnostic page instructs the device server to make available the list of all supported diagnostic pages to be returned by a subsequent RECEIVE DIAGNOSTIC RESULTS command.

The definition of this diagnostic page for the RECEIVE DIAGNOSTIC RESULTS command includes the list of diagnostic pages supported by the device server.

The PAGE CODE field is described in 7.2.2, and shall be set to 00h to indicate a Supported Diagnostics Pages diagnostic page follows.

The PAGE LENGTH field indicates the length in bytes of the following supported page list.

The SUPPORTED PAGE LIST field shall contain a list of all diagnostic page codes, one per byte, implemented by the device server in ascending order beginning with page code 00h.

7.3 Log parameters

7.3.1 Summary of log page codes

The page code assignments for log pages are summarized in table 296.

Table 296 — Summary of log page codes

Log page name	Page code	Subpage code	Reference
Application Client	0Fh	00h	7.3.3
Buffer Over-Run/Under-Run	01h	00h	7.3.4
Cache Memory Statistics	19h	20h	7.3.5
General Statistics and Performance	19h	00h	7.3.6
Group Statistics and Performance (1 to 31)	19h	01h to 1Fh	7.3.6
Informational Exceptions	2Fh	00h	7.3.8
Last *n* Deferred Errors or Asynchronous Events	0Bh	00h	7.3.9
Last *n* Error Events	07h	00h	7.3.10
Non-Medium Error	06h	00h	7.3.11
Power Condition Transitions	1Ah	00h	7.3.12
Protocol Specific Port [a]	18h	00h to FEh	7.3.13
Read Error Counter	03h	00h	7.3.14
Read Reverse Error Counter	04h	00h	7.3.15
Self-Test Results	10h	00h	7.3.16
Start-Stop Cycle Counter	0Eh	00h	7.3.17
Supported Log Pages	00h	00h	7.3.18
Supported Log Pages and Subpages	00h	FFh	7.3.19
Supported Subpages	01h to 3Fh	FFh	7.3.20
Temperature	0Dh	00h	7.3.21
Verify Error Counter	05h	00h	7.3.22
Write Error Counter	02h	00h	7.3.23
Reserved (may be used by specific device types)	08h to 0Ah	00h to FEh	
	0Ch	00h to FEh	
	11h to 17h	00h to FEh	
	1Bh to 2Eh	00h to FEh	
Vendor specific [b]			
Reserved	All other codes		

A numeric ordered listing of log pages codes and subpage codes is provided in D.5.

[a] Each SCSI transport protocol standard (see 3.1.147) may define a different name for these log pages.
[b] The following combinations page codes and subpage codes are vendor specific: 30h to 3Eh/00h to FEh.

7.3.2 Log page structure and log parameter structure for all device types

7.3.2.1 Log page structure

This subclause describes the log page structure that is applicable to all SCSI devices. Log pages specific to each device type are described in the command standard that applies to that device type. The LOG SELECT command (see 6.5) supports the ability to send zero or more log pages. The LOG SENSE command (see 6.6) returns the log page or log pages specified by the combination of the PAGE CODE field and SUBPAGE CODE field in the CDB.

Each log page begins with a four-byte page header followed by zero or more variable-length log parameters defined for that log page. The log page format is shown in table 297.

Table 297 — Log page format

Bit / Byte	7	6	5	4	3	2	1	0
0	DS	SPF	PAGE CODE					
1	SUBPAGE CODE							
2	(MSB)	PAGE LENGTH (n-3)						
3								(LSB)
	Log parameter(s)							
4	Log parameter [first] (see 7.3.2.2) (Length x)							
•••								
x+3								
	⋮							
n-y+1	Log parameter [last] (see 7.3.2.2) (Length y)							
•••								
n								

For the LOG SENSE command (see 6.6), the DS bit indicates whether log parameters in this log page are saved when the SP bit is set to one in the CDB. If the DS bit is set to zero, the log parameters are saved when the SP bit is set to one. If the DS bit is set to one, the log parameters are not saved. For the LOG SELECT command (see 6.5), the disable save (DS) bit operates in conjunction with the PCR bit, the SP bit, the PC field, and the PARAMETER LIST LENGTH field in the CDB.

If the subpage format (SPF) bit is set to zero, then the SUBPAGE CODE field shall contain 00h. If the SPF bit is set to one, then the SUBPAGE CODE field shall contain a value between 01h and FFh.

The PAGE CODE field contains the number of the log page (see 7.3.1) that is being transferred.

The SUBPAGE CODE field contains the subpage number of the log page (see 7.3.1) that is being transferred.

If an application client specifies values in the PAGE CODE field and SUBPAGE CODE field for a log page that is reserved or not implemented by the logical unit, then the device server shall terminate the LOG 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.

If the PARAMETER LIST LENGTH field in a LOG SELECT CDB contains zero, the meanings for the PCR bit, SP bit, and PC field are defined in 6.5.2.

If the PARAMETER LIST LENGTH field in a LOG SELECT CDB contains a non-zero value (i.e., when a parameter list is being sent with the LOG SELECT command), then table 298 defines the meaning for the combinations of values for:

a) the PCR bit, the SP bit, and the PC field in the LOG SELECT CDB; and
b) the PARAMETER CODE field, the FORMAT AND LINKING field, and the DS bit in each log parameter transferred.

Table 298 — LOG SELECT PCR bit, SP bit, and DS bit meanings when parameter list length is not zero

PCR bit	SP bit	DS bit	Description
0b	0b	xb	The device server shall set the specified values[a] to the values in the parameter list and shall not save any values to non-volatile media.
0b	1b	0b	The device server shall set the specified values[a] to the values in the parameter list and shall process the optional saving of log parameter values as follows: a) If default data counter values are specified (see table 149 in 6.5.1), no values shall be saved; b) If values other than default data counter values are specified and the device server implements saving of the specified values[a], then the device server shall save the specified values[a] in the parameter list to non-volatile media; or c) If values other than default values are specified and the device server does not implement saving of one or more of the specified values[a], 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.
0b	1b	1b	The device server shall set the specified values[a] to the values in the parameter list and shall not save any values in the specified log page to non-volatile media.
1b	xb	xb	The device server 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.

[a] The specified parameters are determined by the PARAMETER CODE field contents (see 7.3.2.2.1) in the LOG SELECT parameter data as well as by the PC field contents (see table 149 in 6.5.1) in the LOG SELECT CDB.

The value in the PAGE LENGTH field contains the length in bytes of the log parameters that follow. If the application client sends a log page length that results in the truncation of any parameter, 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.

7.3.2.2 Log parameter structure

7.3.2.2.1 Introduction

Most log pages contain one or more data structures called log parameters (see table 299). Log parameters may be data counters of a particular event(s), the conditions under which certain operations were performed, or list parameters that contain a character string or binary description related to a particular event.

Table 299 — Log parameter

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)	PARAMETER CODE						
1								(LSB)
2	Parameter control byte (see 7.3.2.2.2)							
	DU	Obsolete	TSD	ETC	TMC		FORMAT AND LINKING	
3	PARAMETER LENGTH (n-3)							
4	(MSB)	PARAMETER VALUE						
•••								
n								(LSB)

Each log parameter begins with a four-byte parameter header followed by one or more bytes of parameter value data.

The PARAMETER CODE field identifies the log parameter being transferred. The device server shall return the log parameters in a log page in ascending order based on the value in their PARAMETER CODE field.

If an application client specifies a value in the PARAMETER CODE field in the LOG SELECT command parameter data that is reserved or not implemented by the logical unit, 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 DU bit, TSD bit, ETC bit, TMC field, and FORMAT AND LINKING field are collectively referred to as the parameter control byte. The bits and fields in the parameter control byte are described in 7.3.2.2.2.

The PARAMETER LENGTH field specifies the length in bytes of the PARAMETER VALUE field that follows. If the application client specifies a parameter length that results in the truncation of the PARAMETER VALUE field, 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.

If the application client sends a value in a PARAMETER VALUE field that is outside the range supported by the logical unit, and rounding is implemented for that parameter, the device server may:

a) round to an acceptable value and terminate the command as described in 5.4; or
b) 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.

If the parameter data for one LOG SELECT command contains more than one log page and the log pages are not in ascending order by page code value then subpage code value, 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.

If the parameter data for one LOG SELECT command contains more than one log parameter in any one log page and the log parameters are not in ascending order by parameter code value, 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.

Application clients should send LOG SENSE commands prior to sending LOG SELECT commands to determine supported log pages and page lengths.

The SCSI target device may provide independent sets of log parameters for each logical unit or for each combination of logical units and I_T nexuses. If the SCSI target device does not support independent sets of log parameters and any log parameters are changed that affect other I_T nexuses, then the device server shall establish a unit attention condition (see SAM-4) for the initiator port associated with every I_T nexus except the I_T nexus on which the LOG SELECT command was received, with the additional sense code set to LOG PARAMETERS CHANGED.

7.3.2.2.2 Parameter control byte

7.3.2.2.2.1 Introduction

The bits and fields in the parameter control byte are described in 7.3.2.2.2.

For cumulative log parameter values, indicated by the PC field (see table 149 in 6.5.1) of the LOG SELECT command and LOG SENSE command, the disable update (DU) bit is defined as follows:

a) DU set to zero indicates that the device server shall update the log parameter value to reflect all events that should be noted by that parameter; or
b) DU set to one indicates that the device server shall not update the log parameter value except in response to a LOG SELECT command that specifies a new value for the parameter.

NOTE 46 - When updating cumulative log parameter values, a device server may use volatile memory to hold these values until a LOG SELECT or LOG SENSE command is received with an SP bit set to one or a vendor specific event occurs. As a result the updated cumulative log parameter values may be lost if a power cycle occurs.

If the PC field (see table 149 in 6.5.1) indicates that threshold values or default values are being processed, the device server shall:

a) set the DU bit to zero, if a LOG SENSE command is being processed; and
b) ignore the DU bit, if a LOG SELECT command is being processed.

Regardless of the value in the PC field, the device server shall process ASCII format list log parameters (see 7.3.2.2.2.4) and binary format list log parameters (see 7.3.2.2.2.5) by:

a) setting the DU bit to zero, if a LOG SENSE command is being processed; and
b) ignoring the DU bit, if a LOG SELECT command is being processed.

A target save disable (TSD) bit set to zero indicates that the logical unit implicitly saves the log parameter at vendor specific intervals. This implicit saving operation shall be done frequently enough to ensure that the cumulative parameter values retain statistical significance (i.e., across power cycles). A TSD bit set to one indicates that either the logical unit does not implicitly save the log parameter or implicit saving of the log parameter has been disabled individually by an application client setting the TSD bit to one. An application client may disable the implicit saving for all log parameters without changing any TSD bits using the GLTSD bit in the Control mode page (see 7.5.7).

An enable threshold comparison (ETC) bit set to one indicates that a comparison to the threshold value is performed whenever the cumulative value is updated. An ETC bit set to zero indicates that a comparison is not performed. The value of the ETC bit is the same for cumulative and threshold parameters.

The threshold met criteria (TMC) field (see table 300) defines the basis for comparison of the cumulative and threshold values. The TMC field is valid only if the ETC bit is set to one. The value of the TMC field is the same for cumulative and threshold parameters.

Table 300 — Threshold met criteria (TMC) field

Code	Basis for comparison		
00b	Every update of the cumulative value		
01b	Cumulative value	equal to	threshold value
10b	Cumulative value	not equal to	threshold value
11b	Cumulative value	greater than	threshold value

If the ETC bit is set to one and the result of the comparison is true, the device server shall establish a unit attention condition (see SAM-4) for the initiator port associated with every I_T nexus, with the additional sense code set to THRESHOLD CONDITION MET.

The FORMAT AND LINKING field (see table 301) indicates the type of log parameter.

Table 301 — FORMAT AND LINKING field

Code	Log parameter type	Reference
00b	Bounded data counter	7.3.2.2.2.2
01b	ASCII format list	7.3.2.2.2.4
10b	Bounded data counter or unbounded data counter	7.3.2.2.2.2 or 7.3.2.2.2.3
11b	Binary format list	7.3.2.2.2.5

7.3.2.2.2.2 Parameter control byte values for bounded data counter parameters

The device server shall return LOG SENSE parameter control byte values and process LOG SELECT parameter control byte values as shown in table 302 for any log parameter that is defined to be a bounded data counter log parameter.

Table 302 — Parameter control byte values for bounded data counter parameters

Field or bit	Value for LOG SENSE	Value for LOG SELECT	Description
DU	0 or 1	0 or 1	When the DU bit is set to zero, the device server shall update the log parameter value to reflect all events that should be noted by that parameter. When the DU bit is set to one, the device server shall not update the log parameter value except in response to a LOG SELECT command that specifies a new value for the parameter.
TSD	0 or 1	0 or 1	When the TSD bit is set to zero, the device server shall save the log parameter to its medium at vendor specific intervals. When the TSD bit is set to one, implicit saving of the log parameter is disabled by an application client.
ETC	0 or 1	0 or 1	When the ETC bit is set to one, a comparison to the threshold value is performed whenever the cumulative value is updated. When the ETC bit is set to zero, a comparison is not performed.
TMC	any	any	The TMC field (see table 300 in 7.3.2.2.2.1) defines the basis for comparison of the cumulative and threshold values. The TMC field is valid only if the ETC bit is set to one.
FORMAT AND LINKING	00b or 10b	00b or 10b	The log parameter is a data counter (see table 301 in 7.3.2.2.2.1) and the handling of a parameter that reaches its maximum value is described in this subclause

If a LOG SELECT command contains a bounded data counter log parameter in which the parameter control byte values differ from those shown in table 302, 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 PARAMETER LIST.

Each bounded data counter log parameter contains one saturating counter that is:

a) associated with one or more events; and
b) incremented whenever one of these events occurs.

If the counter in a bounded data counter log parameter has associated with it a vendor specific maximum value, then upon reaching this maximum value, the data counter shall not be incremented (i.e., its value does not wrap).

When the counter in a bounded data counter log parameter reaches its maximum value (i.e., saturates), the device server shall:

a) set the DU bit to one;
b) handle other bounded data counter log parameters in the log page based on the contents of the FORMAT AND LINKING field in each other log parameter as follows:
   A) if the FORMAT AND LINKING field is set to 00b, then that other log parameter shall stop incrementing until reinitialized by a LOG SELECT command; or

B) if the FORMAT AND LINKING field is set to 10b, then that other log parameter shall not stop incrementing, but may be reinitialized by a LOG SELECT command.

and

c) not alter the handling of other log parameters in the log page that are:
   A) unbounded data counter log parameters (see 7.3.2.2.2.3);
   B) ASCII format list log parameters (see 7.3.2.2.2.4); and
   C) binary format list log parameters (see 7.3.2.2.2.5).

The processing of a command shall not be altered because the counter in a bounded data counter log parameter reaches its maximum value (i.e., saturates). If the RLEC bit is set to one in the Control mode page (see 7.5.7) and the processing of a command encounters no exception conditions other than the counter in a bounded data counter log parameter reaching its maximum value, then the command shall be terminated with CHECK CONDITION status, with the sense key set to RECOVERED ERROR, and the additional sense code set to LOG COUNTER AT MAXIMUM.

7.3.2.2.2.3 Parameter control byte values for unbounded data counter parameters

The device server shall return LOG SENSE parameter control byte values and process LOG SELECT parameter control byte values as shown in table 303 for any log parameter that is defined to be an unbounded data counter log parameter.

Table 303 — Parameter control byte values for unbounded data counter parameters

Field or bit	Value for LOG SENSE	Value for LOG SELECT	Description
DU	0 or 1	0 or 1	When the DU bit is set to zero, the device server shall update the log parameter value or values to reflect all events that should be noted by that parameter. When the DU bit is set to one, the device server shall not update the log parameter value or values except in response to a LOG SELECT command that specifies a new value for the parameter.
TSD	0 or 1	0 or 1	When the TSD bit is set to zero, the device server shall save the log parameter to its medium at vendor specific intervals. When the TSD bit is set to one, implicit saving of the log parameter is disabled by an application client.
ETC	0	0	Threshold comparisons are not performed for unbounded data counter parameters
TMC	00b	ignored	Threshold comparisons are not performed for unbounded data counter parameters
FORMAT AND LINKING	10b	10b	The log parameter is a data counter for which saturation of another log parameter does not affect the incrementing of this log parameter (see table 301 in 7.3.2.2.2.1).

If a LOG SELECT command contains an unbounded data counter log parameter in which the parameter control byte values differ from those shown in table 303, 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 PARAMETER LIST.

Each unbounded data counter log parameter contains one or more saturating counters or wrapping counters. The description of each counter field in the log parameter defines when the device server modifies the contents of the counter that is transferred in that field.

Changes in an unbounded data counter (e.g., a counter reaching saturation or another maximum value) shall not affect the handling of other log parameters in the log page. The processing of a command and the status returned by that command shall not be altered because a counter in an unbounded data counter log parameter saturates or reaches its maximum value.

The device server shall not change the value in the DU bit in an unbounded data counter log parameter unless requested to do so by a LOG SELECT command.

7.3.2.2.2.4 Parameter control byte values for ASCII format list log parameters

The device server shall return LOG SENSE parameter control byte values and process LOG SELECT parameter control byte values as shown in table 304 for any log parameter that is defined to be an ASCII format (see 4.4.1) list log parameter.

Table 304 — Parameter control byte values for ASCII format list log parameters

Field or bit	Value for LOG SENSE	Value for LOG SELECT	Description
DU	0	ignored	The DU bit is not defined for list parameters.
TSD	0 or 1	0 or 1	When the TSD bit is set to zero, the device server shall save the log parameter to its medium at vendor specific intervals. When the TSD bit is set to one, implicit saving of the log parameter is disabled by an application client.
ETC	0	0	Threshold comparisons are not performed for list parameters.
TMC	00b	ignored	Threshold comparisons are not performed for list parameters.
FORMAT AND LINKING	01b	01b	The log parameter is an ASCII format list parameter (see table 301 in 7.3.2.2.2.1).

If a LOG SELECT command contains an ASCII format list log parameter in which the parameter control byte values differ from those shown in table 304, 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 PARAMETER LIST.

7.3.2.2.2.5 Parameter control byte values for binary format list log parameters

The device server shall return LOG SENSE parameter control byte values and process LOG SELECT parameter control byte values as shown in table 305 for any log parameter that is defined to be a binary format list log parameter.

Table 305 — Parameter control byte values for binary format list log parameters

Field or bit	Value for LOG SENSE	Value for LOG SELECT	Description
DU	0	ignored	The DU bit is not defined for list parameters.
TSD	0 or 1	0 or 1	When the TSD bit is set to zero, the device server shall save the log parameter to its medium at vendor specific intervals. When the TSD bit is set to one, implicit saving of the log parameter is disabled by an application client.
ETC	0	0	Threshold comparisons are not performed for list parameters.
TMC	00b	ignored	Threshold comparisons are not performed for list parameters.
FORMAT AND LINKING	11b	11b	The log parameter is an binary format list parameter (see table 301 in 7.3.2.2.2.1).

If a LOG SELECT command contains a binary format list log parameter in which the parameter control byte values differ from those shown in table 305, 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 PARAMETER LIST.

7.3.2.2.2.6 Resetting and setting log parameters

In a LOG SELECT command, an application client may specify that:

a) all the parameters in a log page or pages are to be reset (i.e., the PCR bit set to one); or
b) individual parameters in log page are to be changed to specified new values (i.e., the PCR bit set to zero and the PARAMETER LIST LENGTH field not set to zero).

The device server handling of these requests depends on the log parameter that is being reset or changed, and is specified in the table that defines the log parameter using the keywords defined in table 306.

Table 306 — Keywords for resetting or changing log parameters

Keyword	Device server handling when	
	PCR bit is set to one [a]	PCR bit is set to zero [b]
Always	Reset the log parameter	Change the log parameter
Reset Only	Reset the log parameter	If any changes are requested in the parameter value field of the log parameter, then a) 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; and b) do not make any requested changes in any field in any log parameter in any log page
Never	Do not reset the log parameter; table 151 (see 6.5.1) describes error conditions that may apply	

[a] If the PCR bit is set to one and the PARAMETER LIST LENGTH field is not set to zero. then the LOG SELECT command shall be terminated (see table 298 in 7.3.2.1).
[b] If the PCR bit is set to zero and the PARAMETER LIST LENGTH field is set to zero. then no log parameters are changed (see 6.5.2).

7.3.3 Application Client log page

7.3.3.1 Overview

Using the format shown in table 308, the Application Client log page (page code 0Fh) provides a place for application clients to store information. The parameter codes for the Application Client log page are listed in table 307.

Table 307 — Application Client log page parameter codes

Parameter code	Description	Resettable or Changeable [a]	Reference	Support requirements
0000h to 003Fh	General Usage Application Client	Always	7.3.3.2	Mandatory
0040h to 0FFFh				Optional
all others	Reserved			

[a] The keywords in this column – Always, Reset Only, and Never – are defined in 7.3.2.2.2.6.

The Application Client log page has the format shown in table 308.

Table 308 — Application Client log page

Bit / Byte	7	6	5	4	3	2	1	0
0	DS	SPF (0b)	PAGE CODE (0Fh)					
1	SUBPAGE CODE (00h)							
2	(MSB)	PAGE LENGTH (n-3)						
3								(LSB)
	Application client log parameters							
4 •••	Application client log parameter (see 7.3.3.2) [first]							
	⋮							
••• n	Application client log parameter (see 7.3.3.2) [last]							

The DS bit, SPF bit, PAGE CODE field, SUBPAGE CODE field, and PAGE LENGTH field are described in 7.3.2. The SPF bit, PAGE CODE field, and SUBPAGE CODE field shall be set as shown in table 308 for the Application Client log page.

Each application client log parameter contains the information described in 7.3.3.2.

7.3.3.2 General Usage Application Client log parameter

The General Usage Application Client log parameter has the format shown in table 309. This information may be used to describe the system configuration and system problems, but the specific definition of the data is application client specific.

Table 309 — General Usage Application Client log parameter

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)	PARAMETER CODE (see table 307)						
1								(LSB)
2	Parameter control byte – binary format list log parameter (see 7.3.2.2.2.5)							
	DU	Obsolete	TSD	ETC	TMC		FORMAT AND LINKING	
3	PARAMETER LENGTH (FCh)							
4 ••• 255	GENERAL USAGE PARAMETER BYTES							

The PARAMETER CODE field is described in 7.3.2.2.1, and shall be set as shown in table 307 for the General Usage Application Client log parameter.

The DU bit, TSD bit, ETC bit, TMC field, and FORMAT AND LINKING field are described in 7.3.2.2.2.1. These fields shall be set as described for a binary format list log parameter (see 7.3.2.2.2.5) for the General Usage Application Client log parameter.

The contents of the GENERAL USAGE PARAMETER BYTES field represent data sent to the device server in a previous LOG SELECT command. If a previous LOG SELECT command has not occurred, the contents of the GENERAL USAGE PARAMETER BYTES field are vendor specific.

7.3.4 Buffer Over-Run/Under-Run log page

7.3.4.1 Overview

Using the format shown in table 311, the Buffer Over-Run/Under-Run log page (page code 01h) defines bounded data counters that record the number of buffer over-runs or under-runs detected by the device server. The parameter codes for the Buffer Over-Run/Under-Run log page are listed in table 310.

Table 310 — Buffer Over-Run/Under-Run log page parameter codes (part 1 of 2)

Parameter code [a]	Incremented once per:			Resettable or Changeable [b]	Reference	Support requirements
	Over- or Under-run	Experienced by (or during)	Problem detected			
0000h	Under-run	Undefined	Undefined	Reset Only	7.3.4.2	At least one [c]
0001h	Over-run					
0020h	Under-run	A command				
0021h	Over-run					
0040h	Under-run	An I_T nexus				
0041h	Over-run					
0080h	Under-run	A unit of time [d]				
0081h	Over-run					
0002h	Under-run	Undefined	Service delivery subsystem busy	Reset Only	7.3.4.2	At least one [c]
0003h	Over-run					
0022h	Under-run	A command				
0023h	Over-run					
0042h	Under-run	An I_T nexus				
0043h	Over-run					
0082h	Under-run	A unit of time [d]				
0083h	Over-run					

[a] See SPC-2 for a description of how these parameter codes are derived.
[b] The keywords in this column – Always, Reset Only, and Never – are defined in 7.3.2.2.2.6.
[c] If the Buffer Over-Run/Under-Run log page is supported, at least one of the parameter codes listed in this table shall be supported.
[d] The size of the unit of time is vendor specific.

Table 310 — Buffer Over-Run/Under-Run log page parameter codes (part 2 of 2)

Parameter code [a]	Incremented once per:			Resettable or Changeable [b]	Reference	Support requirements
	Over- or Under-run	Experienced by (or during)	Problem detected			
0004h	Under-run	Undefined	Transfer rate too slow	Reset Only	7.3.4.2	At least one [c]
0005h	Over-run					
0024h	Under-run	A command				
0025h	Over-run					
0044h	Under-run	An I_T nexus				
0045h	Over-run					
0084h	Under-run	A unit of time [d]				
0085h	Over-run					
all others	Reserved					

[a] See SPC-2 for a description of how these parameter codes are derived.
[b] The keywords in this column – Always, Reset Only, and Never – are defined in 7.3.2.2.2.6.
[c] If the Buffer Over-Run/Under-Run log page is supported, at least one of the parameter codes listed in this table shall be supported.
[d] The size of the unit of time is vendor specific.

A buffer over-run or under-run may occur when a SCSI initiator device does not transmit data to or from the logical unit's buffer fast enough to keep up with reading or writing the media. A buffer over-run condition may occur during a read operation when a buffer full condition prevents continued transfer of data from the media to the buffer. A buffer under-run condition may occur during a write operation when a buffer empty condition prevents continued transfer of data to the media from the buffer. Most devices incur a delay at this point while the media is repositioned.

The Buffer Over-Run/Under-Run log page has the format shown in table 311.

Table 311 — Buffer Over-Run/Under-Run log page

Bit / Byte	7	6	5	4	3	2	1	0
0	DS	SPF (0b)	PAGE CODE (01h)					
1	SUBPAGE CODE (00h)							
2	(MSB)	PAGE LENGTH (n-3)						
3								(LSB)
	Buffer over-run/under-run log parameters							
4	Buffer Over-run/Under-run log parameter (see 7.3.4.2) [first]							
•••								
	⋮							
	Buffer Over-run/Under-run log parameter (see 7.3.4.2) [last]							
•••								
n								

The DS bit, SPF bit, PAGE CODE field, SUBPAGE CODE field, and PAGE LENGTH field are described in 7.3.2. The SPF bit, PAGE CODE field, and SUBPAGE CODE field shall be set as shown in table 311 for the Buffer Over-Run/Under-Run log page.

Each Buffer Over-run/Under-run log parameter contains the information described in 7.3.4.2.

7.3.4.2 Buffer Over-run/Under-run log parameter

The Buffer Over-run/Under-run log parameter has the format shown in table 312.

Table 312 — Buffer Over-run/Under-run log parameter

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)	PARAMETER CODE (see table 310)						
1								(LSB)
2	Parameter control byte – bounded data counter log parameter (see 7.3.2.2.2.2)							
	DU	Obsolete	TSD	ETC	TMC		FORMAT AND LINKING	
3	PARAMETER LENGTH (n-3)							
4	(MSB)	OVER-RUN/UNDER-RUN COUNTER						
•••								
n								(LSB)

The PARAMETER CODE field is described in 7.3.2.2.1, and shall be set as shown in table 310 for the Buffer Over-run/Under-run log parameter log parameter.

The DU bit, TSD bit, ETC bit, TMC field, and FORMAT AND LINKING field are described in 7.3.2.2.2.1. These fields shall be set as described for a bounded data counter log parameter (see 7.3.2.2.2.2) for the Buffer Over-run/Under-run log parameter log parameter.

The PARAMETER LENGTH field is described in 7.3.2.2.2.1.

The OVER-RUN/UNDER-RUN COUNTER field contains the value for the counter described by the contents of the PARAMETER CODE field.

Each counter contains the total number of times buffer over-run or under-run conditions have occurred since the last time the counter was reset. The counter shall be incremented for each occurrence of a buffer under-run or over-run condition and may be incremented more than once for multiple occurrences during the processing of a single command.

7.3.5 Cache Memory Statistics log page

7.3.5.1 Overview

Using the format shown in table 315, the Cache Memory Statistics log page (page code 19h, subpage code 20h) contains statistics and performance results identified by the parameter codes listed in table 313.

Table 313 — Cache Memory Statistics log page parameter codes

Parameter code	Description	Resettable or Changeable [a]	Reference	Support requirements
0001h	Read Cache Memory Hits	Reset Only	7.3.5.2	At least one [b]
0002h	Reads To Cache Memory	Reset Only	7.3.5.3	
0003h	Write Cache Memory Hits	Reset Only	7.3.5.4	
0004h	Writes From Cache Memory	Reset Only	7.3.5.5	
0005h	Time From Last Hard Reset	Never	7.3.5.6	
0006h	Time Interval	Never	7.3.5.7	
all others	Reserved			

[a] The keywords in this column – Always, Reset Only, and Never – are defined in 7.3.2.2.2.6.
[b] If the Cache Memory Statistics log page is supported, at least one of the parameter codes listed in this table shall be supported.

The Cache Memory Statistics log page provides the following statistics and performance results associated with the addressed logical unit:

a) Cache Memory Statistics log parameters:
   A) number of read cache memory hits;
   B) number of reads to cache memory;
   C) number of write cache memory hits; and
   D) number of writes from cache memory;
b) Hard Reset log parameter:
   A) time from last hard reset (see SAM-4);
   and
c) Time Interval log parameter:
   A) time interval.

In the Cache Memory Statistics log page, read commands and write commands are those shown in table 314.

Table 314 — Cache Memory Statistics log page commands

Read commands [a]	Write commands [a]
READ(6) READ(10) READ(12) READ(16) READ(32) PRE-FETCH(10) PRE-FETCH(16)	SYNCHRONIZE CACHE(10) SYCNHRONIZE CACHE(16) WRITE(6) WRITE(10) WRITE(12) WRITE(16) WRITE(32) WRITE AND VERIFY(10) WRITE AND VERIFY(12) WRITE AND VERIFY(16) WRITE AND VERIFY(32)
[a] See SBC-3.	

The Cache Memory Statistics log page has the format shown in table 315.

Table 315 — Cache Memory Statistics log page

Bit / Byte	7	6	5	4	3	2	1	0
0	DS	SPF (1b)	PAGE CODE (19h)					
1	SUBPAGE CODE (20h)							
2	(MSB)	PAGE LENGTH (n-3)						
3								(LSB)
	Cache memory statistics log parameters							
4 ••• 	Cache memory statistics log parameter (see table 313) [first]							
	⋮							
 ••• n	Cache memory statistics log parameter (see table 313) [last]							

The DS bit, SPF bit, PAGE CODE field, SUBPAGE CODE field, and PAGE LENGTH field are described in 7.3.2. The SPF bit, PAGE CODE field, and SUBPAGE CODE field shall be set as shown in table 315 for the Cache Memory Statistics log page.

The contents of each cache memory statistics log parameter depends on the value in its PARAMETER CODE field (see table 313).

7.3.5.2 Read Cache Memory Hits log parameter

The Read Cache Memory Hits log parameter has the format shown in table 316.

Table 316 — Read Cache Memory Hits log parameter

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)	PARAMETER CODE (0001h)						
1								(LSB)
2	Parameter control byte – bounded data counter log parameter (see 7.3.2.2.2.2)							
	DU	Obsolete	TSD	ETC	TMC		FORMAT AND LINKING	
3	PARAMETER LENGTH (08h)							
4	(MSB)	NUMBER OF READ CACHE MEMORY HITS						
•••								
11								(LSB)

The PARAMETER CODE field is described in 7.3.2.2.1, and shall be set as shown in table 316 for the Read Cache Memory Hits log parameter.

The DU bit, TSD bit, ETC bit, TMC field, and FORMAT AND LINKING field are described in 7.3.2.2.2.1. These fields shall be set as described for a bounded data counter log parameter (see 7.3.2.2.2.2) for the Read Cache Memory Hits log parameter.

The PARAMETER LENGTH field is described in 7.3.2.2.2.1, and shall be set as shown in table 316 for the Read Cache Memory Hits log parameter.

The NUMBER OF READ CACHE MEMORY HITS field indicates the number of read commands (see table 314 in 7.3.5.1) received on an I_T nexus that resulted in:

a) user data being read from cache memory; and
b) no user data read from the medium before the read command being processed is completed.

The NUMBER OF READ CACHE MEMORY HITS field shall not be modified as a result of any read command that contains an FUA bit (see SBC-3) set to one or an FUA_NV bit (see SBC-3) set to one.

The contents of the NUMBER OF READ CACHE MEMORY HITS field shall be set to zero as part of processing a hard reset condition (see SAM-4).

7.3.5.3 Reads To Cache Memory Hits log parameter

The Reads To Cache Memory log parameter has the format shown in table 317.

Table 317 — Reads To Cache Memory log parameter

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)	PARAMETER CODE (0002h)						
1								(LSB)
2	Parameter control byte – bounded data counter log parameter (see 7.3.2.2.2.2)							
	DU	Obsolete	TSD	ETC	TMC		FORMAT AND LINKING	
3	PARAMETER LENGTH (08h)							
4	(MSB)	NUMBER OF READS TO CACHE MEMORY						
•••								
11								(LSB)

The PARAMETER CODE field is described in 7.3.2.2.1, and shall be set as shown in table 317 for the Reads To Cache Memory log parameter.

The DU bit, TSD bit, ETC bit, TMC field, and FORMAT AND LINKING field are described in 7.3.2.2.2.1. These fields shall be set as described for a bounded data counter log parameter (see 7.3.2.2.2.2) for the Reads To Cache Memory log parameter.

The PARAMETER LENGTH field is described in 7.3.2.2.2.1, and shall be set as shown in table 317 for the Reads To Cache Memory log parameter.

The NUMBER OF READS TO CACHE MEMORY field indicates the number of read commands (see table 314 in 7.3.5.1) initiated to move user data from the medium to cache memory.

The NUMBER OF READS TO CACHE MEMORY field shall not be modified as a result of any read command that contains an FUA bit (see SBC-3) set to one or an FUA_NV bit (see SBC-3) set to one.

The contents of the NUMBER OF READS TO CACHE MEMORY field shall be set to zero as part of processing a hard reset condition (see SAM-4).

7.3.5.4 Write Cache Memory Hits log parameter

The Write Cache Memory Hits log parameter has the format shown in table 318.

Table 318 — Write Cache Memory Hits log parameter

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)	PARAMETER CODE (0003h)						
1								(LSB)
2	Parameter control byte – bounded data counter log parameter (see 7.3.2.2.2.2)							
	DU	Obsolete	TSD	ETC	TMC		FORMAT AND LINKING	
3	PARAMETER LENGTH (08h)							
4	(MSB)	NUMBER OF WRITES FROM CACHE MEMORY						
•••								
11								(LSB)

The PARAMETER CODE field is described in 7.3.2.2.1, and shall be set as shown in table 318 for the Write Cache Memory Hits log parameter.

The DU bit, TSD bit, ETC bit, TMC field, and FORMAT AND LINKING field are described in 7.3.2.2.2.1. These fields shall be set as described for a bounded data counter log parameter (see 7.3.2.2.2.2) for the Writes From Cache Memory log parameter.

The PARAMETER LENGTH field is described in 7.3.2.2.2.1, and shall be set as shown in table 318 for the Write Cache Memory Hits log parameter.

The NUMBER OF WRITES FROM CACHE MEMORY field indicates the number of write commands (see table 314 in 7.3.5.1) received on an I_T nexus that resulted in:

a) user data being written to cache memory; and
b) no user data written to the medium before the write command being processed is completed.

The NUMBER OF WRITES FROM CACHE MEMORY field shall not be modified as a result of any read command that contains an FUA bit (see SBC-3) set to one or an FUA_NV bit (see SBC-3) set to one.

The contents of the NUMBER OF WRITES FROM CACHE MEMORY field shall be set to zero as part of processing a hard reset condition (see SAM-4).

7.3.5.5 Writes From Cache Memory Hits log parameter

The Writes From Cache Memory log parameter has the format shown in table 319.

Table 319 — Writes From Cache Memory log parameter

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)	PARAMETER CODE (0004h)						
1								(LSB)
2	Parameter control byte – bounded data counter log parameter (see 7.3.2.2.2.2)							
	DU	Obsolete	TSD	ETC	TMC		FORMAT AND LINKING	
3	PARAMETER LENGTH (08h)							
4	(MSB)	NUMBER OF WRITES FROM CACHE MEMORY						
•••								
11								(LSB)

The PARAMETER CODE field is described in 7.3.2.2.1, and shall be set as shown in table 319 for the Writes From Cache Memory log parameter.

The DU bit, TSD bit, ETC bit, TMC field, and FORMAT AND LINKING field are described in 7.3.2.2.2.1. These fields shall be set as described for a bounded data counter log parameter (see 7.3.2.2.2.2) for the Writes From Cache Memory log parameter.

The PARAMETER LENGTH field is described in 7.3.2.2.2.1, and shall be set as shown in table 319 for the Writes From Cache Memory log parameter.

The NUMBER OF WRITES FROM CACHE MEMORY field indicates the number of write commands (see table 314 in 7.3.5.1) initiated to move user data from cache memory to the medium.

The NUMBER OF WRITES FROM CACHE MEMORY field shall not be modified as a result of any read command that contains an FUA bit (see SBC-3) set to one or an FUA_NV bit (see SBC-3) set to one.

The contents of the NUMBER OF WRITES FROM CACHE MEMORY field shall be set to zero as part of processing a hard reset condition (see SAM-4).

7.3.5.6 Time From Last Hard Reset log parameter

The Time From Last Hard Reset log parameter has the format shown in table 320.

Table 320 — Time From Last Hard Reset log parameter

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)	PARAMETER CODE (0005h)						
1								(LSB)
2	Parameter control byte – bounded data counter log parameter (see 7.3.2.2.2.2)							
	DU	Obsolete	TSD	ETC	TMC		FORMAT AND LINKING	
3	PARAMETER LENGTH (08h)							
4	(MSB)	LAST HARD RESET INTERVALS						
•••								
11								(LSB)

The PARAMETER CODE field is described in 7.3.2.2.1, and shall be set as shown in table 320 for the Time From Last Hard Reset log parameter.

The DU bit, TSD bit, ETC bit, TMC field, and FORMAT AND LINKING field are described in 7.3.2.2.2.1. These fields shall be set as described for a bounded data counter log parameter (see 7.3.2.2.2.2) for the Time From Last Hard Reset log parameter.

The PARAMETER LENGTH field is described in 7.3.2.2.2.1, and shall be set as shown in table 320 for the Time From Last Hard Reset log parameter.

The LAST HARD RESET INTERVALS field contains the number of time intervals that have occurred since a hard reset was processed by the logical unit.

The time since a hard reset was processed by the logical unit is calculated as follows:

time = (time intervals since last hard reset × time interval)

where:

time intervals since last hard reset is the contents of the LAST HARD RESET INTERVALS field; and

time interval is the value represented in the time interval descriptor of the Time Interval log parameter (see table 321 in 7.3.5.7).

7.3.5.7 Time Interval log parameter

The Time Interval log parameter has the format shown in table 321.

Table 321 — Time Interval log parameter

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)	PARAMETER CODE						
1								(LSB)
2	Parameter control byte – binary format list log parameter (see 7.3.2.2.2.5)							
	DU	Obsolete	TSD	ETC	TMC		FORMAT AND LINKING	
3	PARAMETER LENGTH (08h)							
4	Time interval descriptor (see table 322)							
•••								
11								

The PARAMETER CODE field is described in 7.3.2.2.1. A PARAMETER CODE field set to:

a) 0003h identifies the log parameter being transferred as the Time Interval log parameter in the General Statistics and Performance log page (see 7.3.6); or
b) 0006h identifies the log parameter being transferred as the Time Interval log parameter in the Cache Memory Statistics log page (see 7.3.5).

The DU bit, TSD bit, ETC bit, TMC field, and FORMAT AND LINKING field are described in 7.3.2.2.2.1. These fields shall be set as described for a binary format list log parameter (see 7.3.2.2.2.5) for the log parameters described in this subclause.

The PARAMETER LENGTH field is described in 7.3.2.2.2.1, and shall be set as shown in table 321 for the log parameters described in this subclause.

The time interval descriptor (see table 322) contains the time interval in seconds used in various time interval fields in the:

a) Time From Last Hard Reset log parameter (see 7.3.5.6);
b) General Access Statistics and Performance log parameter (see 7.3.6.2);
c) Force Unit Access Statistics and Performance log parameter (see 7.3.6.4);
d) Group n Statistics and Performance log parameter (see 7.3.7.2); and
e) Group n Force Unit Access Statistics and Performance log parameter (see 7.3.7.3).

Table 322 — Time interval descriptor

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)	EXPONENT						
•••								
3								(LSB)
4	(MSB)	INTEGER						
•••								
7								(LSB)

The EXPONENT field contains the negative power of 10 exponent to multiply with the INTEGER field (e.g., a value of 9 represents 10^{-9})

When multiplied by the exponent, the INTEGER field contains the value that represents one time interval (e.g., a value of 5 in the INTEGER field and a value of 9 in the EXPONENT field represents a time interval of 5×10^{-9} seconds or 5 nanoseconds).

7.3.6 General Statistics and Performance log pages

7.3.6.1 Overview

Using the format shown in table 325, the General Statistics and Performance log page (page code 19h, subpage code 00h) collects statistics and performance information for all read CDBs and write CDBs based on the parameter codes listed in table 323.

Table 323 — General Statistics and Performance log page parameter codes

Parameter code	Description	Resettable or Changeable [a]	Reference	Support requirements
0000h	General Access Statistics and Performance	Reset Only	7.3.6.2	Mandatory
0001h	Idle Time	Reset Only	7.3.6.3	Mandatory
0002h	Time Interval	Never	7.3.5.7	Mandatory
0003h	Force Unit Access Statistics and Performance	Reset Only	7.3.6.4	Optional
all others	Reserved			

[a] The keywords in this column – Always, Reset Only, and Never – are defined in 7.3.2.2.2.6.

The General Statistics and Performance log page provides the following statistics and performance results associated to the addressed logical unit:

a) Statistics and Performance log parameters:
   A) number of read commands;
   B) number of write commands;
   C) number of read logical blocks transmitted by a target port;
   D) number of write logical blocks received by a target port;
   E) read command processing time;
   F) write command processing time;
   G) sum of the command weights of the read commands plus write commands;
   H) sum of the weighted command time of the read commands plus write commands;
b) Idle Time log parameter:
   A) idle time;
c) Time Interval log parameter:
   A) time interval;
   and
d) Force Unit Access Statistics and Performance log parameters:
   A) number of read commands with the FUA bit (see SBC-3) set to one;
   B) number of write commands with the FUA bit set to one;
   C) number of read commands with the FUA_NV bit (see SBC-3) set to one;
   D) number of write commands with the FUA_NV bit set to one; and
   E) read command with the FUA bit set to one processing intervals;
   F) write command with the FUA bit set to one processing intervals;
   G) read command with the FUA_NV bit set to one processing intervals; and

H) write command with the FUA_NV bit set to one processing intervals.

In the General Statistics and Performance log page and the Group Statistics and Performance (n) log pages (see 7.3.7), read commands and write commands are those shown in table 324.

Table 324 — Statistics and Performance log pages commands

Read commands [a]	Write commands [a]
READ(6) READ(10) READ(12) READ(16) READ(32)	UNMAP WRITE(6) WRITE(10) WRITE(12) WRITE(16) WRITE(32) WRITE AND VERIFY(10) WRITE AND VERIFY(12) WRITE AND VERIFY(16) WRITE AND VERIFY(32)
[a] See SBC-3.	

The General Statistics and Performance log page has the format shown in table 325.

Table 325 — General Statistics and Performance log page

Bit / Byte	7	6	5	4	3	2	1	0
0	DS	SPF (0b)	PAGE CODE (19h)					
1	SUBPAGE CODE (00h)							
2	(MSB)	PAGE LENGTH (n-3)						
3								(LSB)
	General statistics and performance log parameters							
4 ••• 	General statistics and performance log parameter (see table 323) [first]							
	⋮							
 ••• n	General statistics and performance log parameter (see table 323) [last]							

The DS bit, SPF bit, PAGE CODE field, SUBPAGE CODE field, and PAGE LENGTH field are described in 7.3.2. The SPF bit, PAGE CODE field, and SUBPAGE CODE field shall be set as shown in table 325 for the General Statistics and Performance log page.

The contents of each general statistics and performance log parameter depends on the value in its PARAMETER CODE field (see table 323).

7.3.6.2 General Access Statistics and Performance log parameter

The General Access Statistics and Performance log parameter has the format shown in table 326.

Table 326 — General Access Statistics and Performance log parameter

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)			PARAMETER CODE (0001h)				
1								(LSB)
2	Parameter control byte – unbounded data counter log parameter (see 7.3.2.2.2.3)							
	DU	Obsolete	TSD	ETC	TMC		FORMAT AND LINKING	
3	PARAMETER LENGTH (40h)							
4	(MSB)			NUMBER OF READ COMMANDS				
•••								
11								(LSB)
12	(MSB)			NUMBER OF WRITE COMMANDS				
•••								
19								(LSB)
20	(MSB)			NUMBER OF LOGICAL BLOCKS RECEIVED				
•••								
27								(LSB)
28	(MSB)			NUMBER OF LOGICAL BLOCKS TRANSMITTED				
•••								
35								(LSB)
36	(MSB)			READ COMMAND PROCESSING INTERVALS				
•••								
43								(LSB)
44	(MSB)			WRITE COMMAND PROCESSING INTERVALS				
•••								
51								(LSB)
52	(MSB)			WEIGHTED NUMBER OF READ COMMANDS PLUS WRITE COMMANDS				
•••								
59								(LSB)
60	(MSB)			WEIGHTED READ COMMAND PROCESSING PLUS WRITE COMMAND PROCESSING				
•••								
67								(LSB)

The PARAMETER CODE field is described in 7.3.2.2.1, and shall be set as shown in table 326 for the General Access Statistics and Performance log parameter.

The DU bit, TSD bit, ETC bit, TMC field, and FORMAT AND LINKING field are described in 7.3.2.2.2.1. These fields shall be set as described for an unbounded data counter log parameter (see 7.3.2.2.2.3) for the General Access Statistics and Performance log parameter.

The PARAMETER LENGTH field is described in 7.3.2.2.2.1, and shall be set as shown in table 326 for the General Access Statistics and Performance log parameter.

The NUMBER OF READ COMMANDS field contains the number of read commands (see table 324 in 7.3.6.1) received by the logical unit.

The NUMBER OF WRITE COMMANDS field contains the number of write commands (see table 324 in 7.3.6.1) received by the logical unit.

The NUMBER OF LOGICAL BLOCKS RECEIVED field contains the number of logical blocks received by any SCSI target port for the logical unit as a result of write commands (see table 324 in 7.3.6.1).

The NUMBER OF LOGICAL BLOCKS TRANSMITTED field contains the number of logical blocks transmitted by any SCSI target port for the logical unit as a result of read commands (see table 324 in 7.3.6.1).

The READ COMMAND PROCESSING INTERVALS field contains the cumulative number of time intervals (see 7.3.5.7) spent by the logical unit processing read commands (see table 324 in 7.3.6.1).

The WRITE COMMAND PROCESSING INTERVALS field contains the cumulative number of time intervals (see 7.3.5.7) spent by the logical unit processing write commands (see table 324 in 7.3.6.1).

If command priority is supported (see SAM-4), then the WEIGHTED NUMBER OF READ COMMANDS PLUS WRITE COMMANDS field contains the cumulative command weight of the read commands and write commands (see table 324 in 7.3.6.1) processed by the logical unit.

Command weight is calculated as follows:

command weight = (360 360 ÷ command priority)

where:

command priority is as defined in SAM-4. However, if the computed command priority is zero, then the command priority shall be set to seven (i.e., a mid-range command priority value).

If command priority is not supported, then the WEIGHTED NUMBER OF READ COMMANDS PLUS WRITE COMMANDS field shall be set to zero.

If command priority is supported (see SAM-4), then the WEIGHTED READ COMMAND PROCESSING PLUS WRITE COMMAND PROCESSING field contains the cumulative weighted command time of the time intervals (see 7.3.5.7) spent processing read commands and write commands (see table 324 in 7.3.6.1) by the logical unit.

Weighted command time is calculated as follows:

weighted command time = (time increments processing the command × time interval)
× (360 360 ÷ command priority)

where:

time increments processing a command is the number of time intervals from the time the task manager places the command into a task set until the device server sends a SCSI transport protocol service response for the command;

time interval is the value represented in the TIME INTERVAL DESCRIPTOR field of the Time Interval log parameter (see 7.3.5.7), and

command priority is as defined in SAM-4. However, if the computed command priority is zero, then the command priority time shall be set to seven (i.e., a mid-range command priority value).

If command priority is not supported, then the WEIGHTED READ COMMAND PROCESSING PLUS WRITE COMMAND PROCESSING field shall be set to zero.

7.3.6.3 Idle Time log parameter

The Idle Time log parameter has the format shown in table 327.

Table 327 — Idle Time log parameter

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)	PARAMETER CODE (0002h)						
1								(LSB)
2	Parameter control byte – unbounded data counter log parameter (see 7.3.2.2.2.3)							
	DU	Obsolete	TSD	ETC	TMC		FORMAT AND LINKING	
3	PARAMETER LENGTH (08h)							
4	(MSB)	IDLE TIME INTERVALS						
•••								
11								(LSB)

The PARAMETER CODE field is described in 7.3.2.2.1, and shall be set as shown in table 327 for the Idle Time log parameter.

The DU bit, TSD bit, ETC bit, TMC field, and FORMAT AND LINKING field are described in 7.3.2.2.2.1. These fields shall be set as described for an unbounded data counter log parameter (see 7.3.2.2.2.3) for the Idle Time log parameter.

The PARAMETER LENGTH field is described in 7.3.2.2.2.1, and shall be set as shown in table 327 for the Idle Time log parameter.

The IDLE TIME INTERVALS field contains the cumulative number of idle times spent when there are no commands in the task set and there are no commands being processed by the logical unit.

Idle time is calculated as follows:

idle time = (time increments not processing commands × time interval)

where:

time increments not processing commands is the number of time intervals when there are no commands in the task set and the device server has sent a SCSI transport protocol service response for all commands being processed (i.e., there are no commands to be processed or being processed); and

time interval is the value represented in the time interval descriptor of the Time Interval log parameter (see 7.3.5.7).

7.3.6.4 Force Unit Access Statistics and Performance log parameter

The Force Unit Access Statistics and Performance log parameter has the format shown in table 328.

Table 328 — Force Unit Access Statistics and Performance log parameter

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)			PARAMETER CODE (0004h)				
1								(LSB)
2	Parameter control byte – unbounded data counter log parameter (see 7.3.2.2.2.3)							
	DU	Obsolete	TSD	ETC	TMC		FORMAT AND LINKING	
3	PARAMETER LENGTH (40h)							
4	(MSB)			NUMBER OF READ FUA COMMANDS				
•••								
11								(LSB)
12	(MSB)			NUMBER OF WRITE FUA COMMANDS				
•••								
19								(LSB)
20	(MSB)			NUMBER OF READ FUA_NV COMMANDS				
•••								
27								(LSB)
28	(MSB)			NUMBER OF WRITE FUA_NV COMMANDS				
•••								
35								(LSB)
36	(MSB)			READ FUA COMMAND PROCESSING INTERVALS				
•••								
43								(LSB)
44	(MSB)			WRITE FUA COMMAND PROCESSING INTERVALS				
•••								
51								(LSB)
52	(MSB)			READ FUA_NV COMMAND PROCESSING INTERVALS				
•••								
59								(LSB)
60	(MSB)			WRITE FUA_NV COMMAND PROCESSING INTERVALS				
•••								
67								(LSB)

The PARAMETER CODE field is described in 7.3.2.2.1, and shall be set as shown in table 328 for the Force Unit Access Statistics and Performance log parameter.

The DU bit, TSD bit, ETC bit, TMC field, and FORMAT AND LINKING field are described in 7.3.2.2.2.1. These fields shall be set as described for an unbounded data counter log parameter (see 7.3.2.2.2.3) for the Force Unit Access Statistics and Performance log parameter.

The PARAMETER LENGTH field is described in 7.3.2.2.2.1, and shall be set as shown in table 328 for the Force Unit Access Statistics and Performance log parameter.

The NUMBER OF READ FUA COMMANDS field contains the number of read commands (see table 324 in 7.3.6.1) with the FUA bit (see SBC-3) set to one received by the logical unit.

The NUMBER OF WRITE FUA COMMANDS field contains the number of write commands (see table 324 in 7.3.6.1) with the FUA bit (see SBC-3) set to one received by the logical unit.

The NUMBER OF READ FUA_NV COMMANDS field contains the number of read commands (see table 324 in 7.3.6.1) with the FUA_NV bit (see SBC-3) set to one received by the logical unit.

The NUMBER OF WRITE FUA_NV COMMANDS field contains the number of write commands (see table 324 in 7.3.6.1) with the FUA_NV bit (see SBC-3) set to one received by the logical unit.

The READ FUA COMMAND PROCESSING INTERVALS field contains the cumulative number of time intervals (see 7.3.5.7) spent by the logical unit processing read commands (see table 324 in 7.3.6.1) with the FUA bit (see SBC-3) set to one.

The WRITE FUA COMMAND PROCESSING INTERVALS field contains the cumulative number of time intervals (see 7.3.5.7) spent by the logical unit processing write commands (see table 324 in 7.3.6.1) with the FUA bit (see SBC-3) set to one.

The READ FUA_NV COMMAND PROCESSING INTERVALS field contains the cumulative number of time intervals (see 7.3.5.7) spent by the logical unit processing read commands (see table 324 in 7.3.6.1) with the FUA_NV bit (see SBC-3) set to one.

The WRITE FUA_NV COMMAND PROCESSING INTERVALS field contains the cumulative number of time intervals (see 7.3.5.7) spent by the logical unit processing write commands (see table 324 in 7.3.6.1) with the FUA_NV bit (see SBC-3) set to one.

7.3.7 Group Statistics and Performance (n) log pages

7.3.7.1 Overview

Using the format shown in table 330, Group Statistics and Performance (n) log pages (page code 19h, subpage codes 01h to 1Fh) collect statistics and performance information for the group number specified in a read CDB or a write CDB based on the parameter codes listed in table 329.

Table 329 — Group Statistics and Performance log page parameter codes

Parameter code	Description	Resettable or Changeable [a]	Reference	Support requirements
0001h	Group n Statistics and Performance	Reset Only	7.3.7.2	Mandatory
0004h	Group n Force Unit Access Statistics and Performance	Reset Only	7.3.7.3	Optional
all others	Reserved			

[a] The keywords in this column – Always, Reset Only, and Never – are defined in 7.3.2.2.2.6.

The Group Statistics and Performance (n) log pages provide the following statistics and performance results associated to the addressed logical unit and the GROUP NUMBER field:

a) Statistics and Performance log parameters:
   A) number of read commands;
   B) number of write commands;

C) number of read logical blocks transmitted by a target port;
D) number of write logical blocks received by a target port;
E) read command processing time;
F) write command processing time;
and

b) Force Unit Access Statistics and Performance log parameters:
A) number of read commands with the FUA bit (see SBC-3) set to one;
B) number of write commands with the FUA bit set to one;
C) number of read commands with the FUA_NV bit (see SBC-3) set to one;
D) number of write commands with the FUA_NV bit set to one;
E) read command with the FUA bit set to one processing intervals;
F) write command with the FUA bit set to one processing intervals;
G) read command with the FUA_NV bit set to one processing intervals; and
H) write command with the FUA_NV bit set to one processing intervals.

In the Group Statistics and Performance (n) log pages, read commands and write commands are those shown in table 324 (see 7.3.6.1).

The Group Statistics and Performance (n) log pages provide logging of statistics and performance of read and write operations based on group numbers. There are 31 Group Statistics and Performance (n) log pages one for each group number. The statistics and performance information associated with each group number is collected in the corresponding Group Statistics and Performance (n) log page (e.g., operations associated with group number 16 are logged in the Group Statistics and Performance (16) log page).

Each Group Statistics and Performance (n) log page has the format shown in table 330.

Table 330 — Group Statistics and Performance (n) log page

Bit / Byte	7	6	5	4	3	2	1	0
0	DS	SPF (1b)	PAGE CODE (19h)					
1	SUBPAGE CODE (01h to 1Fh) (see table 331) [a]							
2	(MSB)	PAGE LENGTH (n-3)						
3								(LSB)
	Group statistics and performance log parameters							
4	General statistics and performance log parameter (see table 329) [first]							
•••								
	⋮							
	General statistics and performance log parameter (see table 329) [last]							
•••								
n								

[a] The log parameter associated with the specific group number as specified by the value of n is collected in the corresponding log parameter (e.g., the count of read commands with the GROUP NUMBER field set to 9 is logged in the GROUP N NUMBER OF READ COMMANDS field in the Group n Statistics and Performance log parameter of the Group Statistics and Performance (9) log page).

The DS bit, SPF bit, PAGE CODE field, SUBPAGE CODE field, and PAGE LENGTH field are described in 7.3.2. The SPF bit, PAGE CODE field, and SUBPAGE CODE field shall be set as shown in table 330 for the Group Statistics and Performance (n) log pages.

The SUBPAGE CODE field (see table 331) associates the Group Statistics and Performance (n) log page being transferred with the contents of the GROUP NUMBER field in a read command CDB or write command CDB (see table 324 in 7.3.6.1 and SBC-3).

Table 331 — Group Statistics and Performance (n) subpage codes (part 1 of 2)

Subpage code	Log page name [a]	Group number [b]
01h	Group Statistics and Performance (1)	00001b
02h	Group Statistics and Performance (2)	00010b
03h	Group Statistics and Performance (3)	00011b
04h	Group Statistics and Performance (4)	00100b
05h	Group Statistics and Performance (5)	00101b
06h	Group Statistics and Performance (6)	00110b
07h	Group Statistics and Performance (7)	00111b
08h	Group Statistics and Performance (8)	01000b
09h	Group Statistics and Performance (9)	01001b
0Ah	Group Statistics and Performance (10)	01010b
0Bh	Group Statistics and Performance (11)	01011b
0Ch	Group Statistics and Performance (12)	01100b
0Dh	Group Statistics and Performance (13)	01101b
0Eh	Group Statistics and Performance (14)	01110b
0Fh	Group Statistics and Performance (15)	01111b
10h	Group Statistics and Performance (16)	10000b
11h	Group Statistics and Performance (17)	10001b
12h	Group Statistics and Performance (18)	10010b
13h	Group Statistics and Performance (19)	10011b
14h	Group Statistics and Performance (20)	10100b
15h	Group Statistics and Performance (21)	10101b

[a] The statistics and performance information associated with a group number is collected in the corresponding Group Statistics and Performance (n) log page (e.g., operations associated with group number 10000b are logged in the Group Statistics and Performance (16) log page).

[b] The GROUP NUMBER field is from the read command CDB or the write command CDB (see table 324 in 7.3.6.1 and SBC-3).

Table 331 — Group Statistics and Performance (n) subpage codes (part 2 of 2)

Subpage code	Log page name [a]	Group number [b]
16h	Group Statistics and Performance (22)	10110b
17h	Group Statistics and Performance (23)	10111b
18h	Group Statistics and Performance (24)	11000b
19h	Group Statistics and Performance (25)	11001b
1Ah	Group Statistics and Performance (26)	11010b
1Bh	Group Statistics and Performance (27)	11011b
1Ch	Group Statistics and Performance (28)	11100b
1Dh	Group Statistics and Performance (29)	11101b
1Eh	Group Statistics and Performance (30)	11110b
1Fh	Group Statistics and Performance (31)	11111b

[a] The statistics and performance information associated with a group number is collected in the corresponding Group Statistics and Performance (n) log page (e.g., operations associated with group number 10000b are logged in the Group Statistics and Performance (16) log page).

[b] The GROUP NUMBER field is from the read command CDB or the write command CDB (see table 324 in 7.3.6.1 and SBC-3).

The contents of each group statistics and performance log parameter depends on the value in its PARAMETER CODE field (see table 329).

7.3.7.2 Group n Statistics and Performance log parameter

The Group n Statistics and Performance log parameter has the format shown in table 332.

Table 332 — Group n Statistics and Performance log parameter

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)			PARAMETER CODE (0001h)				
1								(LSB)
2	Parameter control byte – unbounded data counter log parameter (see 7.3.2.2.2.3)							
	DU	Obsolete	TSD	ETC	TMC		FORMAT AND LINKING	
3	PARAMETER LENGTH (30h)							
4	(MSB)			GROUP N NUMBER OF READ COMMANDS				
•••								
11								(LSB)
12	(MSB)			GROUP N NUMBER OF WRITE COMMANDS				
•••								
19								(LSB)
20	(MSB)			GROUP N NUMBER OF LOGICAL BLOCKS RECEIVED				
•••								
27								(LSB)
28	(MSB)			GROUP N NUMBER OF LOGICAL BLOCKS TRANSMITTED				
•••								
35								(LSB)
36	(MSB)			GROUP N READ COMMAND PROCESSING INTERVALS				
•••								
43								(LSB)
44	(MSB)			GROUP N WRITE COMMAND PROCESSING INTERVALS				
•••								
51								(LSB)

The PARAMETER CODE field is described in 7.3.2.2.1, and shall be set as shown in table 332 for the Group n Statistics and Performance log parameter.

The DU bit, TSD bit, ETC bit, TMC field, and FORMAT AND LINKING field are described in 7.3.2.2.2.1. These fields shall be set as described for an unbounded data counter log parameter (see 7.3.2.2.2.3) for the Group n Statistics and Performance log parameter.

The PARAMETER LENGTH field is described in 7.3.2.2.2.1, and shall be set as shown in table 332 for the Group n Statistics and Performance log parameter.

The GROUP N NUMBER OF READ COMMANDS field contains the number of read commands (see table 324 in 7.3.6.1) received by the logical unit.

The GROUP N NUMBER OF WRITE COMMANDS field contains the number of write commands (see table 324 in 7.3.6.1) received by the logical unit.

The GROUP N NUMBER OF LOGICAL BLOCKS RECEIVED field contains the number of logical blocks received by any SCSI target port for the logical unit as a result of write commands (see table 324 in 7.3.6.1).

The GROUP N NUMBER OF LOGICAL BLOCKS TRANSMITTED field contains the number of logical blocks transmitted by any SCSI target port for the logical unit as a result of read commands (see table 324 in 7.3.6.1).

The GROUP N READ COMMAND PROCESSING INTERVALS field contains the cumulative number of time intervals spent by the logical unit processing read commands (see table 324 in 7.3.6.1). Time intervals are defined in the Time Interval log parameter (see 7.3.5.7) as returned in the General Statistics and Performance log page (see 7.3.6).

The GROUP N WRITE COMMAND PROCESSING INTERVALS field contains the cumulative number of time intervals spent by the logical unit processing write commands (see table 324 in 7.3.6.1). Time intervals are defined in the Time Interval log parameter (see 7.3.5.7) as returned in the General Statistics and Performance log page (see 7.3.6).

7.3.7.3 Group n Force Unit Access Statistics and Performance log parameter

The Group n Force Unit Access Statistics and Performance log parameter has the format shown in table 333.

Table 333 — Group n Force Unit Access Statistics and Performance log parameter

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)			PARAMETER CODE (0004h)				
1								(LSB)
2	Parameter control byte – unbounded data counter log parameter (see 7.3.2.2.2.3)							
	DU	Obsolete	TSD	ETC	TMC		FORMAT AND LINKING	
3	PARAMETER LENGTH (40h)							
4	(MSB)			GROUP N NUMBER OF READ FUA COMMANDS				
•••								
11								(LSB)
12	(MSB)			GROUP N NUMBER OF WRITE FUA COMMANDS				
•••								
19								(LSB)
20	(MSB)			GROUP N NUMBER OF READ FUA_NV COMMANDS				
•••								
27								(LSB)
28	(MSB)			GROUP N NUMBER OF WRITE FUA_NV COMMANDS				
•••								
35								(LSB)
36	(MSB)			GROUP N READ FUA COMMAND PROCESSING INTERVALS				
•••								
43								(LSB)
44	(MSB)			GROUP N WRITE FUA COMMAND PROCESSING INTERVALS				
•••								
51								(LSB)
52	(MSB)			GROUP N READ FUA_NV COMMAND PROCESSING INTERVALS				
•••								
59								(LSB)
60	(MSB)			GROUP N WRITE FUA_NV COMMAND PROCESSING INTERVALS				
•••								
67								(LSB)

The PARAMETER CODE field is described in 7.3.2.2.1, and shall be set as shown in table 333 for the Group n Force Unit Access Statistics and Performance log parameter.

The DU bit, TSD bit, ETC bit, TMC field, and FORMAT AND LINKING field are described in 7.3.2.2.2.1. These fields shall be set as described for an unbounded data counter log parameter (see 7.3.2.2.2.3) for the Group n Force Unit Access Statistics and Performance log parameter.

The PARAMETER LENGTH field is described in 7.3.2.2.2.1, and shall be set as shown in table 333 for the Group n Force Unit Access Statistics and Performance log parameter.

The GROUP N NUMBER OF READ FUA COMMANDS field contains the number of read commands (see table 324 in 7.3.6.1) with the FUA bit (see SBC-3) set to one received by the logical unit.

The GROUP N NUMBER OF WRITE FUA COMMANDS field contains the number of write commands (see table 324 in 7.3.6.1) with the FUA bit (see SBC-3) set to one received by the logical unit.

The GROUP N NUMBER OF READ FUA_NV COMMANDS field contains the number of read commands (see table 324 in 7.3.6.1) with the FUA_NV bit (see SBC-3) set to one received by the logical unit.

The GROUP N NUMBER OF WRITE FUA_NV COMMANDS field contains the number of write commands (see table 324 in 7.3.6.1) with the FUA_NV bit (see SBC-3) set to one received by the logical unit.

The GROUP N READ FUA COMMAND PROCESSING INTERVALS field contains the cumulative number of time intervals spent by the logical unit processing read commands (see table 324 in 7.3.6.1) with the FUA bit (see SBC-3) set to one. Time intervals are defined in the Time Interval log parameter (see 7.3.5.7) as returned in the General Statistics and Performance log page (see 7.3.6).

The GROUP N WRITE FUA COMMAND PROCESSING INTERVALS field contains the cumulative number of time intervals spent by the logical unit processing write commands (see table 324 in 7.3.6.1) with the FUA bit (see SBC-3) set to one. Time intervals are defined in the Time Interval log parameter (see 7.3.5.7) as returned in the General Statistics and Performance log page (see 7.3.6).

The GROUP N READ FUA_NV COMMAND PROCESSING INTERVALS field contains the cumulative number of time intervals spent by the logical unit processing read commands (see table 324 in 7.3.6.1) with the FUA_NV bit (see SBC-3) set to one. Time intervals are defined in the Time Interval log parameter (see 7.3.5.7) as returned in the General Statistics and Performance log page (see 7.3.6).

The GROUP N WRITE FUA_NV COMMAND PROCESSING INTERVALS field contains the cumulative number of time intervals spent by the logical unit processing write commands (see table 324 in 7.3.6.1) with the FUA_NV bit (see SBC-3) set to one. Time intervals are defined in the Time Interval log parameter (see 7.3.5.7) as returned in the General Statistics and Performance log page (see 7.3.6).

7.3.8 Informational Exceptions log page

7.3.8.1 Overview

Using the format shown in table 335, the Informational Exceptions log page (page code 2Fh) reports details about informational exceptions identified by the parameter codes listed in table 334.

Table 334 — Informational Exceptions log page parameter codes

Parameter code	Description	Resettable or Changeable [a]	Reference	Support requirements
0000h	Informational Exceptions General	Reset Only	7.3.8.2	Mandatory
0001h to FFFFh	Vendor specific			

[a] The keywords in this column – Always, Reset Only, and Never – are defined in 7.3.2.2.2.6.

The Informational Exceptions log page has the format shown in table 335.

Table 335 — Informational Exceptions log page

Bit / Byte	7	6	5	4	3	2	1	0
0	DS	SPF (0b)	PAGE CODE (2Fh)					
1	SUBPAGE CODE (00h)							
2	(MSB)	PAGE LENGTH (n-3)						
3								(LSB)
	Informational exceptions log parameters							
4 •••	Informational exceptions log parameter (see table 334) [first]							
	⋮							
••• n	Informational exceptions log parameter (see table 334) [last]							

The DS bit, SPF bit, PAGE CODE field, SUBPAGE CODE field, and PAGE LENGTH field are described in 7.3.2. The SPF bit, PAGE CODE field, and SUBPAGE CODE field shall be set as shown in table 335 for the Informational Exceptions log page.

The contents of each informational exceptions log parameter depends on the value in its PARAMETER CODE field (see table 334).

7.3.8.2 Informational Exceptions General log parameter

The Informational Exceptions General log parameter has the format shown in table 336.

Table 336 — Informational Exceptions General log parameter

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)	PARAMETER CODE (0000h)						
1								(LSB)
2	Parameter control byte – binary format list log parameter (see 7.3.2.2.2.5)							
	DU	Obsolete	TSD	ETC	TMC		FORMAT AND LINKING	
3	PARAMETER LENGTH (n-3)							
4	INFORMATIONAL EXCEPTION ADDITIONAL SENSE CODE							
5	INFORMATIONAL EXCEPTION ADDITIONAL SENSE CODE QUALIFIER							
6	MOST RECENT TEMPERATURE READING							
7 ••• n	Vendor specific							

The PARAMETER CODE field is described in 7.3.2.2.1, and shall be set as shown in table 336 for the Informational Exceptions General log parameter.

The DU bit, TSD bit, ETC bit, TMC field, and FORMAT AND LINKING field are described in 7.3.2.2.2.1. These fields shall be set as described for a binary format list log parameter (see 7.3.2.2.2.5) for the Informational Exceptions General log parameter.

The PARAMETER LENGTH field is described in 7.3.2.2.2.1. The parameter length shall be at least 04h.

If the INFORMATIONAL EXCEPTION ADDITIONAL SENSE CODE field contains zero, no informational exception condition exists and contents of the INFORMATIONAL EXCEPTION ADDITIONAL SENSE CODE QUALIFIER field are unspecified. If the INFORMATIONAL EXCEPTION ADDITIONAL SENSE CODE field contains any value other than zero, an informational exception condition exists that has an additional sense code indicated by INFORMATIONAL EXCEPTION ADDITIONAL SENSE CODE field and an ADDITIONAL SENSE CODE QUALIFIER indicated by the INFORMATIONAL EXCEPTION ADDITIONAL SENSE CODE QUALIFIER field.

The MOST RECENT TEMPERATURE READING field indicates the temperature in degrees Celsius of the SCSI target device at the time the LOG SENSE command is performed. Temperatures equal to or less than zero degrees Celsius shall be indicated by a value of zero. If the device server is unable to detect a valid temperature because of a sensor failure or other condition, the value returned shall be FFh. The temperature should be reported with an accuracy of plus or minus three Celsius degrees while the device is operating at a steady state within the environmental limits specified for the device.

7.3.9 Last *n* Deferred Errors or Asynchronous Events log page

7.3.9.1 Overview

Using the format shown in table 338, the Last *n* Deferred Errors or Asynchronous Events log page (page code 0Bh) provides one or more Deferred Error or Asynchronous Event log parameters (see 7.3.9.2). The number of Deferred Error or Asynchronous Event log parameters supported (i.e., *n*) is vendor specific. The parameter codes for the Last *n* Deferred Errors or Asynchronous Events log page are listed in table 337.

Table 337 — Last *n* Deferred Errors or Asynchronous Events log page parameter codes

Parameter code	Description	Resettable or Changeable [a]	Reference	Support requirements
0000h	Deferred Error or Asynchronous Event	Reset Only	7.3.9.2	Mandatory
0001h to FFFFh				Optional

[a] The keywords in this column – Always, Reset Only, and Never – are defined in 7.3.2.2.2.6.

The Last *n* Deferred Errors or Asynchronous Events log page has the format shown in table 338.

Table 338 — Last *n* Deferred Errors or Asynchronous Events log page

Bit / Byte	7	6	5	4	3	2	1	0
0	DS	SPF (0b)	PAGE CODE (0Bh)					
1	SUBPAGE CODE (00h)							
2	(MSB)	PAGE LENGTH (n-3)						
3								(LSB)
	Deferred error or asynchronous event log parameters							
4 •••	Deferred error or asynchronous event log parameter (see 7.3.9.2) [first]							
	⋮							
••• n	Deferred error or asynchronous event log parameter (see 7.3.9.2) [last]							

The DS bit, SPF bit, PAGE CODE field, SUBPAGE CODE field, and PAGE LENGTH field are described in 7.3.2. The SPF bit, PAGE CODE field, and SUBPAGE CODE field shall be set as shown in table 338 for the Last *n* Deferred Errors or Asynchronous Events log page.

The contents of each deferred error or asynchronous event log parameter are described in 7.3.9.2. The device server shall assign parameter codes to log parameters as follows:

a) if the vendor specific number of supported deferred error or asynchronous event log parameters has not been exceeded, then the parameter code in each log parameter shall indicate the relative time at which the deferred error or asynchronous event occurred. A higher parameter code indicates that the deferred error or asynchronous event occurred at a more recent time; or
b) if the vendor specific number of supported deferred error or asynchronous event log parameters has been exceeded, then:
   A) the log parameter with the oldest data shall be overwritten with the newest data (i.e., after the highest supported parameter code is used, reporting wraps so that the next deferred error or asynchronous event is reported in the log parameter with parameter code zero); and
   B) if the RLEC bit is set to one in the Control mode page (see 7.5.7) and a LOG SELECT command that transfers the Last *n* Deferred Errors or Asynchronous Events log page completes without error, except for the parameter code being at its maximum value, then the command shall be terminated with CHECK CONDITION status, with the sense key set to RECOVERED ERROR, and the additional sense code set to LOG LIST CODES EXHAUSTED.

7.3.9.2 Deferred Error or Asynchronous Event log parameters

Each Deferred Error or Asynchronous Event log parameter has the format shown in table 339.

Table 339 — Deferred Error or Asynchronous Event log parameter

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)	PARAMETER CODE (see table 337)						
1								(LSB)
2	Parameter control byte – binary format list log parameter (see 7.3.2.2.2.5)							
	DU	Obsolete	TSD	ETC	TMC		FORMAT AND LINKING	
3	PARAMETER LENGTH (n-3)							
4	SENSE DATA (see 4.5)							
•••								
n								

The PARAMETER CODE field is described in 7.3.2.2.1, and shall be set as shown in table 337 for the Deferred Error or Asynchronous Event log parameter.

The DU bit, TSD bit, ETC bit, TMC field, and FORMAT AND LINKING field are described in 7.3.2.2.2.1. These fields shall be set as described for a binary format list log parameter (see 7.3.2.2.2.5) for the Deferred Error or Asynchronous Event log parameter.

The PARAMETER LENGTH field is described in 7.3.2.2.2.1.

The SENSE DATA field contains the sense data (see 4.5) for a deferred error or asynchronous event that has occurred.

7.3.10 Last *n* Error Events log page

7.3.10.1 Overview

Using the format shown in table 341, the Last *n* Error Events log page (page code 07h) provides one or more Error Event log parameters (see 7.3.10.2). The number of these Error Event log parameters supported (i.e., *n*) is vendor specific. The parameter codes for the Last *n* Error Events log page are listed in table 340.

Table 340 — Last *n* Error Events log page parameter codes

Parameter code	Description	Resettable or Changeable [a]	Reference	Support requirements
0000h	Error Event	Reset Only	7.3.10.2	Mandatory
0001h to FFFFh				Optional

[a] The keywords in this column – Always, Reset Only, and Never – are defined in 7.3.2.2.2.6.

The Last *n* Error Events log page has the format shown in table 341.

Table 341 — Last *n* Error Events log page

Bit / Byte	7	6	5	4	3	2	1	0
0	DS	SPF (0b)	PAGE CODE (07h)					
1	SUBPAGE CODE (00h)							
2	(MSB)	PAGE LENGTH (n-3)						
3								(LSB)
	Error event log parameters							
4 •••	Error event log parameter (see 7.3.10.2) [first]							
	⋮							
••• n	Error event log parameter (see 7.3.10.2) [last]							

The DS bit, SPF bit, PAGE CODE field, SUBPAGE CODE field, and PAGE LENGTH field are described in 7.3.2. The SPF bit, PAGE CODE field, and SUBPAGE CODE field shall be set as shown in table 341 for the Last *n* Error Events log page.

The contents of each error event log parameter is described are 7.3.10.2. The device server shall assign parameter codes to log parameters as follows:

a) if the vendor specific number of supported error event log parameters has not been exceeded, then the parameter code in each log parameter shall indicate the relative time at which the error event occurred. A higher parameter code indicates that the error event occurred later in time; or
b) if the vendor specific number of supported error event log parameters has been exceeded, then:
   A) the log parameter with the oldest data shall be overwritten with the newest data (i.e., after the highest supported parameter code is used, reporting wraps so that the next error event is reported in the log parameter with parameter code zero); and
   B) if the RLEC bit is set to one in the Control mode page (see 7.5.7) and a LOG SELECT command that transfers the Last *n* Error Events log page completes without error, except for the parameter code being at its maximum value, then the command shall be terminated with CHECK CONDITION status, with the sense key set to RECOVERED ERROR, and the additional sense code set to LOG LIST CODES EXHAUSTED.

7.3.10.2 Error Event log parameters

Each Error Event log parameter has the format shown in table 342.

Table 342 — Error Event log parameter

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)	PARAMETER CODE (see table 340)						
1								(LSB)
2	Parameter control byte – ASCII format list log parameter (see 7.3.2.2.2.4)							
	DU	Obsolete	TSD	ETC	TMC		FORMAT AND LINKING	
3	PARAMETER LENGTH (n-3)							
4	ERROR EVENT DATA							
•••								
n								

The PARAMETER CODE field is described in 7.3.2.2.1, and shall be set as shown in table 340 for the Error Event log parameter.

The DU bit, TSD bit, ETC bit, TMC field, and FORMAT AND LINKING field are described in 7.3.2.2.2.1. These fields shall be set as described for a ASCII format list log parameter (see 7.3.2.2.2.4) for the Error Event log parameter.

The PARAMETER LENGTH field is described in 7.3.2.2.2.1.

The ERROR EVENT DATA field contains ASCII data (see 4.4.1) that may describe the error event. The contents of the ERROR EVENT DATA field are not defined by this standard.

7.3.11 Non-Medium Error log page

7.3.11.1 Overview

Using the format shown in table 344, the Non-Medium Error log page (page code 06h) provides for counting the occurrences of recoverable error events other than read (see 7.3.14), read reverse (see 7.3.15), verify (see 7.3.22), or write (see 7.3.23) failures. No discrimination among the various types of events is provided. Vendor specific discrimination may be provided through the vendor specific parameter codes. The parameter codes for the Non-Medium Error log page are listed in table 343.

Table 343 — Non-Medium Error log page parameter codes

Parameter code	Description	Resettable or Changeable [a]	Reference	Support requirements
0000h	Non-medium error count	Reset Only	7.3.11.2	Mandatory
8000h to FFFFh	Vendor specific error counts			
all others	Reserved			

[a] The keywords in this column – Always, Reset Only, and Never – are defined in 7.3.2.2.2.6.

The Non-Medium Error log page has the format shown in table 344.

Table 344 — Non-Medium Error log page

Bit / Byte	7	6	5	4	3	2	1	0
0	DS	SPF (0b)	PAGE CODE (06h)					
1	SUBPAGE CODE (00h)							
2	(MSB)	PAGE LENGTH (n-3)						
3								(LSB)
	Non-medium error log parameters							
4	Non-medium error log parameter (see table 343) [first]							
•••								
	⋮							
	Non-medium error log parameter (see table 343) [last]							
•••								
n								

The DS bit, SPF bit, PAGE CODE field, SUBPAGE CODE field, and PAGE LENGTH field are described in 7.3.2. The SPF bit, PAGE CODE field, and SUBPAGE CODE field shall be set as shown in table 344 for the Non-Medium Error log page.

The contents of each non-medium error log parameter depends on the value in its PARAMETER CODE field (see table 343).

7.3.11.2 Non-Medium Error Count log parameter

The Non-Medium Error Count log parameter has the format shown in table 345.

Table 345 — Non-Medium Error Count log parameter

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)	PARAMETER CODE (0000h)						
1								(LSB)
2	Parameter control byte – bounded data counter log parameter (see 7.3.2.2.2.2)							
	DU	Obsolete	TSD	ETC	TMC		FORMAT AND LINKING	
3	PARAMETER LENGTH (n-3)							
4	(MSB)	NON-MEDIUM ERROR COUNT						
•••								
n								(LSB)

The PARAMETER CODE field is described in 7.3.2.2.1, and shall be set as shown in table 345 for the Non-Medium Error Count log parameter.

The DU bit, TSD bit, ETC bit, TMC field, and FORMAT AND LINKING field are described in 7.3.2.2.2.1. These fields shall be set as described for a bounded data counter log parameter (see 7.3.2.2.2.2) for the Non-Medium Error Count log parameter.

The PARAMETER LENGTH field is described in 7.3.2.2.2.1.

The NON-MEDIUM ERROR COUNT field indicates the number of recoverable error events other than read, read reverse, verify, or write failures.

7.3.12 Power Condition Transitions log page

7.3.12.1 Overview

Using the format shown in table 347, the Power Condition Transitions log page (page code 1Ah) provides a count of the occurrences of power condition transition events using the parameter codes listed in table 346.

Table 346 — Power Conditions Transitions log page parameter codes

Parameter code	Description	Resettable or Changeable [a]	Reference	Support requirements
0001h	Accumulated Transitions to active	Never	7.3.12.2	Mandatory
0002h	Accumulated Transitions to idle_a			At least one [b]
0003h	Accumulated Transitions to idle_b			
0004h	Accumulated Transitions to idle_c			
0008h	Accumulated Transitions to standby_z			
0009h	Accumulated Transitions to standby_y			
all others	Reserved			

[a] The keywords in this column – Always, Reset Only, and Never – are defined in 7.3.2.2.2.6.

[b] If the Power Conditions Transitions log page is supported, at least one of these parameter codes shall be supported.

The Power Conditions Transitions log page has the format shown in table 347.

Table 347 — Power Condition Transitions log page

Bit / Byte	7	6	5	4	3	2	1	0
0	DS	SPF (0b)	PAGE CODE (1Ah)					
1	SUBPAGE CODE (00h)							
2	(MSB)	PAGE LENGTH (n-3)						
3								(LSB)
	Power condition transitions log parameters							
4	Power condition transitions log parameter (see table 346) [first]							
•••								
	⋮							
•••	Power condition transitions log parameter (see table 346) [last]							
n								

The DS bit, SPF bit, PAGE CODE field, SUBPAGE CODE field, and PAGE LENGTH field are described in 7.3.2. The SPF bit, PAGE CODE field, and SUBPAGE CODE field shall be set as shown in table 347 for the Power Conditions Transitions log page.

The contents of each power condition transitions log parameter depends on the value in its PARAMETER CODE field (see table 346).

7.3.12.2 Accumulated Transitions log parameter

The Accumulated Transitions log parameter has the format shown in table 348.

Table 348 — Accumulated Transitions log parameter

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)	PARAMETER CODE (see table 349)						
1								(LSB)
2	Parameter control byte – binary format list log parameter (see 7.3.2.2.2.5)							
	DU	Obsolete	TSD	ETC	TMC		FORMAT AND LINKING	
3	PARAMETER LENGTH (04h)							
4	(MSB)	ACCUMULATED TRANSITIONS TO						
•••								
7								(LSB)

The PARAMETER CODE field is described in 7.3.2.2.1, and shall be set as shown in table 348 for the Accumulated Transitions log parameter.

The DU bit, TSD bit, ETC bit, TMC field, and FORMAT AND LINKING field are described in 7.3.2.2.2.1. These fields shall be set as described for a binary format list log parameter (see 7.3.2.2.2.5) for the Accumulated Transitions log parameter.

The PARAMETER LENGTH field is described in 7.3.2.2.2.1, and shall be set as shown in table 348 for the Accumulated Transitions log parameter.

The ACCUMULATED TRANSITIONS TO field contains a saturating counter that is incremented by one at a time defined by the contents of the parameter code field as described in table 349. The time in the transition at which the count is incremented is vendor specific.

Table 349 — Accumulated Transitions parameter codes and saturating counters

Parameter code	Saturating counter incremented when the device server transitions from any other power condition to this power condition
0001h	active power condition (see 5.12.2)
0002h	idle_a power condition (see 5.12.3)
0003h	idle_b power condition (see 5.12.3)
0004h	idle_c power condition (see 5.12.3)
0008h	standby_z power condition (see 5.12.4)
0009h	standby_y power condition (see 5.12.4)

All values in the ACCUMULATED TRANSITIONS TO field are accumulated from the time the SCSI target device is manufactured.

7.3.13 Protocol Specific Port log page

7.3.13.1 Overview

Using the format shown in table 350, the Protocol Specific Port log page (page code 18h) provides SCSI transport protocol specific parameters that are associated with the SCSI target ports in the SCSI target device. This log page may be implemented in any logical unit, including the TARGET LOG PAGES well-known logical unit (see 8.4).

Protocol Specific Port log pages do not identify the information being logged using the PARAMETER CODE field in each log parameter. Instead, the SUBPAGE CODE field in the log page header (see table 350) serves as the indicator of what logged information is present. The PARAMETER CODE field identifies the SCSI Target Port to which the logged information applies.

The Protocol Specific Port log page has the format shown in table 350.

Table 350 — Protocol Specific Port log page

Bit / Byte	7	6	5	4	3	2	1	0
0	DS	SPF	PAGE CODE (18h)					
1	SUBPAGE CODE (00h to FEh)							
2	(MSB)	PAGE LENGTH (n-3)						
3								(LSB)
	Protocol specific port log parameters							
4	Protocol specific port log parameter [first]							
•••								
	⋮							
	Protocol specific port log parameter [last]							
•••								
n								

The DS bit, SPF bit, PAGE CODE field, SUBPAGE CODE field, and PAGE LENGTH field are described in 7.3.2. The SPF bit, PAGE CODE field, and SUBPAGE CODE field shall be set as shown in table 350 for the Protocol Specific Port log page.

The contents of each protocol specific port log parameter is defined by the corresponding SCSI transport protocol standard.

7.3.13.2 Generic protocol specific port log parameter

The generic format of a protocol specific port log parameter is shown in table 351.

Table 351 — Generic protocol specific port log parameter

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)	PARAMETER CODE						
1								(LSB)
2	Parameter control byte – binary format list log parameter (see 7.3.2.2.2.5)							
	DU	Obsolete	TSD	ETC	TMC		FORMAT AND LINKING	
3	PARAMETER LENGTH (n-3)							
4	Reserved				PROTOCOL IDENTIFIER			
5	SCSI transport protocol specific							
•••								
n								

For the Generic protocol specific port log parameter, the PARAMETER CODE field is described in 7.3.2.2.1, and contains the relative target port identifier (see 3.1.127) of the target port for which the parameter data applies.

The DU bit, TSD bit, ETC bit, TMC field, and FORMAT AND LINKING field are described in 7.3.2.2.2.1. These fields shall be set as described for a binary format list log parameter (see 7.3.2.2.2.5) for the Generic protocol specific port log parameter.

The PARAMETER LENGTH field is described in 7.3.2.2.2.1.

The PROTOCOL IDENTIFIER field contains one of the values shown in table 424 (see 7.6.1) to identify the SCSI transport protocol standard that defines the SCSI transport protocol specific data in this log parameter.

The SCSI transport protocol specific data is defined by the corresponding SCSI transport protocol standard.

7.3.14 Read Error Counters log page

7.3.14.1 Overview

Using the format shown in table 353, the Read Error Counters log page (page code 03h) contains log parameters that report bounded data counters for detected events (e.g., total bytes processed) identified by the parameter codes listed in table 352.

Table 352 — Read Error Counters log page parameter codes

Parameter code	Description	Resettable or Changeable [a]	Reference	Support requirements
0000h	Errors corrected without substantial delay [c]	Reset Only	7.3.14.2	At least one [b]
0001h	Errors corrected with possible delays [c]			
0002h	Total (e.g., rewrites or rereads) [c]			
0003h	Total errors corrected [c]			
0004h	Total times correction algorithm processed [c]			
0005h	Total bytes processed [c]			
0006h	Total uncorrected errors [c]			
8000h to FFFFh	Vendor specific			
all others	Reserved			

[a] The keywords in this column – Always, Reset Only, and Never – are defined in 7.3.2.2.2.6.

[b] If the Read Error Counters log page is supported, at least one of the parameter codes listed in this table shall be supported.

[c] The exact definition of this error counter is not part of this standard. This counter should not be used to compare products because the products may define errors differently.

The Read Error Counters log page has the format shown in table 353.

Table 353 — Read Error Counters log page

Bit / Byte	7	6	5	4	3	2	1	0
0	DS	SPF (0b)	PAGE CODE (03h)					
1	SUBPAGE CODE (00h)							
2	(MSB)	PAGE LENGTH (n-3)						
3								(LSB)
	Read Error Counter log parameters							
4 •••	Read Error Counter log parameter (see 7.3.14.2) [first]							
	⋮							
••• n	Read Error Counter log parameter (see 7.3.14.2) [last]							

The DS bit, SPF bit, PAGE CODE field, SUBPAGE CODE field, and PAGE LENGTH field are described in 7.3.2. The SPF bit, PAGE CODE field, and SUBPAGE CODE field shall be set as shown in table 353 for the Read Error Counters log page.

Each Read Error Counter log parameter contains the information described in 7.3.14.2.

7.3.14.2 Read Error Counter log parameter

The Read Error Counter log parameter has the format shown in table 354.

Table 354 — Read Error Counter log parameter

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)	PARAMETER CODE (see table 352)						
1								(LSB)
2	Parameter control byte – bounded data counter log parameter (see 7.3.2.2.2.2)							
	DU	Obsolete	TSD	ETC	TMC		FORMAT AND LINKING	
3	PARAMETER LENGTH (n-3)							
4 •••	(MSB)	READ ERROR COUNTER						
n								(LSB)

The PARAMETER CODE field is described in 7.3.2.2.1, and shall be set as shown in table 352 for the Read Error Counter log parameter.

The DU bit, TSD bit, ETC bit, TMC field, and FORMAT AND LINKING field are described in 7.3.2.2.2.1. These fields shall be set as described for a bounded data counter log parameter (see 7.3.2.2.2.2) for the Read Error Counter log parameter.

The PARAMETER LENGTH field is described in 7.3.2.2.2.1.

The READ ERROR COUNTER field contains the value for the counter described by the contents of the PARAMETER CODE field.

7.3.15 Read Reverse Error Counters log page

7.3.15.1 Overview

Using the format shown in table 356, the Read Reverse Error Counters log page (page code 04h) contains log parameters that report bounded data counters for detected events (e.g., total bytes processed) identified by the parameter codes listed in table 355.

Table 355 — Read Reverse Error Counters log page parameter codes

Parameter code	Description	Resettable or Changeable [a]	Reference	Support requirements
0000h	Errors corrected without substantial delay [c]	Reset Only	7.3.15.2	At least one [b]
0001h	Errors corrected with possible delays [c]			
0002h	Total (e.g., rewrites or rereads) [c]			
0003h	Total errors corrected [c]			
0004h	Total times correction algorithm processed [c]			
0005h	Total bytes processed [c]			
0006h	Total uncorrected errors [c]			
8000h to FFFFh	Vendor specific			
all others	Reserved			

[a] The keywords in this column – Always, Reset Only, and Never – are defined in 7.3.2.2.2.6.

[b] If the Read Reverse Error Counters log page is supported, at least one of the parameter codes listed in this table shall be supported.

[c] The exact definition of this error counter is not part of this standard. This counter should not be used to compare products because the products may define errors differently.

The Read Reverse Error Counters log page has the format shown in table 356.

Table 356 — Read Reverse Error Counters log page

Bit / Byte	7	6	5	4	3	2	1	0
0	DS	SPF (0b)	PAGE CODE (04h)					
1	SUBPAGE CODE (00h)							
2	(MSB)	PAGE LENGTH (n-3)						
3								(LSB)
	Read reverse error counter log parameters							
4	Read Reverse Error Counter log parameter (see 7.3.15.2) [first]							
•••								
	⋮							
	Read Reverse Error Counter log parameter (see 7.3.15.2) [last]							
•••								
n								

The DS bit, SPF bit, PAGE CODE field, SUBPAGE CODE field, and PAGE LENGTH field are described in 7.3.2. The SPF bit, PAGE CODE field, and SUBPAGE CODE field shall be set as shown in table 356 for the Read Reverse Error Counters log page.

Each Read Reverse Error Counter log parameter contains the information described in 7.3.15.2.

7.3.15.2 Read Reverse Error Counter log parameter

The Read Reverse Error Counter log parameter has the format shown in table 357.

Table 357 — Read Reverse Error Counter log parameter

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)	PARAMETER CODE (see table 355)						
1								(LSB)
2	Parameter control byte – bounded data counter log parameter (see 7.3.2.2.2.2)							
	DU	Obsolete	TSD	ETC	TMC		FORMAT AND LINKING	
3	PARAMETER LENGTH (n-3)							
4	(MSB)	READ REVERSE ERROR COUNTER						
•••								
n								(LSB)

The PARAMETER CODE field is described in 7.3.2.2.1, and shall be set as shown in table 355 for the Read Reverse Error Counter log parameter.

The DU bit, TSD bit, ETC bit, TMC field, and FORMAT AND LINKING field are described in 7.3.2.2.2.1. These fields shall be set as described for a bounded data counter log parameter (see 7.3.2.2.2.2) for the Read Reverse Error Counter log parameter.

The PARAMETER LENGTH field is described in 7.3.2.2.2.1.

The READ REVERSE ERROR COUNTER field contains the value for the counter described by the contents of the PARAMETER CODE field.

7.3.16 Self-Test Results log page

7.3.16.1 Overview

Using the format shown in table 359, the Self-Test Results log page (page code 10h) reports the results from the 20 most recent self-tests (see 5.8). The parameter codes for the Self-Test Results log page are listed in table 358.

Table 358 — Self-Test Results log page parameter codes

Parameter code	Description	Resettable or Changeable [a]	Reference	Support requirements
0001h to 0014h	Self-test results	Reset Only	7.3.16.2	Mandatory
all others	Reserved			

[a] The keywords in this column – Always, Reset Only, and Never – are defined in 7.3.2.2.2.6.

The Self-Test Results log page has the format shown in table 359.

Table 359 — Self-Test Results log page

Bit / Byte	7	6	5	4	3	2	1	0
0	DS	SPF (0b)	PAGE CODE (10h)					
1	SUBPAGE CODE (00h)							
2	(MSB)	PAGE LENGTH (190h)						
3								(LSB)
	Self-test results log parameters							
4 ••• 23	Self-test results log parameter [first]							
	⋮							
384 ••• 403	Self-test results log parameter [twentieth]							

The DS bit, SPF bit, PAGE CODE field, SUBPAGE CODE field, and PAGE LENGTH field are described in 7.3.2. The SPF bit, PAGE CODE field, SUBPAGE CODE field, and PAGE LENGTH field shall be set as shown in table 359 for the Self-Test Results log page.

The results of the most recent self-test shall be contained in the log parameter with the PARAMETER CODE field set to 0001h; the results for the second most recent self-test shall be contained in the log parameter with the PARAMETER CODE field set to 0002h; etc. If fewer than 20 self-tests have been performed, then:

a) unused self-test log parameters shall have parameter code values higher than those of any used self-test log parameter; and
b) each unused self-test log parameter entry shall have the format shown in table 360.

Table 360 — Unused Self-Test Results log parameter

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)	PARAMETER CODE (see table 358)						
1								(LSB)
2	Parameter control byte – binary format list log parameter (see 7.3.2.2.2.5)							
	DU	Obsolete	TSD	ETC	TMC		FORMAT AND LINKING	
3	PARAMETER LENGTH (10h)							
4	ZERO							
•••								
19								

The PARAMETER CODE field is described in 7.3.2.2.1, and shall be set as shown in table 358 for each Unused Self-Test Results log parameter.

The DU bit, TSD bit, ETC bit, TMC field, and FORMAT AND LINKING field are described in 7.3.2.2.2.1. These fields shall be set as described for a binary format list log parameter (see 7.3.2.2.2.5) for each Unused Self-Test Results log parameter.

The PARAMETER LENGTH field is described in 7.3.2.2.2.1, and shall be set as shown in table 360 for each Unused Self-Test Results log parameter.

The ZERO field contains zero.

7.3.16.2 Self-Test Results log parameters

Each Self-Test Results log parameter has the format shown in table 361.

Table 361 — Self-Test Results log parameter

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)	PARAMETER CODE (see table 358)						
1								(LSB)
2	Parameter control byte – binary format list log parameter (see 7.3.2.2.2.5)							
	DU	Obsolete	TSD	ETC	TMC		FORMAT AND LINKING	
3	PARAMETER LENGTH (10h)							
4	SELF-TEST CODE			Reserved	SELF-TEST RESULTS			
5	SELF-TEST NUMBER							
6	(MSB)	ACCUMULATED POWER ON HOURS						
7								(LSB)
8	(MSB)	ADDRESS OF FIRST FAILURE						
•••								
15								(LSB)
16	Reserved				SENSE KEY			
17	ADDITIONAL SENSE CODE							
18	ADDITIONAL SENSE CODE QUALIFIER							
19	Vendor specific							

The PARAMETER CODE field is described in 7.3.2.2.1, and shall be set as shown in table 358 for the Self-Test Results log parameter.

The DU bit, TSD bit, ETC bit, TMC field, and FORMAT AND LINKING field are described in 7.3.2.2.2.1. These fields shall be set as described for a binary format list log parameter (see 7.3.2.2.2.5) for the Unused Self-Test Results log parameter.

The PARAMETER LENGTH field is described in 7.3.2.2.2.1, and shall be set as shown in table 361 for the Self-Test Results log parameter.

The SELF-TEST CODE field contains the value in the SELF-TEST CODE field of the SEND DIAGNOSTIC command (see 6.32) that initiated this self-test.

Table 362 defines the content of the SELF-TEST RESULTS field.

Table 362 — SELF-TEST RESULTS field

Code	Description
0h	The self-test completed without error.
1h	The background self-test was aborted by the application client using a SEND DIAGNOSTIC command (see 6.32) with the SELF-TEST CODE field set to 100b (i.e., abort background self-test).
2h	The self-test routine was aborted by an application client using a method other than a SEND DIAGNOSTIC command with the SELF-TEST CODE field set to 100b (e.g., by a task management function, or by issuing an exception command as defined in 5.8.4).
3h	An unknown error occurred while the device server was processing the self-test and the device server was unable to complete the self-test.
4h	The self-test completed with a failure in a test segment, and the test segment that failed is not known.
5h	The first segment of the self-test failed.
6h	The second segment of the self-test failed.
7h	Another segment of the self-test failed and which test is indicated by the contents of the SELF-TEST NUMBER field.
8h to Eh	Reserved
Fh	The self-test is in progress.

The SELF-TEST NUMBER field identifies the self-test that failed and consists of either:

a) the number of the segment that failed during the self-test; or
b) the number of the test that failed and the number of the segment in which the test was run, using a vendor specific method for placing the two values in the one field.

When the segment in which the failure occurred is not able to be identified or need not be identified, the SELF-TEST NUMBER field shall contain 00h.

The ACCUMULATED POWER ON HOURS field contains the total hours the device server has been powered on since manufacturing at the time the self-test is completed. If the self-test is still in progress, the ACCUMULATED POWER ON HOURS field shall be set to zero. If the number of hours is greater than FFFFh, the ACCUMULATED POWER ON HOURS field shall be set to FFFFh.

The ADDRESS OF FIRST FAILURE field contains information that locates the failure on the media. If the logical unit implements logical blocks, the content of the ADDRESS OF FIRST FAILURE field is the first logical block address where a self-test error occurred. This implies nothing about the quality of any other logical block on the logical unit, since the testing during which the error occurred may not have been performed in a sequential manner. This value shall not change (e.g., as the result of block reassignment). The content of the ADDRESS OF FIRST FAILURE field shall be FFFF FFFF FFFF FFFFh if no errors occurred during the self-test or if the error that occurred is not related to an identifiable media address.

The SENSE KEY field, ADDITIONAL SENSE CODE field, and ADDITIONAL SENSE CODE QUALIFIER field may contain a hierarchy of additional information relating to error or exception conditions that occurred during the self-test represented in the same format used by the sense data (see 4.5).

7.3.17 Start-Stop Cycle Counter log page

7.3.17.1 Overview

Using the format shown in table 364, the Start-Stop Cycle Counter log page (page code 0Eh) provides information about manufacturing dates and cycle counts since date of manufacture using the parameter codes listed in table 363.

Table 363 — Start-Stop Cycle Counter log page parameter codes

Parameter code	Description	Resettable or Changeable [a]	Reference	Support requirements
0001h	Date of Manufacture	Never	7.3.17.2	At least one [b]
0002h	Accounting Date	Always	7.3.17.3	
0003h	Specified Cycle Count Over Device Lifetime	Never	7.3.17.4	
0004h	Accumulated Start-Stop Cycles	Never	7.3.17.5	
0005h	Specified Load-Unload Count Over Device Lifetime	Never	7.3.17.6	
0006h	Accumulated Load-Unload Cycles	Never	7.3.17.7	
all others	Reserved			

[a] The keywords in this column – Always, Reset Only, and Never – are defined in 7.3.2.2.2.6.

[b] If the Start-Stop Cycle Counter log page is supported, at least one of the parameter codes listed in this table shall be supported.

The Start-Stop Cycle Counter log page has the format shown in table 364.

Table 364 — Start-Stop Cycle Counter log page

Bit / Byte	7	6	5	4	3	2	1	0
0	DS	SPF (0b)	PAGE CODE (0Eh)					
1	SUBPAGE CODE (00h)							
2	(MSB)	PAGE LENGTH (n-3)						
3								(LSB)
	Start stop cycle log parameters							
4	Start stop cycle log parameter (see table 363) [first]							
•••								
	⋮							
	Start stop cycle log parameter (see table 363) [last]							
•••								
n								

The DS bit, SPF bit, PAGE CODE field, SUBPAGE CODE field, and PAGE LENGTH field are described in 7.3.2. The SPF bit, PAGE CODE field, and SUBPAGE CODE field shall be set as shown in table 364 for the Start-Stop Cycle Counter log page.

7.3.17.2 Date of Manufacture log parameter

The Date of Manufacture log parameter has the format shown in table 365.

Table 365 — Date of Manufacture log parameter

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)	PARAMETER CODE (0001h)						
1								(LSB)
2	Parameter control byte – ASCII format list log parameter (see 7.3.2.2.2.4)							
	DU	Obsolete	TSD	ETC	TMC		FORMAT AND LINKING	
3	PARAMETER LENGTH (06h)							
4	(MSB)	YEAR OF MANUFACTURE						
•••								
7								(LSB)
8	(MSB)	WEEK OF MANUFACTURE						
9								(LSB)

The PARAMETER CODE field is described in 7.3.2.2.1, and shall be set as shown in table 365 for the Date of Manufacture log parameter.

The DU bit, TSD bit, ETC bit, TMC field, and FORMAT AND LINKING field are described in 7.3.2.2.2.1. These fields shall be set as described for a ASCII format list log parameter (see 7.3.2.2.2.4) for the Date of Manufacture log parameter.

The PARAMETER LENGTH field is described in 7.3.2.2.2.1, and shall be set as shown in table 365 for the Date of Manufacture log parameter.

The YEAR OF MANUFACTURE field indicates the year in which the SCSI target device was manufactured and contains four numeric ASCII characters (e.g., 30h for zero and 39h for nine).

The WEEK OF MANUFACTURE field indicates the week of the year in which the SCSI target device was manufactured and contains two numeric ASCII characters.

7.3.17.3 Accounting Date log parameter

The Accounting Date log parameter has the format shown in table 366.

Table 366 — Accounting Date log parameter

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)	PARAMETER CODE (0002h)						
1								(LSB)
2	Parameter control byte – ASCII format list log parameter (see 7.3.2.2.2.4)							
	DU	Obsolete	TSD	ETC	TMC		FORMAT AND LINKING	
3	PARAMETER LENGTH (06h)							
4	(MSB)	ACCOUNTING DATE YEAR						
•••								
7								(LSB)
8	(MSB)	ACCOUNTING DATE WEEK						
9								(LSB)

The PARAMETER CODE field is described in 7.3.2.2.1, and shall be set as shown in table 366 for the Accounting Date log parameter.

The DU bit, TSD bit, ETC bit, TMC field, and FORMAT AND LINKING field are described in 7.3.2.2.2.1. These fields shall be set as described for a ASCII format list log parameter (see 7.3.2.2.2.4) for the Accounting Date log parameter.

The PARAMETER LENGTH field is described in 7.3.2.2.2.1, and shall be set as shown in table 366 for the Accounting Date log parameter.

The ACCOUNTING DATE YEAR field indicates the year in which the SCSI target device was placed in service and contains four numeric ASCII characters (e.g., 30h for zero and 39h for nine).

The ACCOUNTING DATE WEEK field indicates the week of the year in which the SCSI target device was placed in service and contains two numeric ASCII characters.

A LOG SELECT command may be used to change the value of the ACCOUNTING DATE YEAR field and ACCOUNTING DATE WEEK field. If the Accounting Date log parameter is not yet set, then the value placed:

a) in the ACCOUNTING DATE YEAR field shall be four ASCII space characters (i.e., 20h); and
b) in the ACCOUNTING DATE WEEK field shall be two ASCII space characters (i.e., 20h).

The ACCOUNTING DATE YEAR field and ACCOUNTING DATE WEEK field shall not be checked for validity by the device server.

7.3.17.4 Specified Cycle Count Over Device Lifetime log parameter

The Specified Cycle Count Over Device Lifetime log parameter has the format shown in table 367.

Table 367 — Specified Cycle Count Over Device Lifetime log parameter

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)	PARAMETER CODE (0003h)						
1								(LSB)
2	Parameter control byte – binary format list log parameter (see 7.3.2.2.2.5)							
	DU	Obsolete	TSD	ETC	TMC		FORMAT AND LINKING	
3	PARAMETER LENGTH (04h)							
4	(MSB)	SPECIFIED CYCLE COUNT OVER DEVICE LIFETIME						
•••								
7								(LSB)

The PARAMETER CODE field is described in 7.3.2.2.1, and shall be set as shown in table 367 for the Specified Cycle Count Over Device Lifetime log parameter.

The DU bit, TSD bit, ETC bit, TMC field, and FORMAT AND LINKING field are described in 7.3.2.2.2.1. These fields shall be set as described for a binary format list log parameter (see 7.3.2.2.2.5) for the Specified Cycle Count Over Device Lifetime log parameter.

The PARAMETER LENGTH field is described in 7.3.2.2.2.1, and shall be set as shown in table 367 for the Specified Cycle Count Over Device Lifetime log parameter.

The contents of the SPECIFIED CYCLE COUNT OVER DEVICE LIFETIME field indicate the number of stop-start cycles that may be performed over the lifetime of the SCSI target device without degrading the SCSI target device's operation or reliability outside the limits specified by the manufacturer of the SCSI target device.

7.3.17.5 Accumulated Start-Stop Cycles log parameter

The Accumulated Start-Stop Cycles log parameter has the format shown in table 368.

Table 368 — Accumulated Start-Stop Cycles log parameter

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)	PARAMETER CODE (0004h)						
1								(LSB)
2	Parameter control byte – binary format list log parameter (see 7.3.2.2.2.5)							
	DU	Obsolete	TSD	ETC	TMC		FORMAT AND LINKING	
3	PARAMETER LENGTH (04h)							
4	(MSB)	ACCUMULATED START-STOP CYCLES						
•••								
7								(LSB)

The PARAMETER CODE field is described in 7.3.2.2.1, and shall be set as shown in table 368 for the Accumulated Start-Stop Cycles log parameter.

The DU bit, TSD bit, ETC bit, TMC field, and FORMAT AND LINKING field are described in 7.3.2.2.2.1. These fields shall be set as described for a binary format list log parameter (see 7.3.2.2.2.5) for the Accumulated Start-Stop Cycles log parameter.

The PARAMETER LENGTH field is described in 7.3.2.2.2.1, and shall be set as shown in table 368 for the Accumulated Start-Stop Cycles log parameter.

The contents of the ACCUMULATED START-STOP CYCLES field indicate the number of stop-start cycles the SCSI target device has detected since its date of manufacture. This saturating counter is incremented by one for each complete cycle. The time in the cycle at which the counter is incremented is vendor specific.

For rotating magnetic storage devices (see SBC-3), a single start-stop cycle is defined as an operational cycle that:

a) begins with the disk spindle at rest;
b) continues while the disk accelerates to its normal operational rotational rate;
c) continues during the entire period the disk is rotating;
d) continues as the disk decelerates toward a resting state; and
e) ends when the disk is no longer rotating.

For devices without a spindle or with multiple spindles, the definition of a single start-stop cycle is vendor specific.

The device server shall not compare the contents of the ACCUMULATED START-STOP CYCLES field to contents of the SPECIFIED CYCLE COUNT OVER DEVICE LIFETIME field (see 7.3.17.4).

7.3.17.6 Specified Load-Unload Count Over Device Lifetime log parameter

The Specified Load-Unload Count Over Device Lifetime log parameter has the format shown in table 369.

Table 369 — Specified Load-Unload Count Over Device Lifetime log parameter

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)	PARAMETER CODE (0005h)						
1								(LSB)
2	Parameter control byte – binary format list log parameter (see 7.3.2.2.2.5)							
	DU	Obsolete	TSD	ETC	TMC		FORMAT AND LINKING	
3	PARAMETER LENGTH (04h)							
4	(MSB)	SPECIFIED LOAD-UNLOAD COUNT OVER DEVICE LIFETIME						
•••								
7								(LSB)

The PARAMETER CODE field is described in 7.3.2.2.1, and shall be set as shown in table 369 for the Specified Load-Unload Count Over Device Lifetime log parameter.

The DU bit, TSD bit, ETC bit, TMC field, and FORMAT AND LINKING field are described in 7.3.2.2.2.1. These fields shall be set as described for a binary format list log parameter (see 7.3.2.2.2.5) for the Specified Load-Unload Count Over Device Lifetime log parameter.

The PARAMETER LENGTH field is described in 7.3.2.2.2.1, and shall be set as shown in table 369 for the Specified Load-Unload Count Over Device Lifetime log parameter.

The contents of the SPECIFIED LOAD-UNLOAD COUNT OVER DEVICE LIFETIME field indicate the number of load-unload cycles that may be performed over the lifetime of the SCSI target device without degrading the SCSI target device's operation or reliability outside the limits specified by the manufacturer of the SCSI target device.

7.3.17.7 Accumulated Load-Unload Cycles log parameter

The Accumulated Load-Unload Cycles log parameter has the format shown in table 370.

Table 370 — Accumulated Load-Unload Cycles log parameter

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)	PARAMETER CODE (0006h)						
1								(LSB)
2	Parameter control byte – binary format list log parameter (see 7.3.2.2.2.5)							
	DU	Obsolete	TSD	ETC	TMC		FORMAT AND LINKING	
3	PARAMETER LENGTH (04h)							
4	(MSB)	ACCUMULATED LOAD-UNLOAD CYCLES						
•••								
7								(LSB)

The PARAMETER CODE field is described in 7.3.2.2.1, and shall be set as shown in table 370 for the Accumulated Load-Unload Cycles log parameter.

The DU bit, TSD bit, ETC bit, TMC field, and FORMAT AND LINKING field are described in 7.3.2.2.2.1. These fields shall be set as described for a binary format list log parameter (see 7.3.2.2.2.5) for the Accumulated Load-Unload Cycles log parameter.

The PARAMETER LENGTH field is described in 7.3.2.2.2.1, and shall be set as shown in table 370 for the Accumulated Load-Unload Cycles log parameter.

The contents of the ACCUMULATED LOAD-UNLOAD CYCLES field indicate the number of load-unload cycles the SCSI target device has detected since its date of manufacture. This saturating counter is incremented by one for each complete cycle. The time in the cycle at which the counter is incremented is vendor specific.

For rotating magnetic storage devices (see SBC-3), a single load-unload cycle is defined as an operational cycle that:

a) begins with the heads unloaded from the medium;
b) continues while the heads are loaded onto the spinning medium; and
c) ends when the heads are unloaded from the medium.

The Accumulated Load-Unload Cycles log parameter is not applicable to rotating magnetic storage devices without unloadable heads.

The device server shall not compare the contents of the ACCUMULATED LOAD-UNLOAD CYCLES field to contents of the SPECIFIED LOAD-UNLOAD COUNT OVER DEVICE LIFETIME field (see 7.3.17.6).

7.3.18 Supported Log Pages log page

For the LOG SENSE command, the Supported Log Pages log page (see table 371) returns the list of log pages implemented by the logical unit. Logical units that implement the LOG SENSE command shall implement this log page. This log page is not defined for the LOG SELECT command.

Table 371 — Supported Log Pages log page

Bit / Byte	7	6	5	4	3	2	1	0
0	DS (0b)	SPF (0b)	PAGE CODE (00h)					
1	SUBPAGE CODE (00h)							
2	(MSB)	PAGE LENGTH (n-3)						
3								(LSB)
	Supported pages list							
4	Supported page descriptor (see table 372) [first]							
	⋮							
n	Supported page descriptor (see table 372) [last]							

The DS bit, SPF bit, PAGE CODE field, SUBPAGE CODE field, and PAGE LENGTH field are described in 7.3.2. The DS bit, SPF bit, SUBPAGE CODE field shall be set as shown in table 371 for the Supported Log Pages log page.

The supported page descriptors shall contain a list of all log page codes (see table 372) with a subpage code of zero implemented by the logical unit in ascending order beginning with page code 00h.

Table 372 — Supported page descriptor

Bit / Byte	7	6	5	4	3	2	1	0
0	Reserved		PAGE CODE					

The PAGE CODE field contains the number of a supported log page.

7.3.19 Supported Log Pages and Subpages log page

For the LOG SENSE command, the Supported Log Pages and Subpages log page (see table 373) returns the list of log pages and subpages implemented by the logical unit. If log subpages are supported this page shall be supported. This log page is not defined for the LOG SELECT command.

Table 373 — Supported Log Pages and Subpages log page

Bit / Byte	7	6	5	4	3	2	1	0
0	DS (0b)	SPF (1b)	PAGE CODE (00h)					
1	SUBPAGE CODE (FFh)							
2	(MSB)	PAGE LENGTH (n-3)						
3								(LSB)
	Supported page/subpage descriptors							
4	Supported page/subpage descriptor (see table 374) [first]							
5								
	⋮							
n-1	Supported page/subpage descriptor (see table 374) [last]							
n								

The DS bit, SPF bit, PAGE CODE field, SUBPAGE CODE field, and PAGE LENGTH field are described in 7.3.2. The DS bit, SPF bit, PAGE CODE field, and SUBPAGE CODE field shall be set as shown in table 373 for the Supported Log Pages and Subpages log page.

The supported page/subpage descriptors (see table 374) shall be in ascending order sorted by page code then subpage code.

Table 374 — Supported page/subpage descriptor

Bit / Byte	7	6	5	4	3	2	1	0
0	Reserved		PAGE CODE					
1	SUBPAGE CODE							

The PAGE CODE field contains the number of a supported log page.

The SUBPAGE CODE field contains the subpage number of a supported log page.

7.3.20 Supported Subpages log page

For the LOG SENSE command, the Supported Subpages log page (see table 375) returns the list of all subpage codes that are implemented by the logical unit for a specified page code. If log subpages are supported this page shall be supported. This log page is not defined for the LOG SELECT command.

Table 375 — Supported Subpages log page

Bit / Byte	7	6	5	4	3	2	1	0
0	DS (0b)	SPF (1b)	PAGE CODE					
1	SUBPAGE CODE (FFh)							
2	(MSB)	PAGE LENGTH (n-3)						
3								(LSB)
	Supported subpage descriptors							
4	Supported subpage descriptor (see table 376) [first]							
5								
	⋮							
n-1	Supported subpage descriptor (see table 376) [last]							
n								

The DS bit, SPF bit, SUBPAGE CODE field, and PAGE LENGTH field are described in 7.3.2. The DS bit, SPF bit, SUBPAGE CODE field shall be set as shown in table 375 for the Supported Subpages log page.

The PAGE CODE field (see 7.3.2) indicates the log page for which log page and subpage codes are being returned.

The supported subpage descriptors (see table 376) shall be in ascending order sorted by page code then subpage code.

Table 376 — Supported subpage descriptor

Bit / Byte	7	6	5	4	3	2	1	0
0	Reserved		PAGE CODE					
1	SUBPAGE CODE							

The PAGE CODE field contains the number of a supported log page.

NOTE 47 - The page code is the same in the page header (see table 375) and in each supported subpage descriptor (see table 376).

The SUBPAGE CODE field contains the subpage number of a supported log page.

7.3.21 Temperature log page

7.3.21.1 Overview

Using the format shown in table 378, the Temperature log page (page code 0Dh) provides information about the current operating temperature of the SCSI Target Device using the parameter codes listed in table 377.

Table 377 — Temperature log page parameter codes

Parameter code	Description	Resettable or Changeable [a]	Reference	Support requirements
0000h	Temperature	Never	7.3.21.2	Mandatory
0001h	Reference temperature	Never	7.3.21.3	Optional
all others	Reserved			

[a] The keywords in this column – Always, Reset Only, and Never – are defined in 7.3.2.2.2.6.

The Temperature log page has the format shown in table 378.

Table 378 — Temperature log page

Bit / Byte	7	6	5	4	3	2	1	0
0	DS	SPF (0b)	PAGE CODE (0Dh)					
1	SUBPAGE CODE (00h)							
2	(MSB)	PAGE LENGTH (n-3)						
3								(LSB)
	Temperature log parameters							
4	Temperature log parameter (see table 377) [first]							
•••								
	⋮							
•••	Temperature log parameter (see table 377) [last]							
n								

The DS bit, SPF bit, PAGE CODE field, SUBPAGE CODE field, and PAGE LENGTH field are described in 7.3.2. The SPF bit, PAGE CODE field, and SUBPAGE CODE field shall be set as shown in table 378 for the Temperature log page.

The contents of each temperature log parameter depends on the value in its PARAMETER CODE field (see table 377).

7.3.21.2 Temperature log parameter

The Temperature log parameter has the format shown in table 379.

Table 379 — Temperature log parameter

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)	PARAMETER CODE (0000h)						
1								(LSB)
2	Parameter control byte – binary format list log parameter (see 7.3.2.2.2.5)							
	DU	Obsolete	TSD	ETC	TMC		FORMAT AND LINKING	
3	PARAMETER LENGTH (02h)							
4	Reserved							
5	TEMPERATURE							

The PARAMETER CODE field is described in 7.3.2.2.1, and shall be set as shown in table 379 for the Temperature log parameter.

The DU bit, TSD bit, ETC bit, TMC field, and FORMAT AND LINKING field are described in 7.3.2.2.2.1. These fields shall be set as described for a binary format list log parameter (see 7.3.2.2.2.5) for the Temperature log parameter.

The PARAMETER LENGTH field is described in 7.3.2.2.2.1, and shall be set as shown in table 379 for the Temperature log parameter.

The TEMPERATURE field contains the temperature of the SCSI target device in degrees Celsius at the time the LOG SENSE command is performed. Temperatures equal to or less than zero degrees Celsius shall cause the TEMPERATURE field to be set to zero. If the device server is unable to detect a valid temperature because of a sensor failure or other condition, then the TEMPERATURE field shall be set to FFh. The temperature should be reported with an accuracy of plus or minus three Celsius degrees while the SCSI target device is operating at a steady state within its environmental limits.

No comparison is performed between the contents of the TEMPERATURE field and the contents of the optional REFERENCE TEMPERATURE field (see 7.3.21.3).

7.3.21.3 Reference Temperature log parameter

The Reference Temperature log parameter has the format shown in table 380.

Table 380 — Reference Temperature log parameter

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)	PARAMETER CODE (0001h)						
1								(LSB)
2	Parameter control byte – binary format list log parameter (see 7.3.2.2.2.5)							
	DU	Obsolete	TSD	ETC	TMC		FORMAT AND LINKING	
3	PARAMETER LENGTH (02h)							
4	Reserved							
5	REFERENCE TEMPERATURE							

The PARAMETER CODE field is described in 7.3.2.2.1, and shall be set as shown in table 380 for the Reference Temperature log parameter.

The DU bit, TSD bit, ETC bit, TMC field, and FORMAT AND LINKING field are described in 7.3.2.2.2.1. These fields shall be set as described for a binary format list log parameter (see 7.3.2.2.2.5) for the Reference Temperature log parameter.

The PARAMETER LENGTH field is described in 7.3.2.2.2.1, and shall be set as shown in table 380 for the Reference Temperature log parameter.

The REFERENCE TEMPERATURE field contains the maximum reported sensor temperature in degrees Celsius at which the SCSI target device is capable of operating continuously without degrading the SCSI target device's operation or reliability beyond manufacturer accepted limits. If the device server is unable to return a reference temperature and the optional Reference Temperature log parameter is included in the Temperature log page being returned, then REFERENCE TEMPERATURE field is set to FFh.

The reference temperature may change for vendor specific reasons.

7.3.22 Verify Error Counters log page

7.3.22.1 Overview

Using the format shown in table 382, the Verify Error Counters log page (page code 05h) contains log parameters that report bounded data counters for detected events (e.g., total bytes processed) identified by the parameter codes listed in table 381.

Table 381 — Verify Error Counters log page parameter codes

Parameter code	Description	Resettable or Changeable [a]	Reference	Support requirements
0000h	Errors corrected without substantial delay [c]	Reset Only	7.3.22.2	At least one [b]
0001h	Errors corrected with possible delays [c]			
0002h	Total (e.g., rewrites or rereads) [c]			
0003h	Total errors corrected [c]			
0004h	Total times correction algorithm processed [c]			
0005h	Total bytes processed [c]			
0006h	Total uncorrected errors [c]			
8000h to FFFFh	Vendor specific			
all others	Reserved			

[a] The keywords in this column – Always, Reset Only, and Never – are defined in 7.3.2.2.2.6.

[b] If the Verify Error Counters log page is supported, at least one of the parameter codes listed in this table shall be supported.

[c] The exact definition of this error counter is not part of this standard. This counter should not be used to compare products because the products may define errors differently.

The Verify Error Counters log page has the format shown in table 382.

Table 382 — Verify Error Counters log page

Bit / Byte	7	6	5	4	3	2	1	0
0	DS	SPF (0b)	PAGE CODE (05h)					
1	SUBPAGE CODE (00h)							
2	(MSB)	PAGE LENGTH (n-3)						
3								(LSB)
	Verify error counter log parameters							
4 •••	Verify Error Counter log parameter (see 7.3.22.2) [first]							
	⋮							
••• n	Verify Error Counter log parameter (see 7.3.22.2) [last]							

The DS bit, SPF bit, PAGE CODE field, SUBPAGE CODE field, and PAGE LENGTH field are described in 7.3.2. The SPF bit, PAGE CODE field, and SUBPAGE CODE field shall be set as shown in table 382 for the Verify Error Counters log page.

Each Verify Error Counter log parameter contains the information described in 7.3.22.2.

7.3.22.2 Verify Error Counter log parameter

The Verify Error Counter log parameter has the format shown in table 383.

Table 383 — Verify Error Counter log parameter

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)	PARAMETER CODE (see table 381)						
1								(LSB)
2	Parameter control byte – bounded data counter log parameter (see 7.3.2.2.2.2)							
	DU	Obsolete	TSD	ETC	TMC		FORMAT AND LINKING	
3	PARAMETER LENGTH (n-3)							
4 •••	(MSB)	VERIFY ERROR COUNTER						
n								(LSB)

The PARAMETER CODE field is described in 7.3.2.2.1, and shall be set as shown in table 381 for the Verify Error Counter log parameter.

The DU bit, TSD bit, ETC bit, TMC field, and FORMAT AND LINKING field are described in 7.3.2.2.2.1. These fields shall be set as described for a bounded data counter log parameter (see 7.3.2.2.2.2) for the Verify Error Counter log parameter.

The PARAMETER LENGTH field is described in 7.3.2.2.2.1.

The VERIFY ERROR COUNTER field contains the value for the counter described by the contents of the PARAMETER CODE field.

7.3.23 Write Error Counters log page

7.3.23.1 Overview

Using the format shown in table 385, the Write Error Counters log page (page code 02h) contains log parameters that report bounded data counters for detected events (e.g., total bytes processed) identified by the parameter codes listed in table 384.

Table 384 — Write Error Counters log page parameter codes

Parameter code	Description	Resettable or Changeable [a]	Reference	Support requirements
0000h	Errors corrected without substantial delay [c]	Reset Only	7.3.23.2	At least one [b]
0001h	Errors corrected with possible delays [c]			
0002h	Total (e.g., rewrites or rereads) [c]			
0003h	Total errors corrected [c]			
0004h	Total times correction algorithm processed [c]			
0005h	Total bytes processed [c]			
0006h	Total uncorrected errors [c]			
8000h to FFFFh	Vendor specific			
all others	Reserved			

[a] The keywords in this column – Always, Reset Only, and Never – are defined in 7.3.2.2.2.6.

[b] If the Write Error Counters log page is supported, at least one of the parameter codes listed in this table shall be supported.

[c] The exact definition of this error counter is not part of this standard. This counter should not be used to compare products because the products may define errors differently.

The Write Error Counters log page has the format shown in table 385.

Table 385 — Write Error Counters log page

Bit / Byte	7	6	5	4	3	2	1	0
0	DS	SPF (0b)	PAGE CODE (02h)					
1	SUBPAGE CODE (00h)							
2	(MSB)	PAGE LENGTH (n-3)						
3								(LSB)
	Write error counter log parameters							
4	Write Error Counter log parameter (see 7.3.23.2) [first]							
•••								
	⋮							
•••	Write Error Counter log parameter (see 7.3.23.2) [last]							
n								

The DS bit, SPF bit, PAGE CODE field, SUBPAGE CODE field, and PAGE LENGTH field are described in 7.3.2. The SPF bit, PAGE CODE field, and SUBPAGE CODE field shall be set as shown in table 385 for the Write Error Counters log page.

Each Write Error Counter log parameter contains the information described in 7.3.23.2.

7.3.23.2 Write Error Counter log parameter

The Write Error Counter log parameter has the format shown in table 386.

Table 386 — Write Error Counter log parameter

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)	PARAMETER CODE (see table 384)						
1								(LSB)
2	Parameter control byte – bounded data counter log parameter (see 7.3.2.2.2.2)							
	DU	Obsolete	TSD	ETC	TMC		FORMAT AND LINKING	
3	PARAMETER LENGTH (n-3)							
4	(MSB)	WRITE ERROR COUNTER						
•••								
n								(LSB)

The PARAMETER CODE field is described in 7.3.2.2.1, and shall be set as shown in table 384 for the Write Error Counter log parameter.

The DU bit, TSD bit, ETC bit, TMC field, and FORMAT AND LINKING field are described in 7.3.2.2.2.1. These fields shall be set as described for a bounded data counter log parameter (see 7.3.2.2.2.2) for the Write Error Counter log parameter.

The PARAMETER LENGTH field is described in 7.3.2.2.2.1.

The WRITE ERROR COUNTER field contains the value for the counter described by the contents of the PARAMETER CODE field.

7.4 Medium auxiliary memory attributes

7.4.1 Attribute format

Each medium auxiliary memory attribute shall be communicated between the application client and device server in the format shown in table 387. This format shall be used in the parameter data for the WRITE ATTRIBUTE command (see 6.38) and the READ ATTRIBUTE command (see 6.15). The attribute format in this standard implies nothing about the physical representation of an attribute in the medium auxiliary memory.

Table 387 — MAM ATTRIBUTE format

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)	ATTRIBUTE IDENTIFIER						
1								(LSB)
2	READ ONLY	Reserved					FORMAT	
3	(MSB)	ATTRIBUTE LENGTH (n-4)						
4								(LSB)
5		ATTRIBUTE VALUE						
•••								
n								

The ATTRIBUTE IDENTIFIER field contains a code value identifying the attribute (see 7.4.2).

The READ ONLY bit indicates whether the attribute is in the read only state (see 5.13). If the READ ONLY bit is set to one, the attribute is in the read only state. If the READ ONLY bit is set to zero, the attribute is in the read/write state.

The FORMAT field (see table 388) specifies the format of the data in the ATTRIBUTE VALUE field.

Table 388 — MAM 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 4.4.1).
10b	TEXT	The attribute contains textual data. The character set is as described in the TEXT LOCALIZATION IDENTIFIER attribute (see 7.4.2.4.6).
11b		Reserved

The ATTRIBUTE LENGTH field specifies the length in bytes of the ATTRIBUTE VALUE field.

The ATTRIBUTE VALUE field contains the current value, for the READ ATTRIBUTE command (see 6.15), or intended value, for the WRITE ATTRIBUTE command (see 6.38), of the attribute.

7.4.2 Attribute identifier values

7.4.2.1 Attribute identifier values overview

The values in the ATTRIBUTE IDENTIFIER field (see 7.4.1) are assigned according to the attribute type (see 5.13) and whether the attribute is standard or vendor specific (see table 389).

Table 389 — MAM attribute identifier range assignments

Attribute Identifiers	Attribute Type	Standardized	Reference
0000h to 03FFh	Device	Yes	7.4.2.2
0400h to 07FFh	Medium	Yes	7.4.2.3
0800h to 0BFFh	Host	Yes	7.4.2.4
0C00h to 0FFFh	Device	Vendor specific	
1000h to 13FFh	Medium	Vendor specific	
1400h to 17FFh	Host	Vendor specific	
1800h to FFFFh	Reserved		

Device servers may accept and process a WRITE ATTRIBUTE command containing standardized host type attribute identifier values (i.e., 0800h-0BFFh) or vendor specific host type attribute identifier values (i.e., 1400h-17FFh). Standardized host type attribute identifier values may be checked as described in 7.4.2.4.

7.4.2.2 Device type attributes

Device type attributes (see table 390) shall be maintained and updated by the device server when the medium and associated medium auxiliary memory are present. All supported medium type attributes shall have a status of read only (see 5.13).

Table 390 — Device type attributes

Attribute Identifier	Name	Attribute Length (in bytes)	Format	Reference
0000h	REMAINING CAPACITY IN PARTITION	8	BINARY	7.4.2.2.1
0001h	MAXIMUM CAPACITY IN PARTITION	8	BINARY	7.4.2.2.1
0002h	Restricted (see SSC-3)			
0003h	LOAD COUNT	8	BINARY	7.4.2.2.2
0004h	MAM SPACE REMAINING	8	BINARY	7.4.2.2.3
0005h to 0006h	Restricted (see SSC-3)			
0007h	INITIALIZATION COUNT	2	BINARY	7.4.2.2.4
0008h	VOLUME IDENTIFIER	32	ASCII	7.4.2.2.5
0009h to 0209h	Reserved			
020Ah	DEVICE VENDOR/SERIAL NUMBER AT LAST LOAD	40	ASCII	7.4.2.2.6
020Bh	DEVICE VENDOR/SERIAL NUMBER AT LOAD-1	40	ASCII	7.4.2.2.6
020Ch	DEVICE VENDOR/SERIAL NUMBER AT LOAD-2	40	ASCII	7.4.2.2.6
020Dh	DEVICE VENDOR/SERIAL NUMBER AT LOAD-3	40	ASCII	7.4.2.2.6
020Eh to 021Fh	Reserved			
0220h	TOTAL MBYTES WRITTEN IN MEDIUM LIFE	8	BINARY	7.4.2.2.7
0221h	TOTAL MBYTES READ IN MEDIUM LIFE	8	BINARY	7.4.2.2.7
0222h	TOTAL MBYTES WRITTEN IN CURRENT/LAST LOAD	8	BINARY	7.4.2.2.8
0223h	TOTAL MBYTES READ IN CURRENT/LAST LOAD	8	BINARY	7.4.2.2.8
0224h	LOGICAL POSITION OF FIRST ENCRYPTED BLOCK	8	BINARY	7.4.2.2.9
0225h	LOGICAL POSITION OF FIRST UNENCRYPTED BLOCK AFTER THE FIRST ENCRYPTED BLOCK	8	BINARY	7.4.2.2.10
0226h to 033Fh	Reserved			
0340h	MEDIUM USAGE HISTORY	90	BINARY	7.4.2.2.11
0341h	PARTITION USAGE HISTORY	60	BINARY	7.4.2.2.12
0342h to 03FFh	Reserved			

7.4.2.2.1 REMAINING CAPACITY IN PARTITION and MAXIMUM CAPACITY IN PARTITION: Are native capacities (i.e., assuming no data compression for the specified medium partition). These values are expressed in increments of 1 048 576 bytes (e.g., a value of one means 1 048 576 bytes and a value of two means 2 097 152 bytes).

7.4.2.2.2 LOAD COUNT: Indicates how many times this medium has been fully loaded. This attribute should not be reset to zero by any action of the device server. The load counter is a saturating counter.

7.4.2.2.3 MAM SPACE REMAINING: Indicates the space currently available in the medium auxiliary memory. The total medium auxiliary memory capacity is reported in the MAM CAPACITY attribute (see 7.4.2.3.4).

NOTE 48 - It may not always be possible to utilize all of the available space in a given medium auxiliary memory implementation. Depending on the internal organization of the memory and the software that controls it, fragmentation issues may mean that certain attribute sizes may not be fully accommodated as the medium auxiliary memory nears its maximum capacity.

7.4.2.2.4 INITIALIZATION COUNT: Indicates the number of times that a device server has logically formatted the medium. This value is cumulative over the life of the medium and shall not be reset to zero. The initialization counter is a saturating counter.

7.4.2.2.5 VOLUME IDENTIFIER: Indicates the volume identifier (see SMC-3) of the medium. If the device server supports this attribute but does not have access to the volume identifier, it shall report this attribute with an attribute length value of zero.

7.4.2.2.6 DEVICE VENDOR/SERIAL NUMBER AT LAST LOAD, DEVICE VENDOR/SERIAL NUMBER AT LOAD –1, DEVICE VENDOR/SERIAL NUMBER AT LOAD –2 and DEVICE VENDOR/SERIAL NUMBER AT LOAD –3:

Give a history of the last four device servers in which the medium has been loaded. The format of the attributes is shown in table 391.

Table 391 — DEVICE VENDOR/SERIAL NUMBER attribute format

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)							
•••				T10 VENDOR IDENTIFICATION				
7								(LSB)
8	(MSB)							
•••				PRODUCT SERIAL NUMBER				
39								(LSB)

The T10 VENDOR IDENTIFICATION field shall be the same value returned in the Standard INQUIRY data (see 6.4.2).

The PRODUCT SERIAL NUMBER field contains ASCII data (see 4.4.1) that is a vendor specific serial number. If the product serial number is not available, the PRODUCT SERIAL NUMBER field shall contain ASCII spaces (20h).

7.4.2.2.7 TOTAL MBYTES WRITTEN IN MEDIUM LIFE and TOTAL MBYTES READ IN MEDIUM LIFE: Indicate the total number of data bytes that are transferred to or from the medium, after any data compression has been applied, over the entire medium life. These values are cumulative and shall not be reset to zero. These values are expressed in increments of 1 048 576 bytes (e.g., a value of one means 1 048 576 bytes and a value of two means 2 097 152 bytes).

7.4.2.2.8 TOTAL MBYTES WRITTEN IN CURRENT/LAST LOAD and TOTAL MBYTES READ IN CURRENT/ LAST LOAD: Indicate the total number of data bytes that are transferred to or from the medium, after any data compression has been applied, during the current load if the medium is currently loaded, or the last load if the medium is currently unloaded. The device server should reset these attributes to zero when the medium is loaded. These values are expressed in increments of 1 048 576 bytes (e.g., a value of one means 1 048 576 bytes and a value of two means 2 097 152 bytes).

7.4.2.2.9 LOGICAL POSITION OF FIRST ENCRYPTED BLOCK: Indicates the address of the first logical block on the medium that contains encrypted data. If no logical block in the partition contains encrypted data, then the attribute value shall be set to FFFF FFFF FFFF FFFFh. If it is unknown whether any logical block in the partition contains encrypted data, then the attribute value shall be set to FFFF FFFF FFFF FFFEh.

7.4.2.2.10 LOGICAL POSITION OF FIRST UNENCRYPTED BLOCK AFTER THE FIRST ENCRYPTED BLOCK: Indicates the address of the first logical block in the partition that contains unencrypted data and follows the first logical block in the partition that contains encrypted data. If this attribute is supported, then the LOGICAL POSITION OF FIRST ENCRYPTED BLOCK (see 7.4.2.2.9) attribute shall be supported.

The attribute value shall be set to FFFF FFFF FFFF FFFFh if the attribute value for the LOGICAL POSITION OF FIRST ENCRYPTED BLOCK is set to:

a) FFFF FFFF FFFF FFFFh; or
b) any value other than FFFF FFFF FFFF FFFFh or FFFF FFFF FFFF FFFEh and no logical block in the partition after the first encrypted logical block contains unencrypted data.

The attribute value shall be set to FFFF FFFF FFFF FFFEh if the attribute value for the LOGICAL POSITION OF FIRST ENCRYPTED BLOCK is set to:

a) FFFF FFFF FFFF FFFEh; or
b) any value other than FFFF FFFF FFFF FFFFh or FFFF FFFF FFFF FFFEh and it is unknown whether any logical block in the partition after the first encrypted logical block contains unencrypted data.

7.4.2.2.11 MEDIUM USAGE HISTORY: Provides saturating counters (see table 392) for the entire medium. The value in each field is the sum for all partitions. If a field is not used, it should be set to zero.

Table 392 — MEDIUM USAGE HISTORY attribute format (part 1 of 2)

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)							
•••				CURRENT AMOUNT OF DATA WRITTEN				
5								(LSB)
6	(MSB)							
•••				CURRENT WRITE RETRIES COUNT				
11								(LSB)
12	(MSB)							
•••				CURRENT AMOUNT OF DATA READ				
17								(LSB)
18	(MSB)							
•••				CURRENT READ RETRIES COUNT				
23								(LSB)
24	(MSB)							
•••				PREVIOUS AMOUNT OF DATA WRITTEN				
29								(LSB)
30	(MSB)							
•••				PREVIOUS WRITE RETRIES COUNT				
35								(LSB)

Table 392 — MEDIUM USAGE HISTORY attribute format (part 2 of 2)

Bit / Byte	7	6	5	4	3	2	1	0
36	(MSB)							
•••				PREVIOUS AMOUNT OF DATA READ				
41								(LSB)
42	(MSB)							
•••				PREVIOUS READ RETRIES COUNT				
47								(LSB)
48	(MSB)							
•••				TOTAL AMOUNT OF DATA WRITTEN				
53								(LSB)
54	(MSB)							
•••				TOTAL WRITE RETRIES COUNT				
59								(LSB)
60	(MSB)							
•••				TOTAL AMOUNT OF DATA READ				
65								(LSB)
66	(MSB)							
•••				TOTAL READ RETRIES COUNT				
71								(LSB)
72	(MSB)							
•••				LOAD COUNT				
77								(LSB)
78	(MSB)							
•••				TOTAL CHANGE PARTITION COUNT				
83								(LSB)
84	(MSB)							
•••				TOTAL PARTITION INITIALIZE COUNT				
89								(LSB)

The CURRENT AMOUNT OF DATA WRITTEN field indicates the amount of data written to the medium during this load of the medium. This value is expressed in mebibytes (see 3.5.2).

The CURRENT WRITE RETRIES COUNT field indicates the total number of times a write retry occurred during this load of the medium.[1)]

The CURRENT AMOUNT OF DATA READ field indicates the amount of data read from the medium during this load of the medium. This value is expressed in mebibytes (see 3.5.2).

The CURRENT READ RETRIES COUNT field indicates the number of times a read retry occurred during this load of the medium.[1)]

1) The definition of one retry as counted by this attribute field is not part of this standard. This count should not be used to compare products because the products may define errors differently.

The PREVIOUS AMOUNT OF DATA WRITTEN field indicates the amount of data written to the medium during the previous load of the medium. This value is expressed in mebibytes (see 3.5.2).

The PREVIOUS WRITE RETRIES COUNT field indicates the total number of times a write retry occurred during the previous load of the medium.[1)]

The PREVIOUS AMOUNT OF DATA READ field indicates the amount of data read from the medium during the previous load of the medium. This value is expressed in mebibytes (see 3.5.2).

The PREVIOUS READ RETRIES COUNT field indicates the number of times a read retry occurred during the previous load of the medium.[1)]

The TOTAL AMOUNT OF DATA WRITTEN field indicates the amount of data written to the medium since the last medium format. This value is expressed in mebibytes (see 3.5.2).

The TOTAL WRITE RETRIES COUNT field indicates the total number of times a write retry occurred since the last medium format.[1)]

The TOTAL AMOUNT OF DATA READ field indicates the amount of data read from the medium since the last medium format. This value is expressed in mebibytes (see 3.5.2).

The TOTAL READ RETRIES COUNT field indicates the number of times a read retry occurred since the last medium format.[1)]

The LOAD COUNT field indicates the number of loads since the last medium format. This count accumulates over the life of the medium but it is reset to zero after a medium format.

The TOTAL CHANGE PARTITION COUNT field indicates the number of times that switches between partitions have been performed on the medium. This count accumulates over the life of the medium but it is reset to zero after a medium format.

The TOTAL PARTITION INITIALIZE COUNT field indicates number of times that any of the partitions on the medium have been erased. This count accumulates over the life of the medium but it is reset to zero after a medium format.

7.4.2.2.12 PARTITION USAGE HISTORY: Provides saturating counters (see table 393) for the partition specified by the PARTITION NUMBER field in the CDB. If a field is not used, it should be set to zero.

Table 393 — PARTITION USAGE HISTORY attribute format (part 1 of 2)

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)							
•••				CURRENT AMOUNT OF DATA WRITTEN				
3								(LSB)
4	(MSB)							
•••				CURRENT WRITE RETRIES COUNT				
7								(LSB)
8	(MSB)							
•••				CURRENT AMOUNT OF DATA READ				
11								(LSB)

Table 393 — PARTITION USAGE HISTORY attribute format (part 2 of 2)

Bit / Byte	7	6	5	4	3	2	1	0
12	(MSB)							
•••				CURRENT READ RETRIES COUNT				
15								(LSB)
16	(MSB)							
•••				PREVIOUS AMOUNT OF DATA WRITTEN				
19								(LSB)
20	(MSB)							
•••				PREVIOUS WRITE RETRIES COUNT				
23								(LSB)
24	(MSB)							
•••				PREVIOUS AMOUNT OF DATA READ				
27								(LSB)
28	(MSB)							
•••				PREVIOUS READ RETRIES COUNT				
31								(LSB)
32	(MSB)							
•••				TOTAL AMOUNT OF DATA WRITTEN				
35								(LSB)
36	(MSB)							
•••				TOTAL WRITE RETRIES COUNT				
39								(LSB)
40	(MSB)							
•••				TOTAL AMOUNT OF DATA READ				
43								(LSB)
44	(MSB)							
•••				TOTAL READ RETRIES COUNT				
47								(LSB)
48	(MSB)							
•••				LOAD COUNT				
51								(LSB)
52	(MSB)							
•••				CHANGE PARTITION COUNT				
55								(LSB)
56	(MSB)							
•••				PARTITION INITIALIZE COUNT				
59								(LSB)

The CURRENT AMOUNT OF DATA WRITTEN field indicates the amount of data written to the medium in the partition specified by the PARTITION NUMBER field in the CDB during this load of the medium. This value is expressed in mebibytes (see 3.5.2).

The CURRENT WRITE RETRIES COUNT field indicates the total number of times a write retry occurred in the partition specified by the PARTITION NUMBER field in the CDB during this load of the medium.[2)]

The CURRENT AMOUNT OF DATA READ field indicates the amount of data read from the medium in the partition specified by the PARTITION NUMBER field in the CDB during this load of the medium. This value is expressed mebibytes (see 3.5.2).

The CURRENT READ RETRIES COUNT field indicates the number of times a read retry occurred in the partition specified by the PARTITION NUMBER field in the CDB during this load of the medium.[2)]

The PREVIOUS AMOUNT OF DATA WRITTEN field indicates the amount of data written to the medium in the partition specified by the PARTITION NUMBER field in the CDB during the previous load of the medium. This value is expressed in mebibytes (see 3.5.2).

The PREVIOUS WRITE RETRIES COUNT field indicates the total number of times a write retry occurred in the partition specified by the PARTITION NUMBER field in the CDB during the previous load of the medium.[2)]

The PREVIOUS AMOUNT OF DATA READ field indicates the amount of data read from the medium in the partition specified by the PARTITION NUMBER field in the CDB during the previous load of the medium. This value is expressed in mebibytes (see 3.5.2).

The PREVIOUS READ RETRIES COUNT field indicates the number of times a read retry occurred in the partition specified by the PARTITION NUMBER field in the CDB during the previous load of the medium.[2)]

The TOTAL AMOUNT OF DATA WRITTEN field indicates the amount of data written to the medium in the partition specified by the PARTITION NUMBER field in the CDB since the last medium format. This value is expressed in mebibytes (see 3.5.2).

The TOTAL WRITE RETRIES COUNT field indicates the total number of times a write retry occurred in the partition specified by the PARTITION NUMBER field in the CDB since the last medium format.[2)]

The TOTAL AMOUNT OF DATA READ field indicates the amount of data read from the medium in the partition specified by the PARTITION NUMBER field in the CDB since the last medium format. This value is expressed in mebibytes (see 3.5.2).

The TOTAL READ RETRIES COUNT field indicates the number of times a read retry occurred in the partition specified by the PARTITION NUMBER field in the CDB since the last medium format.[2)]

The LOAD COUNT field indicates the number of loads in the partition specified by the PARTITION NUMBER field in the CDB since the last medium format. This count accumulates over the life of the medium but it is reset to zero after a medium format.

The TOTAL CHANGE PARTITION COUNT field indicates the number of times that switches to the partition specified by the PARTITION NUMBER field in the CDB have been performed on the medium. This count accumulates over the life of the medium but it is reset to zero after a medium format.

The TOTAL PARTITION INITIALIZE COUNT field indicates number of times that the partition specified by the PARTITION NUMBER field in the CDB has been initialized. This count accumulates over the life of the medium but it is reset to zero after a medium format.

---

2) The definition of one retry as counted by this attribute field is not part of this standard. This count should not be used to compare products because the products may define errors differently.

7.4.2.3 Medium type attributes

Medium type attributes (see table 394) are stored in the medium auxiliary memory by the manufacturer. The device server shall not alter medium type attributes. All supported medium type attributes shall have a status of read only (see 5.13).

Table 394 — Medium type attributes

Attribute Identifier	Name	Attribute Length (in bytes)	Format	Reference
0400h	MEDIUM MANUFACTURER	8	ASCII	7.4.2.3.1
0401h	MEDIUM SERIAL NUMBER	32	ASCII	7.4.2.3.2
0402h to 0405h	Restricted (see SSC-3)			
0406h	MEDIUM MANUFACTURE DATE	8	ASCII	7.4.2.3.3
0407h	MAM CAPACITY	8	BINARY	7.4.2.3.4
0408h	MEDIUM TYPE	1	BINARY	7.4.2.3.5
0409h	MEDIUM TYPE INFORMATION	2	BINARY	7.4.2.3.5
040Ah	NUMERIC MEDIUM SERIAL NUMBER	unspecified	unspecified	7.4.2.3.6
040Bh to 07FFh	Reserved			

7.4.2.3.1 MEDIUM MANUFACTURER: Contains eight bytes of left-aligned ASCII data (see 4.4.1) identifying the vendor of the media. The medium manufacturer shall be a T10 vendor identification assigned by INCITS. A list of assigned T10 vendor identifications is in Annex E and on the T10 web site (http://www.T10.org).

NOTE 49 - The T10 web site (http://www.t10.org) provides a convenient means to request an identification code.

7.4.2.3.2 MEDIUM SERIAL NUMBER: Contains the manufacturer's serial number for the medium.

7.4.2.3.3 MEDIUM MANUFACTURE DATE: Contains the date of manufacture of the medium. 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).

7.4.2.3.4 MAM CAPACITY: Is the total capacity of the medium auxiliary memory, in bytes, at manufacture time. It does not indicate the available space of an unused medium auxiliary memory because some of the medium auxiliary memory space may be reserved for device-specific use making it inaccessible to the application client.

7.4.2.3.5 MEDIUM TYPE and MEDIUM TYPE INFORMATION: Give information about non-data media and other types of media. The MEDIUM TYPE INFORMATION attribute is interpreted according to the type of medium indicated by the MEDIUM TYPE (see table 395).

Table 395 — MEDIUM TYPE and MEDIUM TYPE INFORMATION attributes

MEDIUM TYPE	Description	MEDIUM TYPE INFORMATION
00h	Data medium	Reserved
01h	Cleaning medium	Maximum number of cleaning cycles permitted
02h to 7Fh	Reserved	Reserved
80h	Write-once medium	Reserved
81h to FFh	Reserved	Reserved

7.4.2.3.6 NUMERIC MEDIUM SERIAL NUMBER: Contains the manufacturer's serial number for the medium in a vendor specific format.

7.4.2.4 Host type attributes

Application clients may use the WRITE ATTRIBUTE and READ ATTRIBUTE commands to maintain the attributes shown in table 396. All existent host type attributes shall have a status of read/write (see 5.13).

Table 396 — Host type attributes

Attribute Identifier	Name	Attribute Length (in bytes)	Format	Reference
0800h	APPLICATION VENDOR	8	ASCII	7.4.2.4.1
0801h	APPLICATION NAME	32	ASCII	7.4.2.4.2
0802h	APPLICATION VERSION	8	ASCII	7.4.2.4.3
0803h	USER MEDIUM TEXT LABEL	160	TEXT	7.4.2.4.4
0804h	DATE AND TIME LAST WRITTEN	12	ASCII	7.4.2.4.5
0805h	TEXT LOCALIZATION IDENTIFIER	1	BINARY	7.4.2.4.6
0806h	BARCODE	32	ASCII	7.4.2.4.7
0807h	OWNING HOST TEXTUAL NAME	80	TEXT	7.4.2.4.8
0808h	MEDIA POOL	160	TEXT	7.4.2.4.9
0809h	PARTITION USER TEXT LABEL	16	ASCII	7.4.2.4.10
080Ah	LOAD/UNLOAD AT PARTITION	1	BINARY	7.4.2.4.11
080Bh	APPLICATION FORMAT VERSION	16	ASCII	7.4.2.4.12
080Ch	Restricted (see SSC-4)			
080Dh to BFFh	Reserved			

7.4.2.4.1 APPLICATION VENDOR: Contains eight bytes of left-aligned ASCII data (see 4.4.1) identifying the manufacturer of the application client (e.g., class driver or backup program) that last sent a WRITE ATTRIBUTE command to the device server while this medium auxiliary memory was accessible. The application vendor shall be a T10 vendor identification assigned by INCITS. A list of assigned T10 vendor identifications is in Annex E and on the T10 web site (http://www.T10.org).

NOTE 50 - The T10 web site (http://www.t10.org) provides a convenient means to request an identification code.

7.4.2.4.2 APPLICATION NAME: Contains the name of the application client.

7.4.2.4.3 APPLICATION VERSION: Contains the version of the application client.

7.4.2.4.4 USER MEDIUM TEXT LABEL: Is the user level identifier for the medium.

7.4.2.4.5 DATE & TIME LAST WRITTEN: Contains when the application client last wrote to the medium auxiliary memory. The format is YYYYMMDDHHMM (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 followed by two numeric ASCII characters between 00 and 24 for the hour followed by two numeric ASCII characters for the minute with no intervening spaces).

7.4.2.4.6 TEXT LOCALIZATION IDENTIFIER: Defines the character set (see table 397) used for attributes with a TEXT format (see 7.4.1).

Table 397 — TEXT LOCALIZATION IDENTIFIER attribute values

Value	Meaning
00h	No code specified (ASCII)
01h	ISO/IEC 8859-1 (Europe, Latin America)
02h	ISO/IEC 8859-2 (Eastern Europe)
03h	ISO/IEC 8859-3 (SE Europe/miscellaneous)
04h	ISO/IEC 8859-4 (Scandinavia/Baltic)
05h	ISO/IEC 8859-5 (Cyrillic)
06h	ISO/IEC 8859-6 (Arabic)
07h	ISO/IEC 8859-7 (Greek)
08h	ISO/IEC 8859-8 (Hebrew)
09h	ISO/IEC 8859-9 (Latin 5)
0Ah	ISO/IEC 8859-10 (Latin 6)
0Bh to 7Fh	Reserved
80h	ISO/IEC 10646-1 (UCS-2BE)
81h	ISO/IEC 10646-1 (UTF-8)
82h to FFh	Reserved

7.4.2.4.7 BARCODE: Is contents of a barcode associated with the medium in the medium auxiliary memory.

7.4.2.4.8 OWNING HOST TEXTUAL NAME: Indicates the host from which that USER MEDIUM TEXT LABEL (see 7.4.2.4.4) originates.

7.4.2.4.9 MEDIA POOL: Indicates the media pool to which this medium belongs.

7.4.2.4.10 PARTITION USER TEXT LABEL: Is a user level identifier for the partition specified by the PARTITION NUMBER field in the CDB.

7.4.2.4.11 LOAD/UNLOAD AT PARTITION: Indicates whether the media is capable of being loaded or unloaded at the partition specified by the PARTITION NUMBER field in the CDB. If loads and unloads are enabled for the specified partition, the value of this attribute shall be one. If loads and unloads are not enabled for the specified partition, the value of this attribute shall be zero. All attribute values other than zero and one are reserved. If LOAD/UNLOAD AT PARTITION is disabled, then loads and unloads are performed at the beginning of the media instead of at the specified partition. If this attribute is in the nonexistent state (see 5.13), then the default action shall be to load and unload at the beginning of media.

7.4.2.4.12 APPLICATION FORMAT VERSION: Indicates the version of the format being used by the application that set this attribute.

7.5 Mode parameters

7.5.1 Summary of mode page codes

The page code assignments for mode pages are summarized in table 398.

Table 398 — Summary of mode page codes

Mode page name	Page code	Subpage code	Reference
Control	0Ah	n/a [a]	7.5.7
Control Extension	0Ah	01h	7.5.8
Disconnect-Reconnect	02h	n/a [a]	7.5.9
Extended	15h	n/a [a]	7.5.10
Extended Device-Type Specific	16h	n/a [a]	7.5.11
Power Condition	1Ah	n/a [a]	7.5.12
Protocol Specific Logical Unit	18h	n/a [a]	7.5.13
Protocol Specific Port	19h	n/a [a]	7.5.14
Restricted (see applicable protocol standard)	18h	01h to FEh	
	19h	01h to FEh	
Restricted (see applicable command standard)	01h	00h to FEh	
	03h to 08h	00h to FEh	
	0Ah	F0h to FEh	
	0Bh to 14h	00h to FEh	
	1Ah	F0h to FEh	
	1Bh to 1Fh	00h to FEh	
	20h to 3Eh	00h to FEh	
Return all pages	3Fh [b]	00h [b]	6.11 and 6.12
Return all pages and subpages	3Fh [b]	FFh [b]	6.11 and 6.12
Return all subpages	00h to 3Eh [b]	FFh [b]	6.11 and 6.12
Obsolete [c]			
Vendor specific [d]			
Reserved	All other codes		
A numeric ordered listing of mode page and subpage codes is provided in D.6.			

[a] Applicable only in the MODE SENSE command (see 6.11 and 6.12).
[b] The use of these page codes and subpage codes is described in table 164 (see 6.11).
[c] The following page codes are obsolete: 09h.
[d] Page code 00h is vendor specific and does not require a page format or a subpage code.

7.5.2 Mode parameters overview

This subclause describes the mode parameter headers, block descriptors, and mode pages used with MODE SELECT command (see 6.9 and 6.10) and MODE SENSE command (see 6.11 and 6.12) that are applicable to all

SCSI devices. Subpages are identical to mode pages except that they include a SUBPAGE CODE field that further differentiates the mode page contents. Mode pages specific to each device type are described in the command standard (see 3.1.31) that applies to that device type.

7.5.3 Mode parameter list format

The mode parameter list shown in table 399 contains a header, followed by zero or more block descriptors, followed by zero or more variable-length mode pages. Parameter lists are defined for each device type.

Table 399 — Mode parameter list

Bit / Byte	7	6	5	4	3	2	1	0
	Mode parameter header							
	Block descriptor(s)							
	Mode page(s) or vendor specific (e.g., page code set to zero)							

7.5.4 Mode parameter header formats

The mode parameter header that is used by the MODE SELECT(6) command (see 6.9) and the MODE SENSE(6) command (see 6.11) is defined in table 400.

Table 400 — Mode parameter header(6)

Bit / Byte	7	6	5	4	3	2	1	0
0	MODE DATA LENGTH							
1	MEDIUM TYPE							
2	DEVICE-SPECIFIC PARAMETER							
3	BLOCK DESCRIPTOR LENGTH							

The mode parameter header that is used by the MODE SELECT(10) command (see 6.10) and the MODE SENSE(10) command (see 6.12) is defined in table 401.

Table 401 — Mode parameter header(10)

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)	MODE DATA LENGTH						
1								(LSB)
2	MEDIUM TYPE							
3	DEVICE-SPECIFIC PARAMETER							
4	Reserved							LONGLBA
5	Reserved							
6	(MSB)	BLOCK DESCRIPTOR LENGTH						
7								(LSB)

When using the MODE SENSE command, the MODE DATA LENGTH field indicates the length in bytes of the following data that is available to be transferred. The mode data length does not include the number of bytes in the MODE DATA LENGTH field. When using the MODE SELECT command, this field is reserved.

NOTE 51 - Logical units that support more than 256 bytes of block descriptors and mode pages may need to implement ten-byte mode commands. The mode data length field in the six-byte CDB header limits the returned data to 256 bytes.

The contents of the MEDIUM TYPE field are unique for each device type. Refer to the mode parameters subclause of the specific device type command standard (see 3.1.31) for definition of these values. Some device types reserve this field.

The DEVICE-SPECIFIC PARAMETER field is unique for each device type. Refer to the mode parameters subclause of the specific device type command standard (see 3.1.31) for definition of this field. Some device types reserve all or part of this field.

If the Long LBA (LONGLBA) bit is set to zero, the mode parameter block descriptor(s), if any, are each eight bytes long and have the format described in 7.5.5.1. If the LONGLBA bit is set to one, the mode parameter block descriptor(s), if any, are each sixteen bytes long and have a format described in a command standard (see 3.1.31).

The BLOCK DESCRIPTOR LENGTH field contains the length in bytes of all the block descriptors. It is equal to the number of block descriptors times eight if the LONGLBA bit is set to zero or times sixteen if the LONGLBA bit is set to one, and does not include mode pages or vendor specific parameters (e.g., page code set to zero), if any, that may follow the last block descriptor. A block descriptor length of zero indicates that no block descriptors are included in the mode parameter list. This condition shall not be considered an error.

7.5.5 Mode parameter block descriptor formats

7.5.5.1 General block descriptor format

When the LONGLBA bit is set to zero (see 7.5.4), the mode parameter block descriptor format for all device types except direct access block devices (see SBC-2) is shown in table 402.

Table 402 — General mode parameter block descriptor

Bit / Byte	7	6	5	4	3	2	1	0
0	DENSITY CODE							
1	(MSB)							
2	NUMBER OF BLOCKS							
3								(LSB)
4	Reserved							
5	(MSB)							
6	BLOCK LENGTH							
7								(LSB)

Block descriptors specify some of the medium characteristics for all or part of a logical unit. Support for block descriptors is optional. Each block descriptor contains a DENSITY CODE field, a NUMBER OF BLOCKS field, and a BLOCK LENGTH field. Block descriptor values are always current (i.e., saving is not supported). Whenever any block descriptor values are changed, the device server shall establish a unit attention condition (see SAM-4) for the initiator port associated with every I_T nexus except the I_T nexus on which the MODE SELECT command (see

6.9) was received, with the additional sense code set to MODE PARAMETERS CHANGED. Command standards (see 3.1.31) may place additional requirements on the general mode parameter block descriptor. Requirements in the command standards that conflict with requirements defined in this subclause shall take precedence over the requirements defined in this subclause.

The DENSITY CODE field is unique for each device type. Refer to the mode parameters subclause of the specific device type command standard (see 3.1.31) for definition of this field. Some device types reserve all or part of this field.

The NUMBER OF BLOCKS field specifies the number of logical blocks on the medium to which the DENSITY CODE field and BLOCK LENGTH field apply. A value of zero indicates that all of the remaining logical blocks of the logical unit shall have the medium characteristics specified.

If the number of logical blocks on the medium exceeds the maximum value that may be specified in the NUMBER OF BLOCKS field, a value of FFFFFFh indicates that all of the remaining logical blocks of the logical unit shall have the medium characteristics specified.

NOTES
52 There may be implicit association between parameters defined in the mode pages and block descriptors. In this case, the device server may change parameters not explicitly sent with the MODE SELECT command. A subsequent MODE SENSE command may be used to detect these changes.
53 The number of remaining logical blocks may be unknown for some device types.

The BLOCK LENGTH field specifies the length in bytes of each logical block described by the block descriptor. For sequential-access devices, a block length of zero indicates that the logical block size written to the medium is specified by the TRANSFER LENGTH field in the CDB (see SSC-2).

7.5.6 Mode page and subpage formats and page codes

The page_0 mode page format is defined in table 403.

Table 403 — Page_0 mode page format

Bit / Byte	7	6	5	4	3	2	1	0
0	PS	SPF (0b)	PAGE CODE					
1	PAGE LENGTH (n-1)							
2	Mode parameters							
•••								
n								

The sub_page mode page format is defined in table 404.

Table 404 — Sub_page mode page format

Bit / Byte	7	6	5	4	3	2	1	0
0	PS	SPF (1b)	PAGE CODE					
1	SUBPAGE CODE							
2	(MSB)	PAGE LENGTH (n-3)						
3								(LSB)
4	Mode parameters							
•••								
n								

Each mode page contains a PS bit, an SPF bit, a PAGE CODE field, a PAGE LENGTH field, and a set of mode parameters. The page codes are defined in this subclause and in the mode parameter subclauses in the command standard (see 3.1.31) for the specific device type. Each mode page with a SPF bit set to one contains a SUBPAGE CODE field.

A SubPage Format (SPF) bit set to zero indicates that the page_0 mode page format is being used. A SPF bit set to one indicates that the sub_page mode page format is being used.

When using the MODE SENSE command, a parameters saveable (PS) bit set to one indicates that the mode page may be saved by the logical unit in a nonvolatile, vendor specific location. A PS bit set to zero indicates that the device server is not able to save the supported parameters. When using the MODE SELECT command, the PS bit is reserved.

The PAGE CODE field and SUBPAGE CODE field (see 7.5.1) identify the format and parameters defined for that mode page. Some page codes are defined as applying to all device types and other page codes are defined for the specific device type. The page codes that apply to a specific device type are defined in the command standard for that device type. The applicability of each subpage code matches that of the page code with which it is associated.

If a mode page is in the page_0 mode page format, then the mode page shall be the one for which the contents of a MODE SENSE command SUBPAGE CODE field (see 6.11.1) are set to zero.

When using the MODE SENSE command, if page code 00h (vendor specific mode page) is implemented, the device server shall return that mode page last in response to a request to return all mode pages (page code 3Fh). When using the MODE SELECT command, this mode page should be sent last.

The PAGE LENGTH field specifies the length in bytes of the mode parameters that follow. If the application client does not set this value to the value that is returned for the mode page by the MODE SENSE command, 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. The logical unit may implement a mode page that is less than the full mode page length defined, provided no field is truncated and the PAGE LENGTH field correctly specifies the actual length implemented.

The mode parameters for each mode page are defined in the following subclauses, or in the mode parameters subclause in the command standard (see 3.1.31) for the specific device type. Mode parameters not implemented by the logical unit shall be set to zero.

7.5.7 Control mode page

The Control mode page (see table 405) provides controls over SCSI features that are applicable to all device types (e.g., task set management and error logging). If a field in this mode page is changed while there is a command already in the task set, it is vendor specific whether the old or new value of the field applies to that command. The mode page policy (see 6.9) for this mode page shall be shared, or per I_T nexus.

Table 405 — Control mode page

Bit / Byte	7	6	5	4	3	2	1	0
0	PS	SPF (0b)	PAGE CODE (0Ah)					
1	PAGE LENGTH (0Ah)							
2	TST			TMF_ONLY	DPICZ	D_SENSE	GLTSD	RLEC
3	QUEUE ALGORITHM MODIFIER				NUAR	QERR		Obsolete
4	VS	RAC	UA_INTLCK_CTRL		SWP	Obsolete		
5	ATO	TAS	ATMPE	RWWP	Reserved	AUTOLOAD MODE		
6	Obsolete							
7								
8	(MSB)	BUSY TIMEOUT PERIOD						
9								(LSB)
10	(MSB)	EXTENDED SELF-TEST COMPLETION TIME						
11								(LSB)

The PS bit, SPF bit, PAGE CODE field, and PAGE LENGTH field are described in 7.5.6.

A task set type (TST) field (see table 406) specifies the type of task set in the logical unit.

Table 406 — Task set type (TST) field

Code	Description
000b	The logical unit maintains one task set for all I_T nexuses
001b	The logical unit maintains separate task sets for each I_T nexus
010b to 111b	Reserved

Regardless of the mode page policy (see 6.9) for the Control mode page, the shared mode page policy shall be applied to the TST field. If the most recent MODE SELECT changes the setting of this field, then the device server shall establish a unit attention condition (see SAM-4) 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 MODE PARAMETERS CHANGED.

The allow task management functions only (TMF_ONLY) bit set to zero specifies that the device server shall process commands with the ACA task attribute received on the faulted I_T nexus (see 3.1.58) when an ACA condition has been established (see SAM-4). A TMF_ONLY bit set to one specifies that the device server shall complete all commands received on the faulted I_T nexus with an ACA ACTIVE status when an ACA condition has been established.

A disable protection information check if protect field is zero (DPICZ) bit set to zero indicates that checking of protection information bytes is enabled. A DPICZ bit set to one indicates that checking of protection information is disabled on commands with:

a) the RDPROTECT field (see SBC-3) set to zero;
b) the VRPROTECT field (see SBC-3) set to zero; or
c) the ORPROTECT field (see SBC-3) set to zero.

A descriptor format sense data (D_SENSE) bit set to zero specifies that the device server shall return fixed format sense data (see 4.5.3) when returning sense data in the same I_T_L_Q nexus transaction (see 3.1.67) as the status. A D_SENSE bit set to one specifies that the device server shall return descriptor format sense data (see 4.5.2) when returning sense data in the same I_T_L_Q nexus transaction as the status, except as defined in 4.5.1. If an application client enables reporting of referrals in sense data (see SBC-3), then the application client should be able to handle up to 252 bytes of sense data.

A global logging target save disable (GLTSD) bit set to zero specifies that the logical unit implicitly saves, at vendor specific intervals, each log parameter in which the TSD bit (see 7.3) is set to zero. A GLTSD bit set to one specifies that the logical unit shall not implicitly save any log parameters.

A report log exception condition (RLEC) bit set to one specifies that the device server shall report log exception conditions as described in 7.3. A RLEC bit set to zero specifies that the device server shall not report log exception conditions.

The QUEUE ALGORITHM MODIFIER field (see table 407) specifies restrictions on the algorithm used for reordering commands having the SIMPLE task attribute (see SAM-4).

Table 407 — QUEUE ALGORITHM MODIFIER field

Code	Description
0h	Restricted reordering
1h	Unrestricted reordering allowed
2h to 7h	Reserved
8h to Fh	Vendor specific

A value of zero in the QUEUE ALGORITHM MODIFIER field specifies that the device server shall order the processing sequence of commands having the SIMPLE task attribute such that data integrity is maintained for that I_T nexus (i.e., if the transmission of new SCSI transport protocol requests is halted at any time, the final value of all data observable on the medium shall be the same as if all the commands had been processed with the ORDERED task attribute).

A value of one in the QUEUE ALGORITHM MODIFIER field specifies that the device server may reorder the processing sequence of commands having the SIMPLE task attribute in any manner. Any data integrity exposures related to command sequence order shall be explicitly handled by the application client through the selection of appropriate commands and task attributes.

A no unit attention on release (NUAR) bit set to one specifies that the device server shall not establish a unit attention condition as described in 5.9.11.2. A NUAR bit set to zero specifies that the device server shall establish a unit attention condition as described in 5.9.11.2.

The queue error management (QERR) field (see table 408) specifies how the device server shall handle other commands when one command is terminated with CHECK CONDITION status (see SAM-4). The task set type (see the TST field definition in this subclause) defines which other commands are affected. If the TST field equals 000b, then all commands from all I_T nexuses are affected. If the TST field equals 001b, then only commands from the same I_T nexus as the command that is terminated with CHECK CONDITION status are affected.

Table 408 — Queue error management (QERR) field

Code	Definition
00b	If an ACA condition is established, the affected commands in the task set shall resume after the ACA condition is cleared (see SAM-4). Otherwise, all commands other than the command that received the CHECK CONDITION status shall be processed as if no error occurred.
01b	All the affected commands in the task set shall be aborted when the CHECK CONDITION status is sent. If the TAS bit is set to zero, the device server shall establish a unit attention condition (see SAM-4) for the initiator port associated with every I_T nexus that had commands aborted except for the I_T nexus on which the CHECK CONDITION status was returned, with the additional sense code set to COMMANDS CLEARED BY ANOTHER INITIATOR. If the TAS bit is set to one, all affected commands in the task set for I_T nexuses other than the I_T nexus for which the CHECK CONDITION status was sent shall be completed with TASK ABORTED status and no unit attention shall be established. For the I_T nexus to which the CHECK CONDITION status is sent, no status shall be sent for the commands that are aborted.
10b	Reserved
11b	Affected commands in the task set belonging to the I_T nexus on which a CHECK CONDITION status is returned shall be aborted when the status is sent.

The report a check (RAC) bit provides control of reporting long busy conditions or CHECK CONDITION status. A RAC bit set to one specifies that the device server should return CHECK CONDITION status rather than returning BUSY status if the reason for returning the BUSY status may persist for a longer time than that specified by the BUSY TIMEOUT PERIOD field. A RAC bit set to zero specifies that the device server may return BUSY status regardless of the length of time the reason for returning BUSY status may persist.

The unit attention interlocks control (UA_INTLCK_CTRL) field (see table 409) controls the clearing of unit attention conditions reported in the same I_T_L_Q nexus transaction (see 3.1.67) as a CHECK CONDITION status and whether returning a status of BUSY, TASK SET FULL or RESERVATION CONFLICT results in the establishment of a unit attention condition (see SAM-4).

Table 409 — Unit attention interlocks control (UA_INTLCK_CTRL) field

Code	Definition
00b	The logical unit shall clear any unit attention condition reported in the same I_T_L_Q nexus transaction as a CHECK CONDITION status and shall not establish a unit attention condition when a command is completed with BUSY, TASK SET FULL, or RESERVATION CONFLICT status.
01b	Reserved [a]
10b [a]	The logical unit shall not clear any unit attention condition reported in the same I_T_L_Q nexus transaction as a CHECK CONDITION status and shall not establish a unit attention condition when a command is completed with BUSY, TASK SET FULL, or RESERVATION CONFLICT status.
11b [a]	The logical unit shall not clear any unit attention condition reported in the same I_T_L_Q nexus transaction as a CHECK CONDITION status and shall establish a unit attention condition for the initiator port associated with the I_T nexus on which the BUSY, TASK SET FULL, or RESERVATION CONFLICT status is being returned. Depending on the status, the additional sense code shall be set to PREVIOUS BUSY STATUS, PREVIOUS TASK SET FULL STATUS, or PREVIOUS RESERVATION CONFLICT STATUS. Until it is cleared by a REQUEST SENSE command, a unit attention condition shall be established only once for a BUSY, TASK SET FULL, or RESERVATION CONFLICT status regardless to the number of commands completed with one of those status codes.
[a] A REQUEST SENSE command still clears any unit attention condition that it reports.	

A software write protect (SWP) bit set to one specifies that the logical unit shall inhibit writing to the medium after writing all cached or buffered write data, if any. When SWP is one, all commands requiring writes to the medium shall be terminated with CHECK CONDITION status, with the sense key set to DATA PROTECT, and the additional sense code set to WRITE PROTECTED. When SWP is one and the device type's command standard (see 3.1.31) defines a write protect (WP) bit in the DEVICE-SPECIFIC PARAMETER field in the mode parameter header, the WP bit shall be set to one for subsequent MODE SENSE commands. A SWP bit set to zero specifies that the logical unit may allow writing to the medium, depending on other write inhibit mechanisms implemented by the logical unit. When the SWP bit is set to zero, the value of the WP bit, if defined, is device type specific. For a list of commands affected by the SWP bit and details of the WP bit see the command standard for the specific device type.

An application tag owner (ATO) bit set to zero specifies that the device server may modify the contents of the LOGICAL BLOCK APPLICATION TAG field and, depending on the protection type, may modify the contents of the LOGICAL BLOCK REFERENCE TAG field (see SBC-3). If the ATO bit is set to one the device server shall not modify the LOGICAL BLOCK APPLICATION TAG field and, depending on the protection type, shall not modify the contents of the LOGICAL BLOCK REFERENCE TAG field.

A task aborted status (TAS) bit set to zero specifies that aborted commands shall be terminated by the device server without any response to the application client. A TAS bit set to one specifies that commands aborted by the actions of an I_T nexus other than the I_T nexus on which the command was received shall be completed with TASK ABORTED status (see SAM-4).

An application tag mode page enabled (ATMPE) bit set to zero specifies that the Application Tag mode page (see SBC-3) is disabled and the contents of logical block application tags are not defined by this standard. An ATMPE bit set to one specifies that the Application Tag mode page is enabled.

If:

a) the ATMPE is set to one;
b) the ATO bit is set to one;
c) the value in the DPICZ bit allows protection information checking for the specified command; and
d) the APP_CHK bit is set to one in the Extended Inquiry VPD page (see 7.8.7);

then:

knowledge of the value of the Application Tag shall come from the values in the Application Tag mode page as specified by the DPICZ bit.

A reject write without protection (RWWP) bit set to zero specifies that the device server shall process write commands that are specified to include user data without protection information (e.g., a WRITE(10) command with the WRPROTECT field set to 000b or a WRITE (6) command (see SBC-3)). A RWWP bit set to one specifies that the device server in a logical unit that has been formatted with protection information shall terminate with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB any write command that is specified to include user data without protection information.

The AUTOLOAD MODE field specifies the action to be taken by a removable medium device server when a medium is inserted. For devices other than removable medium devices, this field is reserved. Table 410 shows the usage of the AUTOLOAD MODE field.

Table 410 — AUTOLOAD MODE field

Code	Definition
000b	Medium shall be loaded for full access.
001b	Medium shall be loaded for medium auxiliary memory access only.
010b	Medium shall not be loaded.
011b to 111b	Reserved

The BUSY TIMEOUT PERIOD field specifies the maximum time, in 100 milliseconds increments, that the application client allows for the device server to return BUSY status for unanticipated conditions that are not a routine part of commands from the application client. This value may be rounded down as defined in 5.4. A 0000h value in this field is undefined by this standard. An FFFFh value in this field is defined as an unlimited period.

The EXTENDED SELF-TEST COMPLETION TIME field contains advisory data that is the time in seconds that the device server requires to complete an extended self-test when the device server is not interrupted by subsequent commands and no errors occur during processing of the self-test. The application client should expect this time to increase significantly if other commands are sent to the logical unit while a self-test is in progress or if errors occur during the processing of the self-test. Device servers supporting SELF-TEST CODE field values other than 000b for the SEND DIAGNOSTIC command (see 6.32) shall support the EXTENDED SELF-TEST COMPLETION TIME field. The EXTENDED SELF-TEST COMPLETION TIME field is not changeable. A value of FFFFh indicates that the extended self-test takes 65 535 seconds or longer. If the value is FFFFh, then refer to the EXTENDED SELF-TEST COMPLETION MINUTES field in the Extended INQUIRY Data VPD page(see 7.8.7).

7.5.8 Control Extension mode page

The Control Extension mode page (see table 411) is a subpage of the Control mode page (see 7.5.7) and provides controls over SCSI features that are applicable to all device types. The mode page policy (see 6.9) for this mode page shall be shared. If a field in this mode page is changed while there is a command already in the task set, it is vendor specific whether the old or new value of the field applies to that command.

Table 411 — Control Extension mode page

Bit / Byte	7	6	5	4	3	2	1	0
0	PS	SPF (1b)	PAGE CODE (0Ah)					
1	SUBPAGE CODE (01h)							
2	(MSB)	PAGE LENGTH (1Ch)						
3								(LSB)
4	Reserved					TCMOS	SCSIP	IALUAE
5	Reserved				INITIAL COMMAND PRIORITY			
6	Reserved							
•••								
31								

The PS bit, SPF bit, PAGE CODE field, SUBPAGE CODE field, and PAGE LENGTH field are described in 7.5.6.

A SCSI precedence (SCSIP) bit set to one specifies that the timestamp changed using a SET TIMESTAMP command (see 6.36) shall take precedence over methods outside the scope of this standard. A SCSIP bit set to zero specifies that methods outside this standard may change the timestamp and that the SET TIMESTAMP command is illegal.

A timestamp changeable by methods outside this standard (TCMOS) bit set to one specifies that the timestamp may be initialized by methods outside the scope of this standard. A TCMOS bit set to zero specifies that the timestamp shall not be changed by any method except those defined by this standard.

An implicit asymmetric logical unit access enabled (IALUAE) bit set to one specifies that implicitly managed transitions between primary target port asymmetric access states (see 5.11.2) are allowed. An IALUAE bit set to zero specifies that implicitly managed transitions between primary target port asymmetric access states be disallowed and indicates that implicitly managed transitions between primary target port asymmetric access states are disallowed or not supported.

The INITIAL COMMAND PRIORITY field specifies the priority that may be used as the command priority (see SAM-4) for commands received by the logical unit on any I_T nexus (i.e., on any I_T_L nexus) where a priority has not been modified by a SET PRIORITY command (see 6.34). If a MODE SELECT command specifies an initial command priority value that is different than the current initial command priority, then the device server shall set any priorities that have not be set with a SET PRIORITY command to a value different than the new initial command priority value to the new priority. The device server shall establish a unit attention condition for the initiator port associated with every I_T_L nexus that receives a new priority, with the additional sense code set to PRIORITY CHANGED.

7.5.9 Disconnect-Reconnect mode page

The Disconnect-Reconnect mode page (see table 412) provides the application client the means to tune the performance of a service delivery subsystem. The name for this mode page, disconnect-reconnect, comes from the SCSI parallel interface. The mode page policy (see 6.9) for this mode page shall be shared or per target port. If the SCSI target device contains more than one target port, the mode page policy should be per target port.

Table 412 — Disconnect-Reconnect mode page

Bit / Byte	7	6	5	4	3	2	1	0
0	PS	SPF (0b)	PAGE CODE (02h)					
1	PAGE LENGTH (0Eh)							
2	BUFFER FULL RATIO							
3	BUFFER EMPTY RATIO							
4	(MSB)	BUS INACTIVITY LIMIT						
5								(LSB)
6	(MSB)	DISCONNECT TIME LIMIT						
7								(LSB)
8	(MSB)	CONNECT TIME LIMIT						
9								(LSB)
10	(MSB)	MAXIMUM BURST SIZE						
11								(LSB)
12	EMDP	FAIR ARBITRATION			DIMM	DTDC		
13	Reserved							
14	(MSB)	FIRST BURST SIZE						
15								(LSB)

The Disconnect-Reconnect mode page controls parameters that affect one or more target ports. The parameters that may be implemented are specified in the SCSI transport protocol standard (see 3.1.120) for the target port. The MLUS bit (see 7.8.9) shall be set to one in the mode page policy descriptor for this mode page.

The parameters for a target port affect its behavior regardless of which initiator port is forming an I_T nexus with the target port. The parameters may be accessed by MODE SENSE (see 6.11) and MODE SELECT (see 6.9) commands directed to any logical unit accessible through the target port. If a parameter value is changed, all the device servers for all logical units accessible through the target port shall establish a unit attention condition for the initiator port associated with every I_T nexus that includes the target port except the I_T nexus on which the MODE SELECT command was received, with the additional sense code set to MODE PARAMETERS CHANGED.

If a parameter that is not appropriate for the specific SCSI transport protocol implemented by the target port is non-zero, 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.

An interconnect tenancy is a period of time during which a given pair of SCSI ports (i.e., an initiator port and a target port) are accessing the interconnect layer to communicate with each other (e.g., on arbitrated interconnects, a tenancy typically begins when a SCSI port successfully arbitrates for the interconnect and ends when the SCSI

port releases the interconnect for use by other devices). Data and other information transfers take place during interconnect tenancies.

The PS bit, SPF bit, PAGE CODE field, and PAGE LENGTH field are described in 7.5.6.

The BUFFER FULL RATIO field specifies to the target port how full the buffer should be during read operations prior to requesting an interconnect tenancy. Target ports that do not implement the requested ratio should round down to the nearest implemented ratio as defined in 5.4.

The BUFFER EMPTY RATIO field specifies to the target port how empty the buffer should be during write operations prior to requesting an interconnect tenancy. Target ports that do not implement the requested ratio should round down to the nearest implemented ratio as defined in 5.4.

The buffer full and buffer empty ratios are numerators of a fractional multiplier that has 256 as its denominator. A value of zero indicates that the target port determines when to request an interconnect tenancy consistent with the disconnect time limit parameter. These parameters are advisory to the target port.

NOTE 54 - As an example, consider a target port with ten 512-byte buffers and a specified buffer full ratio of 3Fh. The formula is: INTEGER((ratio÷256)×number of buffers). Therefore in this example INTEGER((3Fh÷256)×10) = 2. During the read operations described in this example, the target port should request an interconnect tenancy whenever two or more buffers are full.

The BUS INACTIVITY LIMIT field specifies the maximum time that the target port is permitted to maintain an interconnect tenancy without data or information transfer. If the bus inactivity limit is exceeded, then the target port shall conclude the interconnect tenancy, within the restrictions placed on it by the applicable SCSI transport protocol. The contents of the DTDC field in this mode page also shall affect the duration of an interconnect tenancy. This value may be rounded as defined in 5.4. A value of zero specifies that there is no bus inactivity limit. Different SCSI transport protocols define different units of measure for the bus inactivity limit.

The DISCONNECT TIME LIMIT field specifies the minimum time that the target port shall wait between interconnect tenancies. This value may be rounded as defined in 5.4. A value of zero specifies that there is no disconnect time limit. Different SCSI transport protocols define different units of measure for the disconnect time limit.

The CONNECT TIME LIMIT field specifies the maximum duration of a single interconnect tenancy. If the connect time limit is exceeded, then the target port shall conclude the interconnect tenancy, within the restrictions placed on it by the applicable SCSI transport protocol. The contents of the DTDC field in this mode page also shall affect the duration of an interconnect tenancy. This value may be rounded as defined in 5.4. A value of zero specifies that there is no connect time limit. Different SCSI transport protocols define different units of measure for the connect time limit.

The MAXIMUM BURST SIZE field indicates the maximum amount of data that the target port shall transfer during a single data transfer operation. This value is expressed in increments of 512 bytes (i.e., a value of one means 512 bytes, two means 1 024 bytes, etc.). The relationship, if any, between data transfer operations and interconnect tenancies is defined in the individual SCSI transport protocol standards. A value of zero specifies there is no limit on the amount of data transferred per data transfer operation.

In terms of the SCSI transport protocol services (see SAM-4), the device server shall limit the Request Byte Count argument to the **Receive Data-Out** protocol service and the **Send Data-In** protocol service to the amount specified in the MAXIMUM BURST SIZE field.

The enable modify data pointers (EMDP) bit specifies whether or not the target port may transfer data out of order. If the EMDP bit is set to zero, the target port shall not transfer data out of order. If the EMDP bit is set to one, the target port is allowed to transfer data out of order.

The FAIR ARBITRATION field specifies whether the target port should use fair or unfair arbitration when requesting an interconnect tenancy. The field may be used to specify different fairness methods as defined in the individual SCSI transport protocol standards.

A disconnect immediate (DIMM) bit set to zero specifies that the target port may transfer data for a command during the same interconnect tenancy in which it receives the command. Whether or not the target port does so may depend upon the target port's internal algorithms, the rules of the applicable SCSI transport protocol, and settings of the other parameters in this mode page. A disconnect immediate (DIMM) bit set to one specifies that the target port shall not transfer data for a command during the same interconnect tenancy in which it receives the command.

The data transfer disconnect control (DTDC) field (see table 413) defines other restrictions on when multiple interconnect tenancies are permitted. A non-zero value in the DTDC field shall take precedence over other interconnect tenancy controls represented by other fields in this mode page.

Table 413 — Data transfer disconnect control (DTDC) field

Code	Description
000b	Data transfer disconnect control is not used. Interconnect tenancies are controlled by other fields in this mode page.
001b	All data for a command shall be transferred within a single interconnect tenancy.
010b	Reserved
011b	All data and the response for a command shall be transferred within a single interconnect tenancy.
100b to 111b	Reserved

The FIRST BURST SIZE field specifies the maximum amount of data that may be transferred to the target port for a command along with the command (i.e., the first burst). This value is expressed in increments of 512 bytes (i.e., a value of one means 512 bytes, two means 1 024 bytes, etc.). The meaning of a value of zero is SCSI transport protocol specific. SCSI transport protocols supporting this field shall provide an additional mechanism to enable and disable the first burst function.

In terms of the SCSI transport protocol services (see SAM-4), the **Receive Data-Out** protocol service shall retrieve the first FIRST BURST SIZE amount of data from the first burst.

7.5.10 Extended mode page

The Extended mode page (see table 414) provides a means to specify subpages that are defined for all device types. Subpage code 00h is reserved. All Extended mode pages use the sub_page format.

Table 414 — Extended mode page

Bit / Byte	7	6	5	4	3	2	1	0
0	PS	SPF (1b)	PAGE CODE (15h)					
1	SUBPAGE CODE							
2	(MSB)	PAGE LENGTH (n-3)						
3								(LSB)
4	Subpage specific mode parameters							
•••								
n								

The PS bit, SPF bit, PAGE CODE field, and PAGE LENGTH field are described in 7.5.6.

7.5.11 Extended Device-Type Specific mode page

The Extended Device-Type Specific mode page (see table 415) provides a means to specify subpages that are defined differently for each device type. Subpage code 00h is reserved in the MODE SENSE command (see 6.11.1). All Extended Device-Type Specific mode pages use the sub_page format.

Table 415 — Extended Device-Type Specific mode page

Bit / Byte	7	6	5	4	3	2	1	0
0	PS	SPF (1b)	PAGE CODE (16h)					
1	SUBPAGE CODE							
2	(MSB)	PAGE LENGTH (n-3)						
3								(LSB)
4	Subpage specific mode parameters							
•••								
n								

The PS bit, SPF bit, PAGE CODE field, and PAGE LENGTH field are described in 7.5.6.

7.5.12 Power Condition mode page

The Power Condition mode page provides an application client with a method to control the power condition of a logical unit (see 5.12).

The mode page policy (see 6.9) for this mode page shall be shared.

The logical unit shall use the values in the Power Condition mode page to control its power condition after a power on or a hard reset until a START STOP UNIT command (see SBC-3) setting a power condition is received.

Table 416 defines the Power Condition mode page.

Table 416 — Power Condition mode page

Bit / Byte	7	6	5	4	3	2	1	0
0	PS	SPF (0b)	PAGE CODE (1Ah)					
1	PAGE LENGTH (26h)							
2	PM_BG_PRECEDENCE		Reserved					STANDBY_Y
3	Reserved				IDLE_C	IDLE_B	IDLE_A	STANDBY_Z
4	(MSB)	IDLE_A CONDITION TIMER						
•••								
7								(LSB)
8	(MSB)	STANDBY_Z CONDITION TIMER						
•••								
11								(LSB)
12	(MSB)	IDLE_B CONDITION TIMER						
•••								
15								(LSB)
16	(MSB)	IDLE_C CONDITION TIMER						
•••								
19								(LSB)
20	(MSB)	STANDBY_Y CONDITION TIMER						
•••								
23								(LSB)
24	Reserved							
•••								
38								
39	FIDCPC		FSBCPC		FSTCPC		Reserved	

The PS bit, SPF bit, PAGE CODE field, and PAGE LENGTH field are described in 7.5.6.

The PM_BG_PRECEDENCE field (see table 417) specifies the interactions between background functions (see 3.1.14) and power management.

Table 417 — PM_BG_PRECEDENCE field (part 1 of 2)

Code	Description
00b	Vendor specific

Table 417 — PM_BG_PRECEDENCE field (part 2 of 2)

Code	Description
01b	Performing background functions take precedence over maintaining low power conditions as follows: a) if the logical unit is in a low power condition as the result of a power condition timer associated with that condition expiring, then: A) the logical unit shall change from that power condition, if necessary, to the power condition required to perform the background function, when: a) a timer associated with a background scan operation expires, and that function is enabled (see SBC-3); or b) an event occurs to initiate a device specific background function, and that function is enabled (see 5.7); 2) the logical unit shall perform the background function(s) based on the definitions in this standard and other command standards (e.g., if the device server receives a command while performing a background function, then the logical unit shall suspend the function to process the command); 3) if more than one condition is met to initiate a background function, then: a) all initiated background functions shall be performed; and b) the order of performing the functions is vendor specific; and 4) when all initiated background functions have been completed, the device server shall check to see if any power condition timers have expired. If any power condition timer has expired, then the logical unit shall change to the power condition associated with the highest priority timer that has expired; or b) if the logical unit is performing a background function, and a power condition timer expires, then the logical unit shall perform all initiated background functions before the logical unit changes to a power condition associated with a timer has expired.
10b	Maintaining low power conditions take precedence over performing background functions as follows: a) if the logical unit is in a low power condition, then the logical unit shall not change from that power condition to perform a background function; b) the device server may perform any initiated and enabled background function based on the definitions in this standard or other command standards, if all of the following are true: A) a condition is met to initiate a background function; B) that background function is enabled; C) the logical unit changes to a power condition in which the background function may be performed (e.g., the device server processes a medium access command causing the logical unit to change its power condition to continue processing that command); and D) all outstanding application client requests have been completed; or c) if the logical unit is performing a background function, and a power condition timer expires that causes a change to a power condition in which the logical unit is unable to continue performing the background function, then the logical unit shall: A) suspend the background function; and B) change to the power condition associated with the timer that expired.
11b	Reserved

The behavior of the idle and standby condition timers controlled by this mode page is defined in the power condition overview (see 5.12.1) and the power condition state machine (see 5.12.6).

If the STANDBY_Y bit is set to one, then the standby_y condition timer is enabled. If the STANDBY_Y bit is set to zero, then the device server shall ignore the standby_y condition timer.

If the IDLE_C bit is set to one, then the idle_c condition timer is enabled. If the IDLE_C bit is set to zero, then the device server shall ignore the idle_c condition timer.

If the IDLE_B bit is set to one, then the idle_b condition timer is enabled. If the IDLE_B bit is set to zero, then the device server shall ignore the idle_b condition timer.

If the IDLE_A bit is set to one, then the idle_a condition timer is enabled. If the IDLE_A bit is set to zero, then the device server shall ignore the idle_a condition timer.

If the STANDBY_Z bit is set to one, then the standby_z condition timer is enabled. If the STANDBY_Z bit is set to zero, then the device server shall ignore the standby_z condition timer.

If any of the power condition enable bits (e.g., the IDLE_C bit or the STANDBY_Y bit) are set to zero and are not changeable (see 6.11.3), then the device server does not implement the power condition timer associated with that enable bit (see table 56 in 5.12.6.1).

The IDLE_A CONDITION TIMER field specifies the initial value, in 100 millisecond increments, for the idle_a power condition timer (see 5.12.6.1). This value may be rounded up or down to the nearest implemented time as described in 5.4.

The STANDBY_Z CONDITION TIMER field specifies the initial value, in 100 millisecond increments, for the standby_z power condition timer (see 5.12.6.1). This value may be rounded up or down to the nearest implemented time as described in 5.4.

The IDLE_B CONDITION TIMER field specifies the initial value, in 100 millisecond increments, for the idle_b power condition timer (see 5.12.6.1). This value may be rounded up or down to the nearest implemented time as described in 5.4.

The IDLE_C CONDITION TIMER field specifies the initial value, in 100 millisecond increments, for the idle_c power condition timer (see 5.12.6.1). This value may be rounded up or down to the nearest implemented time as described in 5.4.

The STANDBY_Y CONDITION TIMER field specifies the initial value, in 100 millisecond increments, for the standby_y power condition timer (see 5.12.6.1). This value may be rounded up or down to the nearest implemented time as described in 5.4.

The from idle command processing control (FIDCPC) field is defined in table 418.

Table 418 — FIDCPC field

Code	
00b	Restricted [a]
01b	From idle command processing control is disabled [b]
10b	From idle command processing control is enabled [b]
11b	Reserved

[a] See SAS-2 for command processing in the Active_Wait state and Idle_Wait state.
[b] For direct-access block devices see the Active_Wait state in SBC-3 for the definition of command processing in that state. For devices that are not direct-access block devices, see the Active_Wait state in this standard (i.e., see 5.12.6.6) for the definition of command processing in that state.

The from standby command processing control (FSBCPC) field is defined in table 419.

Table 419 — FSBCPC field

Code	
00b	Restricted [a]
01b	From standby command processing control is disabled [b]
10b	From standby command processing control is enabled [b]
11b	Reserved

[a] See SAS-2 for command processing in the Active_Wait state and Idle_Wait state.
[b] For direct-access block devices see the Active_Wait state and the Idle_Wait state in SBC-3 for the definition of command processing in those states. For devices that are not direct-access block devices, see the Active_Wait state in this standard (i.e., see 5.12.6.6) for the definition of command processing in that state.

The from stopped command processing control (FSTCPC) field is defined in table 420.

Table 420 — FSTCPC field

Code	
00b	Restricted [a]
01b	From stopped command processing control is disabled [b]
10b	From stopped command processing control is enabled [b]
11b	Reserved

[a] See SAS-2 for command processing in the Active_Wait state and Idle_Wait state.
[b] For direct-access block devices see the Active_Wait state, the Idle_Wait state description and the Standby_Wait state in SBC-3 for the definition of command processing in those states.

7.5.13 Protocol Specific Logical Unit mode page

The Protocol Specific Logical Unit mode page (see table 421) provides protocol specific controls that are associated with a logical unit.

Table 421 — Protocol Specific Logical Unit mode page

Bit / Byte	7	6	5	4	3	2	1	0
0	PS	SPF (0b)	PAGE CODE (18h)					
1	PAGE LENGTH (n-1)							
2	Protocol specific mode parameters				PROTOCOL IDENTIFIER			
3	Protocol specific mode parameters							
•••								
n								

During an I_T_L nexus, the Protocol Specific Logical Unit mode page controls parameters that affect both:

a) one or more target ports; and
b) the logical unit.

The parameters that may be implemented are specified in the SCSI transport protocol standard (see 3.1.120) for the target port. The mode page policy (see 6.9) for this mode page shall be shared or per target port and should be per target port.

The parameters for a target port and logical unit affect their behavior regardless of which initiator port is forming an I_T_L nexus with the target port and logical unit. If a parameter value is changed, the device server shall establish a unit attention condition 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 MODE PARAMETERS CHANGED.

The PS bit, SPF bit, PAGE CODE field, and PAGE LENGTH field are described in 7.5.6.

The value in the PROTOCOL IDENTIFIER field (see 7.6.1) defines the SCSI transport protocol to which the mode page applies. For a MODE SENSE command (see 6.11), the device server shall set the PROTOCOL IDENTIFIER field to one of the values shown in table 424 (see 7.6.1) to indicate the SCSI transport protocol used by the target port through which the MODE SENSE command is being processed. For a MODE SELECT command (see 6.9), the application client shall set the PROTOCOL IDENTIFIER field to one of the values shown in table 424 indicating the SCSI transport protocol to which the protocol specific mode parameters apply. If a device server receives a mode page containing a transport protocol identifier value other than the one used by the target port on which the MODE SELECT command was received, 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 PARAMETER LIST.

7.5.14 Protocol Specific Port mode page

The Protocol Specific Port mode page provides protocol specific controls that are associated with a SCSI port. The page_0 format (see table 422) is used when a MODE SENSE command (see 6.11.1) contains zero in the SUBPAGE CODE field, and sub_page format (see table 423) is used for subpages 01h through FEh. See the SCSI transport protocol standard (see 3.1.120) for definition of the protocol specific mode parameters.

Table 422 — Page_0 format Protocol Specific Port mode page

Bit / Byte	7	6	5	4	3	2	1	0
0	PS	SPF (0b)	PAGE CODE (19h)					
1	PAGE LENGTH (n-1)							
2	Protocol specific mode parameters				PROTOCOL IDENTIFIER			
3	Protocol specific mode parameters							
•••								
n								

Table 423 — Sub_page format Protocol Specific Port mode page

Bit / Byte	7	6	5	4	3	2	1	0
0	PS	SPF (1b)	PAGE CODE (19h)					
1	SUBPAGE CODE							
2	(MSB)	PAGE LENGTH (n-3)						
3								(LSB)
4	Reserved							
5	Protocol specific mode parameters				PROTOCOL IDENTIFIER			
6	Protocol specific mode parameters							
•••								
n								

The Protocol Specific Port mode page controls parameters that affect one or more target ports. The parameters that may be implemented are specified in the SCSI transport protocol standard (see 3.1.120) for the target port. The mode page policy (see 6.9) for this mode page shall be shared or per target port. If the SCSI target device contains more than one target port, the mode page policy should be per target port.

The parameters for a target port affect its behavior regardless of which initiator port is forming an I_T nexus with the target port. The MLUS bit (see 7.8.9) shall be set to one in the mode page policy descriptor for this mode page.

The parameters may be accessed by MODE SENSE (see 6.11) and MODE SELECT (see 6.9) commands directed to any logical unit accessible through the target port. If a parameter value is changed, the device servers for all logical units accessible through the target port shall establish a unit attention condition 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 MODE PARAMETERS CHANGED.

The PS bit, SPF bit, PAGE CODE field, SUBPAGE CODE field, and PAGE LENGTH field are described in 7.5.6.

The value in the PROTOCOL IDENTIFIER field (see 7.6.1) defines the SCSI transport protocol to which the mode page applies. For a MODE SENSE command, the device server shall set the PROTOCOL IDENTIFIER field to one of the values shown in table 424 (see 7.6.1) to indicate the SCSI transport protocol used by the target port through which the MODE SENSE command is being processed. For a MODE SELECT command, the application client shall set the PROTOCOL IDENTIFIER field to one of the values shown in table 424 indicating the SCSI transport protocol to which the protocol specific mode parameters apply. If a device server receives a mode page containing a transport protocol identifier value other than the one used by the target port on which the MODE SELECT command was received, then 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.

7.6 Protocol specific parameters

7.6.1 Protocol specific parameters introduction

Some commands use protocol specific information in their CDBs or parameter lists. This subclause describes those protocol specific parameters.

Protocol specific parameters may include a PROTOCOL IDENTIFIER field (see table 424) as a reference for the SCSI transport protocol to which the protocol specific parameter applies.

Table 424 — PROTOCOL IDENTIFIER values

Protocol Identifier	Description	Protocol Standard
0h	Fibre Channel	FCP-4
1h	Parallel SCSI	SPI-5
2h	SSA	SSA-S3P
3h	IEEE 1394	SBP-3
4h	SCSI Remote Direct Memory Access Protocol	SRP
5h	Internet SCSI (iSCSI)	iSCSI
6h	SAS Protocol Layer	SPL
7h	Automation/Drive Interface Transport Protocol	ADT-2
8h	AT Attachment Interface	ACS-2
9h to Eh	Reserved	
Fh	No specific protocol	

7.6.2 Alias entry protocol specific designations

7.6.2.1 Introduction to alias entry protocol specific designations

The alias entries (see 6.2.2) in the parameter data for the CHANGE ALIASES command (see 6.2) and REPORT ALIASES command (see 6.21) include FORMAT CODE, DESIGNATION LENGTH and DESIGNATION fields whose contents and meaning are based on the SCSI transport protocol specified in a PROTOCOL IDENTIFIER field (see 7.6.1). This subclause defines the SCSI transport protocol specific format codes, designation lengths, and designations.

7.6.2.2 Fibre Channel specific alias entry designations

7.6.2.2.1 Introduction to Fibre Channel specific alias entry designations

If an alias entry PROTOCOL IDENTIFIER field contains the Fibre Channel protocol identifier (0h, see table 424), the FORMAT CODE, DESIGNATION LENGTH and DESIGNATION fields shall be as defined in table 425.

Table 425 — Fibre Channel alias entry format codes

Format Code	Description	Designation Length (bytes)	Reference
00h	World Wide Port Name	8	7.6.2.2.2
01h	World Wide Port Name with N_Port checking	12	7.6.2.2.3
02h to FFh	Reserved		

7.6.2.2.2 Fibre Channel world wide port name alias entry designation

If the PROTOCOL IDENTIFIER and FORMAT CODE fields specify a Fibre Channel world wide port name designation, the alias entry DESIGNATION field shall have the format shown in table 426.

Table 426 — Fibre Channel world wide port name alias entry designation

Bit / Byte	7	6	5	4	3	2	1	0
0	See table 100 in 6.2.2							
•••								
15								
16	FIBRE CHANNEL WORLD WIDE PORT NAME							
•••								
23								

The FIBRE CHANNEL WORLD WIDE PORT NAME field shall contain the port world wide name defined by the port login (PLOGI) extended link service (see FC-FS).

A Fibre Channel world wide port name designation is valid (see 6.2.3) if the device server has access to a SCSI domain formed by a Fibre Channel fabric and the fabric contains a port with the specified port world wide name.

7.6.2.2.3 Fibre Channel world wide port name with N_Port checking alias entry designation

If the PROTOCOL IDENTIFIER and FORMAT CODE fields specify a Fibre Channel world wide port name with N_Port checking designation, the alias entry DESIGNATION field shall have the format shown in table 427.

Table 427 — Fibre Channel world wide port name with N_Port checking alias entry designation

Bit / Byte	7	6	5	4	3	2	1	0
0	See table 100 in 6.2.2							
•••								
15								
16	FIBRE CHANNEL WORLD WIDE PORT NAME							
•••								
23								
24	Reserved							
25	(MSB)	N_PORT						
26								
27								(LSB)

The FIBRE CHANNEL WORLD WIDE PORT NAME field shall contain the port world wide name defined by the port login (PLOGI) extended link service (see FC-FS).

The N_PORT field shall contain the FC_FS port D_ID to be used to transport frames including PLOGI and FCP-4 related frames.

A Fibre Channel world wide port name with N_Port checking designation is valid (see 6.2.3) if all of the following conditions are true:

a) the device server has access to a SCSI domain formed by a Fibre Channel fabric;
b) the fabric contains a port with the specified port World Wide Name; and
c) the value in the N_PORT field is the N_Port identifier of a Fibre Channel port whose port world wide name matches that in the FIBRE CHANNEL WORLD WIDE PORT NAME field.

7.6.2.3 RDMA specific alias entry designations

7.6.2.3.1 Introduction to RDMA specific alias entry designations

If an alias entry PROTOCOL IDENTIFIER field contains the SCSI RDMA protocol identifier (4h, see table 424), the FORMAT CODE, DESIGNATION LENGTH and DESIGNATION fields shall be as defined in table 428.

Table 428 — RDMA alias entry format codes

Format Code	Description	Designation Length (bytes)	Reference
00h	Target Port Identifier	16	7.6.2.3.2
01h	InfiniBand™ Global Identifier with Target Port Identifier checking	32	7.6.2.3.3
02h to FFh	Reserved		

7.6.2.3.2 RDMA target port identifier alias entry designation

If the PROTOCOL IDENTIFIER and FORMAT CODE fields specify a SCSI RDMA target port identifier designation, the alias entry DESIGNATION field shall have the format shown in table 429.

Table 429 — RDMA target port identifier alias entry designation

Bit / Byte	7	6	5	4	3	2	1	0
0	See table 100 in 6.2.2							
•••								
15								
16	TARGET PORT IDENTIFIER							
•••								
31								

The TARGET PORT IDENTIFIER field shall contain an SRP target port identifier.

A SCSI RDMA target port identifier designation is valid (see 6.2.3) if the device server has access to an SRP SCSI domain containing the specified SRP target port identifier.

7.6.2.3.3 InfiniBand global identifier with target port identifier checking alias entry designation

If the PROTOCOL IDENTIFIER and FORMAT CODE fields specify an InfiniBand global identifier with target port identifier checking designation, the alias entry designation field shall have the format shown in table 430.

Table 430 — InfiniBand global identifier with target port identifier checking alias entry designation

Bit / Byte	7	6	5	4	3	2	1	0
0	See table 100 in 6.2.2							
•••								
15								
16	INFINIBAND GLOBAL IDENTIFIER							
•••								
31								
32	TARGET PORT IDENTIFIER							
•••								
47								

The INFINIBAND GLOBAL IDENTIFIER field contains an InfiniBand global identifier (GID) of an InfiniBand port connected to an SRP target port.

The TARGET PORT IDENTIFIER field shall contain an SRP target port identifier.

An InfiniBand global identifier with target port identifier checking designation is valid (see 6.2.3) if all of the following conditions are true:

a) the device server has access to an SRP SCSI domain layered on InfiniBand;
b) the device server has access to an SRP target port based on the InfiniBand global identifier specified in the INFINIBAND GLOBAL IDENTIFIER field; and

c) the value in the TARGET PORT IDENTIFIER field is the SRP target port identifier for the SRP target port that is accessible via the InfiniBand global identifier contained in the INFINIBAND GLOBAL IDENTIFIER field.

7.6.2.4 Internet SCSI specific alias entry designations

7.6.2.4.1 Introduction to Internet SCSI specific alias entry designations

If an alias entry PROTOCOL IDENTIFIER field contains the iSCSI protocol identifier (5h, see table 424), the FORMAT CODE, DESIGNATION LENGTH and DESIGNATION fields shall be as defined in table 431.

Table 431 — iSCSI alias entry format codes

Format Code	Description	Designation Length (bytes, maximum)	Reference
00h	iSCSI Name	224	7.6.2.4.2
01h	iSCSI Name with binary IPv4 address	236	7.6.2.4.3
02h	iSCSI Name with IPName	488	7.6.2.4.4
03h	iSCSI Name with binary IPv6 address	248	7.6.2.4.5
04h to FFh	Reserved		

NOTE 55 - A designation that contains no IP addressing information or contains IP addressing information that does not address the named SCSI target device may require a device server to have access to a name server or to other discovery protocols to resolve the given iSCSI Name to an IP address through which the device server may establish iSCSI Login. Access to such a service is protocol specific and vendor specific.

7.6.2.4.2 iSCSI name alias entry designation

If the PROTOCOL IDENTIFIER and FORMAT CODE fields specify iSCSI name designation, the alias entry DESIGNATION field shall have the format shown in table 432.

Table 432 — iSCSI name alias entry designation

Bit / Byte	7	6	5	4	3	2	1	0
0	See table 100 in 6.2.2							
•••								
15								
16	(MSB)	ISCSI NAME						
•••								
4m-1								(LSB)

The null-terminated, null-padded (see 4.4.2) ISCSI NAME field shall contain the iSCSI name of an iSCSI node (see RFC 3720). The number of bytes in the ISCSI NAME field shall be a multiple of four.

An iSCSI name designation is valid if the device server has access to a SCSI domain containing an Internet protocol network and that network contains an iSCSI node with the specified iSCSI name.

7.6.2.4.3 iSCSI name with binary IPv4 address alias entry designation

If the PROTOCOL IDENTIFIER and FORMAT CODE fields specify iSCSI name with binary IPv4 address designation, the alias entry DESIGNATION field shall have the format shown in table 433.

Table 433 — iSCSI name with binary IPv4 address alias entry designation

Bit / Byte	7	6	5	4	3	2	1	0
0	See table 100 in 6.2.2							
•••								
15								
16	(MSB)	ISCSI NAME						
•••								
4m-1								(LSB)
4m	(MSB)	IPV4 ADDRESS						
•••								
4m+3								(LSB)
4m+4	Reserved							
4m+5	Reserved							
4m+6	(MSB)	PORT NUMBER						
4m+7								(LSB)
4m+8	Reserved							
4m+9	Reserved							
4m+10	(MSB)	INTERNET PROTOCOL NUMBER						
4m+11								(LSB)

The null-terminated, null-padded (see 4.4.2) ISCSI NAME field shall contain the iSCSI name of an iSCSI node (see RFC 3720). The number of bytes in the ISCSI NAME field shall be a multiple of four.

The IPV4 ADDRESS field shall contain an IPv4 address (see RFC 791).

The PORT NUMBER field shall contain a TCP port number (see 3.1.180). The TCP port number shall conform to the requirements defined by iSCSI (see RFC 3720).

The INTERNET PROTOCOL NUMBER field shall contain an Internet protocol number (see 3.1.80). The Internet protocol number shall conform to the requirements defined by iSCSI (see RFC 3720).

An iSCSI name designation is valid if the device server has access to a SCSI domain containing an Internet protocol network and that network contains an iSCSI node with the specified iSCSI name.

The IPv4 address, port number, and Internet protocol number provided in the designation may be used by a device server for addressing to discover and establish communication with the named iSCSI node. Alternatively, the device server may use other protocol specific or vendor specific methods to discover and establish communication with the named iSCSI node.

7.6.2.4.4 iSCSI name with IPname alias entry designation

If the PROTOCOL IDENTIFIER and FORMAT CODE fields specify iSCSI name with IPname designation, the alias entry DESIGNATION field shall have the format shown in table 434.

Table 434 — iSCSI name with IPname alias entry designation

Bit / Byte	7	6	5	4	3	2	1	0
0	See table 100 in 6.2.2							
•••								
15								
16	(MSB)	ISCSI NAME						
•••								
k								(LSB)
k+1	(MSB)	IPNAME						
•••								
n								(LSB)
n+1	PAD (if needed)							
4m-1								
4m	Reserved							
4m+1	Reserved							
4m+2	(MSB)	PORT NUMBER						
4m+3								(LSB)
4m+4	Reserved							
4m+5	Reserved							
4m+6	(MSB)	INTERNET PROTOCOL NUMBER						
4m+7								(LSB)

The null-terminated (see 4.4.2) ISCSI NAME field shall contain the iSCSI name of an iSCSI node (see RFC 3720).

The null-terminated (see 4.4.2) IPNAME field shall contain a Internet protocol domain name (see 3.1.79).

The PAD field shall contain zero to three bytes set to zero such that the total length of the ISCSI NAME, IPNAME, and PAD fields is a multiple of four. Device servers shall ignore the PAD field.

The PORT NUMBER field shall contain a TCP port number (see 3.1.180). The TCP port number shall conform to the requirements defined by iSCSI (see RFC 3720).

The INTERNET PROTOCOL NUMBER field shall contain an Internet protocol number (see 3.1.80). The Internet protocol number shall conform to the requirements defined by iSCSI (see RFC 3720).

An iSCSI name designation is valid if the device server has access to a SCSI domain containing an Internet protocol network and that network contains an iSCSI node with the specified iSCSI name.

The Internet protocol domain name, port number, and Internet protocol number provided in the designation may be used by a device server for addressing to discover and establish communication with the named iSCSI node. Alter-

natively, the device server may use other protocol specific or vendor specific methods to discover and establish communication with the named iSCSI node.

7.6.2.4.5 iSCSI name with binary IPv6 address alias entry designation

If the PROTOCOL IDENTIFIER and FORMAT CODE fields specify iSCSI name with binary IPv6 address designation, the alias entry DESIGNATION field shall have the format shown in table 435.

Table 435 — iSCSI name with binary IPv6 address alias entry designation

Bit / Byte	7	6	5	4	3	2	1	0
0	See table 100 in 6.2.2							
•••								
15								
16	(MSB)	ISCSI NAME						
•••								
n								(LSB)
4m	(MSB)	IPV6 ADDRESS						
•••								
4m+15								(LSB)
4m+16	Reserved							
4m+17	Reserved							
4m+18	(MSB)	PORT NUMBER						
4m+19								(LSB)
4m+20	Reserved							
4m+21	Reserved							
4m+22	(MSB)	INTERNET PROTOCOL NUMBER						
4m+23								(LSB)

The null-terminated, null-padded (see 4.4.2) ISCSI NAME field shall contain the iSCSI name of an iSCSI node (see RFC 3720).

The IPV6 ADDRESS field shall contain an IPv6 address (see RFC 4291).

The PORT NUMBER field shall contain a TCP port number (see 3.1.180). The TCP port number shall conform to the requirements defined by iSCSI (see RFC 3720).

The INTERNET PROTOCOL NUMBER field shall contain an Internet protocol number (see 3.1.80). The Internet protocol number shall conform to the requirements defined by iSCSI (see RFC 3720).

An iSCSI name designation is valid if the device server has access to a SCSI domain containing an Internet protocol network and that network contains an iSCSI node with the specified iSCSI name.

The IPv6 address, port number and Internet protocol number provided in the designation may be used by a device server for addressing to discover and establish communication with the named iSCSI node. Alternatively, the device server may use other protocol specific or vendor specific methods to discover and establish communication with the named iSCSI node.

7.6.3 EXTENDED COPY protocol specific target descriptors

7.6.3.1 Introduction to EXTENDED COPY protocol specific target descriptors

The protocol-specific target descriptors in the parameter data of the EXTENDED COPY command (see 6.3) are described in this subclause. An introduction to EXTENDED COPY target descriptors is provided in 6.3.6.1.

NOTE 56 - In the EXTENDED COPY command the target in target descriptor refers to a copy target device (i.e., the source or destination of an EXTENDED COPY operation), not a SCSI target device. Target descriptors specify logical unit numbers or proxy tokens and may also specify initiator ports, target ports, and SCSI target devices used to access those logical units.

7.6.3.2 Fibre Channel N_Port_Name EXTENDED COPY target descriptor format

The target descriptor format shown in table 436 is used by an EXTENDED COPY command to specify an FCP copy target device using its Fibre Channel N_Port_Name.

Table 436 — Fibre Channel N_Port_Name EXTENDED COPY target descriptor format

Bit / Byte	7	6	5	4	3	2	1	0
0	DESCRIPTOR TYPE CODE (E0h)							
1	LU ID TYPE		NUL	PERIPHERAL DEVICE TYPE				
2	(MSB)	RELATIVE INITIATOR PORT IDENTIFIER						
3								(LSB)
4	LU IDENTIFIER							
•••								
11								
12	N_PORT_NAME							
•••								
19								
20	Reserved							
•••								
27								
28	Device type specific parameters							
•••								
31								

The DESCRIPTOR TYPE CODE field, LU ID TYPE field, PERIPHERAL DEVICE TYPE field, NUL bit, RELATIVE INITIATOR PORT IDENTIFIER field, LU IDENTIFIER field, and the device type specific parameters are described in 6.3.6.1.

The N_PORT_NAME field shall contain the N_Port_Name defined by the port login (PLOGI) extended link service (see FC-FS).

NOTE 57 - The N_Port_Name EXTENDED COPY target descriptor format necessitates translating the N_Port_Name to an N_Port_ID (see FC-FS and 7.6.3.3).

7.6.3.3 Fibre Channel N_Port_ID EXTENDED COPY target descriptor format

The target descriptor format shown in table 437 is used by an EXTENDED COPY command to specify an FCP copy target device using its Fibre Channel N_Port_ID.

Table 437 — Fibre Channel N_Port_ID EXTENDED COPY target descriptor format

Bit / Byte	7	6	5	4	3	2	1	0
0	DESCRIPTOR TYPE CODE (E1h)							
1	LU ID TYPE		NUL	PERIPHERAL DEVICE TYPE				
2	(MSB)	RELATIVE INITIATOR PORT IDENTIFIER						
3								(LSB)
4	LU IDENTIFIER							
•••								
11								
12	Reserved							
•••								
20								
21	(MSB)	N_PORT_ID						
22								
23								(LSB)
24	Reserved							
•••								
27								
28	Device type specific parameters							
•••								
31								

The DESCRIPTOR TYPE CODE field, LU ID TYPE field, PERIPHERAL DEVICE TYPE field, NUL bit, RELATIVE INITIATOR PORT IDENTIFIER field, LU IDENTIFIER field, and the device type specific parameters are described in 6.3.6.1.

The N_PORT_ID field shall contain the port D_ID (see FC-FS) to be used to transport frames including PLOGI and FCP-4 related frames.

NOTE 58 - Use of N_Port_ID addressing restricts this target descriptor format to a single Fibre Channel fabric.

7.6.3.4 Fibre Channel N_Port_ID with N_Port_Name checking EXTENDED COPY target descriptor format

The target descriptor format shown in table 438 is used by an EXTENDED COPY command to specify an FCP copy target device using its Fibre Channel N_Port_ID and to require the copy manager to verify that the N_Port_Name of the specified N_Port matches the value in the target descriptor.

Table 438 — Fibre Channel N_Port_ID with N_Port_Name checking target descriptor format

Bit / Byte	7	6	5	4	3	2	1	0
0	DESCRIPTOR TYPE CODE (E2h)							
1	LU ID TYPE		NUL	PERIPHERAL DEVICE TYPE				
2	(MSB)	RELATIVE INITIATOR PORT IDENTIFIER						
3								(LSB)
4	LU IDENTIFIER							
•••								
11								
12	N_PORT_NAME							
•••								
19								
20	Reserved							
21	(MSB)	N_PORT_ID						
22								
23								(LSB)
24	Reserved							
•••								
27								
28	Device type specific parameters							
•••								
31								

The DESCRIPTOR TYPE CODE field, LU ID TYPE field, PERIPHERAL DEVICE TYPE field, NUL bit, RELATIVE INITIATOR PORT IDENTIFIER field, LU IDENTIFIER field, and the device type specific parameters are described in 6.3.6.1.

The N_PORT_NAME field shall contain the N_Port_Name defined by the port login (PLOGI) extended link service (see FC-FS).

The N_PORT field shall contain the port D_ID (see FC-FS) to be used to transport frames including PLOGI and FCP-4 related frames.

NOTE 59 - Use of N_Port addressing restricts this target descriptor format to a single fabric.

When the copy manager first processes a segment descriptor that references this target descriptor, it shall confirm that the D_ID in the N_PORT_ID field is associated with the N_Port_Name in the N_PORT_NAME field. If the confirmation fails, the command shall be terminated because the copy target device is unavailable (see 6.3.3). The SCSI device processing this target descriptor shall track configuration changes that affect the D_ID value for the duration of the command. An application client is responsible for tracking configuration changes between commands.

7.6.3.5 SCSI Parallel T_L EXTENDED COPY target descriptor format

The target descriptor format shown in table 439 is used by an EXTENDED COPY command to specify a SPI copy target device using its SCSI target identifier.

Table 439 — SCSI Parallel T_L EXTENDED COPY target descriptor format

Bit / Byte	7	6	5	4	3	2	1	0
0	DESCRIPTOR TYPE CODE (E3h)							
1	LU ID TYPE		NUL	PERIPHERAL DEVICE TYPE				
2	(MSB)	RELATIVE INITIATOR PORT IDENTIFIER						
3								(LSB)
4	LU IDENTIFIER							
•••								
11								
12	Vendor specific							
13	TARGET IDENTIFIER							
14	Reserved							
•••								
27								
28	Device type specific parameters							
•••								
31								

The DESCRIPTOR TYPE CODE field, LU ID TYPE field, PERIPHERAL DEVICE TYPE field, NUL bit, RELATIVE INITIATOR PORT IDENTIFIER field, LU IDENTIFIER field, and the device type specific parameters are described in 6.3.6.1.

The TARGET IDENTIFIER field specifies the SCSI target identifier (see SPI-5).

7.6.3.6 IEEE 1394 EUI-64 EXTENDED COPY target descriptor format

The target descriptor format shown in table 440 is used by an EXTENDED COPY command to specify an SBP copy target device using its IEEE 1394 Extended Unique Identifier, 64-bits (EUI-64) and configuration ROM (Read-Only Memory) directory identifier.

Table 440 — IEEE 1394 EUI-64 EXTENDED COPY target descriptor format

Bit / Byte	7	6	5	4	3	2	1	0
0	DESCRIPTOR TYPE CODE (E8h)							
1	LU ID TYPE		NUL	PERIPHERAL DEVICE TYPE				
2	(MSB)	RELATIVE INITIATOR PORT IDENTIFIER						
3								(LSB)
4	LU IDENTIFIER							
•••								
11								
12	EUI-64							
•••								
19								
20	DIRECTORY ID							
21								
22								
23	Reserved							
•••								
27								
28	Device type specific parameters							
•••								
31								

The DESCRIPTOR TYPE CODE field, LU ID TYPE field, PERIPHERAL DEVICE TYPE field, NUL bit, RELATIVE INITIATOR PORT IDENTIFIER field, LU IDENTIFIER field, and the device type specific parameters are described in 6.3.6.1.

The EUI-64 field shall contain the node's unique identifier (EUI-64) obtained from the configuration ROM bus information block, as specified by ANSI IEEE 1394a:2000.

NOTE 60 - ANSI IEEE 1394a-2000 separately labels the components of the EUI-64 as NODE_VENDOR_ID, CHIP_ID_HI and CHIP_ID_LO. Collectively these form the node's EUI-64.

The DIRECTORY ID field shall contain the copy target device's directory identifier, as specified by ISO/IEC 13213:1994.

7.6.3.7 RDMA EXTENDED COPY target descriptor format

The target descriptor format shown in table 441 is used by an EXTENDED COPY command to specify an SRP copy target device using its RDMA SRP target port identifier.

Table 441 — RDMA EXTENDED COPY target descriptor format

Bit / Byte	7	6	5	4	3	2	1	0
0	DESCRIPTOR TYPE CODE (E7h)							
1	LU ID TYPE		NUL	PERIPHERAL DEVICE TYPE				
2	(MSB)	RELATIVE INITIATOR PORT IDENTIFIER						
3								(LSB)
4	LU IDENTIFIER							
•••								
11								
12	TARGET PORT IDENTIFIER							
•••								
27								
28	Device type specific parameters							
•••								
31								

The DESCRIPTOR TYPE CODE field, LU ID TYPE field, PERIPHERAL DEVICE TYPE field, NUL bit, RELATIVE INITIATOR PORT IDENTIFIER field, LU IDENTIFIER field, and the device type specific parameters are described in 6.3.6.1.

The TARGET PORT IDENTIFIER field contains the SRP target port identifier (see SRP).

7.6.3.8 iSCSI binary IPv4 address EXTENDED COPY target descriptor format

The target descriptor format shown in table 442 is used by an EXTENDED COPY command to specify an iSCSI copy target device using its binary IPv4 (Internet Protocol version 4) address.

Table 442 — iSCSI binary IPv4 address EXTENDED COPY target descriptor format

Bit / Byte	7	6	5	4	3	2	1	0
0	DESCRIPTOR TYPE CODE (E5h)							
1	LU ID TYPE		NUL	PERIPHERAL DEVICE TYPE				
2	(MSB)			RELATIVE INITIATOR PORT IDENTIFIER				
3								(LSB)
4				LU IDENTIFIER				
•••								
11								
12	(MSB)			IPV4 ADDRESS				
•••								
15								(LSB)
16				Reserved				
•••								
21								
22	(MSB)			PORT NUMBER				
23								(LSB)
24	Reserved							
25	Reserved							
26	(MSB)			INTERNET PROTOCOL NUMBER				
27								(LSB)
28				Device type specific parameters				
•••								
31								

The DESCRIPTOR TYPE CODE field, LU ID TYPE field, PERIPHERAL DEVICE TYPE field, NUL bit, RELATIVE INITIATOR PORT IDENTIFIER field, LU IDENTIFIER field, and the device type specific parameters are described in 6.3.6.1.

The IPV4 ADDRESS field shall contain an IPv4 address (see RFC 791).

The PORT NUMBER field shall contain the TCP port number (see 3.1.180). The TCP port number shall conform to the requirements defined by iSCSI (see RFC 3720).

The INTERNET PROTOCOL NUMBER field shall contain an Internet protocol number (see 3.1.80). The Internet protocol number shall conform to the requirements defined by iSCSI (see RFC 3720).

NOTE 61 - The internet protocol number for TCP is 0006h.

7.6.3.9 iSCSI binary IPv6 address EXTENDED COPY target descriptor format

The target descriptor format shown in table 443 is used by an EXTENDED COPY command to specify an iSCSI copy target device using its binary IPv6 (Internet Protocol version 6) address.

Table 443 — iSCSI binary IPv6 address EXTENDED COPY target descriptor format

Bit / Byte	7	6	5	4	3	2	1	0
0	DESCRIPTOR TYPE CODE (EAh)							
1	LU ID TYPE		NUL	PERIPHERAL DEVICE TYPE				
2	(MSB)	RELATIVE INITIATOR PORT IDENTIFIER						
3								(LSB)
4	LU IDENTIFIER							
•••								
11								
12	(MSB)	IPV6 ADDRESS						
•••								
27								(LSB)
28	Device type specific parameters							
•••								
31								
32	EXTENSION DESCRIPTOR TYPE CODE (FFh)							
33	Reserved							
34								
35								
36	(MSB)	PORT NUMBER						
37								(LSB)
38	(MSB)	INTERNET PROTOCOL NUMBER						
39								(LSB)
40	Reserved							
•••								
63								

The DESCRIPTOR TYPE CODE field, LU ID TYPE field, PERIPHERAL DEVICE TYPE field, NUL bit, RELATIVE INITIATOR PORT IDENTIFIER field, LU IDENTIFIER field, and the device type specific parameters are described in 6.3.6.1.

The IPV6 ADDRESS field shall contain a unicast IPv6 address (see RFC 4291).

The EXTENSION DESCRIPTOR TYPE CODE field is described in 6.3.6.2. If the EXTENSION DESCRIPTOR TYPE CODE field does not contain the value shown in table 443, then the EXTENDED COPY command shall be terminated with CHECK CONDITION status, with the sense key set to COPY ABORTED, and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

The PORT NUMBER field shall contain the TCP port number (see 3.1.180). The TCP port number shall conform to the requirements defined by iSCSI (see RFC 3720).

The INTERNET PROTOCOL NUMBER field shall contain an Internet protocol number (see 3.1.80). The Internet protocol number shall conform to the requirements defined by iSCSI (see RFC 3720).

NOTE 62 - The internet protocol number for TCP is 0006h.

7.6.3.10 SAS serial SCSI protocol target descriptor format

The target descriptor format shown in table 444 is used by an EXTENDED COPY command to specify a SAS copy target device using its SAS address.

Table 444 — SAS serial SCSI protocol EXTENDED COPY target descriptor format

Bit / Byte	7	6	5	4	3	2	1	0
0	DESCRIPTOR TYPE CODE (E9h)							
1	LU ID TYPE		NUL	PERIPHERAL DEVICE TYPE				
2	(MSB)	RELATIVE INITIATOR PORT IDENTIFIER						
3								(LSB)
4	LU IDENTIFIER							
•••								
11								
12	SAS ADDRESS							
•••								
19								
20	Reserved							
•••								
27								
28	Device type specific parameters							
•••								
31								

The DESCRIPTOR TYPE CODE field, LU ID TYPE field, PERIPHERAL DEVICE TYPE field, NUL bit, RELATIVE INITIATOR PORT IDENTIFIER field, LU IDENTIFIER field, and the device type specific parameters are described in 6.3.6.1.

The SAS ADDRESS field contains the SAS address (see SAS).

7.6.4 TransportID identifiers

7.6.4.1 Overview of TransportID identifiers

An application client may use a TransportID to specify an initiator port other than the initiator port that is transporting the command and parameter data (e.g., as an Access identifiers (see 8.3.1.3.2) in ACL ACEs, as the

initiator port in the I_T nexus to which PERSISTENT RESERVE OUT command with REGISTER AND MOVE service action (see 5.9.8) is moving a persistent reservation).

TransportIDs (see table 445) shall be at least 24 bytes long and shall be a multiple of four bytes in length.

Table 445 — TransportID format

Bit / Byte	7	6	5	4	3	2	1	0
0	FORMAT CODE		Reserved		PROTOCOL IDENTIFIER			
1	SCSI transport protocol specific data							
•••								
n								

The FORMAT CODE field specifies the format of the TransportID. All format codes not specified in this standard are reserved.

The PROTOCOL IDENTIFIER field (see table 424 in 7.6.1) specifies the SCSI transport protocol to which the TransportID applies.

The format of the SCSI transport protocol specific data depends on the value in the PROTOCOL IDENTIFIER field. The SCSI transport protocol specific data in a TransportID shall only include initiator port identifiers, initiator port names, or SCSI device names (see SAM-4) that persist across hard resets and I_T nexus losses. TransportID formats specific to SCSI transport protocols are listed in table 446.

Table 446 — TransportID formats for specific SCSI transport protocols

SCSI transport Protocol	Protocol Standard	Reference
Fibre Channel Protocol (FCP)	FCP-4	7.6.4.2
SCSI Parallel Interface (SPI)	SPI-5	7.6.4.3
Serial Bus Protocol (SBP) (i.e., IEEE 1394)	SBP-3	7.6.4.4
Remote Direct Memory Access (RDMA) (e.g., InfiniBand™)	SRP	7.6.4.5
Internet SCSI (iSCSI)	iSCSI	7.6.4.6
SAS Protocol Layer (SPL)	SPL	7.6.4.7

7.6.4.2 TransportID for initiator ports using SCSI over Fibre Channel

A Fibre Channel TransportID (see table 447) specifies an FCP-4 initiator port based on the N_Port_Name belonging to that initiator port.

Table 447 — Fibre Channel TransportID format

Bit / Byte	7	6	5	4	3	2	1	0
0	FORMAT CODE (00b)		Reserved		PROTOCOL IDENTIFIER (0h)			
1 ••• 7	Reserved							
8 ••• 15	N_PORT_NAME							
16 ••• 23	Reserved							

The N_PORT_NAME field shall contain the N_Port_Name defined by the N_Port login (PLOGI) extended link service (see FC-FS).

7.6.4.3 TransportID for initiator ports using a parallel SCSI bus

A parallel SCSI bus TransportID (see table 448) specifies a SPI-5 initiator port based on the SCSI address of an initiator port and the relative port identifier of the target port through which the application client accesses the SCSI target device.

Table 448 — Parallel SCSI bus TransportID format

Bit / Byte	7	6	5	4	3	2	1	0
0	FORMAT CODE (00b)		Reserved		PROTOCOL IDENTIFIER (1h)			
1	Reserved							
2	(MSB)	SCSI ADDRESS						
3								(LSB)
4	Obsolete							
5								
6	(MSB)	RELATIVE TARGET PORT IDENTIFIER						
7								(LSB)
8 ••• 23	Reserved							

The SCSI ADDRESS field specifies the SCSI address (see SPI-5) of the initiator port.

The RELATIVE TARGET PORT IDENTIFIER field specifies the relative port identifier (see 3.1.127) of the target port for which the initiator port SCSI address applies. If the RELATIVE TARGET PORT IDENTIFIER does not reference a target port in the SCSI target device, the TransportID is invalid.

7.6.4.4 TransportID for initiator ports using SCSI over IEEE 1394

An IEEE 1394 TransportID (see table 449) specifies an SBP-3 initiator port based on the EUI-64 initiator port name belonging to that initiator port.

Table 449 — IEEE 1394 TransportID format

Bit / Byte	7	6	5	4	3	2	1	0
0	FORMAT CODE (00b)		Reserved		PROTOCOL IDENTIFIER (3h)			
1 ••• 7	Reserved							
8 ••• 15	EUI-64 NAME							
16 ••• 23	Reserved							

The EUI-64 NAME field shall contain the EUI-64 IEEE 1394 node unique identifier (see SBP-3) for an initiator port.

7.6.4.5 TransportID for initiator ports using SCSI over an RDMA interface

A RDMA TransportID (see table 450) specifies an SRP initiator port based on the world wide unique initiator port name belonging to that initiator port.

Table 450 — RDMA TransportID format

Bit / Byte	7	6	5	4	3	2	1	0
0	FORMAT CODE (00b)		Reserved		PROTOCOL IDENTIFIER (4h)			
1 ••• 7	Reserved							
8 ••• 23	INITIATOR PORT IDENTIFIER							

The INITIATOR PORT IDENTIFIER field shall contain an SRP initiator port identifier (see SRP).

7.6.4.6 TransportID for initiator ports using SCSI over iSCSI

An iSCSI TransportID specifies an iSCSI initiator port using one of the TransportID formats listed in table 451.

Table 451 — iSCSI TransportID formats

Format code	Description
00b	Initiator port is identified using the world wide unique SCSI device name of the iSCSI initiator device containing the initiator port (see table 452).
01b	Initiator port is identified using the world wide unique initiator port identifier (see table 453).
10b to 11b	Reserved

iSCSI TransportIDs with a format code of 00b may be rejected. iSCSI TransportIDs with a format code of 01b should not be rejected.

A iSCSI TransportID with the format code set to 00b (see table 452) specifies an iSCSI initiator port based on the world wide unique SCSI device name of the iSCSI initiator device containing the initiator port.

Table 452 — iSCSI initiator device TransportID format

Bit / Byte	7	6	5	4	3	2	1	0
0	FORMAT CODE (00b)		Reserved		PROTOCOL IDENTIFIER (5h)			
1	Reserved							
2	(MSB)	ADDITIONAL LENGTH (m-3)						
3								(LSB)
4	(MSB)	ISCSI NAME						
•••								
m								(LSB)

The ADDITIONAL LENGTH field specifies the number of bytes that follow in the TransportID. The additional length shall be at least 20 and shall be a multiple of four.

The null-terminated, null-padded (see 4.4.2) ISCSI NAME field shall contain the iSCSI name of an iSCSI initiator node (see RFC 3720). The first ISCSI NAME field byte containing an ASCII null character terminates the ISCSI NAME field without regard for the specified length of the iSCSI TransportID or the contents of the ADDITIONAL LENGTH field.

NOTE 63 - The maximum length of the iSCSI TransportID is 228 bytes because the iSCSI name length does not exceed 223 bytes.

If a iSCSI TransportID with the format code set to 00b appears in a PERSISTENT RESERVE OUT parameter list (see 6.14.3), all initiator ports known to the device server with an iSCSI node name matching the one in the TransportID shall be registered.

If a iSCSI TransportID with the format code set to 00b appears in an ACE access identifier (see 8.3.1.3.2), the logical units listed in the ACE shall be accessible to any initiator port with an iSCSI node name matching the value in the TransportID. The access controls coordinator shall reject any command that attempts to define more than one ACEs with an iSCSI TransportID access identifier containing the same iSCSI name. The command shall be terminated with CHECK CONDITION status, with the sense key ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

A iSCSI TransportID with the format code set to 01b (see table 453) specifies an iSCSI initiator port based on its world wide unique initiator port identifier.

Table 453 — iSCSI initiator port TransportID format

Bit / Byte	7	6	5	4	3	2	1	0
0	FORMAT CODE (01b)		Reserved		PROTOCOL IDENTIFIER (5h)			
1	Reserved							
2	(MSB)	ADDITIONAL LENGTH (m-3)						
3								(LSB)
4	(MSB)	ISCSI NAME						
•••								
n-1								(LSB)
n	(MSB)	SEPARATOR (2C 692C 3078h)						
•••								
n+4								(LSB)
n+5	(MSB)	ISCSI INITIATOR SESSION ID						
•••								
m								(LSB)

The ADDITIONAL LENGTH field specifies the number of bytes that follow in the TransportID encompassing the ISCSI NAME, SEPARATOR, and ISCSI INITIATOR SESSION ID fields. The additional length shall be at least 20 and shall be a multiple of four.

The ISCSI NAME field shall contain the iSCSI name of an iSCSI initiator node (see RFC 3720). The ISCSI NAME field shall not be null-terminated (see 4.4.2) and shall not be padded.

The SEPARATOR field shall contain the five ASCII characters ',i,0x'.

NOTE 64 - The notation used to define the SEPARATOR field is described in 3.5.3.

The null-terminated, null-padded ISCSI INITIATOR SESSION ID field shall contain the iSCSI initiator session identifier (see RFC 3720) in the form of ASCII characters that are the hexadecimal digits converted from the binary iSCSI initiator session identifier value. The first ISCSI INITIATOR SESSION ID field byte containing an ASCII null character terminates the ISCSI INITIATOR SESSION ID field without regard for the specified length of the iSCSI TransportID or the contents of the ADDITIONAL LENGTH field.

7.6.4.7 TransportID for initiator ports using SCSI over SAS Serial SCSI Protocol

A SAS Serial SCSI Protocol (SSP) TransportID (see table 454) specifies a SAS initiator port that is communicating via SSP using the SAS address belonging to that initiator port.

Table 454 — SAS Serial SCSI Protocol TransportID format

Bit / Byte	7	6	5	4	3	2	1	0
0	FORMAT CODE (00b)		Reserved		PROTOCOL IDENTIFIER (6h)			
1 ••• 3	Reserved							
4 ••• 11	SAS ADDRESS							
12 ••• 23	Reserved							

The SAS ADDRESS field specifies the SAS address of the initiator port.

7.7 Security protocol parameters

7.7.1 Security protocol information description

7.7.1.1 Overview

The purpose of security protocol information security protocol (i.e., the SECURITY PROTOCOL field set to 00h in a SECURITY PROTOCOL IN command) is to transfer security protocol related information from the logical unit. A SECURITY PROTOCOL IN command in which the SECURITY PROTOCOL field is set to 00h is not associated with an previous SECURITY PROTOCOL OUT command and shall be processed without regard for whether a SECURITY PROTOCOL OUT command has been processed.

If the SECURITY PROTOCOL IN command is supported, the SECURITY PROTOCOL value of 00h shall be supported as defined in this standard.

7.7.1.2 CDB description

When the SECURITY PROTOCOL field is set to 00h in a SECURITY PROTOCOL IN command, the SECURITY PROTOCOL SPECIFIC field (see table 455) contains a single numeric value as defined in 3.6.

Table 455 — SECURITY PROTOCOL SPECIFIC field for SECURITY PROTOCOL IN protocol 00h

Code	Description	Support	Reference
0000h	Supported security protocol list	Mandatory	7.7.1.3
0001h	Certificate data	Mandatory	7.7.1.4
0002h to FFFFh	Reserved		

All other CDB fields for SECURITY PROTOCOL IN command shall meet the requirements stated in 6.30.

Each time a SECURITY PROTOCOL IN command with the SECURITY PROTOCOL field set to 00h is received, the device server shall transfer the data defined 7.7.1 starting with byte 0.

7.7.1.3 Supported security protocols list description

If the SECURITY PROTOCOL field is set to 00h and the SECURITY PROTOCOL SPECIFIC field is set to 0000h in a SECURITY PROTOCOL IN command, the parameter data shall have the format shown in table 456.

Table 456 — Supported security protocols SECURITY PROTOCOL IN parameter data

Bit / Byte	7	6	5	4	3	2	1	0
0	Reserved							
•••								
5								
6	(MSB)	SUPPORTED SECURITY PROTOCOL LIST LENGTH (m-7)						
7								(LSB)
Supported security protocol list								
8	SUPPORTED SECURITY PROTOCOL [first] (00h)							
	⋮							
m	SUPPORTED SECURITY PROTOCOL [last]							
m+1	Pad bytes (optional)							
n								

The SUPPORTED SECURITY PROTOCOL LIST LENGTH field indicates the total length, in bytes, of the supported security protocol list that follows.

Each SUPPORTED SECURITY PROTOCOL field in the supported security protocols list shall contain one of the security protocol values supported by the logical unit. The values shall be listed in ascending order starting with 00h.

The total data length shall conform to the ALLOCATION LENGTH field requirements (see 6.30). Pad bytes may be appended to meet this length. Pad bytes shall have a value of 00h.

7.7.1.4 Certificate data description

7.7.1.4.1 Certificate overview

A certificate is either an X.509 Public Key Certificate (see 7.7.1.4.2) or an X.509 Attribute Certificate (see 7.7.1.4.3) depending on the capabilities of the logical unit.

If the SECURITY PROTOCOL field is set to 00h and the SECURITY PROTOCOL SPECIFIC field is set to 0001h in a SECURITY PROTOCOL IN command, the parameter data shall have the format shown in table 457.

Table 457 — Certificate data SECURITY PROTOCOL IN parameter data

Bit / Byte	7	6	5	4	3	2	1	0
0	Reserved							
1								
2	(MSB)	CERTIFICATE LENGTH (m-3)						
3								(LSB)
4	CERTIFICATE							
•••								
m								
m+1	Pad bytes (optional)							
n								

The CERTIFICATE LENGTH field indicates the total length, in bytes, of the certificate or certificates that follow. The length may include more than one certificates. If the device server doesn't have a certificate to transfer, the CERTIFICATE LENGTH field shall be set to 0000h.

The contents of the CERTIFICATE field are defined in 7.7.1.4.2 and 7.7.1.4.3.

The total data length shall conform to the ALLOCATION LENGTH field requirements (see 6.30). Pad bytes may be appended to meet this length. Pad bytes shall have a value of 00h.

7.7.1.4.2 Public Key certificate description

RFC 5280 defines the certificate syntax for certificates consistent with X.509v3 Public Key Certificate Specification. Any further restrictions beyond the requirements of RFC 5280 are yet to be defined by T10.

7.7.1.4.3 Attribute certificate description

RFC 3281 defines the certificate syntax for certificates consistent with X.509v2 Attribute Certificate Specification. Any further restrictions beyond the requirements of RFC 3281 are yet to be defined by T10.

7.7.2 SA creation capabilities

7.7.2.1 Overview

If the SECURITY PROTOCOL field in a SECURITY PROTOCOL IN command (see 6.30) is set to 40h, then the command returns information related to the SA creation (see 5.16.2.3) capabilities provided by the device server.

The SA creation capabilities protocol is independent of any other SA creation protocols. The device server shall not refuse to process an SA creation capabilities SECURITY PROTOCOL IN command, and this processing shall not affect the state maintained for any SA creation CCS (e.g., an IKEv2-SCSI CCS) on any I_T_L nexus. Except for those cases where an SA creation capabilities SECURITY PROTOCOL IN command reports changed SA creation capabilities, processing of the command shall not affect the concurrent processing of any commands that are part of an SA creation CCS.

If any SA creation protocols are supported, the SA creation capabilities protocol shall be supported as described in 7.7.2.

The SA creation capabilities SECURITY PROTOCOL IN CDB format is described in 7.7.2.2.

As shown in table 458 (see 7.7.2.2), the format of the parameter data returned by a SA creation capabilities SECURITY PROTOCOL IN command depends on the value in the SECURITY PROTOCOL SPECIFIC field in the CDB.

7.7.2.2 SA creation capabilities CDB description

The SA creation capabilities SECURITY PROTOCOL IN CDB has the format defined in 6.30 with the additional requirements described in this subclause.

When the SECURITY PROTOCOL field is set to SA creation capabilities (i.e., 40h) in a SECURITY PROTOCOL IN command, the SECURITY PROTOCOL SPECIFIC field (see table 458) identifies the SA creation protocol (see 5.16.2.3) for which the device server shall return capability information.

Table 458 — SECURITY PROTOCOL SPECIFIC field for the SA creation capabilities

Code	Description	Parameter data format
0000h	Supported device server capabilities formats	7.7.2.3.1
0001h to 0100h	Reserved	
0101h	IKEv2-SCSI device server capabilities	7.7.2.3.2
0102h to EFFFh	Reserved	
F000h to FFFFh	Vendor Specific	

If an SA creation capabilities SECURITY PROTOCOL IN command is received with the INC_512 bit is set to one, then the SECURITY PROTOCOL IN 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.

7.7.2.3 SA creation capabilities parameter data formats

7.7.2.3.1 Supported device server capabilities formats parameter data format

The supported device server capabilities formats parameter data (see table 459) indicates which device server capabilities parameter data formats are supported by the device server.

Table 459 — Supported device server capabilities formats parameter data

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)							
•••				PARAMETER DATA LENGTH (n-3)				
3								(LSB)
Supported capabilities parameter data formats								
4				CAPABILITIES PARAMETER DATA FORMAT [first] (0000h)				
5								
				⋮				
n-1				CAPABILITIES PARAMETER DATA FORMAT [last]				
n								

The PARAMETER DATA LENGTH field indicates the number of bytes that follow in the parameter data.

Each CAPABILITIES PARAMETER DATA FORMAT field in the supported capabilities parameter data formats list shall contain one of the SECURITY PROTOCOL SPECIFIC field values (see table 458) supported by the device server. The values shall be listed in ascending order starting with 0000h.

7.7.2.3.2 IKEv2-SCSI device server capabilities parameter data format

The IKEv2-SCSI device server capabilities parameter data (see table 460) indicates the IKEv2 transforms (i.e., key exchange protocols and authentication protocols) supported by the device server for IKEv2-SCSI.

Table 460 — IKEv2-SCSI device server capabilities parameter data

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)							
•••				PARAMETER DATA LENGTH (n-3)				
3								(LSB)
4				IKEv2-SCSI SA Creation Capabilities payload (see 7.7.3.5.12)				
•••								
n								

The PARAMETER DATA LENGTH field indicates the number of bytes that follow in the parameter data.

The IKEv2-SCSI SA Creation Capabilities payload (see 7.7.3.5.12) indicates the algorithms supported by IKEv2-SCSI in the Key Exchange step (see 5.16.4.6) and Authentication step (see 5.16.4.7).

NOTE 65 - The primary content of the IKEv2-SCSI device server capabilities SA creation capabilities parameter data is an IKEv2-SCSI payload (see 7.7.3.5) because the IKEv2-SCSI SA Creation Capabilities payload is used by the device server and application client in the construction of other IKEv2-SCSI payloads (see 5.16.4).

7.7.3 IKEv2-SCSI

7.7.3.1 Overview

If the SECURITY PROTOCOL field in a SECURITY PROTOCOL OUT command (see 6.31) or a SECURITY PROTOCOL IN command (see 6.30) is set to 41h, then the command is part of an IKEv2-SCSI CCS (see 5.16.4) and is used to transfer IKEv2-SCSI protocol information to or from the device server.

In an IKEv2-SCSI CCS, a defined sequence of SECURITY PROTOCOL OUT and SECURITY PROTOCOL IN commands are sent by the application client and processed by the device server as summarized in 5.16.4.1.

The IKEv2-SCSI SECURITY PROTOCOL OUT CDB format is described in 7.7.3.3.

The IKEv2-SCSI SECURITY PROTOCOL IN CDB format is described in 7.7.3.2.

The IKEv2-SCSI SECURITY PROTOCOL OUT command and the IKEv2-SCSI SECURITY PROTOCOL IN command use the same parameter data format, and this format is described in 7.7.3.4. A significant component of the IKEv2-SCSI parameter data format is one or more IKE payloads, and the format of IKE payloads is described in 7.7.3.5.

If the IKEv2-SCSI SA creation protocol is supported (see 7.7.1), the SA creation capabilities protocol (see 7.7.2) shall also be supported.

7.7.3.2 IKEv2-SCSI SECURITY PROTOCOL IN CDB description

The IKEv2-SCSI SECURITY PROTOCOL IN CDB has the format defined in 6.30 with the additional requirements described in this subclause.

When the SECURITY PROTOCOL field is set to IKEv2-SCSI (i.e., 41h) in a SECURITY PROTOCOL IN command, the SECURITY PROTOCOL SPECIFIC field (see table 461) identifies the IKEv2-SCSI step (see 5.16.4.1) that the device server is to process. If the IKEv2-SCSI SA creation protocol is supported (see 7.7.1), the SECURITY PROTOCOL IN command support requirements are shown in table 461.

Table 461 — SECURITY PROTOCOL SPECIFIC field for the IKEv2-SCSI SECURITY PROTOCOL IN command

Code	Description	Support	References	
			Usage	Data format
0000h to 00FFh	Restricted		RFC 4306	RFC 4306
0100h to 0101h	Reserved			
0102h	Key Exchange step	Mandatory	5.16.4.6.3	7.7.3.4
0103h	Authentication step	Mandatory	5.16.4.7.3	7.7.3.4
0104h to EFFFh	Reserved			
F000h to FFFFh	Vendor Specific			

If an IKEv2-SCSI SECURITY PROTOCOL IN command is received with the INC_512 bit is set to one while the device server is maintaining state for an IKEv2-SCSI CCS on the I_T_L nexus on which the command was received (see 5.16.4.1), then:

a) the SECURITY PROTOCOL IN command shall be terminated with CHECK CONDITION status, with the sense key set to NOT READY, and the additional sense code set to LOGICAL UNIT NOT READY, SA CREATION IN PROGRESS; and
b) the device server shall continue the IKEv2-SCSI CCS by preparing to receive another SECURITY PROTOCOL IN command or SECURITY PROTOCOL OUT command, as appropriate.

If an IKEv2-SCSI SECURITY PROTOCOL IN command is received with the INC_512 bit set to one while the device server is not maintaining state for an IKEv2-SCSI CCS (see 5.16.4.1), then the SECURITY PROTOCOL IN 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.

7.7.3.3 IKEv2-SCSI SECURITY PROTOCOL OUT CDB description

The IKEv2-SCSI SECURITY PROTOCOL OUT CDB has the format defined in 6.31 with the additional requirements described in this subclause.

When the SECURITY PROTOCOL field is set to IKEv2-SCSI (i.e., 41h) in a SECURITY PROTOCOL OUT command, the SECURITY PROTOCOL SPECIFIC field (see table 462) identifies the IKEv2-SCSI step (see 5.16.4.1) that the device server is to process. If the IKEv2-SCSI SA creation protocol is supported (see 7.7.1), the SECURITY PROTOCOL IN command support requirements are shown in table 462.

Table 462 — SECURITY PROTOCOL SPECIFIC field for the IKEv2-SCSI SECURITY PROTOCOL OUT command

Code	Description	Support	References	
			Usage	Data format
0000h to 00FFh	Restricted		RFC 4306	RFC 4306
0100h to 0101h	Reserved			
0102h	Key Exchange step	Mandatory	5.16.4.6.3	7.7.3.4
0103h	Authentication step	Mandatory	5.16.4.7.3	7.7.3.4
0104h	Delete operation	Mandatory	5.16.4.11	7.7.3.4
0105h to EFFFh	Reserved			
F000h to FFFFh	Vendor Specific			

If an IKEv2-SCSI SECURITY PROTOCOL OUT command is received with the INC_512 bit is set to one while the device server is maintaining state for an IKEv2-SCSI CCS on the I_T_L nexus on which the command was received (see 5.16.4.1), then:

a) the SECURITY PROTOCOL OUT command shall be terminated with CHECK CONDITION status, with the sense key set to NOT READY, and the additional sense code set to LOGICAL UNIT NOT READY, SA CREATION IN PROGRESS; and
b) the device server shall continue the IKEv2-SCSI CCS by preparing to receive another SECURITY PROTOCOL OUT command or SECURITY PROTOCOL IN command, as appropriate.

If an IKEv2-SCSI SECURITY PROTOCOL OUT command is received with the INC_512 bit is set to one while the device server is not maintaining state for an IKEv2-SCSI CCS (see 5.16.4.1), then the SECURITY PROTOCOL OUT command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB.

Any IKEv2-SCSI SECURITY PROTOCOL OUT command with an transfer length of up to 16 384 bytes shall not be terminated with an error due to the number of bytes to be transferred and processed.

7.7.3.4 IKEv2-SCSI parameter data format

Table 463 shows the parameter list format used by a SECURITY PROTOCOL OUT command and the parameter data format used by a SECURITY PROTOCOL IN command when the SECURITY PROTOCOL field is set to IKEv2-SCSI (i.e., 41h).

Table 463 — IKEv2-SCSI SECURITY PROTOCOL OUT and SECURITY PROTOCOL IN parameter data

Bit / Byte	7	6	5	4	3	2	1	0
IKEv2-SCSI header								
0 ••• 3	Restricted (see RFC 4306)							
4 ••• 7	(MSB)	IKE_SA APPLICATION CLIENT SAI						(LSB)
8 ••• 11	Restricted (see RFC 4306)							
12 ••• 15	(MSB)	IKE_SA DEVICE SERVER SAI						(LSB)
16	NEXT PAYLOAD							
17	MAJOR VERSION (2h)				MINOR VERSION			
18	EXCHANGE TYPE							
19	Reserved			INTTR	VERSION	RSPNS	Reserved	
20 ••• 23	(MSB)	MESSAGE ID						(LSB)
24 ••• 27	(MSB)	IKE LENGTH (n+1)						(LSB)
IKEv2-SCSI payloads								
28 •••	IKEv2-SCSI payload [first] (see 7.7.3.5)							
	⋮							
••• n	IKEv2-SCSI payload [last] (see 7.7.3.5)							

The IKE_SA APPLICATION CLIENT SAI field contains the value that is or is destined to become the AC_SAI SA parameter (see 5.16.2.2) in the generated SA (see 5.16.4.9). The AC_SAI is chosen by the application client to uniquely identify its representation of the SA that is being negotiated or managed (e.g., deleted).

If the device server receives an IKEv2-SCSI header with the IKE_SA APPLICATION CLIENT SAI field set to zero, then the error shall be processed as described in 7.7.3.8.

To increase procedural integrity checking, the application client should compare the IKE_SA APPLICATION CLIENT SAI field contents in any SECURITY PROTOCOL IN parameter data it receives to the value that the application client is maintaining for the IKEv2-SCSI CCS or SA management. If the two values are not identical, the application client should abandon the IKEv2-SCSI CCS, if any, and notify the device server that the IKEv2-SCSI CCS, if any, is being abandoned as described in 5.16.4.10.

Except in the Key Exchange step SECURITY PROTOCOL OUT command (see 5.16.4.6.2), the IKE_SA DEVICE SERVER SAI field contains the value that is or is destined to become the DS_SAI SA parameter in the generated SA. The DS_SAI is chosen by the device server in accordance with the requirements in 5.16.2.1 to uniquely identify its representation of the SA that is being negotiated. In the Key Exchange step SECURITY PROTOCOL OUT command the IKE_SA DEVICE SERVER SAI field is reserved.

To increase procedural integrity checking, the application client should compare the IKE_SA DEVICE SERVER SAI field contents in the Authentication step SECURITY PROTOCOL IN parameter data it receives to the value that the application client is maintaining for the IKEv2-SCSI CCS. If the two values are not identical, the application client should abandon the IKEv2-SCSI CCS and notify the device server that the IKEv2-SCSI CCS is being abandoned as described in 5.16.4.10.

The device server shall validate the contents of the IKE_SA APPLICATION CLIENT SAI field and the IKE_SA DEVICE SERVER SAI field as shown in table 464.

Table 464 — IKEv2-SCSI header checking of SAIs

Contents of SECURITY PROTOCOL SPECIFIC field in SECURITY PROTOCOL OUT CDB	Expected contents for …		Device server action if expected field contents not received
	IKE_SA APPLICATION CLIENT SAI field	IKE_SA DEVICE SERVER SAI field	
0102h (i.e., Key Exchange step)	any value	reserved	No actions taken based on expected field contents
0103h (i.e., Authentication step)	A match with the SAI values maintained for an IKEv2-SCSI CCS on the I_T_L nexus on which the command was received		The device server shall: a) 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 SA CREATION PARAMETER VALUE REJECTED; and b) continue the IKEv2-SCSI CCS by preparing to receive another Authentication step SECURITY PROTOCOL OUT command
0104h (i.e., Delete operation)	If at least one IKEv2-SCSI CCS is being maintained for the I_T_L nexus on which the command was received, then: a) a match with the SAI values maintained for an IKEv2-SCSI CCS; or b) a match with the SAI values maintained for any active SA		The device server shall: a) 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 SA CREATION PARAMETER VALUE REJECTED; and b) continue the IKEv2-SCSI CCS by preparing to receive another Authentication step SECURITY PROTOCOL OUT command
	If no IKEv2-SCSI CCS is being maintained for the I_T_L nexus on which the command was received, then a match with the SAI values maintained for any active SA		The SECURITY PROTOCOL OUT command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST

The NEXT PAYLOAD field (see table 465) identifies the first IKEv2-SCSI payload that follows the IKEv2-SCSI header.

Table 465 — NEXT PAYLOAD field

Code	IKE Payload Name	Support requirements in SECURITY PROTOCOL …		Reference
		IN	OUT	
00h	No Next Payload	Mandatory		7.7.3.5.2
01h to 20h		Reserved		
21h	Security Association	Prohibited [a]		RFC 4306
22h	Key Exchange	Mandatory		7.7.3.5.3
23h	Identification – Application Client	Prohibited	Mandatory	7.7.3.5.4
24h	Identification – Device Server	Mandatory	Prohibited	7.7.3.5.4
25h	Certificate	Optional		7.7.3.5.5
26h	Certificate Request	Optional		7.7.3.5.6
27h	Authentication	Mandatory		7.7.3.5.7
28h	Nonce	Mandatory		7.7.3.5.8
29h	Notify	Prohibited	Mandatory	7.7.3.5.9
2Ah	Delete	Prohibited	Mandatory	7.7.3.5.10
2Bh	Vendor ID	Prohibited		RFC 4306
2Ch	Traffic Selector – Application Client	Prohibited		RFC 4306
2Dh	Traffic Selector – Device Server	Prohibited		RFC 4306
2Eh	Encrypted	Mandatory		7.7.3.5.11
2Fh	Configuration	Prohibited		RFC 4306
30h	Extensible Authentication	Prohibited		RFC 4306
31h to 7Fh		Restricted		RFC 4306
80h	IKEv2-SCSI SA Creation Capabilities	Mandatory		7.7.3.5.12
81h	IKEv2-SCSI SA Cryptographic Algorithms	Mandatory		7.7.3.5.13
82h	IKEv2-SCSI SAUT Cryptographic Algorithms	Mandatory		7.7.3.5.14
83h	IKEv2-SCSI Timeout Values	Mandatory		7.7.3.5.15
84h to BFh		Reserved		
C0h to FFh	Vendor Specific			

[a] The Security Association payload type value is not used in IKEv2-SCSI. The IKEv2-SCSI SA Cryptographic Algorithms payload (i.e., 81h) and IKEv2-SCSI SAUT Cryptographic Algorithms payload (i.e., 82h) are used instead.

The MAJOR VERSION field shall contain the value 2h. If a device server receives an IKEv2-SCSI header with a MAJOR VERSION field containing a value other than 2h, then the error shall be processed as described in 7.7.3.8.

The MINOR VERSION field is reserved.

The EXCHANGE TYPE field is reserved.

The initiator (INTTR) bit shall be set to:

a) one for SECURITY PROTOCOL OUT commands; and
b) zero for SECURITY PROTOCOL IN commands.

If a device server receives an IKEv2-SCSI header with the INTTR bit set to zero, then the error shall be processed as described in 7.7.3.8.

If an application client receives an IKEv2-SCSI header with the INTTR bit set to one, it should abandon the IKEv2-SCSI CCS and notify the device server that the IKEv2-SCSI CCS is being abandoned as described in 5.16.4.10.

The VERSION bit is reserved.

The response (RSPNS) bit shall be set to:

a) zero for SECURITY PROTOCOL OUT commands; and
b) one for SECURITY PROTOCOL IN commands.

If a device server receives an IKEv2-SCSI header with the RSPNS bit set to one, then the error shall be processed as described in 7.7.3.8.

If an application client receives an IKEv2-SCSI header with the RSPNS bit set to zero, it should abandon the IKEv2-SCSI CCS and notify the device server that the IKEv2-SCSI CCS is being abandoned as described in 5.16.4.10.

The MESSAGE ID field (see table 466) identifies the function of the parameter data.

Table 466 — MESSAGE ID field

Code	Description
0000 0000h	Key Exchange step SECURITY PROTOCOL OUT command (see 5.16.4.6.2) or SECURITY PROTOCOL IN command (see 5.16.4.6.3)
0000 0001h	Authentication step SECURITY PROTOCOL OUT command (see 5.16.4.7.2) or SECURITY PROTOCOL IN command (see 5.16.4.7.3)
0000 0002h	Delete operation in a SECURITY PROTOCOL OUT command
0000 0003h to 0000 FFFFh	Reserved

If the device server receives a SECURITY PROTOCOL OUT command with an invalid MESSAGE ID field in its IKEv2-SCSI header, then the error shall be processed as described in 7.7.3.8.

If the application client receives an invalid MESSAGE ID field in the parameter data for a SECURITY PROTOCOL IN command, the application client should abandon the IKEv2-SCSI CCS and notify the device server that the IKEv2-SCSI CCS is being abandoned as described in 5.16.4.10.

The IKE LENGTH field contains the total number of bytes in the parameter data, including the IKEv2-SCSI header and all the IKEv2-SCSI payloads.

NOTE 66 - The contents of the IKE LENGTH field differ from those found in most SCSI length fields, however, they are consistent with the IKEv2 usage (see RFC 4306).

Each IKEv2-SCSI payload (see 7.7.3.5) contains specific data related to the operation being performed. A specific combination of IKEv2-SCSI payloads is specified for each operation (e.g., Key Exchange) as summarized in 5.16.4.2. The Encryption payload (see 7.7.3.6.2) nests one set of IKEv2-SCSI payloads inside another.

Based on the contents of the SECURITY PROTOCOL SPECIFIC field in the SECURITY PROTOCOL OUT CDB, the device server shall:

a) validate the contents of the NEXT PAYLOAD field in the IKEv2-SCSI header as shown in table 467, before performing any decryption or integrity checking; and
b) validate that the number of instances of each next payload value shown in table 467 occur in the parameter data, after the encrypted data, if any, is decrypted and integrity checked (i.e., if the NEXT PAYLOAD field in the IKEv2-SCSI header contains 2Eh, after the Encrypted payload is decrypted and integrity checked).

If the next payload values in the IKEv2-SCSI header and the unencrypted SECURITY PROTOCOL OUT parameter data do not meet the requirements shown in table 467 for the listed SECURITY PROTOCOL SPECIFIC field contents, then the error shall be processed as described in 7.7.3.8.

Based on the contents of the SECURITY PROTOCOL SPECIFIC field in the SECURITY PROTOCOL IN CDB, the device server shall:

a) place one of the next payload values allowed by table 467 in the NEXT PAYLOAD field in the IKEv2-SCSI header; and
b) include the number of instances of each next payload value shown in table 467 in the parameter data either before or after encryption, if applicable (i.e., if the NEXT PAYLOAD field in the IKEv2-SCSI header contains 2Eh, before the contents of the Encrypted payload are encrypted and integrity check value is computed).

Table 467 — Next payload values in SECURITY PROTOCOL OUT/IN parameter data (part 1 of 3)

Next payload value	Next payload value allowed in IKEv2-SCSI header	Number of times a next payload value is allowed
SECURITY PROTOCOL OUT command with SECURITY PROTOCOL SPECIFIC field set to 0102h (i.e., Key Exchange step)		**Authentication step not skipped (see 5.16.4.1)**
83h (i.e., IKEv2-SCSI Timeout Values)	Yes	1
81h (i.e., IKEv2-SCSI SA Cryptographic Algorithms)	Yes	1
82h (i.e., IKEv2-SCSI SAUT Cryptographic Algorithms)	No	0
22h (i.e., Key Exchange)	Yes	1
28h (i.e., Nonce)	Yes	1
00h (i.e., No Next Payload)	No	1
All other next payload codes	No	0

Table 467 — Next payload values in SECURITY PROTOCOL OUT/IN parameter data (part 2 of 3)

Next payload value	Next payload value allowed in IKEv2-SCSI header	Number of times a next payload value is allowed
SECURITY PROTOCOL OUT command with SECURITY PROTOCOL SPECIFIC field set to 0102h (i.e., Key Exchange step)		**Authentication step skipped (see 5.16.4.1)**
83h (i.e., IKEv2-SCSI Timeout Values)	Yes	1
81h (i.e., IKEv2-SCSI SA Cryptographic Algorithms)	Yes	1
82h (i.e., IKEv2-SCSI SAUT Cryptographic Algorithms)	Yes	1
22h (i.e., Key Exchange)	Yes	1
28h (i.e., Nonce)	Yes	1
00h (i.e., No Next Payload)	No	1
All other next payload codes	No	0
SECURITY PROTOCOL IN command with SECURITY PROTOCOL SPECIFIC field set to 0102h (i.e., Key Exchange step)		**Authentication step not skipped (see 5.16.4.1)**
81h (i.e., IKEv2-SCSI SA Cryptographic Algorithms)	Yes	1
82h (i.e., IKEv2-SCSI SAUT Cryptographic Algorithms)	No	0
22h (i.e., Key Exchange)	Yes	1
28h (i.e., Nonce)	Yes	1
26h (i.e., Certificate Request)	Yes	0 or more
00h (i.e., No Next Payload)	No	1
All other next payload codes	No	0
SECURITY PROTOCOL IN command with SECURITY PROTOCOL SPECIFIC field set to 0102h (i.e., Key Exchange step)		**Authentication step skipped (see 5.16.4.1)**
81h (i.e., IKEv2-SCSI SA Cryptographic Algorithms)	Yes	1
82h (i.e., IKEv2-SCSI SAUT Cryptographic Algorithms)	Yes	1
22h (i.e., Key Exchange)	Yes	1
28h (i.e., Nonce)	Yes	1
26h (i.e., Certificate Request)	Yes	0 or more
00h (i.e., No Next Payload)	No	1
All other next payload codes	No	0

Table 467 — Next payload values in SECURITY PROTOCOL OUT/IN parameter data (part 3 of 3)

Next payload value	Next payload value allowed in IKEv2-SCSI header	Number of times a next payload value is allowed
SECURITY PROTOCOL OUT command with SECURITY PROTOCOL SPECIFIC field set to 0103h (i.e., Authentication step)		
2Eh (i.e., Encrypted)	Yes	1
23h (i.e., Identification – Application Client)	No	1
82h (i.e., IKEv2-SCSI SAUT Cryptographic Algorithms)	No	1
25h (i.e., Certificate)	No	0 or more
26h (i.e., Certificate Request)	No	0 or more
29h (i.e., Notify)	No	0 or 1
27h (i.e., Authentication)	No	1
00h (i.e., No Next Payload)	No	1
All other next payload codes	No	0
SECURITY PROTOCOL IN command with SECURITY PROTOCOL SPECIFIC field set to 0103h (i.e., Authentication step)		
2Eh (i.e., Encrypted)	Yes	1
24h (i.e., Identification – Device Server)	No	1
82h (i.e., IKEv2-SCSI SAUT Cryptographic Algorithms)	No	1
25h (i.e., Certificate)	No	0 or more
27h (i.e., Authentication)	No	1
00h (i.e., No Next Payload)	No	1
All other next payload codes	No	0
SECURITY PROTOCOL OUT command with SECURITY PROTOCOL SPECIFIC field set to 0104h (i.e., Delete operation)		
2Eh (i.e., Encrypted)	Yes	1
2Ah (i.e., Delete)	No	1
00h (i.e., No Next Payload)	No	1
All other next payload codes	No	0

7.7.3.5 IKEv2-SCSI payloads

7.7.3.5.1 IKEv2-SCSI payload format

Each IKEv2-SCSI payload (see table 468) is composed of a header and data that is specific to the payload type.

Table 468 — IKEv2-SCSI payload format

Bit / Byte	7	6	5	4	3	2	1	0
IKEv2-SCSI payload header								
0	NEXT PAYLOAD							
1	CRIT	Reserved						
2	(MSB)	IKE PAYLOAD LENGTH (n+1)						
3								(LSB)
IKEv2-SCSI payload-specific data								
4								
•••								
n								

The NEXT PAYLOAD field identifies the IKEv2-SCSI payload that follows this IKEv2-SCSI payload using one of the code values shown in table 465 (see 7.7.3.4).

If a device server receives an IKEv2-SCSI payload that it does not recognize (e.g., an IKEv2-SCSI payload identified by a next payload value of 01h) with the critical (CRIT) bit set to one, then the error shall be processed as described in 7.7.3.8.

If an application client receives an IKEv2-SCSI payload that it does not recognize with the CRIT bit set to one, then the application client should abandon the IKEv2-SCSI CCS and notify the device server that it is abandoning the IKEv2-SCSI CCS as described in 5.16.4.10.

If an application client or device server receives an IKEv2-SCSI payload that it does not recognize with the CRIT bit set to zero, then it should use the NEXT PAYLOAD field and the IKE PAYLOAD LENGTH field to skip processing of the unrecognized IKEv2-SCSI payload and continue processing at the next IKEv2-SCSI payload.

The IKE PAYLOAD LENGTH field contains the total number of bytes in the payload, including the IKEv2-SCSI payload header. The value in the IKE PAYLOAD LENGTH field need not be a multiple of two or four (i.e., no byte alignment is maintained among IKEv2-SCSI payloads).

NOTE 67 - The contents of the IKE PAYLOAD LENGTH field differ from those found in most SCSI length fields, however, they are consistent with the IKEv2 usage (see RFC 4306).

The format and contents of the IKEv2-SCSI payload-specific data depends on the value in the NEXT PAYLOAD field of:

a) the IKEv2-SCSI header (see 7.7.3.4), if this is the first IKEv2-SCSI payload in the parameter data; or
b) the previous IKEv2-SCSI payload, in all other cases.

7.7.3.5.2 No Next payload

A NEXT PAYLOAD field that is set to 00h (i.e., No Next payload) specifies that no more IKEv2-SCSI payloads follow the current payload. The IKEv2-SCSI No Next payload contains no bytes and has no format.

7.7.3.5.3 Key Exchange payload

The Key Exchange payload (see table 469) transfers Diffie-Hellman shared key exchange data between an application client and a device server or vice versa.

Table 469 — Key Exchange payload format

Bit / Byte	7	6	5	4	3	2	1	0
0	NEXT PAYLOAD							
1	CRIT (1b)	Reserved						
2	(MSB)	IKE PAYLOAD LENGTH (n+1)						
3								(LSB)
4	(MSB)	DIFFIE-HELLMAN GROUP NUMBER						
5								(LSB)
6		Reserved						
7								
8		KEY EXCHANGE DATA						
•••								
n								

The NEXT PAYLOAD field, CRIT bit, and IKE PAYLOAD LENGTH field are described in 7.7.3.5.1.

The CRIT bit is set to one in the Key Exchange payload.

The DIFFIE-HELLMAN GROUP NUMBER field contains the least significant 16 bits from ALGORITHM IDENTIFIER field in the D-H IKEv2-SCSI algorithm descriptor (see 7.7.3.6.5) in the IKEv2-SCSI SA Cryptographic Algorithms payload (see 7.7.3.5.13).

The KEY EXCHANGE DATA field contains the sender's Diffie-Hellman public value for this key exchange. The binary representation of key exchange data is specified in the reference cited for the Diffie-Hellman group that is used (see table 494).

When a prime modulus (i.e., mod p) Diffie-Hellman group is used, the length of the Diffie-Hellman public value shall be equal to the length of the prime modulus over which the exponentiation was performed as shown in table 494 (see 7.7.3.6.5). Zero bits shall be prepended to the KEY EXCHANGE DATA field if necessary.

Diffie-Hellman exponential reuse and reuse of analogous Diffie-Hellman public values for Diffie-Hellman mechanisms not based on exponentiation should not be performed, but if performed it shall be constrained (e.g., requirements regarding when Diffie-Hellman information is discarded) as specified in RFC 4306. The freshness and randomness of the random nonces are critical to the security of IKEv2-SCSI when Diffie-Hellman exponentials and public values are reused (see RFC 4306).

7.7.3.5.4 Identification – Application Client payload and Identification – Device Server payload

The Identification – Application Client payload (see table 470) transfers identification information from the application client to the device server. The Identification – Device Server payload (see table 470) transfers identification information from the device server to the application client.

Table 470 — Identification payload format

Bit / Byte	7	6	5	4	3	2	1	0
0	NEXT PAYLOAD							
1	CRIT (1b)	Reserved						
2	(MSB)	IKE PAYLOAD LENGTH (n+1)						
3								(LSB)
4	ID TYPE							
5	Reserved							
7								
8	IDENTIFICATION DATA							
•••								
n								

The NEXT PAYLOAD field, CRIT bit, and IKE PAYLOAD LENGTH field are described in 7.7.3.5.1.

The CRIT bit is set to one in the Identification – Application Client payload and the Identification – Device Server payload.

The ID TYPE field describes the contents of the IDENTIFICATION DATA field and shall contain one of the named values shown in table 471.

Table 471 — ID TYPE field

Code	Name [a]	Contents of the IDENTIFICATION DATA field
09h	ID_DER_ASN1_DN	The value of a certificate subject field (see RFC 5280)
0Ah	ID_DER_ASN1_GN	The value of a name contained in a Subject Alternative Name (i.e., SubjectAltName) certificate extension (see RFC 5280)
0Bh	ID_KEY_ID	Arbitrary identity data (e.g., SCSI port names, SCSI device names)
0Ch	ID_FC_NAME	FC-SP certificates that certify a Fibre Channel name as an identity to be used (see RFC 4595 and FC-SP)
all other values	Prohibited	

[a] See RFC 4306 and RFC 4595 for additional information about the associated identification data for these names.

The contents of the IDENTIFICATION DATA field depend on the value in the ID TYPE field.

When the Certificate payload is included in the parameter data, the identity in the Identification – Application Client payload or Identification – Device Server payload is not required to match anything in the Certificate payload (see RFC 4306). A mechanism outside the scope of this standard shall be provided to configure any application client or device server to require a match between the identity in an Identification payload and the subject name or subject alternative name in a Certificate payload.

If a device server receives an Identification – Application Client payload that does not conform to the requirements in RFC 4306 or the requirements in this subclause, then the IKEv2-SCSI CCS state maintained for the I_T_L nexus shall be abandoned as described in 7.7.3.8.3.

If an application client receives an Identification – Device Server payload that does not conform to the requirements in RFC 4306 or the requirements in this subclause, then the application client should abandon the IKEv2-SCSI CCS and notify the device server that it is abandoning the IKEv2-SCSI CCS as described in 5.16.4.10.

7.7.3.5.5 Certificate payload

The Certificate payload (see table 472) delivers a requested identity authentication certificate. The protocol for using Certificate payloads is described in 5.16.4.3.3.4.

Table 472 — Certificate payload format

Bit / Byte	7	6	5	4	3	2	1	0
0	NEXT PAYLOAD							
1	CRIT (1b)	Reserved						
2	(MSB)	IKE PAYLOAD LENGTH (n+1)						
3								(LSB)
4	CERTIFICATE ENCODING							
5	CERTIFICATE DATA							
•••								
n								

The NEXT PAYLOAD field, CRIT bit, and IKE PAYLOAD LENGTH field are described in 7.7.3.5.1.

The CRIT bit is set to one in the Certificate payload.

The CERTIFICATE ENCODING field describes the contents of the CERTIFICATE DATA field and shall contain one of the values shown in table 473.

Table 473 — CERTIFICATE ENCODING field

Code	Description	Reference
00h	Reserved	
01h to 03h	Prohibited	Annex C
04h	X.509 Certificate - Signature	RFC 4306
05h to 0Ah	Prohibited	Annex C
0Bh	Raw RSA Key	RFC 4306 and RFC 4718
0Ch to 0Dh	Prohibited	Annex C
0Eh to C8h	Restricted	IANA
C9h to FFh	Reserved	

The contents of the CERTIFICATE DATA field depend on the value in the CERTIFICATE ENCODING field.

The relationship between the Certificate payload and the Identification payload is described in 7.7.3.5.4.

Device servers that support certificates should support a mechanism outside the scope of this standard for replacing certificates and have the ability to store more than one certificate to facilitate such replacements.

7.7.3.5.6 Certificate Request payload

The Certificate Request payload (see table 474) allows an application client or device server to request the use of certificates as part of identity authentication and to name one or more trust anchors (see RFC 4306) for the certificate verification process. The Certificate payload (see table 472) delivers a requested identity authentication certificate. The protocol for using Certificate Request payloads is described in 5.16.4.3.3.4.

Table 474 — Certificate Request payload format

Bit / Byte	7	6	5	4	3	2	1	0
0	NEXT PAYLOAD							
1	CRIT (1b)	Reserved						
2	(MSB)	IKE PAYLOAD LENGTH (n+1)						
3								(LSB)
4	CERTIFICATE ENCODING							
Certification authority list								
5	(MSB)	CERTIFICATE AUTHORITY [first]						
•••								
24								(LSB)
	⋮							
n-19	(MSB)	CERTIFICATE AUTHORITY [last]						
•••								
n								(LSB)

The NEXT PAYLOAD field, CRIT bit, and IKE PAYLOAD LENGTH field are described in 7.7.3.5.1.

The CRIT bit is set to one in the Certificate Request payload.

The value in the CERTIFICATE ENCODING field (see table 473 in 7.7.3.5.5) indicates the type or format of certificate being requested. Multiple Certificate Request payloads may be included in the parameter data transferred by a single command. If the parameter data contains more than one Certificate Request payload, each Certificate Request payload should have a different value in the CERTIFICATE ENCODING field.

Each CERTIFICATION AUTHORITY field contains an indicator of a trusted authority for the certificate type indicated by the CERTIFICATE ENCODING field in this Certificate Request payload. The indicator is a SHA-1 hash of the public key of a trusted Certification Authority. The indicator is encoded as the SHA-1 hash of the Subject Public Key Info element from the Trust Anchor certificate (see RFC 5280).

Device servers that support certificates should support a mechanism outside the scope of this standard for replacing certification authority values, and shall have the ability to store one or more certification authority values to facilitate such replacements.

7.7.3.5.7 Authentication payload

The Authentication payload (see table 475) allows the application client and a device server to verify that the data transfers in their IKEv2-SCSI CCS have not be compromised by a man-in-the-middle attack (see 5.16.1.4).

Table 475 — Authentication payload format

Bit / Byte	7	6	5	4	3	2	1	0
0	NEXT PAYLOAD							
1	CRIT (1b)	Reserved						
2	(MSB)	IKE PAYLOAD LENGTH (n+1)						
3								(LSB)
4	AUTH METHOD							
5		Reserved						
7								
8		AUTHENTICATION DATA						
•••								
n								

The NEXT PAYLOAD field, CRIT bit, and IKE PAYLOAD LENGTH field are described in 7.7.3.5.1.

The CRIT bit is set to one in the Authentication payload.

The AUTH METHOD field indicates the authentication algorithm to be applied to this Authentication payload. The AUTH METHOD field contains the least significant eight bits of the ALGORITHM IDENTIFIER field in an SA_AUTH_OUT or an SA_AUTH_IN algorithm descriptor (see 7.7.3.6.6) from the Key Exchange step (see 5.16.4.6).

If the contents of the AUTH METHOD field in the parameter data for the Authentication step SECURITY PROTOCOL OUT command (see 5.16.4.7.2) do not match the least significant eight bits in the ALGORITHM IDENTIFIER field in the

SA_AUTH_OUT algorithm descriptor in the IKEv2-SCSI SA Cryptographic Algorithms payload (see 7.7.3.5.13) in the Key Exchange step, then:

a) the SECURITY PROTOCOL OUT command shall be terminated with CHECK CONDITION status, with the sense key set to ABORTED COMMAND and the additional sense code set to AUTHENTICATION FAILED; and
b) the device server shall abandon the IKEv2-SCSI CCS (see 5.16.4.10).

If the contents of the AUTH METHOD field in the parameter data for the Authentication step SECURITY PROTOCOL IN command (see 5.16.4.7.3) do not match the least significant eight bits in the ALGORITHM IDENTIFIER field in the SA_AUTH_IN algorithm descriptor in the IKEv2-SCSI SA Cryptographic Algorithms payload in the Key Exchange step, then application client should abandon the IKEv2-SCSI CCS and notify the device server that it is abandoning the IKEv2-SCSI CCS as described in 5.16.4.10.

In the Authentication step SECURITY PROTOCOL OUT command (see 5.16.4.7.2) parameter list, the AUTHENTICATION DATA field contains the result of applying the algorithm specified by the ALGORITHM IDENTIFIER field in the SA_AUTH_OUT IKEv2-SCSI cryptographic algorithm descriptor in the IKEv2-SCSI SA Cryptographic Algorithms payload (see 7.7.3.5.13) in the Key Exchange step (see 5.16.4.6) as described in table 496 (see 7.7.3.6.6) and this subclause to the following concatenation of bytes:

1) all the bytes in the Data-In Buffer returned by the Device Server Capabilities step (see 5.16.4.5) SECURITY PROTOCOL IN command;
2) all the bytes in the Data-Out Buffer sent by the Key Exchange step SECURITY PROTOCOL OUT command (see 5.16.4.6.2) in the same IKEv2-SCSI CCS for which GOOD status was returned;
3) all the bytes in the IKEv2-SCSI payload-specific data part (see 7.7.3.5.1) of the Nonce payload (see 7.7.3.5.8) that was received in the Key Exchange step SECURITY PROTOCOL IN command in the same IKEv2-SCSI CCS; and
4) all the bytes produced by applying the PRF selected by the PRF IKEv2-SCSI cryptographic algorithm descriptor (see 7.7.3.6.3) in the IKEv2-SCSI SA Cryptographic Algorithms payload (see 7.7.3.5.13) to the following inputs:
   1) the SK_pi shared key (see 5.16.4.4); and
   2) all the bytes in the IKEv2-SCSI payload-specific data part (see 7.7.3.5.1) of the Identification – Application Client payload (see 7.7.3.5.4).

When processing the Authentication step SECURITY PROTOCOL OUT command, the device server shall verify the contents of the AUTHENTICATION DATA field by applying the algorithm specified by the ALGORITHM IDENTIFIER field in the SA_AUTH_OUT IKEv2-SCSI cryptographic algorithm descriptor in the IKEv2-SCSI SA Cryptographic Algorithms payload (see 7.7.3.5.13) in the Key Exchange step (see 5.16.4.6) as described in table 496 (see 7.7.3.6.6) and this subclause to the following concatenation of bytes:

1) all the bytes in the Data-In Buffer that the device server returned to any application client in response to the last received Device Server Capabilities step SECURITY PROTOCOL IN command;
2) all the bytes in the Data-Out Buffer received in the Key Exchange step SECURITY PROTOCOL OUT command (see 5.16.4.6.2) in the same IKEv2-SCSI CCS for which GOOD status was returned;
3) all the bytes in the IKEv2-SCSI payload-specific data part (see 7.7.3.5.1) of the Nonce payload (see 7.7.3.5.8) sent in the Key Exchange step SECURITY PROTOCOL IN command in the same IKEv2-SCSI CCS; and
4) all the bytes produced by applying the PRF selected by the PRF IKEv2-SCSI cryptographic algorithm descriptor (see 7.7.3.6.3) in the IKEv2-SCSI SA Cryptographic Algorithms payload (see 7.7.3.5.13) to the following inputs:
   1) the SK_pi shared key (see 5.16.4.4); and
   2) all the bytes received in the IKEv2-SCSI payload-specific data part (see 7.7.3.5.1) of the Identification – Application Client payload (see 7.7.3.5.4) of the parameter list being processed.

If the verification of the contents of the AUTHENTICATION DATA field is not successful, then:

a) the SECURITY PROTOCOL OUT command shall be terminated with CHECK CONDITION status, with the sense key set to ABORTED COMMAND and the additional sense code set to AUTHENTICATION FAILED; and
b) the device server shall abandon the IKEv2-SCSI CCS (see 5.16.4.10).

For the Authentication step SECURITY PROTOCOL IN command (see 5.16.4.7.3) parameter list, the device server shall compute the AUTHENTICATION DATA field contents by applying the algorithm specified by the ALGORITHM IDENTIFIER field in the SA_AUTH_IN IKEv2-SCSI cryptographic algorithm descriptor in the IKEv2-SCSI SA Cryptographic Algorithms payload (see 7.7.3.5.13) in the Key Exchange step (see 5.16.4.6) as described in table 496 (see 7.7.3.6.6) and this subclause to the following concatenation of bytes:

1) all the bytes in the Data-In Buffer that the device server returned to any application client in response to the last received Device Server Capabilities step SECURITY PROTOCOL IN command;
2) all the bytes in the Data-In Buffer sent by the most recent Key Exchange step SECURITY PROTOCOL IN command (see 5.16.4.6.3) in the same IKEv2-SCSI CCS for which GOOD status was returned;
3) all the bytes in the IKEv2-SCSI payload-specific data part (see 7.7.3.5.1) of the Nonce payload (see 7.7.3.5.8) received in the Key Exchange step SECURITY PROTOCOL OUT command in the same IKEv2-SCSI CCS; and
4) all the bytes produced by applying the PRF selected by the PRF IKEv2-SCSI cryptographic algorithm descriptor (see 7.7.3.6.3) in the IKEv2-SCSI SA Cryptographic Algorithms payload (see 7.7.3.5.13) to the following inputs:
   1) the SK_pr shared key (see 5.16.4.4); and
   2) all the bytes in the IKEv2-SCSI payload-specific data part (see 7.7.3.5.1) of the Identification – Device Server payload (see 7.7.3.5.4).

After GOOD status is received for the Authentication step SECURITY PROTOCOL IN command, the application client should verify the contents of the AUTHENTICATION DATA field by applying the algorithm specified by the ALGORITHM IDENTIFIER field in the SA_AUTH_IN IKEv2-SCSI cryptographic algorithm descriptor in the IKEv2-SCSI SA Cryptographic Algorithms payload (see 7.7.3.5.13) in the Key Exchange step (see 5.16.4.6) as described in table 496 (see 7.7.3.6.6) and this subclause to the following concatenation of bytes:

1) all the bytes in the Data-In Buffer returned by the Device Server Capabilities step (see 5.16.4.5) SECURITY PROTOCOL IN command;
2) all the bytes in the Data-In Buffer returned by the most recent Key Exchange step SECURITY PROTOCOL IN command (see 5.16.4.6.3) in the same IKEv2-SCSI CCS for which GOOD status was returned;
3) all the bytes in the IKEv2-SCSI payload-specific data part (see 7.7.3.5.1) of the Nonce payload (see 7.7.3.5.8) sent in the Key Exchange step SECURITY PROTOCOL OUT command in the same IKEv2-SCSI CCS; and
4) all the bytes produced by applying the PRF selected by the PRF IKEv2-SCSI cryptographic algorithm descriptor (see 7.7.3.6.3) in the IKEv2-SCSI SA Cryptographic Algorithms payload (see 7.7.3.5.13) to the following inputs:
   1) the SK_pr shared key (see 5.16.4.4); and
   2) all the bytes in the IKEv2-SCSI payload-specific data part (see 7.7.3.5.1) of the Identification – Device Server payload (see 7.7.3.5.4) received in the SECURITY PROTOCOL IN parameter data.

If the verification of the contents of the AUTHENTICATION DATA field is not successful, then the application client should abandon the IKEv2-SCSI CCS and notify the device server that it is abandoning the IKEv2-SCSI CCS as described in 5.16.4.10.

If the AUTH METHOD field contains 01h (i.e., RSA Digital Signature) the RSA digital signature shall be encoded with the EMSA-PKCS1-v1_5 signature encoding method as specified in RFC 2437 (see RFC 4718).

7.7.3.5.8 Nonce payload

The Nonce payload (see table 476) transfers one random nonce (see 3.1.122) from the application client to the device server or from the device server to the application client.

Table 476 — Nonce payload format

Bit / Byte	7	6	5	4	3	2	1	0
0	NEXT PAYLOAD							
1	CRIT (1b)	Reserved						
2	(MSB)	IKE PAYLOAD LENGTH (n+1)						
3								(LSB)
4	NONCE DATA							
•••								
n								

The NEXT PAYLOAD field, CRIT bit, and IKE PAYLOAD LENGTH field are described in 7.7.3.5.1.

The CRIT bit is set to one in the Nonce payload.

In the Key Exchange step SECURITY PROTOCOL OUT command (see 5.16.4.6.2) the NONCE DATA field contains the application client's random nonce.

In the Key Exchange step SECURITY PROTOCOL IN command (see 5.16.4.6.3) the NONCE DATA field contains the device server's random nonce.

The requirements that RFC 4306 places on the nonce data shall apply to this standard.

7.7.3.5.9 Notify payload

This standard uses the Notify payload (see table 477) only to provide initial contact notification from the application client to the device server.

Table 477 — Notify payload format

Bit / Byte	7	6	5	4	3	2	1	0
0	NEXT PAYLOAD							
1	CRIT (1b)	Reserved						
2	(MSB)	IKE PAYLOAD LENGTH (0010h)						
3								(LSB)
4	PROTOCOL ID (01h)							
5	SAI SIZE (08h)							
6	(MSB)	NOTIFY MESSAGE TYPE (4000h)						
7								(LSB)
8	Restricted (see RFC 4306)							
•••								
11								
12	(MSB)	SAI						
•••								
15								(LSB)

The NEXT PAYLOAD field, CRIT bit, and IKE PAYLOAD LENGTH field are described in 7.7.3.5.1.

The CRIT bit is set to one in the Notify payload.

The PROTOCOL ID field contains one. If the device server receives a PROTOCOL ID field set to a value other than one, then the IKEv2-SCSI CCS state maintained for the I_T_L nexus shall be abandoned as described in 7.7.3.8.3.

The SAI SIZE field contains eight. If the device server receives a value other than eight in the SAI SIZE field, then the IKEv2-SCSI CCS state maintained for the I_T_L nexus shall be abandoned as described in 7.7.3.8.3.

The NOTIFY MESSAGE TYPE field contains 16 384 (i.e., INITIAL_CONTACT). If the device server receives a value other than 16 384 in the NOTIFY MESSAGE TYPE field, then the IKEv2-SCSI CCS state maintained for the I_T_L nexus shall be abandoned as described in 7.7.3.8.3.

The SAI field contains the device server's SAI. If the contents of the SAI field are not identical to the contents of the IKE_SA DEVICE SERVER SAI field in the IKEv2-SCSI header (see 7.7.3.4), then the IKEv2-SCSI CCS state maintained for the I_T_L nexus shall be abandoned as described in 7.7.3.8.3.

Unless an error is detected, the device server shall process the Notify payload as described in 5.16.4.7.2.

7.7.3.5.10 Delete payload

The Delete payload (see table 478) requests the deletion of an existing SA or the abandonment of an IKEv2-SCSI CCS that is in progress.

Table 478 — Delete payload format

Bit / Byte	7	6	5	4	3	2	1	0
0	NEXT PAYLOAD							
1	CRIT (1b)	Reserved						
2	(MSB)	IKE PAYLOAD LENGTH (0018h)						
3								(LSB)
4	PROTOCOL ID (01h)							
5	SAI SIZE (08h)							
6	(MSB)	NUMBER OF SAIS (0002h)						
7								(LSB)
8	Restricted (see RFC 4306)							
•••								
11								
12	(MSB)	AC_SAI						
•••								
15								(LSB)
16	Restricted (see RFC 4306)							
•••								
19								
20	(MSB)	DS_SAI						
•••								
23								(LSB)

The NEXT PAYLOAD field, CRIT bit, and IKE PAYLOAD LENGTH field are described in 7.7.3.5.1.

The CRIT bit is set to one in the Delete payload.

The PROTOCOL ID field contains one. If the device server receives a PROTOCOL ID field set to a value other than one, then the error shall be processed as described in 7.7.3.8.3.

The SAI SIZE field contains eight. If the device server receives a value other than eight in the SAI SIZE field, then the error shall be processed as described in 7.7.3.8.3.

The AC_SAI field contains the AC_SAI SA parameter value (see 3.1.133) for the SA to be deleted. If the contents of the AC_SAI field do not match the contents of the IKE_SA APPLICATION CLIENT SAI field in the IKEv2-SCSI header (see 7.7.3.4), then the error shall be processed as described in 7.7.3.8.3.

The DS_SAI field contains the DS_SAI SA parameter value (see 3.1.133) for the SA to be deleted. If the contents of the DS_SAI field do not match the contents of the IKE_SA DEVICE SERVER SAI field in the IKEv2-SCSI header (see 7.7.3.4), then the error shall be processed as described in 7.7.3.8.3.

If the device server is maintaining SA parameters for which the AC_SAI matches the contents of the AC_SAI field and the DS_SAI matches the contents of the DS_SAI field, that set of SA parameters shall be deleted.

If the device server is maintaining state for an IKEv2-SCSI CCS on the I_T_L nexus on which the command was received, the IKEv2-SCSI CCS shall be abandoned (see 5.16.4.10) if:

a) the contents of the AC_SAI field match the application client's SAI in the maintained state; and
b) the contents of the DS_SAI field match the device server's SAI in the maintained state.

7.7.3.5.11 Encrypted payload

7.7.3.5.11.1 Combined mode encryption

The following types of encryption algorithms are used in the Encrypted payload:

a) non-combined modes that use separate algorithms to encrypt/decrypt and integrity check the Encrypted payload; and
b) combined modes in which a single encryption algorithm does both encryption/decryption and integrity checking.

The ALGORITHM IDENTIFIER field in the INTEG IKEv2-SCSI cryptographic algorithm descriptor (see 7.7.3.6.4) in the IKEv2-SCSI SA Cryptographic Algorithms payload (see 7.7.3.5.13) in the Key Exchange step (see 5.16.4.6) indicates the type of encryption algorithm to be applied to the Encrypted payload for IKEv2-SCSI functions as follows:

a) if the ALGORITHM IDENTIFIER field is not set to AUTH_COMBINED, a non-combined mode encryption algorithm is being used; or
b) if the ALGORITHM IDENTIFIER field is set to AUTH_COMBINED, a combined mode encryption algorithm is being used.

If the Encrypted payload is in parameter data that is not associated with an active IKEv2-SCSI CCS (see 3.1.71), then the integrity checking algorithm identifier that selects between combined and non-combined mode encryption is found in the MGMT_DATA SA parameter (see 3.1.133 and 5.16.4.9).

7.7.3.5.11.2 Encrypted payload introduction

The Encrypted payload transfers (see table 479) one or more other IKEv2-SCSI payloads that are encrypted and integrity checked from the application client to the device server and vice versa.

If IKEv2-SCSI parameter data contains the Encrypted payload, then the Encrypted payload is the first payload in the parameter data (i.e., the NEXT PAYLOAD field in the IKEv2-SCSI header (see 7.7.3.5.1) contains 2Eh). Since the NEXT PAYLOAD field in an Encrypted payload identifies the first payload in the CIPHERTEXT field, there is no way to identify a payload following the Encrypted payload, and none are allowed.

Table 479 — Encrypted payload format

Bit / Byte	7	6	5	4	3	2	1	0
0	NEXT PAYLOAD							
1	CRIT (1b)	Reserved						
2	(MSB)	IKE PAYLOAD LENGTH (n+1)						
3								(LSB)
4	INITIALIZATION VECTOR							
•••								
i-1								
i	CIPHERTEXT							
•••								
k-1								
k	INTEGRITY CHECK VALUE							
•••								
n								

The NEXT PAYLOAD field identifies the first IKEv2-SCSI payload in the CIPHERTEXT field using the coded values shown in table 465 (see 7.7.3.5.1).

The CRIT bit and IKE PAYLOAD LENGTH field are described in 7.7.3.5.1.

The CRIT bit is set to one in the Encrypted payload.

The IKE PAYLOAD LENGTH field contains the number of bytes that follow in the Encrypted payload. The number of bytes in the CIPHERTEXT field is equal to the number of bytes in the plaintext (see table 480).

The INITIALIZATION VECTOR field contains the initialization vector encryption algorithm input value. The size of the initialization vector is defined by the encryption algorithm as shown in table 488 (see 7.7.3.6.2).

The CIPHERTEXT field contains an output of the encryption algorithm specified by the ALGORITHM IDENTIFIER field in the ENCR IKEv2-SCSI cryptographic algorithm descriptor (see 7.7.3.6.2) in the IKEv2-SCSI SA Cryptographic Algorithms payload (see 7.7.3.5.13) in the Key Exchange step (see 5.16.4.6) using the inputs:

a) IKEv2-SCSI AAD is an encryption algorithm input as follows:
   A) if a non-combined mode encryption algorithm is being used (see 7.7.3.5.11.1), then no AAD input is needed or provided; or
   B) if a combined mode encryption algorithm is being used (see 7.7.3.5.11.1), then the AAD described in 7.7.3.5.11.3 is an input;

b) the contents of the INITIALIZATION VECTOR field (see table 488);
c) the plaintext data shown in table 480; and
d) the shared key value, key length, and salt value (see table 488 in 7.7.3.6.2) if any, for one of the following shared keys:
   A) if the Encrypted payload appears in the parameter data for an Authentication step SECURITY PROTOCOL OUT command (see 5.16.4.7.2) or the parameter data for a Delete operation SECURITY PROTOCOL OUT command (see 5.16.4.11) that affects an active IKEv2-SCSI CCS (see 3.1.71), then the SK_ei shared key (see 5.16.4.4) for the IKEv2-SCSI CCS;
   B) if the Encrypted payload appears in the parameter data for an Authentication step SECURITY PROTOCOL IN command (see 5.16.4.7.3), then the SK_er shared key (see 5.16.4.4) for the IKEv2-SCSI CCS.; or
   C) if the Encrypted payload appears in the parameter data for a Delete operation SECURITY PROTOCOL OUT command (see 5.16.4.11) that does not affect an active IKEv2-SCSI CCS, then the SK_ei shared key (see 5.16.4.4) from the MGMT_DATA SA parameter (see 3.1.133 and 5.16.4.9).

NOTE 68 - Salt values (see table 488 in 7.7.3.6.2) are used only by combined mode encryption algorithms (see 7.7.3.5.11.1).

If the Encrypted payload appears in the parameter data for a Delete operation SECURITY PROTOCOL OUT command that does not affect an active IKEv2-SCSI CCS, then the encryption algorithm identifier stored in the MGMT_DATA SA parameter indicates the encryption algorithm to use.

The INTEGRITY CHECK VALUE field contains the integrity check value that is computed as described in 7.7.3.5.11.1 (e.g., if the integrity algorithm is AUTH_COMBINED, then the integrity check value is an output of the encryption algorithm). The size of the integrity check value is defined by the integrity algorithm or encryption algorithm, depending on which algorithm computes the value.

If the integrity algorithm specified by the ALGORITHM IDENTIFIER field in the INTEG IKEv2-SCSI cryptographic algorithm descriptor in the IKEv2-SCSI SA Cryptographic Algorithms payload in the Key Exchange step is not AUTH_COMBINED, then the contents of the INTEGRITY CHECK VALUE field are computed by processing the integrity check algorithm specified by the ALGORITHM IDENTIFIER field in the INTEG IKEv2-SCSI cryptographic algorithm descriptor using the following inputs:

a) a byte string composed of:
   1) the AAD described in 7.7.3.5.11.3;
   2) the contents of the INITIALIZATION VECTOR field (see table 479); and
   3) the ciphertext that is the result of encrypting the plaintext data;
   and
b) the shared key value from one of the following shared keys:
   A) if the Encrypted payload appears in the parameter data for an Authentication step SECURITY PROTOCOL OUT command (see 5.16.4.7.2) or the parameter data for a Delete operation SECURITY PROTOCOL OUT command (see 5.16.4.11) that affects an active IKEv2-SCSI CCS (see 3.1.71), then the SK_ai shared key (see 5.16.4.4) for the IKEv2-SCSI CCS;
   B) if the Encrypted payload appears in the parameter data for an Authentication step SECURITY PROTOCOL IN command (see 5.16.4.7.3), then the SK_ar shared key (see 5.16.4.4) for the IKEv2-SCSI CCS.; or
   C) if the Encrypted payload appears in the parameter data for a Delete operation SECURITY PROTOCOL OUT command (see 5.16.4.11) that does not affect an active IKEv2-SCSI CCS, then the SK_ai shared key (see 5.16.4.4) from the MGMT_DATA SA parameter (see 3.1.133 and 5.16.4.9).

If the Encrypted payload appears in the parameter data for a Delete operation SECURITY PROTOCOL OUT command that does not affect an active IKEv2-SCSI CCS, then the integrity checking algorithm identifier stored in the MGMT_DATA SA parameter indicates the integrity checking algorithm to use.

When computing the encrypted CIPHERTEXT field contents for an Encrypted payload, the plaintext shown in table 480 is used.

Table 480 — Plaintext format for Encrypted payload CIPHERTEXT field

Bit / Byte	7	6	5	4	3	2	1	0
IKEv2-SCSI payloads								
0	IKEv2-SCSI payload [first] (see 7.7.3.5)							
•••								
	⋮							
	IKEv2-SCSI payload [last] (see 7.7.3.5)							
•••								
n								
p	PADDING BYTES (if any)							
•••								
k-1								
k	PAD LENGTH (k-p)							

Each IKEv2-SCSI payload (see 7.7.3.5) contains specific data related to the operation being performed. A specific combination of IKEv2-SCSI payloads is needed for each operation (e.g., Authentication) as summarized in 5.16.4.2.

The PADDING BYTES field contains zero to 255 bytes. The number of padding bytes is:

a) defined by the encryption algorithm; or
b) any number of bytes that causes the length of all plaintext bytes (i.e., l+2) to be a whole multiple of the alignment (see table 488 in 7.7.3.6.2) for the encryption algorithm being used.

The contents of the padding bytes are:

a) defined by the encryption algorithm; or
b) if the encryption algorithm does not define the padding bytes contents, a series of one byte binary values starting at one and incrementing by one in each successive byte (i.e., 01h in the first padding byte, 02h in the second padding byte, etc.).

If the encryption algorithm does not place requirements on the contents of the padding bytes (i.e., option b) is in effect), then after decryption the contents of the padding bytes shall be verified to match the series of one byte binary values described in this subclause. If this verification is not successful in a device server, the error shall be processed as described in 7.7.3.8.2. If this verification is not successful in an application client, the decrypted data should be ignored.

The PAD LENGTH field contains the number of bytes in the PADDING BYTES field.

7.7.3.5.11.3 IKEv2-SCSI AAD

The AAD defined by this standard for IKEv2-SCSI use is as follows:

1) all the bytes in the IKEv2-SCSI Header (see 7.7.3.4); and

2) all the bytes in the IKEv2-SCSI Payload Header (see 7.7.3.5.1) of the Encrypted payload (see 7.7.3.5.11).

7.7.3.5.11.4 Processing a received Encrypted payload

Before performing any checks of data contained in the CIPHERTEXT field, the contents of the INTEGRITY CHECK VALUE field and CIPHERTEXT field shall be integrity checked and decrypted based on the contents of the IKE_SA APPLICATION CLIENT SAI field and the IKE_SA DEVICE SERVER SAI field in the IKEv2-SCSI header (see 7.7.3.4) as described in this subclause.

Computation of the comparison integrity check value and decryption of an Encrypted payload is performed as follows:

1) if a non-combined mode encryption algorithm is being used (see 7.7.3.5.11.1), then the comparison integrity check value is computed by performing the integrity check algorithm specified by the ALGORITHM IDENTIFIER field in the INTEG IKEv2-SCSI cryptographic algorithm descriptor using the following inputs:
   A) a byte string composed of:
      1) the AAD described in 7.7.3.5.11.3;
      2) the contents of the INITIALIZATION VECTOR field (see table 479) in the Encrypted payload; and
      3) the contents of the ciphertext field (see table 479) in the Encrypted payload;
      and
   B) the shared key value from one of the following shared keys:
      a) if the Encrypted payload appears in the parameter data for an Authentication step SECURITY PROTOCOL OUT command (see 5.16.4.7.2) or the parameter data for a Delete operation SECURITY PROTOCOL OUT command (see 5.16.4.11) that affects an active IKEv2-SCSI CCS (see 3.1.71), then the SK_ai shared key (see 5.16.4.4) for the IKEv2-SCSI CCS;
      b) if the Encrypted payload appears in the parameter data for an Authentication step SECURITY PROTOCOL IN command (see 5.16.4.7.3), then the SK_ar shared key (see 5.16.4.4) for the IKEv2-SCSI CCS; or
      c) if the Encrypted payload appears in the parameter data for a Delete operation SECURITY PROTOCOL OUT command (see 5.16.4.11) that does not affect an active IKEv2-SCSI CCS, then the SK_ai shared key (see 5.16.4.4) from the MGMT_DATA SA parameter (see 3.1.133 and 5.16.4.9);

   and
2) decrypt the CIPHERTEXT field to produce plaintext data – and if a combined mode encryption algorithm is being used (see 7.7.3.5.11.1), produce the comparison integrity check value – using the encryption algorithm specified by the ALGORITHM IDENTIFIER field in the ENCR IKEv2-SCSI cryptographic algorithm descriptor (see 7.7.3.6.2) in the IKEv2-SCSI SA Cryptographic Algorithms payload (see 7.7.3.5.13) in the Key Exchange step (see 5.16.4.6) using the inputs:
   A) IKEv2-SCSI AAD is an input to the encryption algorithm as follows:
      a) if a non-combined mode encryption algorithm is being used (see 7.7.3.5.11.1), then no AAD input is needed or provided; or
      b) if a combined mode encryption algorithm is being used (see 7.7.3.5.11.1), then the AAD described in 7.7.3.5.11.3 is an input;
   B) the contents of the INITIALIZATION VECTOR field (see table 479) in the Encrypted payload;
   C) the contents of the ciphertext field (see table 479) in the Encrypted payload; and
   D) the shared key value, key length, and salt value (see table 488 in 7.7.3.6.2) if any, for one of the following shared keys:
      a) if the Encrypted payload appears in the parameter data for an Authentication step SECURITY PROTOCOL OUT command (see 5.16.4.7.2), or the parameter data for a Delete operation SECURITY PROTOCOL OUT command (see 5.16.4.11) that affects an active IKEv2-SCSI CCS (see 3.1.71), then the SK_ei shared key (see 5.16.4.4) for the IKEv2-SCSI CCS;
      b) if the Encrypted payload appears in the parameter data for an Authentication step SECURITY PROTOCOL IN command (see 5.16.4.7.3), then the SK_er shared key (see 5.16.4.4) for the IKEv2-SCSI CCS; or

c) if the Encrypted payload appears in the parameter data for a Delete operation SECURITY PROTOCOL OUT command (see 5.16.4.11) that does not affect an active IKEv2-SCSI CCS, then the SK_ei shared key (see 5.16.4.4) from the MGMT_DATA SA parameter (see 3.1.133 and 5.16.4.9).

If the Encrypted payload appears in the parameter data for a Delete operation SECURITY PROTOCOL OUT command that does not affect an active IKEv2-SCSI CCS, then the integrity checking algorithm identifier value and encryption algorithm identifier value that are stored in the MGMT_DATA SA parameter indicate the integrity checking algorithm to use.

If the comparison integrity check value differs from the value in the INTEGRITY CHECK VALUE field of the Encrypted payload:

a) the application client should abandon the IKEv2-SCSI CCS and notify the device server that it is abandoning the IKEv2-SCSI CCS as described in 5.16.4.10; and
b) the device server shall respond to the mismatch as follows:
   A) if the IKEv2-SCSI header (see 7.7.3.4) specifies an attempt to provide authentication data for or the deletion of an IKE-v2-SCSI CCS on the I_T_L nexus on which the command was received, then:
      a) the SECURITY PROTOCOL OUT command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to SA CREATION PARAMETER VALUE REJECTED; and
      b) the device server shall continue the IKEv2-SCSI CCS by preparing to receive another Authentication step SECURITY PROTOCOL OUT command;

      or
   B) if the IKEv2-SCSI header (see 7.7.3.4) specifies the deletion of an active SA, then the SECURITY PROTOCOL OUT command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

7.7.3.5.12 IKEv2-SCSI SA Creation Capabilities payload

The IKEv2-SCSI SA Creation Capabilities payload (see table 481) lists all the security algorithms that the device server allows to be used in an IKEv2-SCSI CCS. Events that are outside the scope of this standard may change the contents of the IKEv2-SCSI SA Creation Capabilities payload at any time.

Table 481 — IKEv2-SCSI SA Creation Capabilities payload format

Bit / Byte	7	6	5	4	3	2	1	0
0	NEXT PAYLOAD (00h)							
1	CRIT (1b)	Reserved						
2	(MSB)	IKE PAYLOAD LENGTH (n+1)						
3								(LSB)
4		Reserved						
5								
6								
7	NUMBER OF ALGORITHM DESCRIPTORS							
IKEv2-SCSI cryptographic algorithm descriptors								
8	IKEv2-SCSI cryptographic algorithm descriptor [first] (see 7.7.3.6)							
•••								
	⋮							
	IKEv2-SCSI cryptographic algorithm descriptor [last] (see 7.7.3.6)							
•••								
n								

The NEXT PAYLOAD field, CRIT bit, and IKE PAYLOAD LENGTH field are described in 7.7.3.5.1.

The NEXT PAYLOAD field is set to zero (i.e., No Next Payload) in the IKEv2-SCSI SA Creation Capabilities payload.

The CRIT bit is set to one in the IKEv2-SCSI SA Creation Capabilities payload.

The NUMBER OF ALGORITHM DESCRIPTORS field contains the number of IKEv2-SCSI cryptographic algorithm descriptors that follow in the IKEv2-SCSI SA Creation Capabilities payload.

Each IKEv2-SCSI cryptographic algorithm descriptor (see 7.7.3.6) describes one combination of security algorithm and algorithm attributes that the device server allows to be used in an IKEv2-SCSI CCS. If more than one set of algorithm attributes (e.g., key length) is allowed for any allowed security algorithm, a different SCSI cryptographic algorithms descriptor shall be included for each set of algorithm attributes.

The SCSI cryptographic algorithms descriptors shall be ordered by:

1) increasing algorithm type;
2) increasing algorithm identifier within the same algorithm type; and
3) increasing key length, if any, within the same algorithm identifier.

The algorithms allowed may be a subset of the algorithms supported by the device server.

The method for changing which of the device server supported algorithms are allowed is outside the scope of this standard, but changes in allowed algorithms do not take effect until the new list is returned to any application client in an IKEv2-SCSI SA Creation Capabilities payload.

7.7.3.5.13 IKEv2-SCSI SA Cryptographic Algorithms payload

The IKEv2-SCSI SA Cryptographic Algorithms payload (see table 482) lists the security algorithms that are being used in the creation and management (e.g., deletion) of an SA using an IKEv2-SCSI CCS.

Table 482 — IKEv2-SCSI SA Cryptographic Algorithms payload format

Bit / Byte	7	6	5	4	3	2	1	0
0	NEXT PAYLOAD							
1	CRIT (1b)	Reserved						
2	(MSB)	IKE PAYLOAD LENGTH (n+1)						
3								(LSB)
4		Reserved						
•••								
13								
14	(MSB)	USAGE DATA LENGTH (0000h)						
15								(LSB)
16		Reserved						
•••								
19								
20	NUMBER OF ALGORITHM DESCRIPTORS							
IKEv2-SCSI cryptographic algorithm descriptors								
21		IKEv2-SCSI cryptographic algorithm descriptor [first] (see 7.7.3.6)						
•••								
	⋮							
		IKEv2-SCSI cryptographic algorithm descriptor [last] (see 7.7.3.6)						
•••								
n								

The NEXT PAYLOAD field, CRIT bit, and IKE PAYLOAD LENGTH field are described in 7.7.3.5.1.

The CRIT bit is set to one in the IKEv2-SCSI SA Cryptographic Algorithms payload.

The USAGE DATA LENGTH field is set to zero in the IKEv2-SCSI SA Cryptographic Algorithms payload.

The NUMBER OF ALGORITHM DESCRIPTORS field contains the number of IKEv2-SCSI cryptographic algorithm descriptors that follow in the IKEv2-SCSI SA Cryptographic Algorithms payload. If a device server receives a NUMBER OF ALGORITHM DESCRIPTORS field that does not contain six, then the error shall be processed as described in 7.7.3.8.3.

Each IKEv2-SCSI cryptographic algorithm descriptor (see 7.7.3.6) describes one combination of security algorithm and algorithm attributes to be used during the IKEv2-SCSI CCS.

The IKEv2-SCSI cryptographic algorithm descriptors are ordered as follows:

1) one ENCR IKEv2-SCSI cryptographic algorithm descriptor (see 7.7.3.6.2);
2) one PRF IKEv2-SCSI cryptographic algorithm descriptor (see 7.7.3.6.3);
3) one INTEG IKEv2-SCSI cryptographic algorithm descriptor (see 7.7.3.6.4);
4) one D-H IKEv2-SCSI cryptographic algorithm descriptor (see 7.7.3.6.5);
5) one SA_AUTH_OUT IKEv2-SCSI cryptographic algorithm descriptor (see 7.7.3.6.6).
6) one SA_AUTH_IN IKEv2-SCSI cryptographic algorithm descriptor (see 7.7.3.6.6).

If a device server receives an IKEv2-SCSI SA Cryptographic Algorithms payload that does not contain the IKEv2-SCSI cryptographic algorithm descriptors described in this subclause in the order described in this subclause, then the error shall be processed as described in 7.7.3.8.3.

If the device server receives an IKEv2-SCSI SA Cryptographic Algorithms payload that contains an ENCR IKEv2-SCSI cryptographic algorithm descriptor with the ALGORITHM IDENTIFIER field set to ENCR_NULL, then the error shall be processed as described in 7.7.3.8.3.

In the Key Exchange step SECURITY PROTOCOL IN parameter data (see 5.16.4.6.3), the device server returns the IKEv2-SCSI SA Cryptographic Algorithms payload received during the Key Exchange step SECURITY PROTOCOL OUT command (see 5.16.4.6.2) to confirm acceptance of the algorithms.

7.7.3.5.14 IKEv2-SCSI SAUT Cryptographic Algorithms payload

The IKEv2-SCSI SAUT Cryptographic Algorithms payload (see table 483) lists the usage type of and security algorithms to be used by the SA that is created as a result of an IKEv2-SCSI CCS.

Table 483 — IKEv2-SCSI SAUT Cryptographic Algorithms payload format

Bit / Byte	7	6	5	4	3	2	1	0
0	NEXT PAYLOAD							
1	CRIT (1b)	Reserved						
2	(MSB)	IKE PAYLOAD LENGTH (n+1)						
3								(LSB)
4	Reserved							
11								
12	(MSB)	SA TYPE						
13								(LSB)
14	(MSB)	USAGE DATA LENGTH (j)						
15								(LSB)
16	USAGE DATA							
•••								
16+j-1								
16+j	Reserved							
16+j+1								
16+j+2								
16+j+3	NUMBER OF ALGORITHM DESCRIPTORS							
IKEv2-SCSI cryptographic algorithm descriptors								
16+j+4	IKEv2-SCSI cryptographic algorithm descriptor [first] (see 7.7.3.6)							
•••								
	⋮							
	IKEv2-SCSI cryptographic algorithm descriptor [last] (see 7.7.3.6)							
•••								
n								

The NEXT PAYLOAD field, CRIT bit, and IKE PAYLOAD LENGTH field are described in 7.7.3.5.1.

The CRIT bit is set to one in the IKEv2-SCSI SAUT Cryptographic Algorithms payload.

The SA TYPE field specifies the usage type for the SA and is selected from among those listed in table 62 (see 5.16.2.2). An error shall be processed as described in 7.7.3.8.3 if any of the following conditions occur:

a) the device server receives an SA usage type whose use the device server does not allow;
b) an SA usage type between 8000h and BFFFh inclusive is received in a Key Exchange step SECURITY PROTOCOL OUT command (see 5.16.4.6.2); or

c) an SA usage type between A000h and CFFFh inclusive is received for which the device server is unable to verify the applicable usage type constraints.

The method for changing which of the device server supported SA usage types are allowed is outside the scope of this standard.

The USAGE DATA LENGTH field specifies number of bytes of usage data that follow.

The size and format of the usage data depends on the SA type (see table 62 in 5.16.2.2). If the device server receives a USAGE DATA LENGTH field that contains a value that is inconsistent with the SA type, then the error shall be processed as described in 7.7.3.8.3.

The USAGE DATA field contains information to be stored in the USAGE_DATA SA parameter (see 3.1.133) if the SA is generated (see 5.16.4.9).

The NUMBER OF ALGORITHM DESCRIPTORS field contains the number of IKEv2-SCSI cryptographic algorithm descriptors that follow in the IKEv2-SCSI SAUT Cryptographic Algorithms payload. If a device server receives a NUMBER OF ALGORITHM DESCRIPTORS field that contains a value other than two, then the error shall be processed as described in 7.7.3.8.3.

Each IKEv2-SCSI cryptographic algorithm descriptor (see 7.7.3.6) describes one combination of security algorithm and algorithm attributes to be used by the SA created as a result of the IKEv2-SCSI CCS. The IKEv2-SCSI cryptographic algorithm descriptors are ordered as follows:

1) one ENCR IKEv2-SCSI cryptographic algorithm descriptor (see 7.7.3.6.2); and
2) one INTEG IKEv2-SCSI cryptographic algorithm descriptor (see 7.7.3.6.4).

If a device server receives an IKEv2-SCSI SAUT Cryptographic Algorithms payload that does not contain the IKEv2-SCSI cryptographic algorithm descriptors described in this subclause in the order described in this subclause, then the error shall be processed as described in 7.7.3.8.3.

7.7.3.5.15 IKEv2-SCSI Timeout Values payload

The IKEv2-SCSI Timeout Values payload (see table 484) specifies the timeout intervals associated with an IKEv2-SCSI CCS

Table 484 — IKEv2-SCSI Timeout Values payload format

Bit / Byte	7	6	5	4	3	2	1	0
0	NEXT PAYLOAD							
1	CRIT (1b)	Reserved						
2	(MSB)	IKE PAYLOAD LENGTH (0010h)						
3								(LSB)
4	Reserved							
5								
6								
7	NUMBER OF TIMEOUT VALUES (02h)							
8	(MSB)	IKEV2-SCSI PROTOCOL TIMEOUT						
•••								
11								(LSB)
12	(MSB)	IKEV2-SCSI SA INACTIVITY TIMEOUT						
•••								
15								(LSB)

The NEXT PAYLOAD field, CRIT bit, and IKE PAYLOAD LENGTH field are described in 7.7.3.5.1.

The CRIT bit is set to one in the IKEv2-SCSI Timeout Values payload.

The NUMBER OF TIMEOUT VALUES field specifies the number of four-byte timeout values that follow. If the number of timeout values is less than two, then the IKEv2-SCSI CCS state maintained for the I_T_L nexus shall be abandoned as described in 7.7.3.8.3.

The IKEV2-SCSI PROTOCOL TIMEOUT field specifies the number of seconds that the device server shall wait for the next command in the IKEv2-SCSI CCS. If the timeout expires before the device server receives an IKEv2-SCSI CCS command, the device server shall abandon the IKEv2-SCSI CCS as described in 5.16.4.10.

The IKEV2-SCSI SA INACTIVITY TIMEOUT field specifies the number of seconds that the device server shall wait for the next command that uses an SA. This value is copied to the TIMEOUT SA parameter when the SA is generated (see 5.16.4.9).

The device server shall replace any timeout value that is set to zero with a value of ten (i.e., ten seconds).

The maximum value for the protocol timeout should be long enough to allow the application client to continue the IKEv2-SCSI CCS, but short enough that if an incomplete IKEv2-SCSI CCS is abandoned, the device server discards the state for that IKEv2-SCSI CCS and becomes available to for another IKEv2-SCSI CCS without excessive delay.

7.7.3.6 IKEv2-SCSI cryptographic algorithm descriptors

7.7.3.6.1 Overview

Each IKEv2-SCSI cryptographic algorithm descriptor (see table 485) specifies one algorithm used for encryption, integrity checking, key generation, or authentication.

Table 485 — IKEv2-SCSI cryptographic algorithm descriptor format

Bit / Byte	7	6	5	4	3	2	1	0
0	ALGORITHM TYPE							
1	Reserved							
2	(MSB)	IKE DESCRIPTOR LENGTH (000Ch)						
3								(LSB)
4	(MSB)	ALGORITHM IDENTIFIER						
•••								
7								(LSB)
8		ALGORITHM ATTRIBUTES						
•••								
11								

The ALGORITHM TYPE field (see table 486) specifies the type of cryptographic algorithm to which the IKEv2-SCSI cryptographic algorithm descriptor applies.

Table 486 — ALGORITHM TYPE field

Code	Name	Description	Reference
01h	ENCR	Encryption algorithm	7.7.3.6.2
02h	PRF	Pseudo-random function	7.7.3.6.3
03h	INTEG	Integrity algorithm	7.7.3.6.4
04h	D-H	Diffie-Hellman group	7.7.3.6.5
05h to F0h	Restricted		RFC 4306
F9h	SA_AUTH_OUT	IKEv2-SCSI authentication algorithm for SECURITY PROTOCOL OUT data	7.7.3.6.6
FAh	SA_AUTH_IN	IKEv2-SCSI authentication algorithm for SECURITY PROTOCOL IN data	7.7.3.6.6
All others	Reserved		

The IKE DESCRIPTOR LENGTH field contains 12 (i.e., the total number of bytes in the IKEv2-SCSI SA Cryptographic Algorithms descriptor including the ALGORITHM TYPE field and reserved byte).

NOTE 69 - The contents of the IKE DESCRIPTOR LENGTH field differ from those found in most SCSI length fields, however, they are consistent with the IKEv2 usage (see RFC 4306).

The contents of the ALGORITHM IDENTIFIER field and ALGORITHM ATTRIBUTES field depend on the contents of the ALGORITHM TYPE field (see table 486). The ALGORITHM ATTRIBUTES field is reserved in some IKEv2-SCSI SA Cryptographic Algorithms descriptor formats.

7.7.3.6.2 Encryption algorithm (ENCR) IKEv2-SCSI cryptographic algorithm descriptor

When the ALGORITHM TYPE field is set to ENCR (i.e., 01h) in an IKEv2-SCSI cryptographic algorithm descriptor (see table 487), the descriptor specifies an encryption algorithm to be applied during the IKEv2-SCSI Authentication step (see 5.16.4.7), SA deletion operation (see 5.16.4.11), and when the SA created by the IKEv2-SCSI is applied to user data.

Table 487 — ENCR IKEv2-SCSI cryptographic algorithm descriptor format

Bit / Byte	7	6	5	4	3	2	1	0
0	ALGORITHM TYPE (01h)							
1	Reserved							
2	(MSB)	IKE DESCRIPTOR LENGTH (000Ch)						
3								(LSB)
4	(MSB)	ALGORITHM IDENTIFIER						
•••								
7								(LSB)
8	Reserved							
9								
10	(MSB)	KEY LENGTH						
11								(LSB)

The ALGORITHM TYPE field and IKE DESCRIPTOR LENGTH field are described in 7.7.3.6.1.

The ALGORITHM IDENTIFIER field (see table 488) specifies the encryption algorithm to which the ENCR IKEv2-SCSI cryptographic algorithm descriptor applies.

Table 488 — ENCR ALGORITHM IDENTIFIER field

Code	Description	Salt [a] length (bytes)	IV [b] length (bytes)	Align-ment [c] (bytes)	Key length (bytes)	Support	Reference
8001 000Bh	ENCR_NULL [d]	n/a	0	4	0	Mandatory	
8001 000Ch	AES-CBC [d]	n/a	16	16	16	Optional	RFC 3602
					24	Prohibited	
					32	Optional	
8001 0010h	AES-CCM with a 16 byte MAC [e]	3	8	4	16	Optional	RFC 4309 and RFC 5282
					24	Prohibited	
					32	Optional	
8001 0014h	AES-GCM with a 16 byte MAC [e]	4	8	4	16	Optional	RFC 4106 and RFC 5282
					24	Prohibited	
					32	Optional	
8001 0400h to 8001 FFFFh	Vendor Specific						
0000 0000h to 0000 FFFFh	Restricted						IANA
All other values	Reserved						

[a] See RFC 4106 and RFC 4309.
[b] Initialization Vector.
[c] The alignment required in the plaintext prior to encryption.
[d] If the INTEG cryptographic algorithm descriptor (see 7.7.3.6.4) in the same IKEv2-SCSI SA Cryptographic Algorithms payload or the same IKEv2-SCSI SAUT Cryptographic Algorithms payload as this ENCR cryptographic algorithm descriptor has the ALGORITHM IDENTIFIER field set to AUTH_COMBINED, then the error shall be processed as described in 7.7.3.8.3.
[e] If the INTEG cryptographic algorithm descriptor (see 7.7.3.6.4) in the same IKEv2-SCSI SA Cryptographic Algorithms payload or the same IKEv2-SCSI SAUT Cryptographic Algorithms payload as this ENCR cryptographic algorithm descriptor does not have the ALGORITHM IDENTIFIER field set to AUTH_COMBINED, then the error shall be processed as described in 7.7.3.8.3.

ENCR_NULL indicates that encryption is not to be applied when the SA created by the IKEv2-SCSI is applied to user data.

The IKEv2-SCSI CCS state maintained for the I_T_L nexus shall be abandoned as described in 7.7.3.8.3 if the parameter list contains an IKEv2-SCSI SA Cryptographic Algorithms payload (see 7.7.3.5.13) that contains:

a) an ENCR IKEv2-SCSI cryptographic algorithm descriptor with the ALGORITHM IDENTIFIER field set to ENCR_NULL;
b) the following combination of IKEv2-SCSI cryptographic algorithm descriptors (see 7.7.3.6.4):
   A) an INTEG IKEv2-SCSI cryptographic algorithm descriptor with the ALGORITHM IDENTIFIER field set to a value other than AUTH_COMBINED; and

B) an ENCR IKEv2-SCSI cryptographic algorithm descriptor with the ALGORITHM IDENTIFIER field set to a value that table 488 describes as requiring AUTH_COMBINED as the integrity check algorithm;

or

c) the following combination of IKEv2-SCSI cryptographic algorithm descriptors:
   A) an INTEG IKEv2-SCSI cryptographic algorithm descriptor with the ALGORITHM IDENTIFIER field set to AUTH_COMBINED; and
   B) an ENCR IKEv2-SCSI cryptographic algorithm descriptor with the ALGORITHM IDENTIFIER field set to a value that table 488 does not describe as requiring AUTH_COMBINED as the integrity check algorithm.

The KEY LENGTH field specifies the number of bytes in the shared key (see 3.1.162) for the encryption algorithm to which the ENCR IKEv2-SCSI cryptographic algorithm descriptor applies.

The IKEv2-SCSI CCS state maintained for the I_T_L nexus shall be abandoned as described in 7.7.3.8.3 if the parameter list contains:

a) no ENCR IKEv2-SCSI cryptographic algorithm descriptors;
b) more than one ENCR IKEv2-SCSI cryptographic algorithm descriptor;
c) an ENCR IKEv2-SCSI cryptographic algorithm descriptor that does not appear in the SA Creations Capabilities payload (see 7.7.3.5.12) last returned by the device server to any application client (i.e., an ENCR IKEv2-SCSI cryptographic algorithm descriptor for a combination of algorithm identifier and key length that the device server has not reported as one of its SA creation capabilities); or
d) an ENCR IKEv2-SCSI cryptographic algorithm descriptor that contains:
   A) an algorithm identifier that is not shown in table 488; or
   B) a key length that:
      a) does not match one of the values shown in table 488; or
      b) is not supported by the device server.

7.7.3.6.3 Pseudo-random function (PRF) IKEv2-SCSI cryptographic algorithm descriptor

When the ALGORITHM TYPE field is set to PRF (i.e., 02h) in an IKEv2-SCSI cryptographic algorithm descriptor (see table 489), the descriptor specifies the pseudo-random function and KDF to be used during the Key Exchange step completion (see 5.16.4.6.4).

Table 489 — PRF IKEv2-SCSI cryptographic algorithm descriptor format

Bit / Byte	7	6	5	4	3	2	1	0
0	ALGORITHM TYPE (02h)							
1	Reserved							
2	(MSB)	IKE DESCRIPTOR LENGTH (000Ch)						
3								(LSB)
4	(MSB)	ALGORITHM IDENTIFIER						
•••								
7								(LSB)
8	Reserved							
•••								
11								

The ALGORITHM TYPE field and IKE DESCRIPTOR LENGTH field are described in 7.7.3.6.1.

The ALGORITHM IDENTIFIER field (see table 490) specifies PRF and KDF to which the PRF IKEv2-SCSI cryptographic algorithm descriptor applies.

Table 490 — PRF ALGORITHM IDENTIFIER field

Code	Description	Support	Output length (bytes)	Reference	
				PRF [a]	KDF [b]
8002 0002h	IKEv2-use based on SHA-1	Optional	20	RFC 2104	5.16.3.3
8002 0004h	IKEv2-use based on AES-128 in CBC mode	Optional	16	RFC 4434	5.16.3.4
8002 0005h	IKEv2-use based on SHA-256	Optional	32	RFC 4868	5.16.3.3
8002 0007h	IKEv2-use based on SHA-512	Optional	64	RFC 4868	5.16.3.3
8002 0400h to 8002 FFFFh	Vendor Specific				
0000 0000h to 0000 FFFFh	Restricted			IANA	
All others	Reserved				

[a] PRFs are equivalent to the prf() functions defined in RFC 4306.
[b] KDFs are equivalent to the prf+() functions defined in RFC 4306.

The IKEv2-SCSI CCS state maintained for the I_T_L nexus shall be abandoned as described in 7.7.3.8.3 if the parameter list contains:

a) no PRF IKEv2-SCSI cryptographic algorithm descriptors;
b) more than one PRF IKEv2-SCSI cryptographic algorithm descriptor;
c) a PRF IKEv2-SCSI cryptographic algorithm descriptor that contains an algorithm identifier that does not appear in the SA Creations Capabilities payload (see 7.7.3.5.12) last returned by the device server to any application client; or
d) an PRF IKEv2-SCSI cryptographic algorithm descriptor that contains an algorithm identifier that is not shown in table 490.

7.7.3.6.4 Integrity algorithm (INTEG) IKEv2-SCSI cryptographic algorithm descriptor

When the ALGORITHM TYPE field is set to INTEG (i.e., 03h) in an IKEv2-SCSI cryptographic algorithm descriptor (see table 491), the descriptor specifies an integrity checking (i.e., data authentication) algorithm to be applied during the IKEv2-SCSI Authentication step (see 5.16.4.7) and when the SA created by the IKEv2-SCSI is applied to user data.

Table 491 — INTEG IKEv2-SCSI cryptographic algorithm descriptor format

Bit / Byte	7	6	5	4	3	2	1	0
0	ALGORITHM TYPE (03h)							
1	Reserved							
2	(MSB)	IKE DESCRIPTOR LENGTH (000Ch)						
3								(LSB)
4	(MSB)	ALGORITHM IDENTIFIER						
•••								
7								(LSB)
8	Reserved							
•••								
11								

The ALGORITHM TYPE field and IKE DESCRIPTOR LENGTH field are described in 7.7.3.6.1.

The ALGORITHM IDENTIFIER field (see table 492) specifies integrity checking algorithm and shared key length to which the INTEG IKEv2-SCSI cryptographic algorithm descriptor applies.

Table 492 — INTEG ALGORITHM IDENTIFIER field

Code	IKEv2 Name	ICV [a] length (bytes)	Key length (bytes)	Support	Reference
8003 0002h	AUTH_HMAC_SHA1_96	12	20	Optional	RFC 2404
8003 000Ch	AUTH_HMAC_SHA2_256_128	16	32	Optional	RFC 4868
8003 000Eh	AUTH_HMAC_SHA2_512_256	32	64	Optional	RFC 4868
F003 0001h	AUTH_COMBINED	n/a	0	Optional	this subclause
8003 0400h to 8003 FFFFh	Vendor Specific				
0000 0000h to 0000 FFFFh	Restricted				IANA
All others	Reserved				

[a] Integrity Check Value (see 3.1.77).

The AUTH_COMBINED integrity checking algorithm is used with encryption algorithms that include integrity checking as described in 7.7.3.6.2. The AUTH_COMBINED algorithm identifier specifies that no additional integrity check is performed, as indicated by the zero-length key.

The key length used with an integrity checking algorithm is determined by the algorithm identifier as shown in table 492.

The IKEv2-SCSI CCS state maintained for the I_T_L nexus shall be abandoned as described in 7.7.3.8.3 if the parameter list contains:

a) no INTEG IKEv2-SCSI cryptographic algorithm descriptors;
b) more than one INTEG IKEv2-SCSI cryptographic algorithm descriptor;
c) an INTEG IKEv2-SCSI cryptographic algorithm descriptor that contains an algorithm identifier that does not appear in the SA Creations Capabilities payload (see 7.7.3.5.12) last returned by the device server to any application client; or
d) an INTEG IKEv2-SCSI cryptographic algorithm descriptor that contains an algorithm identifier that is not shown in table 492.

7.7.3.6.5 Diffie-Hellman group (D-H) IKEv2-SCSI cryptographic algorithm descriptor

When the ALGORITHM TYPE field is set to D-H (i.e., 04h) in an IKEv2-SCSI cryptographic algorithm descriptor (see table 493), the descriptor specifies Diffie-Hellman group and Diffie-Hellman algorithm used during the IKEv2-SCSI Key Exchange step (see 5.16.4.6) to derive a shared key (see 3.1.162) that is known only to the application client and device server.

Table 493 — D-H IKEv2-SCSI cryptographic algorithm descriptor format

Bit / Byte	7	6	5	4	3	2	1	0
0	ALGORITHM TYPE (04h)							
1	Reserved							
2	(MSB)	IKE DESCRIPTOR LENGTH (000Ch)						
3								(LSB)
4	(MSB)	ALGORITHM IDENTIFIER						
•••								
7								(LSB)
8	Reserved							
•••								
11								

The ALGORITHM TYPE field and IKE DESCRIPTOR LENGTH field are described in 7.7.3.6.1.

The ALGORITHM IDENTIFIER field (see table 494) specifies Diffie-Hellman algorithm, group, and shared key length to which the D-H IKEv2-SCSI cryptographic algorithm descriptor applies.

Table 494 — D-H ALGORITHM IDENTIFIER field

Code	Description	Key length (bytes)	Support	Reference
8004 000Eh	2 048-bit MODP group (finite field D-H)	256	Optional	RFC 3526
8004 000Fh	3 072-bit MODP group (finite field D-H)	384	Optional	RFC 3526
8004 0010h	4 096-bit MODP group (finite field D-H)	512	Optional	RFC 3526
8004 0013h	256-bit random ECP group	64	Optional	RFC 4753 [a]
8004 0015h	521-bit random ECP group	132	Optional	RFC 4753 [a]
8004 0400h to 8004 FFFFh	Vendor Specific			
0000 0000h to 0000 FFFFh	Restricted			IANA
All others	Reserved			

[a] The RFC Errata for RFC 4753 modify the binary representation of ECP key exchange data. The modified binary representation shall be used, see http://rfc-editor.org/errata_search.php?rfc=4753.

The key length of the public value transferred in the KEY EXCHANGE DATA field (see 7.7.3.5.3) is determined by the algorithm identifier as shown in table 494.

The IKEv2-SCSI CCS state maintained for the I_T_L nexus shall be abandoned as described in 7.7.3.8.3 if the parameter list contains:

a) no D-H IKEv2-SCSI cryptographic algorithm descriptors;
b) more than one D-H IKEv2-SCSI cryptographic algorithm descriptor;
c) a D-H IKEv2-SCSI cryptographic algorithm descriptor that contains an algorithm identifier that does not appear in the SA Creations Capabilities payload (see 7.7.3.5.12) last returned by the device server to any application client; or
d) an D-H IKEv2-SCSI cryptographic algorithm descriptor that contains an algorithm identifier that is not shown in table 494.

7.7.3.6.6 IKEv2-SCSI authentication algorithm IKEv2-SCSI cryptographic algorithm descriptor

When the ALGORITHM TYPE field is set to SA_AUTH_OUT (i.e., F9h) in an IKEv2-SCSI cryptographic algorithm descriptor (see table 495), the descriptor specifies Authentication payload authentication algorithm used by the IKEv2-SCSI Authentication step SECURITY PROTOCOL OUT command (see 5.16.4.7.2).

When the ALGORITHM TYPE field is set to SA_AUTH_IN (i.e., FAh) in an IKEv2-SCSI cryptographic algorithm descriptor (see table 495), the descriptor specifies Authentication payload authentication algorithm used by the IKEv2-SCSI Authentication step SECURITY PROTOCOL IN command (see 5.16.4.7.3).

Table 495 — SA_AUTH_OUT and SA_AUTH_IN IKEv2-SCSI cryptographic algorithm descriptor format

Bit / Byte	7	6	5	4	3	2	1	0
0	ALGORITHM TYPE (F9h or FAh)							
1	Reserved							
2	(MSB)	IKE DESCRIPTOR LENGTH (000Ch)						
3								(LSB)
4	(MSB)	ALGORITHM IDENTIFIER						
•••								
7								(LSB)
8	Reserved							
•••								
11								

The ALGORITHM TYPE field and IKE DESCRIPTOR LENGTH field are described in 7.7.3.6.1.

The ALGORITHM IDENTIFIER field (see table 496) specifies Authentication payload authentication algorithm to which the SA_AUTH_OUT IKEv2-SCSI cryptographic algorithm descriptor or SA_AUTH_IN IKEv2-SCSI cryptographic algorithm descriptor applies.

Table 496 — SA_AUTH_OUT and SA_AUTH_IN ALGORITHM IDENTIFIER field

Code	Description	Support	Reference	Authentication Reference [a]
00F9 0000h	SA_AUTH_NONE	Optional	this subclause	n/a
00F9 0001h	RSA Digital Signature with SHA-1 [b]	Optional	RFC 4306	5.16.4.3.3
00F9 0002h	Shared Key Message Integrity Code	Optional	RFC 4306 [c, d]	5.16.4.3.2
00F9 0009h	ECDSA with SHA-256 on the P-256 curve [b]	Optional	RFC 4754	5.16.4.3.3
00F9 000Bh	ECDSA with SHA-512 on the P-521 curve [b]	Optional	RFC 4754	5.16.4.3.3
00F9 00C9h to 00F9 00FFh	Vendor Specific	Optional		
0000 0000h to 0000 FFFFh	Restricted	Prohibited	IANA	
All others	Reserved			

[a] Description of how the algorithm shall be used to generate and verify the contents of the AUTHENTICATION DATA field of the Authentication Payload.

[b] Use of certificates with this digital signature authentication algorithm is optional.

[c] The 17 ASCII character non-terminated pre-shared key (see 3.1.113) pad string "Key Pad for IKEv2" specified by RFC 4306 is replaced by the 22 ASCII character non-terminated pre-shared key pad string 'Key Pad for IKEv2-SCSI'.

[d] The pre-shared key (see 3.1.113) requirements used by this standard (see 5.16.4.3.2) apply in addition to those found in RFC 4306.

SA_AUTH_NONE specifies the omission of the IKEv2-SCSI Authentication step (see 5.16.4.7) as follows:

a) the presence of an SA_AUTH_OUT IKEv2-SCSI cryptographic algorithm descriptor and an SA_AUTH_IN IKEv2-SCSI cryptographic algorithm descriptor with the ALGORITHM IDENTIFIER field set to SA_AUTH_NONE is an SA Creation Capabilities payload (see 7.7.3.5.12) indicates that the device server is allowed to negotiate the omission of the IKEv2-SCSI Authentication step; and
b) the presence of an SA_AUTH_OUT IKEv2-SCSI cryptographic algorithm descriptor and an SA_AUTH_IN IKEv2-SCSI cryptographic algorithm descriptor with the ALGORITHM IDENTIFIER field set to SA_AUTH_NONE is an IKEv2-SCSI SA Cryptographic Algorithms payload (see 7.7.3.5.13) indicates the following based upon the command whose parameter data carries the payload:
   A) in the parameter list for a Key Exchange SECURITY PROTOCOL OUT command (see 5.16.4.6.2), SA_AUTH_NONE specifies that the application client is requesting that the IKEv2-SCSI Authentication step be skipped; and
   B) in the parameter data for a Key Exchange SECURITY PROTOCOL IN command (see 5.16.4.6.3), SA_AUTH_NONE indicates that the device server has agreed to skip the IKEv2-SCSI Authentication step.

If it is reported by a device server in its capabilities and selected by an application client, then the IKEv2-SCSI Authentication step is skipped and the resulting SAs are not authenticated.

An SA_AUTH_OUT IKEv2-SCSI cryptographic algorithm descriptor with the ALGORITHM IDENTIFIER field set to SA_AUTH_NONE shall not appear in an IKEv2-SCSI SA Cryptographic Algorithms payload except as described in 5.16.4.3.4.

The Shared Key Message Integrity Code is based on a pre-shared key (see 3.1.113 and 5.16.4.3.2) that is associated with the identity in the Identification payload (see 7.7.3.5.4).

The IKEv2-SCSI CCS state maintained for the I_T_L nexus shall be abandoned as described in 7.7.3.8.3 if the parameter list contains:

a) no SA_AUTH_OUT IKEv2-SCSI cryptographic algorithm descriptors;
b) no SA_AUTH_IN IKEv2-SCSI cryptographic algorithm descriptors;
c) more than one SA_AUTH_OUT IKEv2-SCSI cryptographic algorithm descriptor;
d) more than one SA_AUTH_IN IKEv2-SCSI cryptographic algorithm descriptor;
e) an SA_AUTH_OUT IKEv2-SCSI cryptographic algorithm descriptor that contains an algorithm identifier that does not appear in the SA Creations Capabilities payload (see 7.7.3.5.12) last returned by the device server to any application client;
f) an SA_AUTH_IN IKEv2-SCSI cryptographic algorithm descriptor that contains an algorithm identifier that does not appear in the SA Creations Capabilities payload last returned by the device server to any application client
g) an SA_AUTH_OUT IKEv2-SCSI cryptographic algorithm descriptor that contains an algorithm identifier that is not shown in table 496;
h) an SA_AUTH_IN IKEv2-SCSI cryptographic algorithm descriptor that contains an algorithm identifier that is not shown in table 496;
i) an SA_AUTH_OUT IKEv2-SCSI cryptographic algorithm descriptor with the ALGORITHM IDENTIFIER field set to SA_AUTH_NONE, and an SA_AUTH_IN IKEv2-SCSI cryptographic algorithm descriptor with the ALGORITHM IDENTIFIER field set to a value other than SA_AUTH_NONE; or
j) an SA_AUTH_IN IKEv2-SCSI cryptographic algorithm descriptor with the ALGORITHM IDENTIFIER field set to SA_AUTH_NONE, and an SA_AUTH_OUT IKEv2-SCSI cryptographic algorithm descriptor with the ALGORITHM IDENTIFIER field set to a value other than SA_AUTH_NONE.

7.7.3.7 Errors in IKEv2-SCSI security protocol commands

For a single I_T_L nexus, the device server shall ensure that the two or four IKEv2-SCSI CCS commands are processed in the order described in 5.16.4.1 based only on the contents of the CDB (i.e., the SECURITY PROTOCOL OUT parameter data shall not be processed unless the tests in table 497 specify the processing of the command) using the tests and responses shown in table 497.

Table 497 — IKEv2-SCSI command ordering processing requirements on a single I_T_L nexus (part 1 of 2)

IKEv2-SCSI SECURITY PROTOCOL OUT command or SECURITY PROTOCOL IN command received	Time →				
	Before Key Exchange step SECURITY PROTOCOL OUT command returns GOOD status	**After a Key Exchange step SECURITY PROTOCOL OUT command returns GOOD status**	**After a Key Exchange step SECURITY PROTOCOL IN command returns GOOD status**	**After an Authentication step SECURITY PROTOCOL OUT command returns GOOD status**	**After an Authentication step SECURITY PROTOCOL IN command returns GOOD status**
Key Exchange step SECURITY PROTOCOL OUT command	Process the command as described in this standard	Do not process the command [a]	Do not process the command [a]	Do not process the command [a]	Same as before Key Exchange step SECURITY PROTOCOL OUT command
Key Exchange step SECURITY PROTOCOL IN command	No IKEv2-SCSI CCS exists [b]	Process the command as described in this standard	Repeat processing of the command [c]	Do not process the command [a]	Do not process the command [a]

[a] The command shall be terminated and the IKEv2-SCSI CCS shall be continued as follows:
a) the command shall be terminated with CHECK CONDITION status, with the sense key set to NOT READY, and the additional sense code set to LOGICAL UNIT NOT READY, SA CREATION IN PROGRESS; and
b) the device server shall continue the IKEv2-SCSI CCS by preparing to receive another IKEv2-SCSI CCS SECURITY PROTOCOL OUT command.

[b] 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.

[c] Processing of the SECURITY PROTOCOL IN commands in an IKEv2-SCSI CCS may be repeated. The device server should save the information necessary to repeat processing of these commands until the number of seconds specified in the IKEV2-SCSI PROTOCOL TIMEOUT field of the IKEv2-SCSI Timeout Values payload (see 7.7.3.5.15) have elapsed since the processing of the Authentication step SECURITY PROTOCOL OUT command that completed with GOOD status.

Table 497 — IKEv2-SCSI command ordering processing requirements on a single I_T_L nexus (part 2 of 2)

IKEv2-SCSI SECURITY PROTOCOL OUT command or SECURITY PROTOCOL IN command received	Time → Before Key Exchange step SECURITY PROTOCOL OUT command returns GOOD status	After a Key Exchange step SECURITY PROTOCOL OUT command returns GOOD status	After a Key Exchange step SECURITY PROTOCOL IN command returns GOOD status	After an Authentication step SECURITY PROTOCOL OUT command returns GOOD status	After an Authentication step SECURITY PROTOCOL IN command returns GOOD status
Authentication step SECURITY PROTOCOL OUT command	No IKEv2-SCSI CCS exists [b]	Do not process the command [a]	Process the command as described in this standard	Do not process the command [a]	Do not process the command [a]
Authentication step SECURITY PROTOCOL IN command		Do not process the command [a]	Do not process the command [a]	Process the command as described in this standard	Repeat processing of the command [c]
Command with an invalid field in the CDB	No IKEv2-SCSI CCS exists [b]	Do not process the command [a]	Do not process the command [a]	Do not process the command [a]	No IKEv2-SCSI CCS exists [b]

[a] The command shall be terminated and the IKEv2-SCSI CCS shall be continued as follows:
a) the command shall be terminated with CHECK CONDITION status, with the sense key set to NOT READY, and the additional sense code set to LOGICAL UNIT NOT READY, SA CREATION IN PROGRESS; and
b) the device server shall continue the IKEv2-SCSI CCS by preparing to receive another IKEv2-SCSI CCS SECURITY PROTOCOL OUT command.

[b] 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.

[c] Processing of the SECURITY PROTOCOL IN commands in an IKEv2-SCSI CCS may be repeated. The device server should save the information necessary to repeat processing of these commands until the number of seconds specified in the IKEV2-SCSI PROTOCOL TIMEOUT field of the IKEv2-SCSI Timeout Values payload (see 7.7.3.5.15) have elapsed since the processing of the Authentication step SECURITY PROTOCOL OUT command that completed with GOOD status.

The processing shown in table 497 shall be performed before parameter data error handling described in 7.7.3.8.

7.7.3.8 Errors in IKEv2-SCSI security protocol parameter data

7.7.3.8.1 Overview

Errors in the parameter data transferred to the device sever by an IKEv2-SCSI SECURITY PROTOCOL OUT command are classified (see table 498) based on the ease with which they may be used to mount denial of service attacks against IKEv2-SCSI SA creation operations by an attacker that has not participated as the application client or device server in the Key Exchange step SECURITY PROTOCOL OUT command (see 5.16.4.6.2) that started the IKEv2-SCSI CCS.

Table 498 — IKEv2-SCSI parameter error categories

Denial of service attack potential	Applicable SECURITY PROTOCOL OUT commands	Error handling description	Reference
High [a]	Authentication step, and Delete operation	If possible, the IKEv2-SCSI CCS state maintained for an I_T_L nexus is not changed	7.7.3.8.2
Minimal [b]	Key Exchange step, Authentication step, and Delete operation	The IKEv2-SCSI CCS state maintained for an I_T_L nexus is abandoned	7.7.3.8.3

[a] Attacks capable of causing significant harm by sending a malformed IKEv2-SCSI SECURITY PROTOCOL OUT command to the device server.

[b] Attacks that produce no significant harm, or collusive attacks (i.e., attacks that require knowledge of the IKEv2-SCSI CCS shared keys and participation in the IKEv2-SCSI CCS). Collusive attacks depend on collusion between the attacker and the application client (i.e., require the application client to act against its own best interests). Abandoning the IKEv2-SCSI CCS is a justified response to such attacks.

7.7.3.8.2 Errors with high denial of service attack potential

Errors detected before or during the decryption and integrity checking of an Encrypted payload in an Authentication step SECURITY PROTOCOL OUT command or a Delete operation SECURITY PROTOCOL OUT command have a high potential for being a denial of service attack against one or more application clients (see table 498 in 7.7.3.8.1).

The device server shall respond to these SECURITY PROTOCOL OUT command errors as follows:

a) the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to SA CREATION PARAMETER VALUE REJECTED; and
b) the device server shall continue the IKEv2-SCSI CCS, if any, by preparing to receive an Authentication step SECURITY PROTOCOL OUT command.

If a specific field is the cause for returning the SA CREATION PARAMETER VALUE REJECTED additional sense code, then the SKSV bit may be set to one. If the SKSV bit is set to one, then the SENSE KEY SPECIFIC field shall be set as defined in 4.5.2.4.2.

7.7.3.8.3 Errors with minimal denial of service attack potential

The errors that have minimal denial of service attack potential for IKEv2-SCSI SA creation (see table 498 in 7.7.3.8.1) are:

a) all errors detected for a Key Exchange step SECURITY PROTOCOL OUT command or its associated parameter data (e.g., errors detected in the IKEv2-SCSI header) that is attempting to establish a new IKEv2-SCSI CCS on an I_T_L nexus; and
b) all errors that are detected after the Encrypted payload has been successfully decrypted and integrity checked in an Authentication step SECURITY PROTOCOL OUT command (see 5.16.4.7.2) or a Delete operation SECURITY PROTOCOL OUT command (see 5.16.4.11).

The device server shall respond to these SECURITY PROTOCOL OUT command errors by terminating the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to SA CREATION PARAMETER VALUE INVALID. The state being maintained for an IKEv2-SCSI CCS on the I_T_L nexus on which the command was received, if any, shall be abandoned (see 5.16.4.10).

If a specific field is the cause for returning the SA CREATION PARAMETER VALUE INVALID additional sense code, then the SKSV bit shall be set to one and SENSE KEY SPECIFIC field shall be set as defined in 4.5.2.4.2.

7.7.3.9 Translating IKEv2 errors

IKEv2 (see RFC 4306) defines an error reporting mechanism based on the Notify payload. This standard translates such error reports into the device server and application actions defined in this subclause.

If a device server is required by IKEv2 to report an error using a Notify payload, the device server shall translate error into CHECK CONDITION status with the sense key and additional sense code shown in table 499. The device server shall terminate the SECURITY PROTOCOL OUT command (see 6.31) that transferred the parameter list in which the IKE requirements for one or more payloads (see 7.7.3.5) require use of the Notify payload to report an error. The SECURITY PROTOCOL OUT command shall be terminated as described in this subclause. The device server shall not report the IKE errors described in this subclause by terminating a SECURITY PROTOCOL IN command (see 6.30) with CHECK CONDITION status.

Table 499 — IKEv2 Notify payload error translations for IKEv2-SCSI

IKEv2 (see RFC 4306)		IKEv2-SCSI	
Error Type	**Description**	**Additional sense code**	**Sense key**
0000h	Reserved		
0001h	UNSUPPORTED_CRITICAL_PAYLOAD	SA CREATION PARAMETER NOT SUPPORTED	ILLEGAL REQUEST
0004h	INVALID_IKE_SPI	SA CREATION PARAMETER VALUE INVALID or SA CREATION PARAMETER VALUE REJECTED	
0005h	INVALID_MAJOR_VERSION		
0007h	INVALID_SYNTAX [a]		
0009h	INVALID_MESSAGE_ID		
000Bh	INVALID_SPI	SA CREATION PARAMETER VALUE INVALID [b]	
000Eh	NO_PROPOSAL_CHOSEN [c]	SA CREATION PARAMETER VALUE INVALID	
0011h	INVALID_KE_PAYLOAD [c]		
0018h	AUTHENTICATION_FAILED	AUTHENTICATION FAILED	ABORTED COMMAND
0022h to 0027h	See RFC 4306 [d]	n/a	n/a
2000h to 3FFFh	Vendor Specific		
All others	Restricted (see RFC 4306)		

[a] This sense key and one of the additional sense codes shown shall be returned for a syntax error within an Encrypted payload (see 7.7.3.5.11) regardless of conflicting IKEv2 requirements.

[b] SA CREATION PARAMETER VALUE INVALID shall be used for an invalid SAI in an IKEv2-SCSI SECURITY PROTOCOL IN or SECURITY PROTOCOL OUT. The additional sense code for an invalid SAI in all other commands is specified by the applicable usage type definition (see table 62 in 5.16.2.2).

[c] An application client recovers by restarting processing with the Device Capabilities step (see 5.16.4.5) to rediscover the device server's capabilities.

[d] These IKEv2 Error Types are used for features that are not supported by IKEv2-SCSI SA creation.

If an application client detects an IKEv2 error that RFC 4306 requires to be reported with a Notify payload, the application client then the application client should abandon the IKEv2-SCSI CCS and notify the device server that it is abandoning the IKEv2-SCSI CCS as described in 5.16.4.10.

7.7.4 CbCS security protocol

7.7.4.1 Overview

If the SECURITY PROTOCOL field in a SECURITY PROTOCOL IN command (see 6.30) is set to 07h, then the command specifies one of the CbCS pages (see 7.7.4.2) to be returned by the device sever. The information returned by a CbCS SECURITY PROTOCOL IN command indicates the CbCS operating parameters of:

a) the logical unit to which the CbCS SECURITY PROTOCOL IN command is addressed; or
b) the SCSI target device that contains the well-known logical unit to which the CbCS SECURITY PROTOCOL IN command is addressed.

If the SECURITY PROTOCOL field in a SECURITY PROTOCOL OUT command (see 6.31) is set to 07h, then the command specifies one of the CbCS pages (see 7.7.4.4) to be sent to the device sever. The instructions sent in a CbCS SECURITY PROTOCOL OUT command specify the CbCS operating parameters of:

a) the logical unit to which the CbCS SECURITY PROTOCOL OUT command is addressed; or
b) the SCSI target device that contains the well-known logical unit to which the CbCS SECURITY PROTOCOL OUT command is addressed.

7.7.4.2 CbCS SECURITY PROTOCOL IN CDB description

The CbCS SECURITY PROTOCOL IN CDB has the format defined in 6.30 with the additional requirements described in this subclause.

When the SECURITY PROTOCOL field is set to CbCS (i.e., 07h) in a SECURITY PROTOCOL IN command, the SECURITY PROTOCOL SPECIFIC field (see table 500) specifies the CbCS page to be returned in the parameter data (see 7.7.4.3). If the CBCS bit is set to one in the Extended INQUIRY Data VPD page (see 7.8.7), the CbCS SECURITY PROTOCOL IN command support requirements are shown in table 500.

Table 500 — SECURITY PROTOCOL SPECIFIC field for the CbCS SECURITY PROTOCOL IN command

Code [a]	CbCS page returned	Support	Reference
0000h	Supported CbCS SECURITY PROTOCOL IN Pages	Mandatory	7.7.4.3.1
0001h	Supported CbCS SECURITY PROTOCOL OUT Pages	Mandatory	7.7.4.3.2
0002h	Unchangeable CbCS Parameters	Mandatory	7.7.4.3.3
0003h to 003Eh	Reserved		
003Fh	Security Token	Optional [b]	7.7.4.3.4
0040h	Current CbCS Parameters	Mandatory	7.7.4.3.5
0041h to D00Fh	Reserved		
D010h	Set Master Key – Seed Exchange	Optional [b]	7.7.4.3.6
D011h to FFFFh	Reserved		

[a] If the SECURITY PROTOCOL SPECIFIC field contains a value that is less than D000h, then the working key specified by the KEY VERSION field in the CbCS capability descriptor (see 6.19.2.3) shall be used to compute the capability key (see 5.16.6.8.12). If the SECURITY PROTOCOL SPECIFIC field contains a value that is greater than or equal to D000h, then the authentication key component of the master key (see 5.16.6.8.11) shall be used to compute the capability key.
[b] Mandatory if the CAPKEY CbCS method (see 5.16.6.8.8.3) is supported.

If a CbCS SECURITY PROTOCOL IN command is received with the INC_512 bit set to one, 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.

7.7.4.3 CbCS SECURITY PROTOCOL IN parameter data

7.7.4.3.1 Supported CbCS SECURITY PROTOCOL IN Pages CbCS page

The Supported CbCS SECURITY PROTOCOL IN Pages CbCS page (see table 501) lists the CbCS pages that are supported for the (i.e., the values that are allowed in the SECURITY PROTOCOL SPECIFIC field in a) SECURITY PROTOCOL IN command (see 6.30).

Table 501 — Supported CbCS SECURITY PROTOCOL IN Pages CbCS page format

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)
Supported CbCS SECURITY PROTOCOL IN page list								
4	(MSB)			SUPPORTED CBCS SECURITY PROTOCOL IN PAGE [first] (0000h)				
5								(LSB)
				⋮				
n-1	(MSB)			SUPPORTED CBCS SECURITY PROTOCOL IN PAGE [last]				
n								(LSB)

The PAGE CODE field shall be set to 0000h to indicate that the Supported CbCS SECURITY PROTOCOL IN Pages CbCS page is being returned.

The page length field indicates the number of bytes that follow in the Supported CbCS SECURITY PROTOCOL IN Pages CbCS page.

Each SUPPORTED CBCS SECURITY PROTOCOL IN PAGE field indicates the page code (see table 500 in 7.7.4.2) of one CbCS page that is supported by the SECURITY PROTOCOL IN command when the SECURITY PROTOCOL field (see 6.30) is set to 07h (i.e., CbCS). The values in the SUPPORTED CBCS SECURITY PROTOCOL IN PAGE fields shall be returned in ascending order beginning with 0000h (i.e., this CbCS page).

7.7.4.3.2 Supported CbCS SECURITY PROTOCOL OUT pages CbCS page

The Supported CbCS SECURITY PROTOCOL OUT Pages CbCS page (see table 502) lists the CbCS pages that are supported for the (i.e., the values that are allowed in the SECURITY PROTOCOL SPECIFIC field in a) SECURITY PROTOCOL OUT command (see 6.31).

Table 502 — Supported CbCS SECURITY PROTOCOL OUT Pages CbCS page format

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)
Supported CbCS SECURITY PROTOCOL OUT page list								
4	(MSB)			SUPPORTED CBCS SECURITY PROTOCOL OUT PAGE [first]				
5								(LSB)
				⋮				
n-1	(MSB)			SUPPORTED CBCS SECURITY PROTOCOL OUT PAGE [last]				
n								(LSB)

The PAGE CODE field shall be set to 0001h to indicate that the Supported CbCS SECURITY PROTOCOL OUT Pages CbCS page is being returned.

The page length field indicates the number of bytes that follow in the Supported CbCS SECURITY PROTOCOL OUT Pages CbCS page.

Each SUPPORTED CBCS SECURITY PROTOCOL OUT PAGE field indicates the page code (see table 509 in 7.7.4.4) of one CbCS page that is supported by the SECURITY PROTOCOL OUT command when the SECURITY PROTOCOL field (see 6.31) is set to 07h (i.e., CbCS). The values in the SUPPORTED CBCS SECURITY PROTOCOL IN PAGE fields shall be returned in ascending order.

7.7.4.3.3 Unchangeable CbCS Parameters CbCS page

The Unchangeable CbCS Parameters CbCS page (see table 503) indicates the supported CbCS features and algorithms.

Table 503 — Unchangeable CbCS Parameters CbCS page format (part 1 of 2)

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)	PAGE CODE (0002h)						
1								(LSB)
2	(MSB)	PAGE LENGTH (n-3)						
3								(LSB)
4	KEYS SUPPORT		MIN CBCS METHOD SUP		Reserved			
5	Reserved							
6	(MSB)	SUPPORTED INTEGRITY CHECK VALUE ALGORITHM LIST LENGTH (c-7)						
7								(LSB)
Supported integrity check value algorithms list								
8	(MSB)	SUPPORTED INTEGRITY CHECK VALUE ALGORITHM [first]						
•••								
11								(LSB)
	⋮							
c-3	(MSB)	SUPPORTED INTEGRITY CHECK VALUE ALGORITHM [last]						
•••								
c								(LSB)
c+1	Reserved							
c+2								
c+3	(MSB)	SUPPORTED D-H ALGORITHM LIST LENGTH (d-c-4)						
c+4								(LSB)
Supported Diffie-Hellman (D-H) algorithms list								
c+5	(MSB)	SUPPORTED D-H ALGORITHM [first]						
•••								
c+8								(LSB)
	⋮							
d-3	(MSB)	SUPPORTED D-H ALGORITHM [last]						
•••								
d								(LSB)

Table 503 — Unchangeable CbCS Parameters CbCS page format (part 2 of 2)

Bit / Byte	7	6	5	4	3	2	1	0
d+1	(MSB)	SUPPORTED CBCS METHODS LIST LENGTH (n-d-2)						
d+2								(LSB)
Supported CbCS methods list								
d+3	SUPPORTED CBCS METHOD [first]							
	⋮							
n	SUPPORTED CBCS METHOD [last]							

The PAGE CODE field shall be set to 0002h to indicate that the Unchangeable CbCS Parameters CbCS page is being returned.

The page length field indicates the number of bytes that follow in the Unchangeable CbCS Parameters CbCS page.

The KEYS SUPPORT field (see table 504) indicates the type of CbCS master keys and working keys supported.

Table 504 — KEYS SUPPORT field

Code	Description
00b	Reserved
01b	The SCSI target device supports single CbCS master key and a set of CbCS working keys (see 5.16.6.8.11) for the SCSI target device, but the logical units in the SCSI target device do not support CbCS master keys or working keys.
10b	The SCSI target device does not support CbCS master keys or working keys for the SCSI target device, but each logical unit in the SCSI target device supports a single CbCS master key and a set of CbCS working keys for that logical unit.
11b	The SCSI target device supports single CbCS master key and a set of CbCS working keys for the SCSI target device, and each logical unit in the SCSI target device supports a single CbCS master key and a set of CbCS working keys for that logical unit. Keys stored in the logical unit take precedence over keys stored in the SCSI target device (see 5.16.6.8.6).

The MIN CBCS METHOD SUP field (see table 505) indicates how the assignment of the minimum allowable CbCS method is supported.

Table 505 — MIN CBCS METHOD SUP field

Code	Description
00b	Reserved
01b	A single minimum allowed CbCS method (see 5.16.6.8.8) is assigned to all logical units in the SCSI target device, and the SECURITY PROTOCOL well known logical unit is implemented to control its value.
10b	Each logical unit in the SCSI target device is assigned its own minimum allowed CbCS method.
11b	Reserved

The SUPPORTED INTEGRITY CHECK VALUE ALGORITHM LIST LENGTH field indicates the number of bytes that follow in the supported integrity check value algorithms list.

Each SUPPORTED INTEGRITY CHECK VALUE ALGORITHM field indicates one supported algorithm for computing CbCS integrity check values. The values in the SUPPORTED INTEGRITY CHECK VALUE ALGORITHM fields are selected from the codes that table 92 (see 5.16.8) lists as integrity checking (i.e., AUTH) algorithms, except for AUTH_COMBINED.

The SUPPORTED D-H ALGORITHM LIST LENGTH field indicates the number of bytes that follow in the supported Diffie-Hellman (D-H) algorithms list.

Each SUPPORTED D-H ALGORITHM field indicates one supported algorithm for constructing CbCS shared keys. The values in the SUPPORTED D-H ALGORITHM fields are selected from the codes that table 92 (see 5.16.8) lists as Diffie-Hellman algorithms with finite field D-H computations.

The SUPPORTED CBCS METHODS LIST LENGTH field indicates the number of bytes that follow in the supported CbCS methods list.

Each SUPPORTED CBCS METHODS field indicates one supported CbCS method (see 6.19.2.3). The values in the SUPPORTED CBCS METHODS fields are selected from the codes listed in table 222 (see 6.19.2.3).

7.7.4.3.4 Security Token CbCS page

The Security Token CbCS page (see table 506) indicates the value of the security token (see 5.16.6.8.10) for the I_T nexus on which the SECURITY PROTOCOL IN command was received.

Table 506 — Security Token CbCS page format

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)	PAGE CODE (003Fh)						
1								(LSB)
2	(MSB)	PAGE LENGTH (n-3)						
3								(LSB)
4	(MSB)	SECURITY TOKEN						
•••								
n								(LSB)

The PAGE CODE field shall be set to 003Fh to indicate that the Security Token CbCS page is being returned.

The page length field indicates the number of bytes that follow in the Security Token CbCS page.

The SECURITY TOKEN field shall contain the security token (see 5.16.6.8.10) for the I_T nexus on which the command was received.

7.7.4.3.5 Current CbCS Parameters CbCS page

The Current CbCS Parameters CbCS page (see table 507) indicates the current values for the CbCS parameters (see 5.16.6.8.15) used by the SCSI target device or logical unit as follows:

a) if the logical unit to which the SECURITY PROTOCOL IN command is addressed is the SECURITY PROTOCOL well known logical unit (see 8.5), then the contents of the Current CbCS Parameters CbCS page apply to the SCSI target device; or
b) if the logical unit to which the SECURITY PROTOCOL IN command is addressed is not the SECURITY PROTOCOL well known logical unit, then the contents of the Current CbCS Parameters CbCS page apply to the addressed logical unit.

Table 507 — Current CbCS Parameters CbCS page format

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)			PAGE CODE (0040h)				
1								(LSB)
2	(MSB)			PAGE LENGTH (009Ah)				
3								(LSB)
4				Reserved				
5								
6								
7				MINIMUM ALLOWED CBCS METHOD				
8	(MSB)			POLICY ACCESS TAG				
•••								
11								(LSB)
12				Reserved				
•••								
15								
16	(MSB)			MASTER KEY IDENTIFIER				
•••								
23								(LSB)
24	(MSB)			WORKING KEY 0 IDENTIFIER				
•••								
31								(LSB)
32	(MSB)			WORKING KEY 1 IDENTIFIER				
•••								
39								(LSB)
				⋮				
144	(MSB)			WORKING KEY 15 IDENTIFIER				
•••								
151								(LSB)
152	(MSB)			CLOCK				
•••								
157								(LSB)

The PAGE CODE field shall be set to 0040h to indicate that the Current CbCS Parameters CbCS page is being returned.

The page length field indicates the number of bytes that follow in the Current CbCS Parameters CbCS page.

The contents of the MINIMUM ALLOWED CBCS METHOD field depend on the logical unit that returned the Current CbCS Parameters CbCS page as follows:

a) if the logical unit is not the SECURITY PROTCOL well known logical unit, then the MINIMUM ALLOWED CBCS METHOD field indicates the smallest value allowed in the CBCS METHOD field (see 6.19.2.3) of a capability descriptor processed by the enforcement manager (see 5.16.6.8.7) as described in 5.16.6.8.13.2 (i.e., the value of the minimum CbCS method CbCS parameter for the logical unit (see 5.16.6.8.15); or
b) if the SECURITY PROTOCOL well known logical unit is returning the Current CbCS Parameters CbCS page, then the MINIMUM ALLOWED CBCS METHOD field indicates the value that will be copied to the minimum CbCS method CbCS parameter of any dynamically created logical units (i.e., the initial minimum CbCS method CbCS parameter summarized in 5.16.6.8.15).

The value in the MINIMUM ALLOWED CBCS METHOD field is selected from those listed in table 222 (see 6.19.2.3).

The contents of the POLICY ACCESS TAG field depend on the logical unit that returned the Current CbCS Parameters CbCS page as follows:

a) if the logical unit is not the SECURITY PROTCOL well known logical unit, then the POLICY ACCESS TAG field indicates the value required in the POLICY ACCESS TAG field (see 6.19.2.3) of a capability descriptor processed by the enforcement manager (see 5.16.6.8.7) as described in 5.16.6.8.13.2 (i.e., the value of the policy access tag CbCS parameter for the logical unit (see 5.16.6.8.15); or
b) if the SECURITY PROTOCOL well known logical unit is returning the Current CbCS Parameters CbCS page, then the POLICY ACCESS TAG field indicates the value that will be copied to the policy access tag CbCS parameter of any dynamically created logical units (i.e., the initial policy access tag CbCS parameter summarized in 5.16.6.8.15).

The MASTER KEY IDENTIFIER field contains the current CbCS shared key identifier (see 5.16.6.8.11.2) for the master key (see 5.16.6.8.11).

The WORKING KEY 0 IDENTIFIER field, WORKING KEY 1 IDENTIFIER field, WORKING KEY 2 IDENTIFIER field, … and WORKING KEY 15 IDENTIFIER field contain the current CbCS shared key identifier (see 5.16.6.8.11.2) for the working keys (see 5.16.6.8.11).

The CLOCK field shall be set to the TIMESTAMP field format and value defined in 5.15.

7.7.4.3.6 Set Master Key – Seed Exchange CbCS page

The Set Master Key – Seed Exchange CbCS page (see table 508) in a SECURITY PROTOCOL IN command continues a CbCS master key update CCS (see 5.16.6.8.11.4) that was initiated by processing of a Set Master Key – Seed Exchange CbCS page in a SECURITY PROTOCOL OUT command (see 7.7.4.5.5), and completes a Diffie-Hellman key exchange protocol as part of that CCS.

Table 508 — Set Master Key – Seed Exchange CbCS page format

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)	PAGE CODE (D010h)						
1								(LSB)
2	(MSB)	PAGE LENGTH (n-3)						
3								(LSB)
4	(MSB)	D-H DATA						
•••								
n								(LSB)

The Diffie-Hellman (D-H) algorithm being used in the CbCS master key update CCS is determined by the contents of the D-H ALGORITHM field in the Set Master Key – Seed Exchange CbCS page in the SECURITY PROTOCOL OUT command (see 7.7.4.5.5) that initiated the CCS. This Diffie-Hellman algorithm specifies the DH_generator value and DH_prime value to be used in the computation of the D-H DATA field as described in this subclause.

The PAGE CODE field set to D010h specifies that the Set Master Key – Seed Exchange CbCS page follows.

The PAGE LENGTH field specifies the number of bytes that follow in the Set Master Key – Seed Exchange CbCS page. If the PAGE LENGTH field is set to a value that is inconsistent with the Diffie-Hellman algorithm specified for the CbCS master key update CCS, 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 PARAMETER LIST.

The contents of the D-H DATA field are computed as follows:

1) a random number, y, is generated having a value between 0 and DH_prime minus one observing the requirements in RFC 4086; and
2) the D-H DATA field is set to $DH_generator^{y}$ modulo DH_prime, where the DH_generator and DH_prime values are identified by the value in the D-H ALGORITHM field.

As part of the successful completion of processing for the SECURITY PROTOCOL IN command transfer of the Set Master Key – Seed Exchange CbCS page the device server and application client store as part of the state maintained for CbCS master key update CCS (see 5.16.6.8.11.4):

a) the authentication key component of the next master key; and
b) the generation key component of the next master key.

The new master key is computed as follows:

1) an initial_seed value that is the value $DH_generator^{xy}$ modulo DH_prime is computed as follows:
   A) the device server computes $(DH_generator^{x}$ modulo $DH_prime)^{y}$, where $DH_generator^{x}$ is the contents of the D-H DATA field in the SECURITY PROTOCOL OUT command and y is the random number generated by the device server; and

B) the application client computes (DH_generator^y modulo DH_prime)^x, where DH_generator^y is the contents of the D-H DATA field in the SECURITY PROTOCOL IN command and x is the random number generated by the application client (see 7.7.4.5.5);

2) the generation key component of the new master key is computed using the integrity check value algorithm specified by the integrity check value algorithm field in the capability (see 6.19.2.3) in the CbCS extension descriptor (see 5.16.6.8.16) associated with the SECURITY PROTOCOL IN command that transferred the Set Master Key – Seed Exchange CbCS page. The following inputs are used with the specified integrity check value algorithm:
   A) the concatenation of the initial_seed value computed in step 1) and all of the bytes in the Device Identification VPD page (see 7.8.6) for the logical unit that is processing the CbCS master key update CCS (see 5.16.6.8.11.4) as the string for which the integrity check value is to be computed; and
   B) the generation key component of the current master key (see 5.16.6.8.11) for the logical unit that is processing the CbCS master key update CCS as the cryptographic key;

3) a modified_seed value is computed as follows:
   A) if the least significant bit of the initial_seed value is zero, then the modified_seed value is equal to the initial_seed value with the least significant bit set to one; and
   B) if the least significant bit of the initial_seed value is one, then the modified_seed value is equal to the initial_seed value with the least significant bit set to zero;

   and

4) the authentication key component of the new master key is computed using the integrity check value algorithm specified by the integrity check value algorithm field in the capability (see 6.19.2.3) in the CbCS extension descriptor (see 5.16.6.8.16) associated with the SECURITY PROTOCOL IN command that transferred the Set Master Key – Seed Exchange CbCS page. The following inputs are used with the specified integrity check value algorithm:
   A) the concatenation of the modified_seed value computed in step 3) and all of the bytes in the Device Identification VPD page for the logical unit that is processing the CbCS master key update CCS as the string for which the integrity check value is to be computed; and
   B) the generation key component of the current master key for the logical unit that is processing the CbCS master key update CCS as the cryptographic key.

7.7.4.4 CbCS SECURITY PROTOCOL OUT CDB description

The CbCS SECURITY PROTOCOL OUT CDB has the format defined in 6.31 with the additional requirements described in this subclause.

When the SECURITY PROTOCOL field is set to CbCS (i.e., 07h) in a SECURITY PROTOCOL OUT command, the SECURITY PROTOCOL SPECIFIC field (see table 509) specifies the CbCS page to be returned in the parameter data (see 7.7.4.5). If the CBCS bit is set to one in the Extended INQUIRY Data VPD page (see 7.8.7), the CbCS SECURITY PROTOCOL IN command support requirements are shown in table 509.

Table 509 — SECURITY PROTOCOL SPECIFIC field for the CbCS SECURITY PROTOCOL OUT command

Code [a]	CbCS page sent	Support	Reference
0000h to 0040h	Reserved		
0041h	Set Policy Access Tag	Optional	7.7.4.5.1
0042h	Set Minimum CbCS Method	Optional	7.7.4.5.2
0043h to CFFFh	Reserved		
D000h	Invalidate Key	Optional [b]	7.7.4.5.3
D001h	Set Key	Optional [b]	7.7.4.5.4
D003h to D00Fh	Reserved		
D010h	Set Master Key – Seed Exchange	Optional [b]	7.7.4.5.5
D011h	Set Master Key – Change Master Key	Optional [b]	7.7.4.5.6
D012h to FFFFh	Reserved		

[a] If the SECURITY PROTOCOL SPECIFIC field contains a value that is less than D000h, then the working key specified by the KEY VERSION field in the CbCS capability descriptor (see 6.19.2.3) shall be used to compute the capability key (see 5.16.6.8.12). If the SECURITY PROTOCOL SPECIFIC field contains a value that is greater than or equal to D000h, then the authentication key component of the master key (see 5.16.6.8.11) shall be used to compute the capability key.

[b] Mandatory if the CAPKEY CbCS method (see 5.16.6.8.8.3) is supported.

If a CbCS SECURITY PROTOCOL OUT command is received with the INC_512 bit set to one, 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.

7.7.4.5 CbCS SECURITY PROTOCOL OUT parameter data

7.7.4.5.1 Set Policy Access Tag CbCS page

The Set Policy Access Tag CbCS page (see table 510) specifies a new policy access tag CbCS parameter value or a new initial policy access tag CbCS parameter value.

Table 510 — Set Policy Access Tag CbCS page format

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)	PAGE CODE (0041h)						
1								(LSB)
2	(MSB)	PAGE LENGTH (0004h)						
3								(LSB)
4	(MSB)	POLICY ACCESS TAG						
•••								
7								(LSB)

The PAGE CODE field set to 0041h specifies that the Set Policy Access Tag CbCS page follows.

The PAGE LENGTH field specifies the number of bytes that follow in the Set Policy Access Tag CbCS page. If the PAGE LENGTH field is set to a value that is smaller than four, 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 PARAMETER LIST.

The CbCS parameter (see 5.16.6.8.15) that is set to the contents of the POLICY ACCESS TAG field depends on the logical unit that is processing the Set Policy Access Tag CbCS page as follows:

a) if the logical unit is not the SECURITY PROTCOL well known logical unit, then the contents of the POLICY ACCESS TAG field shall be placed in the policy access tag CbCS parameter for the logical unit; or
b) if the SECURITY PROTOCOL well known logical unit is processing the Set Policy Access Tag CbCS page, then the contents of the POLICY ACCESS TAG field shall be placed in the initial policy access tag CbCS parameter.

7.7.4.5.2 Set Minimum CbCS Method CbCS page

The Set Minimum CbCS Method CbCS page (see table 511) specifies a new minimum CbCS method CbCS parameter value or a new initial minimum CbCS method CbCS parameter value.

Table 511 — Set Minimum CbCS Method CbCS page format

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)	PAGE CODE (0042h)						
1								(LSB)
2	(MSB)	PAGE LENGTH (0002h)						
3								(LSB)
4	Reserved							
5	MINIMUM ALLOWED CBCS METHOD							

The PAGE CODE field set to 0042h specifies that the Set Minimum CbCS Method CbCS page follows.

The PAGE LENGTH field specifies the number of bytes that follow in the Set Minimum CbCS Method CbCS page. If the PAGE LENGTH field is set to a value that is smaller than two, 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 PARAMETER LIST.

The CbCS parameter or CbCS parameters (see 5.16.6.8.15) that are set to the contents of the MINIMUM ALLOWED CBCS METHOD field depends on the logical unit that is processing the Set Minimum CbCS Method CbCS page and the contents of the MIN CBCS METHOD SUP field in the Unchangeable CbCS Parameters CbCS page (see 7.7.4.3.3) as shown in table 512.

Table 512 — Minimum CbCS Method CbCS Parameter set

MIN CBCS METHOD SUP field	CbCS Parameter set based on the logical unit that processes the Set Minimum CbCS Method CbCS page	
	Not the SECURITY PROTOCOL well known logical unit	SECURITY PROTOCOL well known logical unit
00b	Reserved (see table 505)	
01b	The command is 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	All minimum CbCS method CbCS parameters for all logical units in the SCSI target device, and the initial minimum CbCS method, if any
10b	The minimum CbCS method CbCS parameter for the logical unit that processes the Set Minimum CbCS Method CbCS page	The initial minimum CbCS method CbCS parameter
11b	Reserved (see table 505)	

If the MINIMUM ALLOWED CBCS METHOD field specifies a value that does not appear in the supported CbCS methods list in the Unchangeable CbCS Parameters CbCS page (see 7.7.4.3.3), 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 PARAMETER LIST.

7.7.4.5.3 Invalidate Key CbCS page

The Invalidate Key CbCS page (see table 513) causes a working key (see 5.16.6.8.11) to be invalidated. After the successful processing of an Invalidate Key CbCS page, the working key with the specified key version shall not be valid for the purposes of enforcement manager (see 5.16.6.8.7) CbCS extension descriptor validation (see 5.16.6.8.13.2).

Table 513 — Invalidate Key CbCS page format

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)	PAGE CODE (D000h)						
1								(LSB)
2	(MSB)	PAGE LENGTH (0004h)						
3								(LSB)
4		Reserved						
5								
6								
7	Reserved				KEY VERSION			

The PAGE CODE field set to D000h specifies that the Invalidate Key CbCS page follows.

The PAGE LENGTH field specifies the number of bytes that follow in the Invalidate Key CbCS page. If the PAGE LENGTH field is set to a value that is smaller than four, 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 PARAMETER LIST.

The KEY VERSION field specifies which working key (e.g., a KEY VERSION field set to four specifies that the working key whose Current CbCS Parameters CbCS page (see 7.7.4.3.5) key identifier is returned in the WORKING KEY 4 IDENTIFIER field) is to be invalidated as follows:

a) if the Invalidate Key CbCS page is processed by a logical unit that is not the SECURITY PROTOCOL well known logical unit, then the specified working key for that logical unit shall be invalidated; or
b) if the Invalidate Key CbCS page is processed by the SECURITY PROTOCOL well known logical unit, then the specified target-wide working key (see 5.16.6.8.11) shall be invalidated.

The CbCS shared key identifier (see 5.16.6.8.11.2) for the invalidated working key shall be set to FFFF FFFF FFFF FFFEh.

It shall not be an error to invalidate a working key that is already invalid.

7.7.4.5.4 Set Key CbCS page

The Set Key CbCS page (see table 514) causes a working key (see 5.16.6.8.11) to be set to a new value. After the successful processing of a Set Key CbCS page, the working key with the specified key version shall be valid for the purposes of enforcement manager (see 5.16.6.8.7) CbCS extension descriptor validation (see 5.16.6.8.13.2).

Table 514 — Set Key CbCS page format

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)	PAGE CODE (D001h)						
1								(LSB)
2	(MSB)	PAGE LENGTH (0020h)						
3								(LSB)
4		Reserved						
5								
6								
7	Reserved				KEY VERSION			
8	(MSB)	KEY IDENTIFIER						
•••								
15								(LSB)
16	(MSB)	SEED						
•••								
35								(LSB)

The PAGE CODE field set to D001h specifies that the Set Key CbCS page follows.

The PAGE LENGTH field specifies the number of bytes that follow in the Set Key CbCS page. If the PAGE LENGTH field is set to a value that is other than 32, 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 PARAMETER LIST.

The KEY VERSION field specifies which working key (e.g., a KEY VERSION field set to six specifies that the working key whose Current CbCS Parameters CbCS page (see 7.7.4.3.5) key identifier is returned in the WORKING KEY 6 IDENTIFIER field) is to be set as follows:

a) if the Set Key CbCS page is processed by a logical unit that is not the SECURITY PROTOCOL well known logical unit, then the specified working key for that logical unit shall be set; or
b) if the Set Key CbCS page is processed by the SECURITY PROTOCOL well known logical unit, then the specified target-wide working key (see 5.16.6.8.11) shall be set.

The KEY IDENTIFIER field specifies the value to which the CbCS shared key identifier (see 5.16.6.8.11.2) shall be set for the working key affected by the Set Key CbCS page. If the KEY IDENTIFIER field is set to a value that table 76 (see 5.16.6.8.11.2) describes as reserved in the CbCS pages that change CbCS shared key values, 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 PARAMETER LIST.

The seed field contains a random number that is an input to the computation of the new working key value.

The value to which the specified working key is set shall be computed using the integrity check value algorithm specified by the INTEGRITY CHECK VALUE ALGORITHM field in the capability (see 6.19.2.3) in the CbCS extension descriptor (see 5.16.6.8.16) associated with the SECURITY PROTOCOL OUT command that sent the Set Key CbCS page. The following inputs shall be used with the specified integrity check value algorithm:

a) the contents of the SEED field in the Set Key CbCS page as the string for which the integrity check value is to be computed; and
b) the generation key component of the master key (see 5.16.6.8.11) for the logical unit that is processing the Set Key CbCS page as the cryptographic key.

7.7.4.5.5 Set Master Key – Seed Exchange CbCS page

The Set Master Key – Seed Exchange CbCS page (see table 515) in a SECURITY PROTOCOL OUT command initiates a CbCS master key update CCS (see 5.16.6.8.11.4) and begins a Diffie-Hellman key exchange protocol as part of that CCS.

Table 515 — Set Master Key – Seed Exchange CbCS page format

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)			PAGE CODE (D010h)				
1								(LSB)
2	(MSB)			PAGE LENGTH (n-3)				
3								(LSB)
4	(MSB)			D-H ALGORITHM				
•••								
7								(LSB)
8	(MSB)			D-H DATA LENGTH (n-11)				
•••								
11								(LSB)
12	(MSB)			D-H DATA				
•••								
n								(LSB)

The PAGE CODE field set to D010h specifies that the Set Master Key – Seed Exchange CbCS page follows.

The PAGE LENGTH field specifies the number of bytes that follow in the Set Master Key – Seed Exchange CbCS page. If the PAGE LENGTH field is set to a value that is smaller than ten, 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 PARAMETER LIST.

The D-H ALGORITHM field specifies the Diffie-Hellman (D-H) algorithm to be used in the seed exchange protocol. The value in the D-H ALGORITHM field is selected from the codes that table 92 (see 5.16.8) lists as Diffie-Hellman algorithms with finite field D-H computations. The Diffie-Hellman algorithm selected specifies the DH_generator value and DH_prime value to be used in the computation of the D-H DATA field as described in this subclause.

If the value in the D-H ALGORITHM field does not appear in the Supported Diffie-Hellman algorithms list in the Unchangeable CbCS Parameters CbCS page (see 7.7.4.3.3), 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 PARAMETER LIST.

The D-H DATA LENGTH field specifies the number of bytes that follow in D-H DATA field. If the D-H DATA LENGTH field is set to a value that is inconsistent with the Diffie-Hellman algorithm specified by the D-H ALGORITHM 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 PARAMETER LIST.

The contents of the D-H DATA field are computed as follows:

1) a random number, x, is generated having a value between 0 and DH_prime minus one observing the requirements in RFC 4086; and
2) the D-H DATA field is set to DH_generator^{x} modulo DH_prime, where the DH_generator and DH_prime values are identified by the value in the D-H ALGORITHM field.

7.7.4.5.6 Set Master Key – Change Master Key CbCS page

The Set Master Key – Change Master Key CbCS page (see table 516) concludes a CbCS master key update CCS (see 5.16.6.8.11.4) and changes the master key components to the values computed as part of processing completion (see 7.7.4.3.6) for the SECURITY PROTOCOL IN command that transferred the Set Master Key – Seed Exchange CbCS page.

Table 516 — Set Master Key – Change Master Key CbCS page format

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)			PAGE CODE (D011h)				
1								(LSB)
2	(MSB)			PAGE LENGTH (n-3)				
3								(LSB)
4				Reserved				
•••								
7								
8	(MSB)			KEY IDENTIFIER				
•••								
15								(LSB)
16	(MSB)			APPLICATION CLIENT D-H DATA LENGTH (k-19)				
•••								
19								(LSB)
20	(MSB)			APPLICATION CLIENT D-H DATA				
•••								
k								(LSB)
k+1	(MSB)			DEVICE SERVER D-H DATA LENGTH (n-k-4)				
•••								
k+4								(LSB)
k+5	(MSB)			DEVICE SERVER D-H DATA				
•••								
n								(LSB)

The PAGE CODE field set to D011h specifies that the Set Master Key – Change Master Key CbCS page follows.

The PAGE LENGTH field specifies the number of bytes that follow in the Set Master Key – Change Master Key CbCS page. If the PAGE LENGTH field is set to a value that is smaller than 24, 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 PARAMETER LIST.

The KEY IDENTIFIER field specifies the value to which the master key CbCS shared key identifier (see 5.16.6.8.11.2) shall be set. If the KEY IDENTIFIER field is set to a value that table 76 (see 5.16.6.8.11.2) describes as reserved in the CbCS pages that change CbCS shared key values, 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 PARAMETER LIST.

The APPLICATION CLIENT D-H DATA LENGTH field specifies the number of bytes that follow in the APPLICATION CLIENT D-H DATA field. If the contents of the APPLICATION CLIENT D-H DATA LENGTH field do not match the number of Diffie-Hellman (D-H) data bytes that the device server received in the SECURITY PROTOCOL OUT command that sent the Set Master Key – Seed Exchange CbCS page (see 7.7.4.5.5) and initiated the current CbCS master key update CCS (see 5.16.6.8.11.4), then the master key shall not be modified and 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.

The APPLICATION CLIENT D-H DATA field contains Diffie-Hellman data field that was sent in the Set Master Key – Seed Exchange CbCS page that initiated the current CbCS master key update CCS. If the contents of the APPLICATION CLIENT D-H DATA field do not match Diffie-Hellman data that the device server received in the SECURITY PROTOCOL OUT command that sent the Set Master Key – Seed Exchange CbCS page that initiated the current CbCS master key update CCS (see 5.16.6.8.11.4), then the master key shall not be modified and 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.

The DEVICE SERVER D-H DATA LENGTH field specifies the number of bytes that follow in the DEVICE SERVER D-H DATA field. If the contents of the DEVICE SERVER D-H DATA LENGTH field do not match the number of Diffie-Hellman data bytes that the device server returned in the SECURITY PROTOCOL IN command that returned the Set Master Key – Seed Exchange CbCS page (see 7.7.4.3.6) in the current CbCS master key update CCS, then the master key shall not be modified and 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.

The DEVICE SERVER D-H DATA field contains Diffie-Hellman data field that was returned in the Set Master Key – Seed Exchange CbCS page that the device server returned in response to a SECURITY PROTOCOL IN COMMAND as part of the current CbCS master key update CCS. If the contents of the DEVICE SERVER D-H DATA field do not match Diffie-Hellman data that the device server sent in the SECURITY PROTOCOL IN command that send the Set Master Key – Seed Exchange CbCS page, then the master key shall not be modified and 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.

7.8 Vital product data parameters

7.8.1 Vital product data parameters overview and page codes

This subclause describes the vital product data (VPD) page structure and the VPD pages (see table 517) that are applicable to all SCSI devices. These VPD pages are returned by an INQUIRY command with the EVPD bit set to one (see 6.4) and contain vendor specific product information about a logical unit and SCSI target device. The vital product data may include vendor identification, product identification, unit serial numbers, device operating definitions, manufacturing data, field replaceable unit information, and other vendor specific information. This standard defines the structure of the vital product data, but not the contents.

Table 517 — Vital product data page codes

VPD Page Name	Page code	Reference	Support requirements
ASCII Information	01h to 7Fh	7.8.3	Optional
ATA Information	89h	SAT-2	Optional [a]
CFA Profile Information	8Ch	7.8.4	Optional
Device Constituents	8Bh	7.8.5	Optional
Device Identification	83h	7.8.6	Mandatory
Extended INQUIRY Data	86h	7.8.7	Optional
Management Network Addresses	85h	7.8.8	Optional
Mode Page Policy	87h	7.8.9	Optional
Power Condition	8Ah	7.8.10	Optional
SCSI Ports	88h	7.8.11	Optional
Protocol Specific Logical Unit Information	90h	7.8.12	Protocol specific [b]
Protocol Specific Port Information	91h	7.8.13	Protocol specific [b]
Software Interface Identification	84h	7.8.14	Optional
Supported VPD Pages	00h	7.8.15	Mandatory
Unit Serial Number	80h	7.8.16	Optional
Restricted (see applicable command standard)	B0h to BFh		
Obsolete [c]			
Vendor specific [d]			
Reserved	All other codes		
A numeric ordered listing of VPD page codes is provided in D.7.			

[a] See SAT-2 for support requirements.
[b] See applicable SCSI transport protocol standard (see 3.1.147) for support requirements.
[c] The following page codes are obsolete: 81h and 82h.
[d] The following page codes are vendor specific: C0h to FFh.

This standard does not define the location or method of storing the vital product data. The retrieval of the vital product data may require completion of initialization operations within the device, that may result in delays before the vital product data is available to the application client. Time-critical requirements are an implementation consideration and are not defined in this standard.

7.8.2 VPD page format and page codes for all device types

This subclause describes the VPD page structure and the VPD pages that are applicable to all SCSI devices. VPD pages specific to each device type are described in the command standard (see 3.1.31) that applies to that device type.

An INQUIRY command with the EVPD bit set to one (see 6.4) specifies that the device server return a VPD page using the format defined in table 518.

Table 518 — VPD page format

Bit / Byte	7	6	5	4	3	2	1	0
0	PERIPHERAL QUALIFIER			PERIPHERAL DEVICE TYPE				
1	PAGE CODE							
2	(MSB)	PAGE LENGTH (n-3)						
3								(LSB)
4	VPD parameters							
•••								
n								

The PERIPHERAL QUALIFIER field and the PERIPHERAL DEVICE TYPE field are the same as defined for standard INQUIRY data (see 6.4.2).

The PAGE CODE field (see 7.8.1) identifies the VPD page and contains the same value as in the PAGE OR OPERATION CODE field in the INQUIRY CDB (see 6.4).

The PAGE LENGTH field indicates the length in bytes of the VPD parameters that follow this field. The relationship between the PAGE LENGTH field and the CDB ALLOCATION LENGTH field is defined in 4.3.5.6.

The VPD parameters are defined for each VPD page code.

7.8.3 ASCII Information VPD page

The ASCII Information VPD page (see table 519) contains information for the field replaceable unit code returned in the sense data (see 4.5).

Table 519 — ASCII Information VPD page

Bit / Byte	7	6	5	4	3	2	1	0
0	PERIPHERAL QUALIFIER			PERIPHERAL DEVICE TYPE				
1	PAGE CODE (01h to 7Fh)							
2	(MSB)	PAGE LENGTH (n-3)						
3								(LSB)
4	ASCII LENGTH (m-4)							
5	(MSB)	ASCII INFORMATION						
•••								
m								(LSB)
m+1	Vendor specific information							
•••								
n								

The PERIPHERAL QUALIFIER field, PERIPHERAL DEVICE TYPE field, PAGE CODE field, and PAGE LENGTH field are defined in 7.8.2.

The value in the PAGE CODE field is associated with the FIELD REPLACEABLE UNIT CODE field returned in the sense data.

NOTE 70 - The FIELD REPLACEABLE UNIT CODE field in the sense data provides for 255 possible codes, while the PAGE CODE field provides for only 127 possible codes. For that reason it is not possible to return ASCII Information VPD pages for the upper code values.

The PAGE LENGTH field is defined in 7.8.2.

The ASCII LENGTH field indicates the length in bytes of the ASCII INFORMATION field that follows. A value of zero in this field indicates that no ASCII information is available for the indicated page code. The relationship between the ASCII LENGTH field and the CDB ALLOCATION LENGTH field is defined in 4.3.5.6.

The ASCII INFORMATION field contains ASCII information concerning the field replaceable unit identified by the page code. The data in this field shall be formatted in one or more character string lines. Each line shall contain only graphic codes (i.e., code values 20h through 7Eh) and shall be terminated with a NULL (00h) character.

The contents of the vendor specific information field is not defined in this standard.

7.8.4 CFA Profile Information VPD page

The CFA Profile Information VPD page (see table 520) provides information on the CFA profiles, if any, that are supported by the device server.

Table 520 — CFA Profile Information VPD page

Bit / Byte	7	6	5	4	3	2	1	0
0	PERIPHERAL QUALIFIER			PERIPHERAL DEVICE TYPE				
1	PAGE CODE (8Ch)							
2	(MSB)	PAGE LENGTH (n-3)						
3								(LSB)
	CFA profile descriptor list							
4 ••• 7	CFA profile descriptor (see table 521) [first]							
	⋮							
n-3 ••• n	CFA profile descriptor (see table 521) [last]							

The PERIPHERAL QUALIFIER field, PERIPHERAL DEVICE TYPE field, and PAGE LENGTH field are defined in 7.8.2.

The PAGE CODE field is defined in 7.8.2 and shall be set as shown in table 520 for the CFA Profile Information VPD page.

Each CFA profile descriptor (see table 521) contains identifying information for one CFA profile supported by the device server described in the standard INQUIRY data (see 6.4.2).

Table 521 — CFA profile descriptor

Bit / Byte	7	6	5	4	3	2	1	0
0	CFA PROFILE SUPPORTED							
1	Reserved							
2	(MSB)	SEQUENTIAL WRITE DATA SIZE						
3								(LSB)

The CFA PROFILE SUPPORTED field indicates a CFA profile number (see VPG1) that the device server supports.

The SEQUENTIAL WRITE DATA SIZE field indicates the preferred number of logical blocks in a stream field write operation (see VPG1) for the CFA profile indicated by the CFA PROFILE SUPPORTED field. If the SEQUENTIAL WRITE DATA SIZE field is set to zero, then no preferred number of logical blocks is indicated for stream field write operations by this CFA profile descriptor.

7.8.5 Device Constituents VPD page

7.8.5.1 Device Constituents VPD page overview

The Device Constituents VPD page (see table 522) provides identifying information for the constituents (e.g., logical units in other SCSI devices, microcode, storage medium, or vendor specific information technology components) of the logical unit described in the standard INQUIRY data (see 6.4.2).

Table 522 — Device Constituents VPD page

Bit / Byte	7	6	5	4	3	2	1	0
0	PERIPHERAL QUALIFIER			PERIPHERAL DEVICE TYPE				
1	PAGE CODE (8Bh)							
2	(MSB)	PAGE LENGTH (n-3)						
3								(LSB)
	Constituent descriptor list							
4	Constituent descriptor (see table 523) [first]							
•••								
	⋮							
	Constituent descriptor (see table 523) [last]							
•••								
n								

The PERIPHERAL QUALIFIER field, PERIPHERAL DEVICE TYPE field, and PAGE LENGTH field are defined in 7.8.2.

The PAGE CODE field is defined in 7.8.2 and shall be set as shown in table 522 for the Device Constituents VPD page.

Each constituent descriptor (see table 523) contains identifying information for one constituent of the logical unit described in the standard INQUIRY data (see 6.4.2).

Table 523 — Constituent descriptor

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)			CONSTITUENT TYPE				
1								(LSB)
2	CONSTITUENT DEVICE TYPE							
3	Reserved							
4	(MSB)			T10 VENDOR IDENTIFICATION				
•••								
11								(LSB)
12	(MSB)			PRODUCT IDENTIFICATION				
•••								
27								(LSB)
28	(MSB)			PRODUCT REVISION LEVEL				
•••								
31								(LSB)
32				Reserved				
33								
34	(MSB)			CONSTITUENT DESCRIPTOR LENGTH (n-35)				
35								(LSB)
	Constituent type specific descriptor list							
36				Constituent type specific descriptor [first]				
•••								
				⋮				
				Constituent type specific descriptor [last]				
•••								
n								

The CONSTITUENT TYPE field (see table 524) indicates the constituent descriptor type.

Table 524 — CONSTITUENT TYPE field

Code	Description	Reference
000h	Reserved	
001h	Virtual tape library	7.8.5.2
002h	Virtual tape drive	7.8.5.3
0003h to FFFFh	Reserved	

The CONSTITUENT DEVICE TYPE field (see table 525) indicates the constituent descriptor type.

Table 525 — CONSTITUENT DEVICE TYPE field

Code	Description	Reference
00h to 1Fh	One of the values defined for the PERIPHERAL DEVICE TYPE field in the standard INQUIRY data	6.4.2
20h to FEh	Reserved	
FFh	Unknown	

The T10 VENDOR IDENTIFICATION field, PRODUCT IDENTIFICATION field, and the PRODUCT REVISION LEVEL field are identifying information for one constituent of the logical unit described in the standard INQUIRY data, and are as defined in 6.4.2.

The CONSTITUENT DESCRIPTOR LENGTH field indicates the length of the constituent type specific descriptor list.

Each constituent type specific descriptor contains information that is specific to the constituent type. The format of each constituent type specific descriptor is defined by the constituent type (see table 524).

7.8.5.2 Virtual tape library constituent type specific descriptor

If the CONSTITUENT TYPE field (see 7.8.5.1) is set to 01h (i.e., virtual tape library), then the format for each constituent type specific descriptor (see 7.8.5.1) is defined in table 526.

Table 526 — Virtual tape library constituent type specific descriptor format

Bit / Byte	7	6	5	4	3	2	1	0
0	VIRTUAL TAPE LIBRARY SPECIFIC TYPE							
1	Reserved							
2								
3	ADDITIONAL LENGTH (n-3)							
4	Virtual tape library specific data							
•••								
n								

The VIRTUAL TAPE LIBRARY SPECIFIC TYPE field (see table 527) indicates the type of virtual tape library specific data in this descriptor.

Table 527 — VIRTUAL TAPE LIBRARY SPECIFIC TYPE field

Code	Description
00h to FEh	Reserved
FFh	Vendor specific

The ADDITIONAL LENGTH field indicates the number of bytes that follow in this descriptor.

The contents of the virtual tape library specific data depend on the value in the VIRTUAL TAPE LIBRARY SPECIFIC TYPE field.

7.8.5.3 Virtual tape drive constituent type specific descriptor

If the CONSTITUENT TYPE field (see 7.8.5.1) is set to 02h (i.e., virtual tape drive), then the format for each constituent type specific descriptor (see 7.8.5.1) is defined in table 528.

Table 528 — Virtual tape drive constituent type specific descriptor format

Bit / Byte	7	6	5	4	3	2	1	0
0	VIRTUAL TAPE DRIVE SPECIFIC TYPE							
1	Reserved							
2								
3	ADDITIONAL LENGTH (n-3)							
4	Virtual tape drive specific data							
•••								
n								

The VIRTUAL TAPE DRIVE SPECIFIC TYPE field (see table 529) indicates the type of virtual tape drive specific data in this descriptor.

Table 529 — VIRTUAL TAPE DRIVE SPECIFIC TYPE field

Code	Description
00h to FEh	Reserved
FFh	Vendor specific

The ADDITIONAL LENGTH field indicates the number of bytes that follow in this descriptor.

The contents of the virtual tape drive specific data depend on the value in the VIRTUAL TAPE DRIVE SPECIFIC TYPE field.

7.8.6 Device Identification VPD page

7.8.6.1 Device Identification VPD page overview

The Device Identification VPD page (see table 530) provides the means to retrieve designation descriptors applying to the logical unit. Logical units may have more than one designation descriptor (e.g., if several types or associations of designator are supported). Designators consist of one or more of the following:

a) logical unit names;
b) SCSI target port identifiers;
c) SCSI target port names;
d) SCSI device names;
e) relative target port identifiers;
f) primary target port group number; or
g) logical unit group number.

Designation descriptors shall be assigned to the peripheral device (e.g., a disk drive) and not to the currently mounted media, in the case of removable media devices. Operating systems are expected to use the designation

descriptors during system configuration activities to determine whether alternate paths exist for the same peripheral device.

Table 530 — Device Identification VPD page

Bit / Byte	7	6	5	4	3	2	1	0
0	PERIPHERAL QUALIFIER			PERIPHERAL DEVICE TYPE				
1	PAGE CODE (83h)							
2	(MSB)	PAGE LENGTH (n-3)						
3								(LSB)
	Designation descriptor list							
4 •••	Designation descriptor [first]							
	⋮							
••• n	Designation descriptor [last]							

The PERIPHERAL QUALIFIER field, PERIPHERAL DEVICE TYPE field, and PAGE LENGTH field are defined in 7.8.2.

The PAGE CODE field is defined in 7.8.2 and shall be set as shown in table 530 for the Device Identification VPD page.

Each designation descriptor (see table 531) contains information identifying the logical unit, SCSI target device containing the logical unit, or access path (i.e., target port) used by the command and returned parameter data. The Device Identification VPD page shall contain the designation descriptors enumerated in 7.8.6.2.

Table 531 — Designation descriptor

Bit / Byte	7	6	5	4	3	2	1	0
0	PROTOCOL IDENTIFIER				CODE SET			
1	PIV	Reserved	ASSOCIATION		DESIGNATOR TYPE			
2	Reserved							
3	DESIGNATOR LENGTH (n-3)							
4 ••• n	DESIGNATOR							

The PROTOCOL IDENTIFIER field may indicate the SCSI transport protocol to which the designation descriptor applies. If the ASSOCIATION field contains a value other than 01b (i.e., target port) or 10b (i.e., SCSI target device) or the PIV bit is set to zero, then the PROTOCOL IDENTIFIER field contents are reserved. If the ASSOCIATION field contains a value of 01b or 10b and the PIV bit is set to one, then the PROTOCOL IDENTIFIER field shall contain one of the values shown in table 424 (see 7.6.1) to indicate the SCSI transport protocol to which the designation descriptor applies.

The CODE SET field contains a code set enumeration (see 4.4.3) that indicates the format of the DESIGNATOR field.

A protocol identifier valid (PIV) bit set to zero indicates the PROTOCOL IDENTIFIER field contents are reserved. If the ASSOCIATION field contains a value of 01b or 10b, then a PIV bit set to one indicates the PROTOCOL IDENTIFIER field contains a valid protocol identifier selected from the values shown in table 424 (see 7.6.1). If the ASSOCIATION field contains a value other than 01b or 10b, then the PIV bit contents are reserved.

The ASSOCIATION field indicates the entity with which the DESIGNATOR field is associated, as described in table 532. If a logical unit returns a designation descriptor with the ASSOCIATION field set to 00b or 10b, it shall return the same descriptor when it is accessed through any other I_T nexus.

Table 532 — ASSOCIATION field

Code	Description
00b	The DESIGNATOR field is associated with the addressed logical unit.
01b	The DESIGNATOR field is associated with the target port that received the request.
10b	The DESIGNATOR field is associated with the SCSI target device that contains the addressed logical unit.
11b	Reserved

The DESIGNATOR TYPE field (see table 533) indicates the format and assignment authority for the designator.

Table 533 — DESIGNATOR TYPE field

Code	Description	Reference
0h	Vendor specific	7.8.6.3
1h	T10 vendor ID based	7.8.6.4
2h	EUI-64 based	7.8.6.5
3h	NAA	7.8.6.6
4h	Relative target port identifier	7.8.6.7
5h	Target port group	7.8.6.8
6h	Logical unit group	7.8.6.9
7h	MD5 logical unit identifier	7.8.6.10
8h	SCSI name string	7.8.6.11
9h to Fh	Reserved	

The DESIGNATOR LENGTH field indicates the length in bytes of the DESIGNATOR field. The relationship between the DESIGNATOR LENGTH field and the CDB ALLOCATION LENGTH field is defined in 4.3.5.6.

The DESIGNATOR field contains the designator as described by the ASSOCIATION, DESIGNATOR TYPE, CODE SET, and DESIGNATOR LENGTH fields.

7.8.6.2 Device designation descriptor requirements

7.8.6.2.1 Designation descriptors for logical units other than well known logical units

For each logical unit that is not a well known logical unit, the Device Identification VPD page shall include at least one designation descriptor in which a logical unit name (see SAM-4) is indicated. The designation descriptor shall have the ASSOCIATION field set to 00b (i.e., logical unit) and the DESIGNATOR TYPE field set to:

a) 1h (i.e., T10 vendor ID based);
b) 2h (i.e., EUI-64-based);
c) 3h (i.e., NAA); or
d) 8h (i.e., SCSI name string).

At least one designation descriptor should have the DESIGNATOR TYPE field set to:

a) 2h (i.e., EUI-64-based);
b) 3h (i.e., NAA); or
c) 8h (i.e., SCSI name string).

In the case of virtual logical units (e.g., volume sets as defined by SCC-2), designation descriptors should contain a DESIGNATOR TYPE field set to:

a) 2h (i.e., EUI-64-based);
b) 3h (i.e., NAA); or
c) 8h (i.e., SCSI name string).

In the case of virtual logical units that have an EUI-64 based designation descriptor (see 7.8.6.5) the DESIGNATOR LENGTH field should be set to:

a) 0Ch (i.e., EUI-64-based 12-byte); or
b) 10h (i.e., EUI-64-based 16-byte).

In the case of virtual logical units that have an NAA designation descriptor (see 7.8.6.6) the NAA field should be set to 6h (i.e., IEEE Registered Extended).

The Device Identification VPD page shall contain the same set of designation descriptors with the ASSOCIATION field set to 00b (i.e., logical unit) regardless of the I_T nexus being used to retrieve the designation descriptors.

For logical units that are not well known logical units, the requirements for SCSI target device designation descriptors are defined in 7.8.6.2.4 and the requirements for SCSI target port designation descriptors are defined in 7.8.6.2.3.

7.8.6.2.2 Designation descriptors for well known logical units

Well known logical units shall not return any designation descriptors with the ASSOCIATION field set to 00b (i.e., logical unit).

The Device Identification VPD page shall contain the same set of designation descriptors with the ASSOCIATION field set to 10b (i.e., SCSI target device) regardless of the I_T nexus being used to retrieve the designation descriptors.

7.8.6.2.3 Designation descriptors for SCSI target ports

7.8.6.2.3.1 Relative target port identifiers

For the target port through which the Device Identification VPD page is accessed, the Device Identification VPD page should include one designation descriptor with the ASSOCIATION field set to 01b (i.e., target port) and the DESIGNATOR TYPE field set to 4h (i.e., relative target port identifier) identifying the target port being used to retrieve the designation descriptors.

7.8.6.2.3.2 Target port names or identifiers

For the SCSI target port through which the Device Identification VPD page is accessed, the Device Identification VPD page should include one designation descriptor in which the target port name or identifier (see SAM-4) is indicated. The designation descriptor, if any, shall have the ASSOCIATION field set to 01b (i.e., target port) and the DESIGNATOR TYPE field set to:

a) 2h (i.e., EUI-64-based);
b) 3h (i.e., NAA); or
c) 8h (i.e., SCSI name string).

If the SCSI transport protocol standard (see 3.1.147) for the target port defines target port names, the designation descriptor, if any, shall contain the target port name. If the SCSI transport protocol for the target port does not define target port names, the designation descriptor, if any, shall contain the target port identifier.

7.8.6.2.4 Designation descriptors for SCSI target devices

If the SCSI target device contains a well known logical unit, the Device Identification VPD page shall have one or more designation descriptors for the SCSI target device. If the SCSI target device does not contain a well known logical unit, the Device Identification VPD page should have one or more designation descriptors for the SCSI target device.

Each SCSI target device designation descriptor, if any, shall have the ASSOCIATION field set to 10b (i.e., SCSI target device) and the DESIGNATOR TYPE field set to:

a) 2h (i.e., EUI-64-based);
b) 3h (i.e., NAA); or
c) 8h (i.e., SCSI name string).

The Device Identification VPD page shall contain designation descriptors, if any, for all the SCSI device names for all the SCSI transport protocols supported by the SCSI target device.

7.8.6.3 Vendor specific designator format

If the designator type is 0h (i.e., vendor specific), no assignment authority was used and there is no guarantee that the designator is globally unique (i.e., the identifier is vendor specific). Table 534 defines the DESIGNATOR field format.

Table 534 — Vendor specific DESIGNATOR field format

Bit / Byte	7	6	5	4	3	2	1	0
0	VENDOR SPECIFIC IDENTIFIER							
•••								
n								

7.8.6.4 T10 vendor ID based designator format

If the designator type is 1h (i.e., T10 vendor ID based), the DESIGNATOR field has the format shown in table 535.

Table 535 — T10 vendor ID based DESIGNATOR field format

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)	T10 VENDOR IDENTIFICATION						
•••								
7								(LSB)
8	VENDOR SPECIFIC IDENTIFIER							
•••								
n								

The T10 VENDOR IDENTIFICATION field contains eight bytes of left-aligned ASCII data (see 4.4.1) identifying the vendor of the product. The data shall be left aligned within this field.The T10 vendor identification shall be one assigned by INCITS. A list of assigned T10 vendor identifications is in Annex E and on the T10 web site (http://www.T10.org).

NOTE 71 - The T10 web site (http://www.t10.org) provides a convenient means to request an identification code.

The organization associated with the T10 vendor identification is responsible for ensuring that the VENDOR SPECIFIC DESIGNATOR field is unique in a way that makes the entire DESIGNATOR field unique. A recommended method of constructing a unique DESIGNATOR field is to concatenate the PRODUCT IDENTIFICATION field from the standard INQUIRY data (see 6.4.2) and the PRODUCT SERIAL NUMBER field from the Unit Serial Number VPD page (see 7.8.16).

7.8.6.5 EUI-64 based designator format

7.8.6.5.1 EUI-64 based designator format overview

If the designator type is 2h (i.e., EUI-64 based identifier), the DESIGNATOR LENGTH field (see table 536) indicates the format of the designation descriptor.

Table 536 — EUI-64 based designator lengths

Designator Length	Description	Reference
08h	EUI-64 identifier	7.8.6.5.2
0Ch	EUI-64 based 12-byte identifier	7.8.6.5.3
10h	EUI-64 based 16-byte identifier	7.8.6.5.4
All other values	Reserved	

7.8.6.5.2 EUI-64 designator format

If the designator type is 2h (i.e., EUI-64 based identifier) and the DESIGNATOR LENGTH field is set to 08h, the DESIGNATOR field has the format shown in table 537. The CODE SET field shall be set to 1h (i.e., binary).

Table 537 — EUI-64 DESIGNATOR field format

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)							
1				IEEE COMPANY_ID				
2								(LSB)
3	(MSB)							
•••				VENDOR SPECIFIC EXTENSION IDENTIFIER				
7								(LSB)

The IEEE COMPANY_ID field contains a 24 bit OUI (see 3.1.103) assigned by the IEEE.

The VENDOR SPECIFIC EXTENSION IDENTIFIER field contains a 40 bit numeric value that is uniquely assigned by the organization associated with the IEEE company_id as required by the IEEE definition of EUI-64 (see 3.1.57).

7.8.6.5.3 EUI-64 based 12-byte designator format

If the designator type is 2h (i.e., EUI-64 based identifier) and the DESIGNATOR LENGTH field is set to 0Ch, the DESIGNATOR field has the format shown in table 538. The CODE SET field shall be set to 1h (i.e., binary).

Table 538 — EUI-64 based 12-byte DESIGNATOR field format

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)							
1				IEEE COMPANY_ID				
2								(LSB)
3	(MSB)							
•••				VENDOR SPECIFIC EXTENSION IDENTIFIER				
7								(LSB)
8	(MSB)							
•••				DIRECTORY ID				
11								(LSB)

The IEEE COMPANY_ID field and VENDOR SPECIFIC EXTENSION IDENTIFIER field are defined in 7.8.6.5.2.

The DIRECTORY ID field contains a directory identifier, as specified by ISO/IEC 13213:1994.

NOTE 72 - The EUI-64 based 12 byte format may be used to report IEEE 1394 target port identifiers (see SBP-3).

7.8.6.5.4 EUI-64 based 16-byte designator format

If the designator type is 2h (i.e., EUI-64 based identifier) and the DESIGNATOR LENGTH field is set to 10h, the DESIGNATOR field has the format shown in table 539. The CODE SET field shall be set to 1h (i.e., binary).

Table 539 — EUI-64 based 16-byte DESIGNATOR field format

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)							
•••				IDENTIFIER EXTENSION				
7								(LSB)
8	(MSB)							
•••				IEEE COMPANY_ID				
10								(LSB)
11	(MSB)							
•••				VENDOR SPECIFIC EXTENSION IDENTIFIER				
15								(LSB)

The IDENTIFIER EXTENSION field contains a 64 bit numeric value.

The IEEE COMPANY_ID field and VENDOR SPECIFIC EXTENSION IDENTIFIER field are defined in 7.8.6.5.2.

NOTE 73 - The EUI-64 based 16-byte format may be used to report SCSI over RDMA target port identifiers (see SRP).

7.8.6.6 NAA designator format

7.8.6.6.1 NAA identifier basic format

If the designator type is 3h (i.e., NAA identifier), the DESIGNATOR field has the format shown in table 540. This format is compatible with the Name_Identifier format defined in FC-FS.

Table 540 — NAA DESIGNATOR field format

Bit / Byte	7	6	5	4	3	2	1	0
0	NAA							
1	NAA specific data							
•••								
n								

The Name Address Authority (NAA) field (see table 541) defines the format of the NAA specific data in the designator.

Table 541 — Name Address Authority (NAA) field

Code	Description	Reference
2h	IEEE Extended	7.8.6.6.2
3h	Locally Assigned	7.8.6.6.3
5h	IEEE Registered	7.8.6.6.4
6h	IEEE Registered Extended	7.8.6.6.5
All others	Reserved	

7.8.6.6.2 NAA IEEE Extended designator format

If NAA is 2h (i.e., IEEE Extended), the eight byte fixed length DESIGNATOR field shall have the format shown in table 542. The CODE SET field shall be set to 1h (i.e., binary) and the DESIGNATOR LENGTH field shall be set to 08h.

Table 542 — NAA IEEE Extended DESIGNATOR field format

Bit / Byte	7	6	5	4	3	2	1	0
0	NAA (2h)				(MSB)			
1	VENDOR SPECIFIC IDENTIFIER A							(LSB)
2	(MSB)	IEEE COMPANY_ID						
3								
4								(LSB)
5	(MSB)	VENDOR SPECIFIC IDENTIFIER B						
6								
7								(LSB)

The IEEE COMPANY_ID field contains a 24 bit canonical form OUI (see 3.1.103) assigned by the IEEE.

The VENDOR SPECIFIC IDENTIFIER A contains a 12 bit numeric value that is uniquely assigned by the organization associated with the IEEE company_id.

The VENDOR SPECIFIC IDENTIFIER B contains a 24 bit numeric value that is uniquely assigned by the organization associated with the IEEE company_id.

NOTE 74 - The EUI-64 format includes a 40 bit vendor specific identifier. The IEEE Extended format includes 36 bits vendor specific identifier in two fields.

7.8.6.6.3 NAA Locally Assigned designator format

If NAA is 3h (i.e., Locally Assigned), the eight byte fixed length DESIGNATOR field shall have the format shown in table 543. The CODE SET field shall be set to 1h (i.e., binary) and the DESIGNATOR LENGTH field shall be set to 08h.

Table 543 — NAA Locally Assigned DESIGNATOR field format

Bit / Byte	7	6	5	4	3	2	1	0
0	NAA (3h)							
1	LOCALLY ADMINISTERED VALUE							
•••								
7								

The LOCALLY ADMINISTERED VALUE field contains a 60 bit value that is assigned by an administrator to be unique within the set of SCSI domains that are accessible by a common instance of an administrative tool or tools.

7.8.6.6.4 NAA IEEE Registered designator format

If NAA is 5h (i.e., IEEE Registered), the eight byte fixed length DESIGNATOR field shall have the format shown in table 544. The CODE SET field shall be set to 1h (i.e., binary) and the DESIGNATOR LENGTH field shall be set to 08h.

Table 544 — NAA IEEE Registered DESIGNATOR field format

Bit / Byte	7	6	5	4	3	2	1	0
0	NAA (5h)				(MSB)			
1	IEEE COMPANY_ID							
2								
3				(LSB)	(MSB)			
4	VENDOR SPECIFIC IDENTIFIER							
•••								
7								(LSB)

The IEEE COMPANY_ID field contains a 24 bit canonical form OUI (see 3.1.103) assigned by the IEEE.

The VENDOR SPECIFIC IDENTIFIER a 36 bit numeric value that is uniquely assigned by the organization associated with the IEEE company_id.

NOTE 75 - The EUI-64 identifier includes a 40 bit vendor specific identifier. The IEEE Registered format includes a 36 bit vendor specific identifier.

7.8.6.6.5 NAA IEEE Registered Extended designator format

If NAA is 6h (i.e., IEEE Registered Extended), the sixteen byte fixed length DESIGNATOR field shall have the format shown in table 545. The CODE SET field shall be set to 1h (i.e., binary) and the DESIGNATOR LENGTH field shall be set to 10h.

Table 545 — NAA IEEE Registered Extended DESIGNATOR field format

Bit / Byte	7	6	5	4	3	2	1	0
0	NAA (6h)				(MSB)			
1	IEEE COMPANY_ID							
2								
3				(LSB)	(MSB)			
4	VENDOR SPECIFIC IDENTIFIER							
•••								
7								(LSB)
8	(MSB)	VENDOR SPECIFIC IDENTIFIER EXTENSION						
•••								
15								(LSB)

The IEEE COMPANY_ID field contains a 24 bit canonical form OUI (see 3.1.103) assigned by the IEEE.

The VENDOR SPECIFIC IDENTIFIER field contains a 36 bit numeric value that is uniquely assigned by the organization associated with the IEEE company_id.

NOTE 76 - The EUI-64 format includes a 40 bit vendor specific identifier. The IEEE Registered Extended format includes a 36 bit vendor specific identifier.

The VENDOR SPECIFIC IDENTIFIER EXTENSION a 64 bit numeric value that is assigned to make the DESIGNATOR field unique.

7.8.6.7 Relative target port designator format

If the designator type is 4h (i.e., relative target port identifier) and the ASSOCIATION field contains 01b (i.e., target port), then the DESIGNATOR field shall have the format shown in table 546. The CODE SET field shall be set to 1h (i.e., binary) and the DESIGNATOR LENGTH field shall be set to 04h. If the ASSOCIATION field does not contain 01b, use of this designator type is reserved.

Table 546 — Relative target port DESIGNATOR field format

Bit / Byte	7	6	5	4	3	2	1	0
0	Reserved							
1								
2	(MSB)	RELATIVE TARGET PORT IDENTIFIER						
3								(LSB)

The RELATIVE TARGET PORT IDENTIFIER field (see table 547) contains the relative port identifier (see 3.1.127) of the target port on which the INQUIRY command was received.

Table 547 — RELATIVE TARGET PORT IDENTIFIER field

Code	Description
0h	Reserved
1h	Relative port 1, historically known as port A
2h	Relative port 2, historically known as port B
3h to FFFFh	Relative port 3 through 65 535

7.8.6.8 Target port group designator format

If the designator type is 5h (i.e., target port group) and the ASSOCIATION value is 01b (i.e., target port), the four byte fixed length DESIGNATOR field shall have the format shown in table 548. The CODE SET field shall be set to 1h (i.e., binary) and the DESIGNATOR LENGTH field shall be set to 04h. If the ASSOCIATION field does not contain 01b, use of this designator type is reserved.

Table 548 — Target port group DESIGNATOR field format

Bit / Byte	7	6	5	4	3	2	1	0
0	Reserved							
1								
2	(MSB)	TARGET PORT GROUP						
3								(LSB)

The TARGET PORT GROUP field indicates the primary target port group to which the target port is a member (see 5.11).

7.8.6.9 Logical unit group designator format

A logical unit group is a group of logical units that share the same primary target port group (see 5.11) definitions. The primary target port groups maintain the same primary target port group asymmetric access states for all logical units in the same logical unit group. A logical unit shall be in no more than one logical unit group.

If the designator type is 6h (i.e., logical unit group) and the ASSOCIATION value is 00b (i.e., logical unit), the four byte fixed length DESIGNATOR field shall have the format shown in table 549. The CODE SET field shall be set to 1h (i.e., binary) and the DESIGNATOR LENGTH field shall be set to 04h. If the ASSOCIATION field does not contain 00b, use of this designator type is reserved.

Table 549 — Logical unit group DESIGNATOR field format

Bit / Byte	7	6	5	4	3	2	1	0
0	Reserved							
1								
2	(MSB)	LOGICAL UNIT GROUP						
3								(LSB)

The LOGICAL UNIT GROUP field indicates the logical unit group to which the logical unit is a member.

7.8.6.10 MD5 logical unit designator format

If the designator type is 7h (i.e., MD5 logical unit identifier) and the ASSOCIATION value is 00b (i.e., logical unit), the DESIGNATOR field has the format shown in table 550. The CODE SET field shall be set to 1h (i.e., binary). The MD5 logical unit designator shall not be used if a logical unit provides unique identification using designator types 2h (i.e., EUI-64 based identifier), 3h (i.e., NAA identifier), or 8h (i.e., SCSI name string). A bridge device may return a MD5 logical unit designator type for that logical unit that does not support the Device Identification VPD page (see 7.8.6).

If the ASSOCIATION field does not contain 00b, use of this designator type is reserved.

Table 550 — MD5 logical unit DESIGNATOR field format

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)							
•••	MD5 LOGICAL UNIT IDENTIFIER							
15								(LSB)

The MD5 LOGICAL UNIT IDENTIFIER field contains the message digest of the supplied message input. The message digest shall be generated using the MD5 message-digest algorithm as specified in RFC 1321 (see 2.5) with the following information as message input:

1) the contents of the T10 VENDOR IDENTIFICATION field in the standard INQUIRY data (see 6.4.2);
2) the contents of the PRODUCT IDENTIFICATION field in the standard INQUIRY data;
3) the contents of the PRODUCT SERIAL NUMBER field in the Unit Serial Number VPD page (see 7.8.16);
4) the contents of a vendor specific DESIGNATOR field (designator type 0h) from the Device Identification VPD page; and
5) the contents of a T10 vendor ID based DESIGNATOR field (designator type 1h) from the Device Identification VPD page.

If a field or page is not available, the message input for that field or page shall be 8 bytes of ASCII space characters (i.e., 20h).

The uniqueness of the MD5 logical unit identifier is dependent upon the relative degree of randomness (i.e., the entropy) of the message input. If it is found that two or more logical units have the same MD5 logical unit identifier, the application client should determine in a vendor specific manner whether the logical units are the same entities.

The MD5 logical unit identifier example described in this paragraph and shown in table 551 and table 552 is not a normative part of this standard. The data available for input to the MD5 algorithm for this example is shown in table 551.

Table 551 — MD5 logical unit identifier example available data

MD5 message input	Available	Contents
T10 VENDOR IDENTIFICATION field	Yes	T10
PRODUCT IDENTIFICATION field	Yes	MD5 Logical Unit
PRODUCT SERIAL NUMBER field	Yes	01234567

Table 551 — MD5 logical unit identifier example available data

MD5 message input	Available	Contents
vendor specific DESIGNATOR field	No	
T10 vendor ID based DESIGNATOR field	No	

The concatenation of the fields in table 551 to form input to the MD5 algorithm is shown in table 552.

Table 552 — Example MD5 input for computation of a logical unit identifier

Bytes	Hexadecimal values				ASCII values
00 to 15	54 31 30 20	20 20 20 20	4D 44 35 20	4C 6F 67 69	T10 MD5 Logi
16 to 31	63 61 6C 20	55 6E 69 74	30 31 32 33	34 35 36 37	cal Unit01234567
32 to 47	20 20 20 20	20 20 20 20	20 20 20 20	20 20 20 20	
NOTE 1 Non-printing ASCII characters are shown as '.'.					

Based on the example inputs shown in table 551 and the concatenation of the inputs shown in table 552, the MD5 base 16 algorithm described in RFC 1321 produces the value 8FAC A22A 0AC0 3839 1255 25F2 0EFE 2E7Eh.

7.8.6.11 SCSI name string designator format

If the designator type is 8h (i.e., SCSI name string), the DESIGNATOR field has the format shown in table 553. The CODE SET field shall be set to 3h (i.e., UTF-8).

Table 553 — SCSI name string DESIGNATOR field format

Bit / Byte	7	6	5	4	3	2	1	0
0	SCSI NAME STRING							
•••								
n								

The null-terminated, null-padded (see 4.4.2) SCSI NAME STRING field contains a UTF-8 format string. The number of bytes in the SCSI NAME STRING field (i.e., the value in the DESIGNATOR LENGTH field) shall be no larger than 256 and shall be a multiple of four.

The SCSI NAME STRING field starts with either:

a) the four UTF-8 characters 'eui.' concatenated with 16, 24, or 32 hexadecimal digits (i.e., the UTF-8 characters 0 through 9 and A through F) for an EUI-64 based identifier (see 7.8.6.5). The first hexadecimal digit shall be the most significant four bits of the first byte (i.e., most significant byte) of the EUI-64 based identifier;
b) the four UTF-8 characters 'naa.' concatenated with 16 or 32 hexadecimal digits for an NAA identifier (see 7.8.6.6). The first hexadecimal digit shall be the most significant four bits of the first byte (i.e., most significant byte) of the NAA identifier; or
c) the four UTF-8 characters 'iqn.' concatenated with an iSCSI Name for an iSCSI-name based identifier (see iSCSI)."

If the ASSOCIATION field is set to 00b (i.e., logical unit) and the SCSI NAME STRING field starts with the four UTF-8 characters 'iqn.', the SCSI NAME STRING field ends with the five UTF-8 characters ',L,0x' concatenated with 16 hexadecimal digits for the logical unit name extension. The logical unit name extension is a UTF-8 string containing

no more than 16 hexadecimal digits. The logical unit name extension is assigned by the SCSI target device vendor and shall be assigned so the logical unit name is worldwide unique.

If the ASSOCIATION field is set to 01b (i.e., target port), the SCSI NAME STRING field ends with the five UTF-8 characters ',t,0x' concatenated with two or more hexadecimal digits as specified in the applicable SCSI transport protocol standard (see 3.1.147).

If the ASSOCIATION field is set to 10b (i.e., SCSI target device), the SCSI NAME STRING field has no additional characters.

NOTE 77 - The notation used in this subclause to specify exact UTF-8 character strings is described in 3.5.3.

7.8.7 Extended INQUIRY Data VPD page

The Extended INQUIRY Data VPD page (see table 554) provides the application client with a means to obtain information about the logical unit.

Table 554 — Extended INQUIRY Data VPD page

Bit / Byte	7	6	5	4	3	2	1	0
0	PERIPHERAL QUALIFIER			PERIPHERAL DEVICE TYPE				
1	PAGE CODE (86h)							
2	(MSB)	PAGE LENGTH (003Ch)						
3								(LSB)
4	ACTIVATE MICROCODE		SPT			GRD_CHK	APP_CHK	REF_CHK
5	Reserved		UASK_SUP	GROUP_SUP	PRIOR_SUP	HEADSUP	ORDSUP	SIMPSUP
6	Reserved				WU_SUP	CRD_SUP	NV_SUP	V_SUP
7	Reserved			P_I_I_SUP	Reserved			LUICLR
8	Reserved			R_SUP	Reserved			CBCS
9	Reserved				MULTI I_T NEXUS MICROCODE DOWNLOAD			
10	(MSB)	EXTENDED SELF-TEST COMPLETION MINUTES						
11								(LSB)
12	Reserved							
•••								
63								

The PERIPHERAL QUALIFIER field and PERIPHERAL DEVICE TYPE field are defined in 7.8.2.

The PAGE CODE field and PAGE LENGTH field are defined in 7.8.2, and shall be set to the value shown in table 554 for the Extended INQUIRY Data VPD page.

The ACTIVATE MICROCODE field (see table 555) indicates how a device server activates microcode and establishes a unit attention condition when a WRITE BUFFER command (see 6.39) with the download microcode mode set to 05h or 07h is processed (see 5.18).

Table 555 — ACTIVATE MICROCODE field

Value	Meaning
00b	The actions of the device server may or may not be as defined for values 01b or 10b.
01b	The device server: 1) activates the microcode before completion of the final command in the WRITE BUFFER sequence; and 2) establishes a unit attention condition for the initiator port associated with every I_T nexus, except the I_T nexus on which the WRITE BUFFER command was received, with the additional sense code set to MICROCODE HAS BEEN CHANGED.
10b	The device server: 1) activates the microcode after: A) a vendor specific event; B) a power on event; or C) a hard reset event; and 2) establishes a unit attention condition for the initiator port associated with every I_T nexus with the additional sense code set to MICROCODE HAS BEEN CHANGED.
11b	Reserved

A supported protection type (SPT) field indicates the type of protection the logical unit supports based on the contents of the PERIPHERAL DEVICE TYPE field. If the PROTECT bit (see 6.4.2) is set to zero, the SPT field is reserved.

If the PERIPHERAL DEVICE TYPE field is set to 00h (i.e., direct access block device) and the PROTECT bit is set to one, then table 556 defines the contents of the SPT field.

Table 556 — SPT field for peripheral device type 00h

Code	Protection type supported		
	Type 1	Type 2	Type 3
000b	yes	no	no
001b	yes	yes	no
010b	no	yes	no
011b	yes	no	yes
100b	no	no	yes
101b	no	yes	yes
110b	Reserved		
111b	yes	yes	yes

If the PERIPHERAL DEVICE TYPE field is set to 01h (i.e., sequential-access device) and the PROTECT bit is set to one, then table 557 defines the contents of the SPT field.

Table 557 — SPT field for peripheral device type 01h

Code	Protection type supported
001b	Logical block protection
all others	Reserved

If the PROTECT bit is set to one and the PERIPHERAL DEVICE TYPE field is set to a value other than 00h or 01h, the SPT field is reserved.

A guard check (GRD_CHK) bit set to zero indicates that the device server does not check the LOGICAL BLOCK GUARD field in the protection information (see SBC-2), if any. A GRD_CHK bit set to one indicates that the device server checks the LOGICAL BLOCK GUARD field in the protection information, if any.

An application tag check (APP_CHK) bit set to zero indicates that the device server does not check the LOGICAL BLOCK APPLICATION TAG field in the protection information (see SBC-2), if any. An APP_CHK bit set to one indicates that the device server checks the LOGICAL BLOCK APPLICATION TAG field in the protection information, if any.

A reference tag check (REF_CHK) bit set to zero indicates that the device server does not check the LOGICAL BLOCK REFERENCE TAG field in the protection information (see SBC-2), if any. A REF_CHK bit set to one indicates that the device server checks the LOGICAL BLOCK REFERENCE TAG field in the protection information, if any.

A unit attention condition sense key specific data supported (UASK_SUP) bit set to one indicates that the device server returns sense-key specific data for the UNIT ATTENTION sense key (see 4.5.2.4.6). A UASK_SUP bit set to zero indicates that the device server does not return sense-key specific data for the UNIT ATTENTION sense key.

A grouping function supported (GROUP_SUP) bit set to one indicates that the grouping function (see SBC-2) is supported by the device server. A GROUP_SUP bit set to zero indicates that the grouping function is not supported.

A priority supported (PRIOR_SUP) bit set to one indicates that command priority (see SAM-4) is supported by the logical unit. A PRIOR_SUP bit set to zero indicates that command priority is not supported.

A head of queue supported (HEADSUP) bit set to one indicates that the HEAD OF QUEUE task attribute (see SAM-4) is supported by the logical unit. A HEADSUP bit set to zero indicates that the HEAD OF QUEUE task attribute is not supported. If the HEADSUP bit is set to zero, application clients should not specify the HEAD OF QUEUE task attribute as an Execute Command (see 4.2) procedure call argument.

An ordered supported (ORDSUP) bit set to one indicates that the ORDERED task attribute (see SAM-4) is supported by the logical unit. An ORDSUP bit set to zero indicates that the ORDERED task attribute is not supported. If the ORDSUP bit is set to zero, application clients should not specify the ORDERED task attribute as an Execute Command procedure call argument.

The simple supported (SIMPSUP) bit shall be set to one indicating that the SIMPLE task attribute (see SAM-4) is supported by the logical unit.

SAM-4 defines how unsupported task attributes are processed.

A write uncorrectable supported (WU_SUP) bit set to zero indicates that the device server does not support application clients setting the WR_UNCOR bit to one in the WRITE LONG command (see SBC-3). A WU_SUP bit set to one indicates that the device server supports application clients setting the WR_UNCOR bit to one in the WRITE LONG command.

A correction disable supported (CRD_SUP) bit set to zero indicates that the device server does not support application clients setting the COR_DIS bit to one in the WRITE LONG command (see SBC-3). A CRD_SUP bit set to one indicates that the device server supports application clients setting the COR_DIS bit to one in the WRITE LONG command.

A non-volatile cache supported (NV_SUP) bit set to one indicates that the device server supports a non-volatile cache (see 3.1.97) and that the applicable command standard (see 3.1.31) defines features using this cache (e.g., the FUA_NV bit in SBC-2). An NV_SUP bit set to zero indicates that the device server may or may not support a non-volatile cache.

A volatile cache supported (V_SUP) bit set to one indicates that the device server supports a volatile cache (see 3.1.191) and that the applicable command standard (see 3.1.31) defines features using this cache (e.g., the FUA bit in SBC-2). An V_SUP bit set to zero indicates that the device server may or may not support a volatile cache.

A protection information interval supported (P_I_I_SUP) bit set to one indicates that the logical unit supports protection information intervals (see SBC-3). A P_I_I_SUP bit set to zero indicates that the logical unit does not support protection information intervals.

A logical unit I_T nexus clear (LUICLR) bit set to one indicates the SCSI target device clears any unit attention condition with an additional sense code of REPORTED LUNS DATA HAS CHANGED in each logical unit accessible to an I_T nexus after reporting the unit attention condition for any logical unit over that I_T nexus (see SAM-4). An LUICLR bit set to zero indicates the SCSI target device clears unit attention conditions according to a previous version of this standard. The LUICLR bit shall be set to one.

A referrals supported (R_SUP) bit set to zero indicates that the device server does not support referrals (see SBC-3). An R_SUP bit set to one indicates that the device server supports referrals.

A capability-based command security (CBCS) bit set to one indicates that the logical unit supports the capability-based command security technique (see 5.16.6.8). A CBCS bit set to zero indicates that the logical unit does not support the capability-based command security technique.

The MULTI I_T NEXUS MICROCODE DOWNLOAD field (see table 96 in 5.18) indicates how the device server handles concurrent attempts to download microcode using the WRITE BUFFER command (see 5.18) from multiple I_T nexuses.

The EXTENDED SELF-TEST COMPLETION MINUTES field contains advisory data that is the time in minutes that the device server requires to complete an extended self-test when the device server is not interrupted by subsequent commands and no errors occur during processing of the self-test. The application client should expect the self-test completion time to exceed the value in this field if other commands are sent to the logical unit while a self-test is in progress or if errors occur during the processing of the self-test. If a device server supports SELF-TEST CODE field values other than 000b for the SEND DIAGNOSTIC command (see 6.32) and the self-test completion time is greater than 18 hours, then the device server shall support the EXTENDED SELF-TEST COMPLETION MINUTES field. A value of 0000h indicates that the EXTENDED SELF-TEST COMPLETION MINUTES field is not supported. A value of FFFFh indicates that the extended self-test takes 65 535 minutes or longer.

7.8.8 Management Network Addresses VPD page

The Management Network Addresses VPD page (see table 558) provides a list of network addresses of management services associated with a SCSI target device, target port, or logical unit.

Table 558 — Management Network Addresses VPD page

Bit / Byte	7	6	5	4	3	2	1	0
0	PERIPHERAL QUALIFIER			PERIPHERAL DEVICE TYPE				
1	PAGE CODE (85h)							
2	(MSB)	PAGE LENGTH (n-3)						
3								(LSB)
	Network services descriptor list							
4 •••		Network services descriptor [first]						
	⋮							
•••		Network services descriptor [last]						
n								

The PERIPHERAL QUALIFIER field, PERIPHERAL DEVICE TYPE field, and PAGE LENGTH field are defined in 7.8.2.

The PAGE CODE field is defined in 7.8.2 and shall be set as shown in table 558 for the Management Network Address VPD page.

Each network service descriptor (see table 559) contains information about one management service.

Table 559 — Network service descriptor format

Bit / Byte	7	6	5	4	3	2	1	0
0	Reserved	ASSOCIATION		SERVICE TYPE				
1	Reserved							
2	(MSB)	NETWORK ADDRESS LENGTH (n-3)						
3								(LSB)
4 •••		NETWORK ADDRESS						
n								

The ASSOCIATION field (see table 532 in 7.8.6.1) indicates the entity (i.e., SCSI target device, target port, or logical unit) with which the service is associated.

The SERVICE TYPE field (see table 560) allows differentiation of multiple services with the same protocol running at different port numbers or paths.

NOTE 78 - A SCSI target device may provide separate HTTP services for configuration and diagnostics. One of these services may use the standard HTTP port 80 (see 3.1.180) and the other service may use a different port (e.g., 8080).

Table 560 — SERVICE TYPE field

Code	Description
00h	Unspecified
01h	Storage Configuration Service
02h	Diagnostics
03h	Status
04h	Logging
05h	Code Download
06h	Copy Service
07h	Administrative Configuration Service
08h to 1Fh	Reserved

The NETWORK ADDRESS LENGTH field contains the length in bytes of the NETWORK ADDRESS field. The network address length shall be a multiple of four.

The null-terminated, null-padded NETWORK ADDRESS field contains the URL form of a URI as defined in RFC 2396.

7.8.9 Mode Page Policy VPD page

The Mode Page Policy VPD page (see table 561) indicates which mode page policy (see 6.9) is in effect for each mode page supported by the logical unit.

Table 561 — Mode Page Policy VPD page

Bit / Byte	7	6	5	4	3	2	1	0
0	PERIPHERAL QUALIFIER			PERIPHERAL DEVICE TYPE				
1	PAGE CODE (87h)							
2	(MSB)	PAGE LENGTH (n-3)						
3								(LSB)
	Mode page policy descriptor list							
4	Mode page policy descriptor [first]							
•••								
7								
	⋮							
n-3	Mode page policy descriptor [last]							
•••								
n								

The PERIPHERAL QUALIFIER field, PERIPHERAL DEVICE TYPE field, and PAGE LENGTH field are defined in 7.8.2.

The PAGE CODE field is defined in 7.8.2 and shall be set as shown in table 561 for the Mode Page Policy VPD page.

Each mode page policy descriptor (see table 562) contains information describing the mode page policy for one or more mode pages or subpages (see 7.5.6). The information in the mode page policy descriptors in this VPD page shall describe the mode page policy for every mode page and subpage supported by the logical unit.

Table 562 — Mode page policy descriptor

Bit / Byte	7	6	5	4	3	2	1	0
0	Reserved		POLICY PAGE CODE					
1	POLICY SUBPAGE CODE							
2	MLUS	Reserved					MODE PAGE POLICY	
3	Reserved							

The POLICY PAGE CODE field and POLICY SUBPAGE CODE field indicate the mode page and subpage to which the descriptor applies. The POLICY PAGE CODE field contains the value that is used in a MODE SENSE command (see table 164 in 6.11.1) to retrieve the mode page being described.

If the first mode page policy descriptor in the list contains a POLICY PAGE CODE field set to 3Fh and a POLICY SUBPAGE CODE field set to FFh, then the descriptor applies to all mode pages and subpages not described by other mode page policy descriptors. The POLICY PAGE CODE field shall be set to 3Fh and the POLICY SUBPAGE CODE field shall be set to FFh only in the first mode page policy descriptor in the list.

If the POLICY PAGE CODE field contains a value other than 3Fh and a POLICY SUBPAGE CODE field contains a value other than FFh, then the POLICY PAGE CODE field and the POLICY SUBPAGE CODE field indicate a single mode page and subpage to which the descriptor applies.

If the POLICY PAGE CODE field contains a value other than 3Fh, then POLICY SUBPAGE CODE field shall contain a value other than FFh. If the POLICY SUBPAGE CODE field contains a value other than FFh, then POLICY PAGE CODE field shall contain a value other than 3Fh.

If the SCSI target device has more than one logical unit, a multiple logical units share (MLUS) bit set to one indicates the mode page and subpage identified by the POLICY PAGE CODE field and POLICY SUBPAGE CODE field is shared by more than one logical unit. A MLUS bit set to zero indicates the logical unit maintains its own copy of the mode page and subpage identified by the POLICY PAGE CODE field and POLICY SUBPAGE CODE field.

The MLUS bit is set to one in the mode page policy descriptors or descriptor that indicates the mode page policy for the:

a) Disconnect-Reconnect mode page (see 7.5.9); and
b) Protocol Specific Port mode page (see 7.5.14).

The MODE PAGE POLICY field (see table 563) indicates the mode page policy for the mode page and subpage identified by the POLICY PAGE CODE field and POLICY SUBPAGE CODE field. The mode page policies are described in table 160 (see 6.9).

Table 563 — MODE PAGE POLICY field

Code	Description
00b	Shared
01b	Per target port
10b	Obsolete
11b	Per I_T nexus

7.8.10 Power Condition VPD page

The Power Condition VPD page (see table 564) indicates which power conditions (see 5.12) are supported by the logical unit and provides information about how those power conditions operate.

Table 564 — Power Condition VPD page

Bit / Byte	7	6	5	4	3	2	1	0
0	PERIPHERAL QUALIFIER			PERIPHERAL DEVICE TYPE				
1	PAGE CODE (8Ah)							
2	(MSB)	PAGE LENGTH (000Eh)						
3								(LSB)
4	Reserved						STANDBY_Y	STANDBY_Z
5	Reserved					IDLE_C	IDLE_B	IDLE_A
6	(MSB)	STOPPED CONDITION RECOVERY TIME						
7								(LSB)
8	(MSB)	STANDBY_Z CONDITION RECOVERY TIME						
9								(LSB)
10	(MSB)	STANDBY_Y CONDITION RECOVERY TIME						
11								(LSB)
12	(MSB)	IDLE_A CONDITION RECOVERY TIME						
13								(LSB)
14	(MSB)	IDLE_B CONDITION RECOVERY TIME						
15								(LSB)
16	(MSB)	IDLE_C CONDITION RECOVERY TIME						
17								(LSB)

The PERIPHERAL QUALIFIER field and PERIPHERAL DEVICE TYPE field are defined in 7.8.2.

The PAGE CODE field and PAGE LENGTH field are defined in 7.8.2, and shall be set to the value shown in table 564 for the Power Condition VPD page.

If set to one, a power condition support bit (i.e., the STANDBY_Y bit, the STANDBY_Z bit, the IDLE_C bit, the IDLE_B bit, and the IDLE_A bit) indicates that:

a) the associated power condition may be entered with the START STOP UNIT command (see SBC-3) if that command is implemented; and
b) the associated power condition may be entered with a power condition timer if the associated timer is supported and enabled (see 7.5.12).

A STANDBY_Y power conditions support bit set to one indicates that the logical unit supports the standby_y power condition as described in this subclause. A STANDBY_Y bit set to zero indicates that the logical unit does not support the standby_y power condition.

A STANDBY_Z power conditions support bit set to one indicates that the logical unit supports the standby_z power condition as described in this subclause. A STANDBY_Z bit set to zero indicates that the logical unit does not support the standby_z power condition.

A IDLE_C power conditions support bit set to one indicates that the logical unit supports the idle_c power condition as described in this subclause. A IDLE_C bit set to zero indicates that the logical unit does not support the idle_c power condition.

A IDLE_B power conditions support bit set to one indicates that the logical unit supports the idle_b power condition as described in this subclause. A IDLE_B bit set to zero indicates that the logical unit does not support the idle_b power condition.

A IDLE_A power conditions support bit set to one indicates that the logical unit supports the idle_a power condition as described in this subclause. A IDLE_A bit set to zero indicates that the logical unit does not support the idle_a power condition.

The STOPPED CONDITION RECOVERY TIME field indicates the time, in one millisecond increments, that the logical unit takes to transition from the stopped power condition to the active power condition. This field is only applicable to SCSI target devices that implement the START STOP UNIT command (see SBC-3). This time does not include the processing time for the command that caused this transition to occur or any SCSI transport protocol specific waiting time (e.g., the NOTIFY (ENABLE SPINUP) requirement described in SPL). A value of zero indicates that the recovery time is not specified. A value of FFFFh indicates that the recovery time is more than 65.534 seconds.

The STANDBY_Z CONDITION RECOVERY TIME field indicates the time, in one millisecond increments, that the logical unit takes to transition from the standby_z power condition to the active power condition. This time does not include the processing time for the command that caused this transition to occur or any SCSI transport protocol specific waiting time (e.g., the NOTIFY (ENABLE SPINUP) requirement described in SPL). A value of zero indicates that the recovery time is not specified. A value of FFFFh indicates that the recovery time is more than 65.534 seconds.

The STANDBY_Y CONDITION RECOVERY TIME field indicates the time, in one millisecond increments, that the logical unit takes to transition from the standby_y power condition to the active power condition. This time does not include the processing time for the command that caused this transition to occur or any SCSI transport protocol specific waiting time (e.g., the NOTIFY (ENABLE SPINUP) requirement described in SPL). A value of zero indicates that the recovery time is not specified. A value of FFFFh indicates that the recovery time is more than 65.534 seconds.

The IDLE_A CONDITION RECOVERY TIME field indicates the time, in one millisecond increments, that the logical unit takes to transition from the idle_a power condition to the active power condition. This time does not include the processing time for the command that caused this transition to occur. A value of zero indicates that the recovery time is not specified. A value of FFFFh indicates that the recovery time is more than 65.534 seconds.

The IDLE_B CONDITION RECOVERY TIME field indicates the time, in one millisecond increments, that the logical unit takes to transition from the idle_b power condition to the active power condition. This time does not include the

processing time for the command that caused this transition to occur. A value of zero indicates that the recovery time is not specified. A value of FFFFh indicates that the recovery time is more than 65.534 seconds.

The IDLE_C CONDITION RECOVERY TIME field indicates the time, in one millisecond increments, that the logical unit takes to transition from the idle_c power condition to the active power condition. This time does not include the processing time for the command that caused this transition to occur. A value of zero indicates that the recovery time is not specified. A value of FFFFh indicates that the recovery time is more than 65.534 seconds.

7.8.11 SCSI Ports VPD page

The SCSI Ports VPD page (see table 565) provides a means to retrieve designation descriptors for all the SCSI ports in a SCSI target device or SCSI target/initiator device.

Table 565 — SCSI Ports VPD page

Bit / Byte	7	6	5	4	3	2	1	0
0	PERIPHERAL QUALIFIER			PERIPHERAL DEVICE TYPE				
1	PAGE CODE (88h)							
2	(MSB)	PAGE LENGTH (n-3)						
3								(LSB)
	Designation descriptor list							
4	SCSI port designation descriptor [first] (see table 566)							
•••								
	⋮							
	SCSI port designation descriptor [last] (see table 566)							
•••								
n								

The SCSI Ports VPD page only reports information on SCSI ports known to the device server processing the INQUIRY command. The REPORT LUNS well-known logical unit (see 8.2) may be used to return information on all SCSI ports in the SCSI device (i.e., all target ports and all initiator ports).

If the device server detects that a SCSI port is added or removed from the SCSI device and the SCSI port designation descriptor list changes, it shall establish a unit attention condition (see SAM-4) for the initiator port associated with every I_T nexus, with the additional sense code set to INQUIRY DATA HAS CHANGED.

The PERIPHERAL QUALIFIER field, PERIPHERAL DEVICE TYPE field, and PAGE LENGTH field are defined in 7.8.2.

The PAGE CODE field is defined in 7.8.2 and shall be set as shown in table 565 for the SCSI Ports VPD page.

Each SCSI port designation descriptor (see table 566) identifies a SCSI port. The SCSI port designation descriptors may be returned in any order.

Table 566 — SCSI port designation descriptor

Bit / Byte	7	6	5	4	3	2	1	0
0	Reserved							
1								
2	(MSB)	RELATIVE PORT IDENTIFIER						
3								(LSB)
4	Reserved							
5								
6	(MSB)	INITIATOR PORT TRANSPORTID LENGTH (k-7)						
7								(LSB)
8	INITIATOR PORT TRANSPORTID, if any							
•••								
k								
k+1	Reserved							
k+2								
k+3	(MSB)	TARGET PORT DESCRIPTORS LENGTH (n-(k+4))						
k+4								(LSB)
	Target port descriptor list							
k+5	Target port descriptor [first] (see table 568)							
•••								
	⋮							
	Target port descriptor [last] (see table 568)							
•••								
n								

The RELATIVE PORT IDENTIFIER field (see table 567) contains the relative port identifier (see 3.1.127) of the SCSI port to which the SCSI port designation descriptor applies.

Table 567 — RELATIVE PORT IDENTIFIER field

Code	Description
0h	Reserved
1h	Relative port 1, historically known as port A
2h	Relative port 2, historically known as port B
3h to FFFFh	Relative port 3 through 65 535

The INITIATOR PORT TRANSPORTID LENGTH field contains the length of the INITIATOR PORT TRANSPORTID field. An INITIATOR PORT TRANSPORTID LENGTH field set to zero indicates no INITIATOR PORT TRANSPORTID field is present (i.e., the SCSI port is not an initiator port and not a target/initiator port).

If the INITIATOR PORT TRANSPORTID LENGTH field contains a non-zero value, the INITIATOR PORT TRANSPORTID field contains a TransportID identifying the initiator port as specified in 7.6.4.

The TARGET PORT DESCRIPTORS LENGTH field contains the length of the target port descriptors, if any. A TARGET PORT DESCRIPTORS LENGTH field set to zero indicates no target port descriptors are present (i.e., the SCSI port is not a target port and not a target/initiator port).

Each target port descriptor (see table 568) contains an identifier for the target port. The target port descriptors may be returned in any order.

Table 568 — Target port descriptor

Bit / Byte	7	6	5	4	3	2	1	0
0	PROTOCOL IDENTIFIER				CODE SET			
1	PIV (1b)	Reserved	ASSOCIATION (01b)		DESIGNATOR TYPE			
2	Reserved							
3	DESIGNATOR LENGTH (n-3)							
4	DESIGNATOR							
•••								
n								

The PROTOCOL IDENTIFIER field indicates the SCSI transport protocol to which the designation descriptor applies as described in 7.8.6.1.

The CODE SET field, PIV field, ASSOCIATION field, DESIGNATOR TYPE field, DESIGNATOR LENGTH field, and DESIGNATOR field are as defined in the Device Identification VPD page designation descriptor (see 7.8.6.1), with the following additional requirements:

a) the PIV bit shall be set to one (i.e., the PROTOCOL IDENTIFIER field always contains a SCSI transport protocol identifier); and
b) the ASSOCIATION field shall be set to 01b (i.e., the descriptor always identifies a target port).

7.8.12 Protocol Specific Logical Unit Information VPD page

The Protocol Specific Logical Unit Information VPD page (see table 569) contains protocol specific parameters associated with a SCSI port that may be different for each logical unit in the SCSI target device.

Table 569 — Protocol Specific Logical Unit Information VPD page

Bit / Byte	7	6	5	4	3	2	1	0
0	PERIPHERAL QUALIFIER			PERIPHERAL DEVICE TYPE				
1	PAGE CODE (90h)							
2	(MSB)	PAGE LENGTH (n-3)						
3								(LSB)
	Logical unit information descriptor list							
4	Logical unit information descriptor [first] (see table 570)							
•••								
	⋮							
	Logical unit information descriptor [last] (see table 570)							
•••								
n								

The PERIPHERAL QUALIFIER field, PERIPHERAL DEVICE TYPE field, and PAGE LENGTH field are defined in 7.8.2.

The PAGE CODE field is defined in 7.8.2 and shall be set as shown in table 569 for the Protocol Specific Logical Unit Information VPD page.

The logical unit information descriptor list contains descriptors for SCSI ports known to the device server that is processing the INQUIRY command. The logical unit information descriptors (see table 570) may be returned in any order.

Table 570 — Logical unit information descriptor

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)	RELATIVE PORT IDENTIFIER						
1								(LSB)
2	Reserved				PROTOCOL IDENTIFIER			
3	Reserved							
4								
5								
6	(MSB)	PROTOCOL SPECIFIC DATA LENGTH (n-7)						
7								(LSB)
8	Per logical unit SCSI transport protocol specific data							
n								

The RELATIVE PORT IDENTIFIER field contains the relative port identifier (see 3.1.127) of the SCSI port to which the logical unit information descriptor applies and is defined in the SCSI Ports VPD page (see 7.8.11).

The PROTOCOL IDENTIFIER field indicates the SCSI transport protocol to which the logical unit information descriptor applies as described in 7.8.6.1.

The PROTOCOL SPECIFIC DATA LENGTH field indicates the length in bytes of the per logical unit SCSI transport protocol specific data.

The per logical unit SCSI transport protocol specific data is defined by the SCSI transport protocol standard (see 3.1.147) that corresponds to the SCSI target port.

7.8.13 Protocol Specific Port Information VPD page

The Protocol Specific Port Information VPD page (see table 571) contains protocol specific parameters associated with a SCSI port that are the same for all logical units in the SCSI target device.

Table 571 — Protocol Specific Port Information VPD page

Bit / Byte	7	6	5	4	3	2	1	0
0	PERIPHERAL QUALIFIER			PERIPHERAL DEVICE TYPE				
1	PAGE CODE (91h)							
2	(MSB)	PAGE LENGTH (n-3)						
3								(LSB)
	Port information descriptor list							
4 •••	Port information descriptor [first] (see table 570)							
	⋮							
••• n	Port information descriptor [last] (see table 570)							

The PERIPHERAL QUALIFIER field, PERIPHERAL DEVICE TYPE field, and PAGE LENGTH field are defined in 7.8.2.

The PAGE CODE field is defined in 7.8.2 and shall be set as shown in table 571 for the Protocol Specific Port Information VPD page.

The port information descriptor list contains descriptors for SCSI ports known to the device server that is processing the INQUIRY command. The port information descriptors (see table 572) may be returned in any order.

Table 572 — Port information descriptor

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)	RELATIVE PORT IDENTIFIER						
1								(LSB)
2	Reserved				PROTOCOL IDENTIFIER			
3	Reserved							
4								
5								
6	(MSB)	PROTOCOL SPECIFIC DATA LENGTH (n-7)						
7								(LSB)
8	Shared SCSI transport protocol specific data							
•••								
n								

The RELATIVE PORT IDENTIFIER field contains the relative port identifier (see 3.1.127) of the SCSI port to which the port information descriptor applies and is defined in the SCSI Ports VPD page (see 7.8.11).

The PROTOCOL IDENTIFIER field indicates the SCSI transport protocol to which the port information descriptor applies as described in 7.8.6.1.

The PROTOCOL SPECIFIC DATA LENGTH field indicates the length in bytes of the shared SCSI transport protocol specific data.

The shared SCSI transport protocol specific data is defined by the SCSI transport protocol standard (see 3.1.147) that corresponds to the SCSI target port.

7.8.14 Software Interface Identification VPD page

The Software Interface Identification VPD page (see table 573) provides identification of software interfaces applicable to the logical unit. Logical units may have more than one associated software interface identifier.

NOTE 79 - Application clients may use the software IDs to differentiate peripheral device function in cases where the command set (e.g., processor devices) is too generic to distinguish different software interfaces implemented.

Table 573 — Software Interface Identification VPD page

Bit / Byte	7	6	5	4	3	2	1	0
0	PERIPHERAL QUALIFIER			PERIPHERAL DEVICE TYPE				
1	PAGE CODE (84h)							
2	(MSB)			PAGE LENGTH (n-3)				
3								(LSB)
	Software interface identifier list							
4				Software interface identifier [first]				
•••								
10								
	⋮							
n-9				Software interface identifier [last]				
•••								
n								

The PERIPHERAL QUALIFIER field, PERIPHERAL DEVICE TYPE field, and PAGE LENGTH field are defined in 7.8.2.

The PAGE CODE field is defined in 7.8.2 and shall be set as shown in table 573 for the Software Interface Identification VPD page.

Each software interface identifier (see table 574) is a six-byte, fixed-length field that contains information identifying a software interface implemented by the logical unit. The contents of software interface identifier are in EUI-48 format (see 3.1.56).

Table 574 — Software interface identifier format

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)							
1				IEEE COMPANY_ID				
2								(LSB)
3	(MSB)							
4				VENDOR SPECIFIC EXTENSION IDENTIFIER				
5								(LSB)

The IEEE COMPANY_ID field contains a 24 bit OUI (see 3.1.103) assigned by the IEEE.

The VENDOR SPECIFIC EXTENSION IDENTIFIER a 24 bit numeric value that is uniquely assigned by the organization associated with the OUI as required by the IEEE definition of EUI-48 (see 3.1.56). The combination of OUI and vendor specific extension identifier shall uniquely identify the document or documents that specify the supported software interface.

7.8.15 Supported VPD Pages VPD page

The Supported VDP Pages VPD page contains a list of the VPD page codes supported by the logical unit (see table 575). If a device server supports any VPD pages, it also shall support this VPD page.

Table 575 — Supported VPD Pages VPD page

Bit / Byte	7	6	5	4	3	2	1	0
0	PERIPHERAL QUALIFIER			PERIPHERAL DEVICE TYPE				
1	PAGE CODE (00h)							
2	(MSB)	PAGE LENGTH (n-3)						
3								(LSB)
4	Supported VPD page list							
•••								
n								

The PERIPHERAL QUALIFIER field, PERIPHERAL DEVICE TYPE field, and PAGE LENGTH field are defined in 7.8.2.

The PAGE CODE field is defined in 7.8.2 and shall be set as shown in table 575 for the Supported VDP Pages VPD page.

The supported VPD page list shall contain a list of all VPD page codes (see 7.8) implemented by the logical unit in ascending order beginning with page code 00h.

7.8.16 Unit Serial Number VPD page

The Unit Serial Number VPD page (see table 576) provides a product serial number for the SCSI target device or logical unit.

Table 576 — Unit Serial Number VPD page

Bit / Byte	7	6	5	4	3	2	1	0
0	PERIPHERAL QUALIFIER			PERIPHERAL DEVICE TYPE				
1	PAGE CODE (80h)							
2	(MSB)	PAGE LENGTH (n-3)						
3								(LSB)
4	(MSB)	PRODUCT SERIAL NUMBER						
•••								
n								(LSB)

The PERIPHERAL QUALIFIER field, PERIPHERAL DEVICE TYPE field, and PAGE LENGTH field are defined in 7.8.2.

The PAGE CODE field is defined in 7.8.2 and shall be set as shown in table 576 for the Unit Serial Number VPD page.

The PRODUCT SERIAL NUMBER field contains right-aligned ASCII data (see 4.4.1) that is vendor-assigned serial number. If the product serial number is not available, the device server shall return ASCII spaces (20h) in this field.

8 Well known logical units

8.1 Model for well known logical units

Well known logical units are addressed using the well known logical unit addressing method of extended logical unit addressing (see SAM-4). Each well known logical unit has a well known logical unit number (W-LUN) as shown in table 577.

Table 577 — Well known logical unit numbers

W-LUN	Description	Reference
00h	Reserved	
01h	REPORT LUNS well known logical unit	8.2
02h	ACCESS CONTROLS well known logical unit	8.3
03h	TARGET LOG PAGES well known logical unit	8.4
04h	SECURITY PROTOCOL well known logical unit	8.5
05h	MANAGEMENT PROTOCOL well known logical unit	8.6
06h-FFh	Reserved	

If a well known logical unit is supported within a SCSI target device, then that logical unit shall support all the commands defined for it.

Access to well known logical units shall not be affected by access controls.

The SCSI device name of the well known logical unit may be determined by issuing an INQUIRY command (see 6.4) requesting the Device Identification VPD page (see 7.8.6).

All well known logical units shall support the INQUIRY command's Device Identification VPD page as specified in 7.8.6.2.2.

8.2 REPORT LUNS well known logical unit

The REPORT LUNS well known logical unit shall only process the commands listed in table 578. If a command is received by the REPORT LUNS well known logical unit that is not listed in table 578, then the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID COMMAND OPERATION CODE.

Table 578 — Commands for the REPORT LUNS well known logical unit

Command name	Operation code	Type	Reference
INQUIRY	12h	M	6.4
REPORT LUNS	A0h	M	6.23
REQUEST SENSE	03h	M	6.29
TEST UNIT READY	00h	M	6.37
Key: M = Command implementation is mandatory.			

8.3 ACCESS CONTROLS well known logical unit

8.3.1 Access controls model

8.3.1.1 Access controls commands

The ACCESS CONTROLS well known logical unit shall only process the commands listed in table 579. If a command is received by the ACCESS CONTROLS well know logical unit that is not listed in table 579, then the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID COMMAND OPERATION CODE.

Table 579 — Commands for the ACCESS CONTROLS well known logical unit

Command name	Operation code	Type	Reference
ACCESS CONTROL IN	86h	M	8.3.2
ACCESS CONTROL OUT	87h	M	8.3.3
INQUIRY	12h	M	6.4
REQUEST SENSE	03h	M	6.29
TEST UNIT READY	00h	M	6.37
Key: M = Command implementation is mandatory.			

8.3.1.2 Access controls overview

Access controls are a SCSI target device feature that application clients may use to restrict logical unit access to specified initiator ports or groups of initiator ports.

Access controls shall not allow restrictions to be placed on access to well known logical units. Access controls shall not cause new well known logical units to be defined.

Access controls are handled in the SCSI target device by an access controls coordinator located at the ACCESS CONTROLS well known logical unit. The access controls coordinator also may be accessible via LUN 0. The access controls coordinator associates a specific LUN to a specific logical unit depending on which initiator port accesses the SCSI target device and whether the initiator port has access rights to the logical unit.

Access rights to a logical unit affects whether the logical unit number appears in the parameter data returned by a REPORT LUNS command and how the logical unit responds to INQUIRY commands.

The access controls coordinator maintains the ACL as described in 8.3.1.3 to supply information about:

a) which initiator ports are allowed access to which logical units; and
b) which LUN value is used by a specific initiator port when accessing a specific logical unit.

The format of the ACL is vendor specific.

To support third party commands (e.g., EXTENDED COPY), the access controls coordinator may provide proxy tokens (see 8.3.1.6.2) to allow an application client to pass its access capabilities to the application client for another initiator port.

An application client manages the access controls state of the SCSI target device using the ACCESS CONTROL IN command (see 8.3.2) and the ACCESS CONTROL OUT command (see 8.3.3).

A SCSI target device has access controls disabled when it is manufactured and after successful completion of the ACCESS CONTROL OUT command with DISABLE ACCESS CONTROLS service action (see 8.3.3.3). When access controls are disabled, the ACL contains no entries and the management identifier key (see 8.3.1.8) is zero.

The first successful ACCESS CONTROL OUT command with MANAGE ACL service action (see 8.3.3.2) shall enable access controls. When access controls are enabled, all logical units, except LUN 0 and all well known logical units, shall be inaccessible to all initiator ports unless the ACL (see 8.3.1.3) allows access.

The ACL allows an initiator port access to a logical unit if the ACL contains an ACE (see 8.3.1.3) with an access identifier (see 8.3.1.3.2) associated with the initiator port and that ACE contains a LUACD (see 8.3.1.3.3) that references the logical unit.

When the ACL allows access to a logical unit, the REPORT LUNS command parameter data bytes representing that logical unit shall contain the LUN value found in the LUACD that references that logical unit and the application client for the initiator port shall use the same LUN value when sending commands to the logical unit.

An initiator port also may be allowed access to a logical unit through the use of a proxy token (see 8.3.1.6.2).

Once access controls are enabled, they shall remain enabled until:

a) successful completion of an ACCESS CONTROL OUT command with DISABLE ACCESS CONTROLS service action; or
b) vendor specific physical intervention.

Successful downloading of microcode (see 6.39) may result in access controls being disabled.

Once access controls are enabled, power cycles, hard resets, logical unit resets, and I_T nexus losses shall not disable them.

8.3.1.3 The access control list (ACL)

8.3.1.3.1 ACL overview

The specific access controls for a SCSI target device are instantiated by the access controls coordinator using data in an ACL. The ACL contains zero or more ACEs and zero or more proxy tokens (see 8.3.1.6.2.1).

Each ACE contains the following:

a) one access identifier (see 8.3.1.3.2) that identifies the initiator port(s) to which the ACE applies; and
b) a list of LUACDs (see 8.3.1.3.3) that identify the logical units to which the identified initiator port(s) have access rights and the LUNs used to access those logical units via those initiator port(s). Each LUACD contains the following:
   A) a vendor specific logical unit reference; and
   B) a LUN value.

Figure 15 shows the logical structure of an ACL.

Figure 15 — ACL Structure

8.3.1.3.2 Access identifiers

Initiator ports are identified in an ACE using one of the following types of access identifiers:

a) AccessID - based on initiator port enrollment;
b) TransportID - based on protocol specific identification of initiator ports; or
c) vendor specific access identifiers.

An initiator port is allowed access to the logical units in an ACE containing an AccessID type access identifier when that initiator port is enrolled as described in 8.3.1.5. An initiator port that has not previously enrolled uses the ACCESS CONTROL OUT command with ACCESS ID ENROLL service action to enroll including the AccessID in parameter data as specified in 8.3.3.4.

An initiator port is associated with an AccessID type access identifier if that initiator port is in the enrolled state or pending-enrolled state with respect to that AccessID (see 8.3.1.5). At any instant in time, an initiator port may be associated with at most one AccessID. All initiator ports enrolled using a specific AccessID share the same ACE and access to all the logical units its LUACDs describe.

TransportID access identifiers are SCSI transport protocol specific as described in 7.6.4.

An initiator port is allowed access to the logical units in an ACE containing a TransportID type access identifier when the identification for the initiator port matches that found in the TransportID in a way that is consistent with the TransportID definition (see 7.6.4). There is no need to process any command to obtain logical unit access based on a Transport ID because the needed information is provided by the SCSI transport protocol layer.

The formats of access identifiers are defined in 8.3.1.13.

8.3.1.3.3 Logical unit access control descriptors

Each LUACD in an ACE identifies one logical unit to which the initiator ports associated with the access identifier are allowed access and specifies the LUN value used when accessing the logical unit via those initiator ports. The format of a LUACD is vendor specific.

The identification of a logical unit in a LUACD is vendor specific. The logical unit identified by a LUACD shall not be a well known logical unit. A logical unit shall be referenced in no more than one LUACD per ACE.

The LUN value shall conform to the requirements specified in SAM-4. A specific LUN value shall appear in no more than one LUACD per ACE.

8.3.1.4 Managing the ACL

8.3.1.4.1 ACL management overview

The contents of the ACL are managed by an application client using the ACCESS CONTROL OUT command with MANAGE ACL and DISABLE ACCESS CONTROLS service actions. The ACCESS CONTROL OUT command with MANAGE ACL service action (see 8.3.3.2) is used to add, remove, or modify ACEs thus adding, revoking, or changing the allowed access of initiator ports to logical units. The ACCESS CONTROL OUT command with DISABLE ACCESS CONTROLS service action (see 8.3.3.3) disables access controls and discards the ACL.

8.3.1.4.2 Authorizing ACL management

To reduce the possibility of applications other than authorized ACL managers changing the ACL, successful completion of specific access controls service actions (e.g., ACCESS CONTROL OUT command with MANAGE ACL or DISABLE ACCESS CONTROLS service action) requires delivery of the correct management identifier key value (see 8.3.1.8) in the ACCESS CONTROL OUT parameter data. The service actions that require the correct management identifier key are shown in table 580 and table 581.

Table 580 — ACCESS CONTROL OUT management identifier key requirements

Service action name	Management Identifier Key Required	Reference
ACCESS ID ENROLL	No	8.3.3.4
ASSIGN PROXY LUN	No	8.3.3.11
CANCEL ENROLLMENT	No	8.3.3.5
CLEAR ACCESS CONTROLS LOG	Yes	8.3.3.6
DISABLE ACCESS CONTROLS	Yes	8.3.3.3
MANAGE ACL	Yes	8.3.3.2
MANAGE OVERRIDE LOCKOUT TIMER	Yes/No	8.3.3.7
OVERRIDE MGMT ID KEY	No	8.3.3.8
RELEASE PROXY LUN	No	8.3.3.12
REVOKE ALL PROXY TOKENS	No	8.3.3.10
REVOKE PROXY TOKEN	No	8.3.3.9

Table 581 — ACCESS CONTROL IN management identifier key requirements

Service action name	Management Identifier Key Required	Reference
REPORT ACCESS CONTROLS LOG	Yes	8.3.2.4
REPORT ACL	Yes	8.3.2.2
REPORT LU DESCRIPTORS	Yes	8.3.2.3
REPORT OVERRIDE LOCKOUT TIMER	Yes	8.3.2.5
REQUEST PROXY TOKEN	No	8.3.2.6

8.3.1.4.3 Identifying logical units during ACL management

The access controls coordinator shall identify every logical unit of a SCSI target device with a unique default LUN value. The default LUN values used by the access controls coordinator shall be the LUN values that would be reported by the REPORTS LUNS command if access controls were disabled.

An application client discovers the default LUN values using the ACCESS CONTROL IN command with REPORT LU DESCRIPTORS (see 8.3.2.3) or REPORT ACL (see 8.3.2.2) service action and then supplies those default LUN values to the access controls coordinator using the ACCESS CONTROL OUT command with MANAGE ACL service action.

The association between default LUN values and logical units is managed by the access controls coordinator and may change due to circumstances that are beyond the scope of this standard. To track such changes, the access controls coordinator maintains the DLgeneration counter as described in 8.3.1.4.4.

8.3.1.4.4 Tracking changes in logical unit identification

The access controls coordinator shall implement a Default LUNs Generation (DLgeneration) counter to track changes in the association between default LUN values and logical units. The DLgeneration counter is a wrapping counter.

When access controls are disabled the DLgeneration counter shall be set to zero. When access controls are first enabled (see 8.3.1.2) the DLgeneration counter shall be set to one. While access controls are enabled, the access controls coordinator shall increment the DLgeneration counter by one every time the association between default LUN values and logical units changes (e.g., following the creation of a new logical unit, deletion of an existing logical unit, or removal and recreation of an existing logical unit).

The access controls coordinator shall include the current DLgeneration value in the parameter data returned by an ACCESS CONTROL IN command with REPORT LU DESCRIPTORS (see 8.3.2.3) or REPORT ACL (see 8.3.2.2) service action. The application client shall supply the DLgeneration value for the default LUN values it is using in the parameter data for an ACCESS CONTROL OUT command with MANAGE ACL service action (see 8.3.3.2).

Before processing the ACL change information in the parameter list provided by an ACCESS CONTROL OUT command with MANAGE ACL service action, the access controls coordinator shall verify that the DLgeneration value in the parameter data matches the current value of the DLgeneration counter. If the DLgeneration value verification finds a mismatch, 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.

8.3.1.5 Enrolling AccessIDs

8.3.1.5.1 Enrollment states

8.3.1.5.1.1 Summary of enrollment states

Application clients enroll an initiator port AccessID with the access controls coordinator to be allowed to access logical units listed in the ACE having the same AccessID type access identifier. The ACCESS CONTROL OUT command with ACCESS ID ENROLL service action (see 8.3.3.4) is used to enroll an AccessID. An initiator port shall be in one of three states with respect to such an enrollment:

a) **Not-enrolled**: the state for an initiator port before the first ACCESS CONTROL OUT command with ACCESS ID ENROLL service action is sent to the access controls coordinator. Also the state for an initiator port following successful completion of an ACCESS CONTROL OUT command with CANCEL ENROLLMENT service action (see 8.3.3.5);
b) **Enrolled**: the state for an initiator port following successful completion of an ACCESS CONTROL OUT command with ACCESS ID ENROLL service action; or
c) **Pending-enrolled**: the state for an enrolled initiator port following:
   A) events described in 8.3.1.12; or
   B) successful completion of an ACCESS CONTROL OUT command with MANAGE ACL service action from any initiator port with the FLUSH bit set to one (see 8.3.3.2).

8.3.1.5.1.2 Not-enrolled state

The access controls coordinator shall place an initiator port in the not-enrolled state when it first detects the receipt of a SCSI command or task management function from that initiator port. The initiator port shall remain in the not-enrolled state until successful completion of an ACCESS CONTROL OUT command with ACCESS ID ENROLL service action (see 8.3.3.5).

When in the not-enrolled state, an initiator port shall only have access to logical units on the basis of a TransportID (see 8.3.1.3.2) or on the basis of proxy tokens (see 8.3.1.6.2.1).

The access controls coordinator changes an initiator port from the enrolled state or pending-enrolled state to the not-enrolled state in response to the following events:

a) successful completion of the ACCESS CONTROL OUT command with CANCEL ENROLLMENT service action (see 8.3.3.5) shall change the state to not-enrolled; or
b) successful completion of an ACCESS CONTROL OUT command with MANAGE ACL service action (see 8.3.3.2) that replaces the ACL entry for the enrolled AccessID as follows:
   A) if the NOCNCL bit (see 8.3.3.2.2) is set to zero in the ACCESS CONTROL OUT command with MANAGE ACL service action parameter data, the state shall change to not-enrolled; or
   B) if the NOCNCL bit is set to one, the state may change to not-enrolled based on vendor specific criteria.

An application client for an enrolled initiator port may discover that the initiator port has transitioned to the not-enrolled state as a result of actions taken by a third party (e.g., an ACCESS CONTROL OUT command with MANAGE ACL service action performed by another initiator port or a logical unit reset).

Placing an enrolled initiator in the not-enrolled state indicates that the ACE defining that initiator port's logical unit access has changed (e.g., previous relationships between logical units and LUN values may no longer apply).

If an application client detects this loss of enrollment on an initiator port, it may take recovery actions. However, such actions may be disruptive for the SCSI initiator device and may not be required. Use of the not-enrolled state is avoidable if the application client that sends the ACCESS CONTROL OUT command with MANAGE ACL

service action determines that its requested changes to the ACL do not alter the existing relationships between logical units and LUN values in any existing ACEs with AccessID type access identifiers and sets the NOCNCL bit to one, recommending that initiator ports be left in their current enrollment state.

The access controls coordinator selects from the following options for responding to a NOCNCL bit set to one in a vendor specific manner:

a) honor the recommendation, causing the minimum effects on SCSI initiator devices and requiring no extra actions on the part of the access controls coordinator;
b) ignore the recommendation and always place initiator ports in the non-enrolled state, causing the maximum disruption for SCSI initiator devices, but requiring no extra resources on the part of the access controls coordinator; or
c) ignore the recommendation and examine the current and new ACEs to determine if an initiator port should be placed in the non-enrolled state.

If the application client that sends the ACCESS CONTROL OUT command with MANAGE ACL service action is unable to determine whether the ACE logical unit relationships are altered as a result of processing the command, then it should set the NOCNCL bit to zero and it should coordinate the ACL change with the application clients for affected initiator ports to ensure proper data integrity. Such coordination is beyond the scope of this standard.

8.3.1.5.1.3 Enrolled state

The access controls coordinator shall place an initiator port in the enrolled state (i.e., enroll the initiator port) following successful completion of the ACCESS CONTROL OUT command with ACCESS ID ENROLL service action (see 8.3.3.4). The ACCESS CONTROL OUT command with ACCESS ID ENROLL service action is successful only if:

a) the initiator port was in the not-enrolled state and the AccessID in the ACCESS CONTROL OUT command with ACCESS ID ENROLL service action parameter data matches the access identifier in an ACE. This results in the initiator port being enrolled and allowed access to the logical units specified in the LUACDs in the ACE (see 8.3.1.3); or
b) the initiator port was in the enrolled state or pending-enrolled state and the AccessID in the ACCESS CONTROL OUT command with ACCESS ID ENROLL service action parameter data matches the current enrolled AccessID for the initiator port.

If the initiator port was in the enrolled state or pending-enrolled state and the AccessID in the ACCESS CONTROL OUT command with ACCESS ID ENROLL service action parameter data does not match the current enrolled AccessID for the initiator port, 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 ACCESS DENIED - ENROLLMENT CONFLICT. If the initiator port was in the enrolled state, it shall be transitioned to the pending-enrolled state.

Transitions from the enrolled state to the not-enrolled state are described in 8.3.1.5.1.2. Transitions from the enrolled state to the pending-enrolled state are described in 8.3.1.5.1.4.

NOTE 80 - This standard does not preclude implicit enrollments through mechanisms in a service delivery subsystem. Such mechanisms should perform implicit enrollments after identification by TransportID and should fail in the case where there are ACL conflicts as described in 8.3.1.5.2.

8.3.1.5.1.4 Pending-enrolled state

The access controls coordinator shall place an initiator port in the pending-enrolled state if that initiator port currently is in the enrolled state, and in response to the following:

a) a logical unit reset;

b) an I_T nexus loss associated with that initiator port; or
c) successful completion of an ACCESS CONTROL OUT command with MANAGE ACL service action where the FLUSH bit is set to one in the parameter data.

While in the pending-enrolled state, the initiator port's access to logical units is limited as described in 8.3.1.7.

8.3.1.5.2 ACL LUN conflict resolution

ACL LUN conflicts may occur if:

a) an application client for an initiator port in the not-enrolled state attempts to enroll an AccessID using the ACCESS CONTROL OUT command with ACCESS ID ENROLL service action (see 8.3.3.4); or
b) an ACCESS CONTROL OUT command with MANAGE ACL service action (see 8.3.3.2) attempts to change the ACL with the result that it conflicts with existing enrollments (see 8.3.1.5) or proxy LUN assignments (see 8.3.1.6.2.2).

Three types of ACL LUN conflicts may occur:

a) the TransportID ACE (see 8.3.1.3) and the AccessID ACE for the initiator port each contain a LUACD with the same LUN value but with different logical unit references;
b) the TransportID ACE and the AccessID ACE for the initiator port each contain a LUACD with the different LUN values but with the same logical unit references; or
c) the enrolling initiator port has proxy access rights to a logical unit addressed with a LUN value that equals a LUN value in a LUACD in the AccessID ACE for the initiator port.

If an ACL LUN conflict occurs during the processing of an ACCESS CONTROL OUT command with MANAGE ACL service action, the command shall be terminated with CHECK CONDITION status (see 8.3.3.2.2).

If an ACL LUN conflict occurs during the processing of an ACCESS CONTROL OUT command with ACCESS ID ENROLL service action, the following actions shall be taken as part of the handling of the enrollment function:

a) the ACCESS CONTROL OUT command with ACCESS ID ENROLL service action shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, the additional sense code set to ACCESS DENIED - ACL LUN CONFLICT;
b) the initiator port shall remain in the not-enrolled state; and
c) if the ACL LUN conflict is not the result of proxy access rights, the access controls coordinator shall record the event in the access controls log as described in 8.3.1.10.

8.3.1.6 Granting and revoking access rights

8.3.1.6.1 Non-proxy access rights

The ACCESS CONTROL OUT command with MANAGE ACL service action (see 8.3.3.2) adds or replaces ACEs in the ACL (see 8.3.1.3). One ACE describes the logical unit access allowed by one access identifier (see 8.3.1.3.2) and the LUN values to be used in addressing the accessible logical units. The access identifier specifies the initiator port(s) permitted access to the logical units described by the ACE.

With the exception of proxy access rights (see 8.3.1.6.2), access rights are granted by:

a) adding a new ACE to the ACL; or
b) replacing an existing ACE with an ACE that includes additional LUACDs.

With the exception of proxy access rights, access rights are revoked by:

a) removing an ACE from the ACL; or
b) replacing an existing ACE with an ACE that removes one or more LUACDs.

When an ACE is added or replaced the requirements stated in 8.3.1.5.1.2 and 8.3.1.11 apply.

8.3.1.6.2 Proxy access

8.3.1.6.2.1 Proxy tokens

An application client with access rights to a logical unit via an initiator port on the basis of an ACE in the ACL (see 8.3.1.6.1) may temporarily share that access with third parties using the proxy access mechanism. The application client uses the ACCESS CONTROL IN command with REQUEST PROXY TOKEN service action (see 8.3.2.6) to request that the access control coordinator generate a proxy token for the logical unit specified by the LUN value in the CDB.

The access controls coordinator generates the proxy token in a vendor specific manner. For a specific SCSI target device, all active proxy token values should be unique. Proxy token values should be reused as infrequently as possible to prevent proxy tokens that have been used and released from being given unintended meaning.

Power cycles, hard resets, logical unit resets, and I_T nexus losses shall not affect the validity and proxy access rights of proxy tokens (see 8.3.1.12). A proxy token shall remain valid and retain the same proxy access rights until one of the following occurs:

a) an application client with access rights to a logical unit via an initiator port based on an ACE in the ACL revokes the proxy token using:
   A) the ACCESS CONTROL OUT command with REVOKE PROXY TOKEN service action (see 8.3.3.9); or
   B) the ACCESS CONTROL OUT command with REVOKE ALL PROXY TOKENS service action (see 8.3.3.10);

   or
b) an application client issues the ACCESS CONTROL OUT command with MANAGE ACL service action (see 8.3.3.2) with parameter data containing the Revoke Proxy Token ACE page (see 8.3.3.2.4) or Revoke All Proxy Tokens ACE page (see 8.3.3.2.5).

8.3.1.6.2.2 Proxy LUNs

To extend proxy access rights to a third party, an application client forwards a proxy token (see 8.3.1.6.2.2) to the third party (e.g., in a target descriptor in the parameter data of the EXTENDED COPY command).

The third party sends the access controls coordinator an ACCESS CONTROL OUT command with ASSIGN PROXY LUN service action (see 8.3.3.11) specifying the proxy token to request creation of a proxy access right to the referenced logical unit. The access controls coordinator determines the referenced logical unit from the proxy token value. The third party is unaware of the exact logical unit to which it is requesting access.

The parameter data for the ACCESS CONTROL OUT command with ASSIGN PROXY LUN service action includes the LUN value that the third party intends to use when accessing the referenced logical unit. The resulting LUN value is called a proxy LUN. If the ACCESS CONTROL OUT command with ASSIGN PROXY LUN service action is successful, the proxy LUN becomes the third party's mechanism for accessing the logical unit by proxy.

Once assigned, a proxy LUN shall remain valid until one of the following occurs:

a) the third party releases the proxy LUN value using the ACCESS CONTROL OUT command with RELEASE PROXY LUN service action (see 8.3.3.12);
b) the proxy token is made invalid as described in 8.3.1.6.2.1; or
c) a logical unit reset or I_T nexus loss of the I_T nexus used to assign the proxy LUN (see 8.3.1.12).

The third party may reissue the ACCESS CONTROL OUT command with ASSIGN PROXY LUN service action in an attempt to re-establish its proxy access rights. If the cause of the proxy token becoming invalid was temporary, the reissued command should succeed. The access controls coordinator shall process the request as described in 8.3.1.6.2.1 without reference to any previous assignment of the proxy LUN value.

8.3.1.7 Verifying access rights

When access controls are enabled (see 8.3.1.2), access rights for an initiator port shall be validated as described in this subclause.

Relationships between access controls and commands in a task set are described in 8.3.1.11.1.

All commands shall be processed as if access controls were not present if the ACL (see 8.3.1.3) allows the initiator port access to the addressed logical unit as a result of one of the following conditions:

a) the ACL contains an ACE containing a TransportID type access identifier (see 8.3.1.3.2) for the initiator port and that ACE includes a LUACD with LUN value matching the addressed LUN;
b) the initiator port is in the enrolled state (see 8.3.1.5.1.3) under an AccessID, the ACL contains an ACE containing that AccessID as an access identifier, and that ACE includes a LUACD with LUN value matching the addressed LUN; or
c) the addressed LUN matches a proxy LUN value (see 8.3.1.6.2.2) assigned using the ACCESS CONTROL OUT command with ASSIGN PROXY LUN service action (see 8.3.3.11) and the proxy token (see 8.3.1.6.2.1) used to assign the proxy LUN value is still valid.

If the initiator port is in the pending-enrolled state (see 8.3.1.5.1.4) under an AccessID, the ACL contains an ACE containing that AccessID as an access identifier, and that ACE includes a LUACD with LUN value matching the addressed LUN, then commands shall be processed as follows:

a) INQUIRY, REPORT LUNS, ACCESS CONTROL OUT and ACCESS CONTROL IN commands shall be processed as if access controls were not present;
b) a REQUEST SENSE command (see 6.29) shall be processed as if access controls were not present, except in cases where parameter data containing pollable sense data (see 5.6.2) would be returned. In these cases, device server shall terminate the REQUEST SENSE command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST and the additional sense code set to ACCESS DENIED - INITIATOR PENDING-ENROLLED; and
c) any other command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to ACCESS DENIED - INITIATOR PENDING-ENROLLED.

An application client should respond to the ACCESS DENIED - INITIATOR PENDING-ENROLLED additional sense code by sending an ACCESS CONTROL OUT command with ACCESS ID ENROLL service action. If the command succeeds, the application client may retry the terminated command.

If an INQUIRY command is addressed to a LUN for which there is no matching LUN value in any LUACD in any ACE allowing the initiator port logical unit access rights, the standard INQUIRY data (see 6.4.2) PERIPHERAL DEVICE TYPE field shall be set to 1Fh and the PERIPHERAL QUALIFIER field shall be set to 011b (i.e., the device server is not capable of supporting a device at this logical unit).

The parameter data returned in response to a REPORT LUNS command addressed to LUN 0 or to the REPORT LUNS well known logical unit shall return only the list of LUN values that are associated to accessible logical units according to the following criteria:

a) if the initiator port is in the enrolled state or pending-enrolled state, the REPORT LUNS parameter data shall include any LUN values found in LUACDs in the ACE containing the AccessID enrolled by the initiator port;
b) if the initiator port, in any enrollment state has a TransportID found in the access identifier of an ACE, then the REPORT LUNS parameter data shall include any LUN values found in LUACDs in that ACE; and
c) if the initiator port, in any enrollment state has access to any proxy LUNs (see 8.3.1.6.2.2), then those LUN values shall be included in the REPORT LUNS parameter data.

The parameter data returned in response to a REPORT LUNS command that describes well known logical units shall not be affected by access controls.

If the initiator port is in the not-enrolled state and is not allowed access to any logical unit as result of its TransportID or as a result of a proxy LUN assignment, then the REPORT LUNS parameter data shall include only LUN 0 and well known logical units, as specified in 6.23.

Except when access controls are disabled, all cases not described previously in this subclause shall result in termination of the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to LOGICAL UNIT NOT SUPPORTED.

8.3.1.8 The management identifier key

8.3.1.8.1 Management identifier key usage

The management identifier key identifies the application that is responsible for managing access controls for a SCSI target device. This identification occurs when the application client specifies a new management identifier key value in each ACCESS CONTROL OUT command with the MANAGE ACL service action (see 8.3.3.2), and when the last specified management identifier key value appears in ACCESS CONTROL IN and ACCESS CONTROL OUT service actions as required in 8.3.1.4.2.

To allow for failure scenarios where the management identifier key value has been lost, an override procedure involving a timer is described in 8.3.1.8.2.

Use of the management identifier key has the following features:

a) management of access controls is associated with those application clients that provide the correct management identifier key without regard for the initiator port from which the command was received; and
b) only an application client that has knowledge of the management identifier key may change the ACL, allowing the management of access controls to be limited to specific applications and application clients.

8.3.1.8.2 Overriding the management identifier key

8.3.1.8.2.1 The OVERRIDE MGMT ID KEY service action

If the management identifier key needs to be replaced and the current management identifier key is not available, then the ACCESS CONTROL OUT command with OVERRIDE MGMT ID KEY service action (see 8.3.3.8) may be used to force the management identifier key to a known value.

The ACCESS CONTROL OUT command with OVERRIDE MGMT ID KEY service action should be used only for failure recovery. If failure recovery is not required, the ACCESS CONTROL OUT command with MANAGE ACL service action should be used.

To protect the management identifier key from unauthorized overrides, the access controls coordinator shall restrict use of the ACCESS CONTROL OUT command with OVERRIDE MGMT ID KEY service action based on the value of the override lockout timer (see 8.3.1.8.2.2).

When the override lockout timer is not zero, an ACCESS CONTROL OUT command with OVERRIDE MGMT ID KEY service action 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.

When the override lockout timer is zero, an ACCESS CONTROL OUT command with OVERRIDE MGMT ID KEY service action shall be processed as described in 8.3.3.8.

The access controls coordinator shall log the receipt of each ACCESS CONTROL OUT command with OVERRIDE MGMT ID KEY service action and its success or failure as described in 8.3.1.10.

8.3.1.8.2.2 The override lockout timer

The access controls coordinator shall maintain the override lockout timer capable of counting up to 65 535 seconds. When the override lockout timer is not zero it shall be decreased by one nominally once per second but no more frequently than once every 800 milliseconds until the value reaches zero. When the override lockout timer is zero, it shall not be changed except as the result of commands sent by an application client.

The ACCESS CONTROL OUT command with MANAGE OVERRIDE LOCKOUT TIMER service action manages the state of the override lockout timer (see 8.3.3.7), performing one of the following functions:

a) if the incorrect management identifier key is supplied or if no parameter data is sent, the access controls coordinator shall reset the override lockout timer to the last received initial override lockout timer value; or
b) if the correct management identifier key is supplied, then the access controls coordinator shall do the following:
   1) save the initial override lockout timer value supplied in the parameter data; and
   2) reset the override lockout timer to the new initial value.

Setting the initial override lockout timer value to zero disables the override lockout timer and allows the ACCESS CONTROL OUT command with OVERRIDE MGMT KEY service action to succeed at any time.

Any application that knows the management identifier key may establish an initial override lockout timer value of sufficient duration (i.e., up to about 18 hours). Maintaining a non-zero override lockout timer value may be accomplished without knowing the management identifier key or transporting the management identifier key on a service delivery subsystem. Attempts to establish a zero initial override lockout timer value that are not accompanied by the correct management identifier key result in decreasing the probability that a subsequent ACCESS CONTROL OUT command with OVERRIDE MGMT ID KEY service action is able to succeed by resetting the override lockout timer.

After a logical unit reset, the override lock timer shall be set to the initial override lockout timer value within ten seconds of the non-volatile memory containing the initial override lockout timer value becoming available.

The ACCESS CONTROL IN command with REPORT OVERRIDE LOCKOUT TIMER may be used to discover the state of the override lockout timer.

8.3.1.9 Reporting access control information

Specific service actions of the ACCESS CONTROL IN command may be used by an application client to request a report from the access controls coordinator about its access controls data and state.

The ACCESS CONTROL IN command with REPORT ACL service action (see 8.3.2.2) returns the ACL (see 8.3.1.3). The information reported includes the following:

a) the list of access identifiers (see 8.3.1.3.2) and the associated LUACDs (see 8.3.1.3.3) currently in effect; and
b) the list of proxy tokens (see 8.3.1.6.2.1) currently in effect.

The ACCESS CONTROL IN command with REPORT ACCESS CONTROLS LOG service action (see 8.3.2.4) returns the contents of the access controls log (see 8.3.1.10).

The ACCESS CONTROL IN command with REPORT OVERRIDE LOCKOUT TIMER service action (see 8.3.2.5) reports on the state of the override lockout timer (see 8.3.1.8.2.2).

8.3.1.10 Access controls log

The access controls log is a record of events maintained by the access controls coordinator.

The access controls log has three portions, each recording a different class of events:

c) **invalid key events**: mismatches between the management identifier key (see 8.3.1.8) specified by a service action and the current value maintained by the access controls coordinator;
d) **key override events**: attempts to override the management identifier key (see 8.3.1.8.2.1), whether the attempt fails or succeeds; and
e) **ACL LUN conflict events**: (see 8.3.1.5.2).

Each portion of the log is required to contain a saturating counter of the corresponding events. When a SCSI target device is manufactured, all event counters shall be set to zero. When access controls are disabled, all event counters except the Key Override events counter shall be set to zero. Each event counter shall be incremented by one whenever the corresponding event occurs.

Each log portion may contain additional records with more specific information about each event. When the resources for additional log records are exhausted, the access controls coordinator shall preserve the most recently added log records in preference to older log records.

Log records contain a TIME STAMP field whose contents are vendor specific. If the access controls coordinator has no time stamp resources the TIME STAMP field shall be set to zero. If time stamp values are provided, the same timing clock and time stamp format shall be used for all access controls log entries.

Invalid key events occur whenever an access controls command requires the checking of an application client supplied management identifier key against the current management identifier key saved by the access controls coordinator and the two values fail to match. When such an event occurs, the access controls coordinator shall increment the Invalid Keys events counter by one. If the log has additional resources to record event details, the access controls coordinator shall add an invalid keys log record (containing the information defined in 8.3.2.4.2.3) describing the event.

Key override events occur when the access controls coordinator receives the ACCESS CONTROL OUT command with OVERRIDE MGMT KEY service action (see 8.3.3.8). When such an event occurs, the access controls coordinator shall increment the Key Overrides events counter by one without regard for whether the command succeeds or fails. If the log has additional resources to record event details, the access controls coordinator shall add a key overrides log record (containing the information defined in 8.3.2.4.2.2) describing the event.

ACL LUN conflict events occur as specified in 8.3.1.5.2. When such an event occurs, the access controls coordinator shall increment the ACL LUN Conflicts events counter by one. If the log has additional resources to record event details, the access controls coordinator shall add an ACL LUN conflicts log record (containing the information defined in 8.3.2.4.2.4) describing the event.

Selected portions of the access controls log may be requested by an application client using the ACCESS CONTROL IN command with REPORT ACCESS CONTROLS LOG service action (see 8.3.2.4). With the exception of the key overrides portion, selected portions of the log may be cleared and the event counters reset to zero using the ACCESS CONTROL OUT command with CLEAR ACCESS CONTROLS LOG service action (see 8.3.3.6).

8.3.1.11 Interactions of access controls and other features

8.3.1.11.1 Task set management and access controls

Upon successful completion of an ACCESS CONTROL OUT command with MANAGE ACL service action (see 8.3.3.2), the specified ACL (see 8.3.1.3) shall apply to all commands that subsequently enter the enabled command state. Commands that have modified SCSI target device state information (e.g., media, mode pages, and log pages) shall not be affected by an ACCESS CONTROL OUT command that subsequently enters the enabled command state. Commands in the enabled command state that have not modified SCSI target device state information may or may not be affected by an ACCESS CONTROL OUT command that subsequently enters the enabled command state. The ACL in effect prior to when the ACCESS CONTROL OUT command with MANAGE ACL or DISABLE ACCESS CONTROLS service action entered the enabled command state shall apply to all commands that are not affected by the ACCESS CONTROL OUT command.

All the operations performed by a command shall complete under the control of a single ACL, either the state in effect prior to processing of the ACCESS CONTROL OUT command or the state in effect following processing of the ACCESS CONTROL OUT command. After a command enters the enabled command state for the first time changing the access control state from disabled to enabled (see 8.3.1.2) shall have no effect on the command.

Multiple access control commands, both ACCESS CONTROL IN and ACCESS CONTROL OUT, may be in the task set concurrently. The order of processing of such commands is defined by the task set management requirements (see SAM-4), but each command shall be processed as a single indivisible command without any interleaving of actions that may be required by other access control commands.

8.3.1.11.2 Existing reservations and ACL changes

If a logical unit is reserved by one I_T nexus and that logical unit becomes accessible to another I_T nexus as a result of an access control command, then there shall be no changes in the reservation of that logical unit.

If a logical unit is reserved by an I_T nexus and that logical unit becomes inaccessible to that I_T nexus as a result of an access control command or other access control related event, then there shall be no changes in the reservation. Existing persistent reservations mechanisms allow for other SCSI initiator devices with access to that logical unit to clear the reservation.

8.3.1.12 Access controls information persistence and memory usage requirements

If a SCSI target device supports access controls, then the SCSI target device shall contain an access controls coordinator that shall maintain the following information in nonvolatile memory:

a) whether access controls are enabled or disabled; and
b) the access controls data that table 582 and table 583 require to persist across power cycles, hard resets, and logical unit resets.

If the access control coordinator's nonvolatile memory is not ready and the access controls coordinator is unable to determine that access controls are disabled, then the device servers for all logical units shall terminate all commands except INQUIRY and REQUEST SENSE commands with CHECK CONDITION status, with the sense key set to NOT READY, and the additional sense code set as described in table 284 (see 6.37). If a device server in any logical unit receives an INQUIRY command or a REQUEST SENSE command while the access control coordinator's nonvolatile memory is not ready and the access controls coordinator is unable to determine that access controls are disabled, then the device server shall process the command.

Following an I_T nexus loss, a previously enrolled initiator port shall be placed in the pending-enrolled state, if that initiator port was associated with the lost I_T nexus. Following a logical unit reset, all previously enrolled initiator ports shall be placed in the pending-enrolled state.

The information shown in table 582 shall be maintained by the access controls coordinator.

Table 582 — Mandatory access controls resources

Information Description	**Size** (in bits)	**Persistent Across Power Cycles, Hard Resets, and Logical Unit Resets**
One ACL (see 8.3.1.3) containing at least one ACE containing one access identifier (see 8.3.1.3.2), and at least one LUACD (see 8.3.1.3.3)	VS	Yes
The Enrollment State for each initiator port (see 8.3.1.5.1)	VS	Yes
Management Identifier Key (see 8.3.1.8)	64	Yes
Default LUNs Generation (DLgeneration, see 8.3.1.4.4)	32	Yes
Override Lockout Timer (see 8.3.1.8.2.2)	16	No
Initial Override Lockout Timer value (see 8.3.1.8.2.2)	16	Yes
Access Controls Log Event Counters (see 8.3.1.10) containing at least the following:		Yes
a) Key Override events counter;	16	Yes
b) Invalid Key events counter; and	16	Yes
c) ACL LUN Conflict events counter	16	Yes

Optionally, the access controls coordinator may maintain the information shown in table 583.

Table 583 — Optional access controls resources

Information Description	Size (in bits)	Persistent Across Power Cycles, Hard Resets, and Logical Unit Resets
One or more proxy tokens (see 8.3.1.6.2.1)	64	Yes
One or more proxy LUNs (see 8.3.1.6.2.2)	64	No
Access controls log event records (see 8.3.1.10) for:		
a) Key Override events;	(see 8.3.2.4.2.2)	Yes
b) Invalid Key events; and	(see 8.3.2.4.2.3)	Yes
c) ACL LUN Conflict events	(see 8.3.2.4.2.4)	Yes

At the time of manufacturer, the ACL shall be empty, all values shown in table 582 shall be zero, additional access control log structures shall be empty and there shall be no valid proxy tokens.

8.3.1.13 Access identifier formats

8.3.1.13.1 Access identifier type

The ACCESS IDENTIFIER TYPE field (see table 584) indicates the format and usage of the access identifier.

Table 584 — ACCESS IDENTIFIER TYPE field

Code	Access Identifier Name	Reference
00h	AccessID	8.3.1.13.2
01h	TransportID	7.6.4
02h to 7Fh	Reserved	
80h to FFh	Vendor specific	

8.3.1.13.2 AccessID access identifiers

AccessID access identifiers shall have the format shown in table 585.

Table 585 — AccessID access identifier format

Bit / Byte	7	6	5	4	3	2	1	0
0	ACCESSID							
•••								
15								
16	Reserved							
•••								
23								

The ACCESSID field contains a value that identifies the AccessID type ACE in which the AccessID access identifier appears. Within the ACL, no two ACEs shall contain the same AccessID.

8.3.2 ACCESS CONTROL IN command

8.3.2.1 ACCESS CONTROL IN introduction

The service actions of the ACCESS CONTROL IN command (see table 586) are used to obtain information about the access controls that are active within the access controls coordinator and to perform other access control functions (see 8.3.1). If the ACCESS CONTROL IN command is implemented, the ACCESS CONTROL OUT command also shall be implemented. The ACCESS CONTROL IN command shall not be affected by access controls.

Table 586 — ACCESS CONTROL IN service actions

Service Action	Name	Type	Reference
00h	REPORT ACL	M	8.3.2.2
01h	REPORT LU DESCRIPTORS	M	8.3.2.3
02h	REPORT ACCESS CONTROLS LOG	M	8.3.2.4
03h	REPORT OVERRIDE LOCKOUT TIMER	M	8.3.2.5
04h	REQUEST PROXY TOKEN	O	8.3.2.6
05h to 17h	Reserved		
18h to 1Fh	Vendor specific		

Key: M = Service action implementation is mandatory if ACCESS CONTROL IN is implemented.
O = Service action implementation is optional.

The ACCESS CONTROL IN command may be addressed to any logical unit whose standard INQUIRY data (see 6.4.2) has the ACC bit set to one (e.g., LUN 0), in which case it shall be processed in the same manner as if the command had been addressed to the ACCESS CONTROLS well known logical unit. If an ACCESS CONTROL IN command is received by a device server whose standard INQUIRY data has the ACC bit set to zero, the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID COMMAND OPERATION CODE.

8.3.2.2 REPORT ACL service action

8.3.2.2.1 REPORT ACL introduction

The ACCESS CONTROL IN command with REPORT ACL service action (see table 587) is used to query the ACL (see 8.3.1.3) maintained by the access controls coordinator. If the ACCESS CONTROL IN command is implemented, the REPORT ACL service action shall be implemented.

Table 587 — ACCESS CONTROL IN command with REPORT ACL service action

Bit / Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (86h)							
1	Reserved			SERVICE ACTION (00h)				
2	(MSB)			MANAGEMENT IDENTIFIER KEY				
•••								
9								(LSB)
10	(MSB)			ALLOCATION LENGTH				
•••								
13								(LSB)
14	Reserved							
15	CONTROL							

The OPERATION CODE field is defined in 4.3.5.1 and shall be set as shown in table 587 for the ACCESS CONTROL IN command with REPORT ACL service action.

The SERVICE ACTION field is defined in 4.3.5.2 and shall be set as shown in table 587 for the ACCESS CONTROL IN command with REPORT ACL service action.

If access controls are disabled, the device server shall ignore the MANAGEMENT IDENTIFIER KEY field and shall complete the command with GOOD status returning the eight byte parameter list header specified in 8.3.2.2.2 subject to the allocation length limitation described in 4.3.5.6.

If access controls are enabled and the contents of the MANAGEMENT IDENTIFIER KEY field do not match the current management identifier key (see 8.3.1.8) maintained by the access controls coordinator, then parameter data shall not be returned and the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to ACCESS DENIED - INVALID MGMT ID KEY. This event shall be recorded in the invalid keys portion of the access controls log (see 8.3.1.10).

The ALLOCATION LENGTH field is defined in 4.3.5.6. The ALLOCATION LENGTH field value should be at least eight.

The CONTROL field is defined in SAM-5.

8.3.2.2.2 REPORT ACL parameter data format

8.3.2.2.2.1 REPORT ACL parameter data introduction

The format of the parameter data returned in response to an ACCESS CONTROL IN command with REPORT ACL service actions is shown in table 588.

Table 588 — ACCESS CONTROL IN with REPORT ACL parameter data format

Bit / Byte	7	6	5	4	3	2	1	0
	Parameter list header							
0	(MSB)							
•••				ACL DATA LENGTH (n-3)				
3								(LSB)
4	(MSB)							
•••				DLGENERATION				
7								(LSB)
	ACL data pages							
8								
•••				ACL data page 0				
				⋮				
•••				ACL data page x				
n								

The ACL DATA LENGTH field shall contain a count of the number of bytes in the remaining parameter data. If access controls are disabled, the ACL DATA LENGTH field shall be set to four. The relationship between the ACL DATA LENGTH field and the CDB ALLOCATION LENGTH field is defined in 4.3.5.6.

The DLGENERATION field shall contain the current DLgeneration value (see 8.3.1.4.4).

The ACL data pages contain a description of the ACL (see 8.3.1.3) maintained by the access controls coordinator. Each ACL data page describes one ACE in the ACL or one proxy token (see 8.3.1.6.2). Every ACE and every proxy token managed by the access controls coordinator shall have an ACL data page in the parameter data. The content and format of an ACL data page is indicated by a page code. Table 589 lists the ACL data page codes.

Table 589 — ACL data page codes

Page Code	ACL Data Page Name	Reference
00h	Granted	8.3.2.2.2.2
01h	Granted All	8.3.2.2.2.3
02h	Proxy Tokens	8.3.2.2.2.4
03h to EFh	Reserved	
F0h to FFh	Vendor specific	

8.3.2.2.2.2 Granted ACL data page format

The Granted ACL data page (see table 590) describes an ACE that allows access to a specific set of logical units via a list of LUACDs (see 8.3.1.3.3).

Table 590 — Granted ACL data page format

Bit / Byte	7	6	5	4	3	2	1	0
0	PAGE CODE (00h)							
1	Reserved							
2	(MSB)	PAGE LENGTH (n-3)						
3								(LSB)
4	Reserved							
5	ACCESS IDENTIFIER TYPE							
6	(MSB)	ACCESS IDENTIFIER LENGTH (m-7)						
7								(LSB)
8	ACCESS IDENTIFIER							
•••								
m								
	LUACD Descriptors							
m+1	LUACD descriptor 0							
•••								
m+20								
	⋮							
n-19	LUACD descriptor x							
•••								
n								

The PAGE LENGTH field indicates the number of additional bytes required for this ACL data page. The relationship between the PAGE LENGTH field and the CDB ALLOCATION LENGTH field is defined in 4.3.5.6.

The ACCESS IDENTIFIER TYPE field (see 8.3.1.13) indicates the format and usage of the access identifier.

The ACCESS IDENTIFIER LENGTH field indicates the number of bytes following taken up by the ACCESS IDENTIFIER field. The access identifier length shall be at least 24 and shall be a multiple of four.

The ACCESS IDENTIFIER field contains the identifier that the access controls coordinator uses to select the initiator port(s) that are allowed access to the logical units named by the LUACD descriptors in this ACL data page. The format of the ACCESS IDENTIFIER field is specified in table 584 (see 8.3.1.13). One Granted or Granted All (see 8.3.2.2.2.3) ACL data page shall be returned for a specific pair of values in the ACCESS IDENTIFIER TYPE and ACCESS IDENTIFIER fields.

Each LUACD descriptor (see table 591) describes the access allowed to one logical unit based on the access identifier. There shall be one LUACD descriptor for each logical unit to which the access identifier allows access.

Table 591 — Granted ACL data page LUACD descriptor format

Bit / Byte	7	6	5	4	3	2	1	0
0	ACCESS MODE							
1	Reserved							
2								
3								
4	LUN VALUE							
•••								
11								
12	DEFAULT LUN							
•••								
19								

The ACCESS MODE field (see table 592) indicates the type of access allowed to the logical unit referenced by the DEFAULT LUN field and addressable at the specified LUN value.

Table 592 — ACCESS MODE field

Code	Description
00h	Normal access
01h to EFh	Reserved
F0h to FFh	Vendor specific

The LUN VALUE field indicates the LUN value an accessing application client uses to access the logical unit via the initiator port to which the LUACD descriptor applies.

The DEFAULT LUN field identifies the logical unit to which access is allowed using the default LUN value described in 8.3.1.4.3. The value in the DEFAULT LUN field shall be consistent with the DLGENERATION field value returned in the parameter list header (see 8.3.2.2.2).

The LUN VALUE and DEFAULT LUN fields may contain the same value.

8.3.2.2.2.3 Granted All ACL data page format

The Granted All ACL data page (see table 593) describes an ACE that allows access to all the SCSI target device's logical units with the default LUN values being used as the accessing LUN values. Initiator ports that have access via the access identifier in a Granted All ACL data page are allowed to access the SCSI target device as if access controls were disabled.

Table 593 — Granted All ACL data page format

Bit / Byte	7	6	5	4	3	2	1	0
0	PAGE CODE (01h)							
1	Reserved							
2	(MSB)	PAGE LENGTH (m-3)						
3								(LSB)
4	Reserved							
5	ACCESS IDENTIFIER TYPE							
6	(MSB)	ACCESS IDENTIFIER LENGTH (m-7)						
7								(LSB)
8	ACCESS IDENTIFIER							
•••								
m								

The PAGE LENGTH, ACCESS IDENTIFIER TYPE, and ACCESS IDENTIFIER LENGTH, are described in 8.3.2.2.2.2.

The ACCESS IDENTIFIER field contains the identifier that the access controls coordinator uses to select the initiator port(s) that are allowed access to all the SCSI target device's logical units with the default LUN values being used as the accessing LUN values. The format of the access identifier field is specified in table 584 (see 8.3.1.13). One Granted (see 8.3.2.2.2.2) or Granted All ACL data page shall be returned for a specific pair of values in the ACCESS IDENTIFIER TYPE and ACCESS IDENTIFIER fields.

8.3.2.2.2.4 Proxy Tokens ACL data page format

The Proxy Tokens ACL data page (see table 594) describes the proxy tokens (see 8.3.1.6.2) maintained by the access controls coordinator.

Table 594 — Proxy Tokens ACL data page format

Bit / Byte	7	6	5	4	3	2	1	0
0	PAGE CODE (02h)							
1	Reserved							
2	(MSB)	PAGE LENGTH (n-3)						
3								(LSB)
	Proxy token descriptors							
4	Proxy token descriptor 0							
•••								
23								
	⋮							
n-19	Proxy token descriptor x							
•••								
n								

The PAGE LENGTH field indicates the number of additional bytes required for this ACL data page. The relationship between the PAGE LENGTH field and the CDB ALLOCATION LENGTH field is defined in 4.3.5.6.

If there are no active proxy tokens, the access controls coordinator may either not include the Proxy Tokens ACL data page in the parameter data or may include one such ACL data page containing no proxy token descriptors.

No more than one Proxy Tokens ACL data page shall be included in the parameter data.

Each proxy token descriptor (see table 595) describes the access allowed to one logical unit based on one proxy token. There shall be one proxy token descriptor for each active proxy token maintained by the access controls coordinator.

Table 595 — Proxy token descriptor format

Bit / Byte	7	6	5	4	3	2	1	0
0	Reserved							
•••								
3								
4	PROXY TOKEN							
•••								
11								
12	DEFAULT LUN							
•••								
19								

The PROXY TOKEN field indicates the proxy token to which this proxy token descriptor applies.

The DEFAULT LUN field identifies the logical unit to which this proxy token allows access using the default LUN value described in 8.3.1.4.3. The value in the DEFAULT LUN field shall be consistent with the DLGENERATION field value returned in the parameter list header (see 8.3.2.2.2).

The same default LUN value may appear in multiple proxy token descriptors, if multiple proxy tokens are valid for the same logical unit.

8.3.2.3 REPORT LU DESCRIPTORS service action

8.3.2.3.1 REPORT LU DESCRIPTORS introduction

The ACCESS CONTROL IN command with REPORT LU DESCRIPTORS service action (see table 596) reports the inventory of logical units for which access controls may be established. If the ACCESS CONTROL IN command is implemented, the REPORT LU DESCRIPTORS service action shall be implemented.

Table 596 — ACCESS CONTROL IN command with REPORT LU DESCRIPTORS service action

Bit / Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (86h)							
1	Reserved			SERVICE ACTION (01h)				
2	(MSB)			MANAGEMENT IDENTIFIER KEY				
•••								
9								(LSB)
10	(MSB)			ALLOCATION LENGTH				
•••								
13								(LSB)
14	Reserved							
15	CONTROL							

The OPERATION CODE field is defined in 4.3.5.1 and shall be set as shown in table 596 for the ACCESS CONTROL IN command with REPORT LU DESCRIPTORS service action.

The SERVICE ACTION field is defined in 4.3.5.2 and shall be set as shown in table 596 for the ACCESS CONTROL IN command with REPORT LU DESCRIPTORS service action.

If access controls are disabled, the device server shall ignore the MANAGEMENT IDENTIFIER KEY field and shall complete the command with GOOD status returning the 20 byte parameter list header as specified in 8.3.2.3.2 subject to the ALLOCATION LENGTH limitation described in 4.3.5.6.

NOTE 81 - When access controls are disabled, the logical unit inventory may be obtained using commands such as REPORT LUNS (see 6.23). To facilitate access controls management, the ACCESS CONTROL IN command with REPORT LU DESCRIPTORS service action returns more information than the REPORT LUNS command. When access controls are disabled additional commands such as INQUIRY (see 6.4) are required to obtain all the information provided by the ACCESS CONTROL IN command with REPORT LU DESCRIPTORS service action.

If access controls are enabled and the contents of the MANAGEMENT IDENTIFIER KEY field do not match the current management identifier key (see 8.3.1.8) maintained by the access controls coordinator, then parameter data shall not be returned and the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to ACCESS DENIED - INVALID MGMT ID KEY. This event shall be recorded in the invalid keys portion of the access controls log (see 8.3.1.10).

The ALLOCATION LENGTH field is defined in 4.3.5.6. The ALLOCATION LENGTH field value should be at least 20.

The CONTROL field is defined in SAM-5.

8.3.2.3.2 REPORT LU DESCRIPTORS parameter data format

The format of the parameter data returned in response to an ACCESS CONTROL IN command with REPORT LU DESCRIPTORS service actions is shown in table 597.

Table 597 — ACCESS CONTROL IN with REPORT LU DESCRIPTORS parameter data format

Bit / Byte	7	6	5	4	3	2	1	0
	Parameter list header							
0	(MSB)							
•••				LU INVENTORY LENGTH (n-3)				
3								(LSB)
4	(MSB)							
•••				NUMBER OF LOGICAL UNITS				
7								(LSB)
8								
•••				SUPPORTED LUN MASK FORMAT				
15								
16	(MSB)							
•••				DLGENERATION				
19								(LSB)
	Logical Unit descriptors							
20								
•••				Logical Unit descriptor 0				
				⋮				
•••				Logical Unit descriptor x				
n								

The LU INVENTORY LENGTH field shall contain a count of the number of bytes in the remaining parameter data. If access controls are disabled, the LU INVENTORY LENGTH field shall be set to 16. The relationship between the LU INVENTORY LENGTH field and the CDB ALLOCATION LENGTH field is defined in 4.3.5.6.

The NUMBER OF LOGICAL UNITS field shall contain a count of the number of logical units managed by the access controls coordinator. The value in NUMBER OF LOGICAL UNITS field shall be the same as the number of Logical Unit descriptors that follow in the parameter data.

The SUPPORTED LUN MASK FORMAT field (see table 598) contains a summary of the LUN values (see 8.3.1.3.3) that the access controls coordinator supports. LUN values are exchanged between application clients and the access controls coordinator by several service actions (e.g., the ACCESS CONTROL IN command with REPORT ACL service action described in 8.3.2.2 and the ACCESS CONTROL OUT command with MANAGE ACL service action described in 8.3.3.2). The format of the SUPPORTED LUN MASK FORMAT field follows the eight byte LUN structure defined for dependent logical units by SAM-4.

Table 598 — SUPPORTED LUN MASK FORMAT field format

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)	FIRST LEVEL LUN MASK						
1								(LSB)
2	(MSB)	SECOND LEVEL LUN MASK						
3								(LSB)
4	(MSB)	THIRD LEVEL LUN MASK						
5								(LSB)
6	(MSB)	FOURTH LEVEL LUN MASK						
7								(LSB)

The LUN MASK at each level indicates the approximate range of the logical unit number values the access controls coordinator supports. A bit set to zero in a LUN MASK field indicates that the access controls coordinator prohibits setting that bit to one in a LUN value. A bit set to one in a LUN MASK field indicates that the access controls coordinator may allow setting that bit to one in a LUN value.

(E.g., if the access controls coordinator only supports level one LUN values with LUN values ranging from 0 to 256, then the SUPPORTED LUN MASK FORMAT field shall contain 00FF 0000 0000 0000h. If only LUN values ranging from 0 to 200 were supported, the SUPPORTED LUN MASK FORMAT field still would contain 00FF 0000 0000 0000h.)

The value in the SUPPORT LUN MASK FORMAT field only summarizes the supported LUN values and is not a complete description. The value in the SUPPORT LUN MASK FORMAT field should be used as a guideline for specifying LUN values in service actions (e.g., ACCESS CONTROL OUT command with MANAGE ACL service action). LUN values that appear valid based on the contents of the SUPPORT LUN MASK FORMAT field may still be rejected.

The DLGENERATION field shall contain the current DLgeneration value (see 8.3.1.4.4).

Each Logical Unit descriptor (see table 599) contains information about one logical unit managed by the access controls coordinator. There shall be one Logical Unit descriptor for every logical unit managed by the access controls coordinator.

Table 599 — Logical Unit descriptor format

Bit / Byte	7	6	5	4	3	2	1	0
0	Reserved			PERIPHERAL DEVICE TYPE				
1	Reserved							
2	(MSB)	DESCRIPTOR LENGTH (n-3)						
3								(LSB)
4	DEFAULT LUN							
•••								
11								
12	Reserved							
13	EVPD DESIGNATION DESCRIPTOR LENGTH							
14	Reserved							
15	DEVICE IDENTIFIER LENGTH							
16	EVPD DESIGNATION DESCRIPTOR							
•••								
47								
48	(MSB)	DEVICE IDENTIFIER						
•••								
79								(LSB)
80	DEVICE TYPE SPECIFIC DATA							
•••								
n								

The PERIPHERAL DEVICE TYPE field is as defined in 6.4.2.

The DESCRIPTOR LENGTH field indicates the total number of bytes remaining in the descriptor. If the PERIPHERAL DEVICE TYPE field contains 0h, 4h, or 7h, the DESCRIPTOR LENGTH field shall contain 92 if the descriptor includes the DEVICE TYPE SPECIFIC DATA field and 80 if it does not. If the PERIPHERAL DEVICE TYPE field contains any value other than 0h, 4h, or 7h, the DESCRIPTOR LENGTH field shall contain 76. The relationship between the DESCRIPTOR LENGTH field and the CDB ALLOCATION LENGTH field is defined in 4.3.5.6.

The DEFAULT LUN field contains the default LUN value (see 8.3.1.4.3) for the logical unit described by this logical unit descriptor. The value in the DEFAULT LUN field shall be consistent with the DLGENERATION field value returned in the parameter list header (see 8.3.2.3.2). The value in the DEFAULT LUN field shall not identify a well known logical unit.

The EVPD DESIGNATION DESCRIPTOR LENGTH field indicates the number of non pad bytes in the EVPD DESIGNATION DESCRIPTOR field.

The DEVICE IDENTIFIER LENGTH field indicates the number of non pad bytes in the DEVICE IDENTIFIER field.

The EVPD DESIGNATION DESCRIPTOR field shall be derived from one of the Device Identification VPD page (see 7.8.6) designation descriptors having 00b in the ASSOCIATION field as follows:

a) if the designation descriptor has a length less than 32 bytes, then the EVPD DESIGNATION DESCRIPTOR field shall be set to the zero-padded (see 4.4.2) designation descriptor value. The EVPD DESIGNATION DESCRIPTOR LENGTH field shall be set to the length of the designation descriptor not including pad bytes; or
b) if the designation descriptor has a length greater than or equal to 32 bytes, then the EVPD DESIGNATION DESCRIPTOR field shall be set to the first 32 bytes of the designation descriptor. The EVPD DESIGNATION DESCRIPTOR LENGTH field shall be set to 32.

If there are several designation descriptors having 00b in the ASSOCIATION field, the choice of which descriptor to copy to the EVPD DESIGNATION DESCRIPTOR field is vendor specific, however, all ACCESS CONTROL IN commands with REPORT LU DESCRIPTORS service action shall return the same EVPD DESIGNATION DESCRIPTOR field contents for a specific logical unit.

If a device identifier has been set for the logical unit using the SET IDENTIFYING INFORMATION command (see 6.33), the DEVICE IDENTIFIER field shall contain that device identifier subject to the following considerations:

a) if the device identifier has length less than 32 bytes, then the DEVICE IDENTIFIER field shall be set to the zero-padded (see 4.4.2) device identifier value. The DEVICE IDENTIFIER LENGTH field shall be set to the length of the device identifier not including pad bytes; or
b) if the device identifier has length greater than or equal to 32 bytes, then the DEVICE IDENTIFIER field shall be set to the first 32 bytes of the identifier. The DEVICE IDENTIFIER LENGTH field shall be set to 32.

If no device identifier has been established by a SET IDENTIFYING INFORMATION command, then the DEVICE IDENTIFIER LENGTH field shall be set to zero and the DEVICE IDENTIFIER field shall be set to zero.

If the PERIPHERAL DEVICE TYPE field contains any value other than 0h, 4h, or 7h, the DEVICE TYPE SPECIFIC DATA field shall not be present in the Logical Unit descriptor.

The Logical Unit descriptor shall include the DEVICE TYPE SPECIFIC DATA field if:

a) the PERIPHERAL DEVICE TYPE field contains 0h, 4h, or 7h;
b) the logical unit supports the READ CAPACITY(16) command (see SBC-3); and
c) the logical unit standard INQUIRY data (see 6.4.2) has the RMB bit set to zero.

If the Logical Unit descriptor includes the DEVICE TYPE SPECIFIC DATA field, then the size of the DEVICE TYPE SPECIFIC DATA field shall be 12 bytes and the field shall contain data equivalent to the first 12 bytes of parameter data returned by a successful READ CAPACITY(16) command.

8.3.2.4 REPORT ACCESS CONTROLS LOG service action

8.3.2.4.1 REPORT ACCESS CONTROLS LOG introduction

The ACCESS CONTROL IN command with REPORT ACCESS CONTROLS LOG service action (see table 600) is used to obtain the access controls log (see 8.3.1.10). If the ACCESS CONTROL IN command is implemented, the REPORT ACCESS CONTROLS LOG service action shall be implemented.

Table 600 — ACCESS CONTROL IN command with REPORT ACCESS CONTROLS LOG service action

Bit / Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (86h)							
1	Reserved			SERVICE ACTION (02h)				
2	(MSB)			MANAGEMENT IDENTIFIER KEY				
•••								
9								(LSB)
10	Reserved						LOG PORTION	
11	Reserved							
12	(MSB)			ALLOCATION LENGTH				
13								(LSB)
14	Reserved							
15	CONTROL							

The OPERATION CODE field is defined in 4.3.5.1 and shall be set as shown in table 600 for the ACCESS CONTROL IN command with REPORT ACCESS CONTROLS LOG service action.

The SERVICE ACTION field is defined in 4.3.5.2 and shall be set as shown in table 600 for the ACCESS CONTROL IN command with REPORT ACCESS CONTROLS LOG service action.

If access controls are disabled, the device server shall ignore the MANAGEMENT IDENTIFIER KEY field and shall complete the command with GOOD status returning the eight byte parameter list header as specified in 8.3.2.4.2.1 subject to the ALLOCATION LENGTH limitation described in 4.3.5.6.

Since the Key Overrides portion of the log is maintained while access controls are disabled (see 8.3.3.3), it may be retrieved by enabling access controls and issuing an ACCESS CONTROL IN command with REPORT ACCESS CONTROLS LOG service action.

If access controls are enabled and table 601 specifies that the management identifier key is not required then the device server shall ignore the contents of the MANAGEMENT IDENTIFIER KEY field.

If access controls are enabled, table 601 specifies that the management key identifier is required, and the contents of the MANAGEMENT IDENTIFIER KEY field do not match the current management identifier key (see 8.3.1.8) maintained by the access controls coordinator, then the parameter data shall not be returned and the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to ACCESS DENIED - INVALID MGMT ID KEY. This event shall be recorded in the invalid keys portion of the access controls log (see 8.3.1.10).

The LOG PORTION field (see table 601) specifies the access controls log portion being requested.

Table 601 — CDB LOG PORTION field

Code	Description	Management Identifier Key Required
00b	Key Overrides portion	No
01b	Invalid Keys portion	Yes
10b	ACL LUN Conflicts portion	Yes
11b	Reserved	

The ALLOCATION LENGTH field is defined in 4.3.5.6. The ALLOCATION LENGTH field value should be at least eight.

The CONTROL field is defined in SAM-5.

8.3.2.4.2 REPORT ACCESS CONTROLS LOG parameter data format

8.3.2.4.2.1 REPORT ACCESS CONTROLS LOG parameter data introduction

The format of the parameter data returned in response to an ACCESS CONTROL IN command with REPORT ACCESS CONTROLS LOG service actions is shown in table 602.

Table 602 — ACCESS CONTROL IN with REPORT ACCESS CONTROLS LOG parameter data format

Bit / Byte	7	6	5	4	3	2	1	0
	Parameter list header							
0	(MSB)							
•••				LOG LIST LENGTH (n-3)				
3								(LSB)
4	Reserved							
5	Reserved						LOG PORTION	
6	(MSB)							
7				COUNTER				(LSB)
	Access Controls Log portion pages							
8								
•••				Access Controls Log portion page 0				
				⋮				
•••				Access Controls Log portion page x				
n								

The LOG LIST LENGTH field shall contain a count of the number of bytes in the remaining parameter data. If access controls are disabled, the LOG LIST LENGTH field shall be set to four. The relationship between the LOG LIST LENGTH field and the CDB ALLOCATION LENGTH field is defined in 4.3.5.6.

The LOG PORTION field (see table 603) indicates the access controls log portion being returned, the contents of the COUNTER field, and the type of access controls log portion pages being returned.

Table 603 — Parameter data LOG PORTION field

Code	Access Controls Log Portion Being Returned	COUNTER Field Contents	Access Controls Log Page Format Reference
00b	Key Overrides portion	Key Override events counter	8.3.2.4.2.2
01b	Invalid Keys portion	Invalid Key events counter	8.3.2.4.2.3
11b	ACL LUN Conflicts portion	ACL LUN Conflict events counter	8.3.2.4.2.4
11b	Reserved		

The COUNTER field contains the events counter value (see 8.3.1.10) for the access controls log portion indicated by the LOG PORTION field (see table 603).

The format of the access controls log portion pages is indicated by the value in the LOG PORTION field (see table 603). All the access controls log portion pages returned in a single parameter list shall have the same format. If the access controls coordinator does not support access controls log portion pages in the portion of the access controls log indicated by the LOG PORTION field, then the parameter data shall only contain the parameter list header.

8.3.2.4.2.2 Key Overrides access controls log portion page format

The Key Overrides access controls log portion page (see table 604) contains details of logged attempts to override the management identifier key (see 8.3.1.10) using the ACCESS CONTROL OUT command with OVERRIDE MGMT ID KEY service action (see 8.3.3.8) whether those attempts were successful or not.

Table 604 — Key Overrides access controls log portion page format

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)			TRANSPORTID ADDITIONAL LENGTH (m-32)				
1								(LSB)
2				Reserved				
3				Reserved				SUCCESS
4	(MSB)			TIME STAMP				
•••								
7								(LSB)
8				TRANSPORTID				
•••								
m-1								
m	(MSB)			INITIAL OVERRIDE LOCKOUT TIMER				
m+1								(LSB)
m+2	(MSB)			OVERRIDE LOCKOUT TIMER				
m+3								(LSB)

The TRANSPORTID ADDITIONAL LENGTH field indicates the additional length of the TRANSPORTID field beyond the minimum length of 24 bytes. The TransportID additional length shall be a multiple of four.

A SUCCESS bit set to one indicates that the specific ACCESS CONTROL OUT command with OVERRIDE MGMT ID KEY service action event recorded in the access controls log successfully overrode the management identifier key. A SUCCESS bit set to zero indicates that the command did not succeed.

The TIME STAMP field shall contain zero or an indication of the time at which the ACCESS CONTROL OUT command with OVERRIDE MGMT ID KEY service action was processed as described in 8.3.1.10.

The TRANSPORTID field shall contain the TransportID of the initiator port from which the command was received.

The INITIAL OVERRIDE LOCKOUT TIMER field shall contain the access controls coordinator's initial override lockout timer value (see 8.3.1.8.2.2) at the time when the key override event was logged.

The OVERRIDE LOCKOUT TIMER field shall contain the access controls coordinator's override lockout timer value (see 8.3.1.8.2.2) at the time when the key override event was logged.

8.3.2.4.2.3 Invalid Keys access controls log portion page format

The Invalid Keys access controls log portion page (see table 605) contains details of logged receipts of ACCESS CONTROL IN or ACCESS CONTROL OUT commands specifying an incorrect management identifier key (see 8.3.1.10).

Table 605 — Invalid Keys access controls log portion page format

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)	TRANSPORTID ADDITIONAL LENGTH (m-32)						
1								(LSB)
2	OPERATION CODE							
3	Reserved			SERVICE ACTION				
4	(MSB)	TIME STAMP						
•••								
7								(LSB)
8		TRANSPORTID						
•••								
m-1								
m	(MSB)	INVALID MANAGEMENT IDENTIFIER KEY						
•••								
m+7								(LSB)

The TRANSPORTID ADDITIONAL LENGTH field indicates the additional length of the TRANSPORTID field beyond the minimum length of 24 bytes. The TransportID additional length shall be a multiple of four.

The OPERATION CODE and SERVICE ACTION fields shall be set to the respective values from the CDB of the access controls command that specified the invalid management identifier key.

The TIME STAMP field shall contain zero or an indication of the time at which the ACCESS CONTROL IN or ACCESS CONTROL OUT command was processed as described in 8.3.1.10.

The TRANSPORTID field shall contain the TransportID of the initiator port from which the command was received.

The INVALID MANAGEMENT IDENTIFIER KEY field shall be set to the value of the invalid management identifier key detected by the access controls coordinator.

NOTE 82 - The management identifier key is typically in the CDB for ACCESS CONTROL IN commands and in the parameter data for ACCESS CONTROL OUT commands.

8.3.2.4.2.4 ACL LUN Conflicts access controls log portion page format

The ACL LUN Conflicts access controls log portion page (see table 606) contains details of logged ACL LUN conflicts (see 8.3.1.10) encountered by the access controls coordinator when a previously not-enrolled initiator port sends an ACCESS CONTROL OUT command with ACCESS ID ENROLL service action (see 8.3.3.4).

Table 606 — ACL LUN Conflicts access controls log portion page format

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)			TRANSPORTID ADDITIONAL LENGTH (m-32)				
1								(LSB)
2				Reserved				
3								
4	(MSB)			TIME STAMP				
•••								
7								(LSB)
8				TRANSPORTID				
•••								
m-1								
m	(MSB)			ACCESSID				
•••								
m+23								(LSB)

The TRANSPORTID ADDITIONAL LENGTH field indicates the additional length of the TRANSPORTID field beyond the minimum length of 24 bytes. The TransportID additional length shall be a multiple of four.

The TIME STAMP field shall contain zero or an indication of the time at which the ACCESS CONTROL OUT command with ACCESS ID ENROLL service action was processed as described in 8.3.1.10.

The TRANSPORTID field shall contain the TransportID of the initiator port from which the command was received that resulted in the ACL LUN conflict.

The ACCESSID field shall be set to the AccessID that the initiator port attempted to enroll. This shall correspond to an access identifier in ACL entry at the time the ACL LUN conflict event occurred.

8.3.2.5 REPORT OVERRIDE LOCKOUT TIMER service action

The ACCESS CONTROL IN command with REPORT OVERRIDE LOCKOUT TIMER service action (see table 607) is used query the value of the override lockout timer (see 8.3.1.8.2.2). If the ACCESS CONTROL IN command is implemented, the REPORT OVERRIDE LOCKOUT TIMER service action shall be implemented.

Table 607 — ACCESS CONTROL IN command with REPORT OVERRIDE LOCKOUT TIMER service action

Bit / Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (86h)							
1	Reserved			SERVICE ACTION (03h)				
2	(MSB)			MANAGEMENT IDENTIFIER KEY				
•••								
9								(LSB)
10	(MSB)			ALLOCATION LENGTH				
•••								
13								(LSB)
14	Reserved							
15	CONTROL							

The OPERATION CODE field is defined in 4.3.5.1 and shall be set as shown in table 607 for the ACCESS CONTROL IN command with REPORT OVERRIDE LOCKOUT TIMER service action.

The SERVICE ACTION field is defined in 4.3.5.2 and shall be set as shown in table 607 for the ACCESS CONTROL IN command with REPORT OVERRIDE LOCKOUT TIMER service action.

If access controls are disabled, eight bytes of zeros shall be returned subject to the allocation length limitations described in 4.3.5.6 and GOOD status shall be returned.

If access controls are enabled and the contents of the MANAGEMENT IDENTIFIER KEY field do not match the current management identifier key (see 8.3.1.8) maintained by the access controls coordinator, then parameter data shall not be returned, the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to ACCESS DENIED - INVALID MGMT ID KEY. This event shall be recorded in the invalid keys portion of the access controls log (see 8.3.1.10).

The ALLOCATION LENGTH field is defined in 4.3.5.6. The ALLOCATION LENGTH field value should be at least eight.

The CONTROL field is defined in SAM-5.

If access controls are enabled, the parameter data returned by the ACCESS CONTROL IN command with REPORT OVERRIDE LOCKOUT TIMER service action shall have the format shown in table 608.

Table 608 — ACCESS CONTROL IN with REPORT OVERRIDE LOCKOUT TIMER parameter data

Bit / Byte	7	6	5	4	3	2	1	0
0	Reserved							
1								
2	(MSB)	CURRENT OVERRIDE LOCKOUT TIMER						
3								(LSB)
4	(MSB)	INITIAL OVERRIDE LOCKOUT TIMER						
5								(LSB)
6	(MSB)	KEY OVERRIDES COUNTER						
7								(LSB)

The CURRENT OVERRIDE LOCKOUT TIMER field shall be set to the current value of the override lockout timer (see 8.3.1.8.2.2).

The INITIAL OVERRIDE LOCKOUT TIMER field shall be set to the initial override lockout timer value (see 8.3.1.8.2.2) established by the last successful ACCESS CONTROL OUT command with MANAGE OVERRIDE LOCKOUT TIMER service action (see 8.3.3.7).

The KEY OVERRIDES COUNTER field shall be set to the value of the Key Override events counter in the access controls log (see 8.3.1.10).

8.3.2.6 REQUEST PROXY TOKEN service action

The ACCESS CONTROL IN command with REQUEST PROXY TOKEN service action (see table 609) is used to obtain a proxy token (see 8.3.1.6.2) for a logical unit to which that initiator port has non-proxy access rights. The returned proxy token may be used to pass temporary access to the logical unit to a third party that may use other proxy related service actions of the ACCESS CONTROL IN and ACCESS CONTROL OUT commands to gain

access to the logical unit. If the ACCESS CONTROL IN command with REQUEST PROXY TOKEN service action is not supported, 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.

Table 609 — ACCESS CONTROL IN command with REQUEST PROXY TOKEN service action

Bit / Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (86h)							
1	Reserved			SERVICE ACTION (04h)				
2	(MSB)			LUN VALUE				
•••								
9								(LSB)
10	(MSB)			ALLOCATION LENGTH				
•••								
13								(LSB)
14	Reserved							
15	CONTROL							

The OPERATION CODE field is defined in 4.3.5.1 and shall be set as shown in table 609 for the ACCESS CONTROL IN command with REQUEST PROXY TOKEN service action.

The SERVICE ACTION field is defined in 4.3.5.2 and shall be set as shown in table 609 for the ACCESS CONTROL IN command with REQUEST PROXY TOKEN service action.

If access controls are disabled, 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.

NOTE 83 - If access controls are disabled, all logical units are accessible and all initiator ports share the same LUN values for addressing. A proxy token is not needed because sharing LUN values is sufficient.

The LUN VALUE field shall contain the LUN value the application client uses to access the logical unit via the initiator port over which the proxy token is requested.

If the LUN value corresponds to a logical unit that is accessible to the requesting initiator port either through a TransportID or through the AccessID under which the initiator port is currently in the enrolled state (see 8.3.1.5.1), and the access controls coordinator has sufficient resources to create and manage a new proxy token, then the parameter data shown in table 610 shall be returned.

If the LUN value does not correspond to an accessible logical unit or corresponds to a logical unit accessible only through a proxy token, then the parameter data shall not be returned and the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to ACCESS DENIED - INVALID LU IDENTIFIER.

If the LUN value corresponds to a logical unit accessible only through an enrolled AccessID and the initiator port is in the pending-enrolled state, then the parameter data shall not be returned and the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to ACCESS DENIED - INITIATOR PENDING-ENROLLED.

If the access controls coordinator does not have enough resources to create and manage a new proxy token, the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INSUFFICIENT ACCESS CONTROL RESOURCES.

The ALLOCATION LENGTH field is defined in 4.3.5.6. The ALLOCATION LENGTH field value should be at least eight.

The CONTROL field is defined in SAM-5.

The format of the parameter data returned by the ACCESS CONTROL IN command with REQUEST PROXY TOKEN service action is shown in table 610.

Table 610 — ACCESS CONTROL IN with REQUEST PROXY TOKEN parameter data

Bit / Byte	7	6	5	4	3	2	1	0
0	PROXY TOKEN							
•••								
7								

8.3.3 ACCESS CONTROL OUT command

8.3.3.1 ACCESS CONTROL OUT introduction

The service actions of the ACCESS CONTROL OUT command (see table 611) are used to request service actions by the access controls coordinator to limit or grant access to the logical units by initiator ports. If the ACCESS CONTROL OUT command is implemented, the ACCESS CONTROL IN command also shall be implemented. The ACCESS CONTROL OUT command shall not be affected by access controls.

Table 611 — ACCESS CONTROL OUT service actions

Service Action	Name	Type	Reference
00h	MANAGE ACL	M	8.3.3.2
01h	DISABLE ACCESS CONTROLS	M	8.3.3.3
02h	ACCESS ID ENROLL	M	8.3.3.4
03h	CANCEL ENROLLMENT	M	8.3.3.5
04h	CLEAR ACCESS CONTROLS LOG	M	8.3.3.6
05h	MANAGE OVERRIDE LOCKOUT TIMER	M	8.3.3.7
06h	OVERRIDE MGMT ID KEY	M	8.3.3.8
07h	REVOKE PROXY TOKEN	O	8.3.3.9
08h	REVOKE ALL PROXY TOKENS	O	8.3.3.10
09h	ASSIGN PROXY LUN	O	8.3.3.11
0Ah	RELEASE PROXY LUN	O	8.3.3.12
0Bh to 17h	Reserved		
18h to 1Fh	Vendor specific		

Key: M = Service action implementation is mandatory if ACCESS CONTROL OUT is implemented.
O = Service action implementation is optional.

The ACCESS CONTROL OUT command may be addressed to any logical unit whose standard INQUIRY data (see 6.4.2) has the ACC bit set to one (e.g., LUN 0), in which case it shall be processed in the same manner as if the command had been addressed to the ACCESS CONTROLS well known logical unit. If an ACCESS CONTROL OUT command is received by a device server whose standard INQUIRY data has the ACC bit set to zero, the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID COMMAND OPERATION CODE.

If an ACCESS CONTROL OUT command is received while an IKEv2-SCSI CCS is in progress (see 5.16.4), the command shall be terminated with CHECK CONDITION status, with the sense key NOT READY, and the additional sense code set to LOGICAL UNIT NOT READY, SA CREATION IN PROGRESS. The sense key specific additional sense data may be set as described in 5.16.5.

The CDB format used by all ACCESS CONTROL OUT service actions is shown in table 612.

Table 612 — ACCESS CONTROL OUT command format

Bit / Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (87h)							
1	Reserved			SERVICE ACTION (see table 611)				
2	Reserved							
•••								
9								
10	(MSB)	PARAMETER LIST LENGTH						
•••								
13								(LSB)
14	Reserved							
15	CONTROL							

The OPERATION CODE field is defined in 4.3.5.1 and shall be set as shown in table 612 for the ACCESS CONTROL OUT command.

The SERVICE ACTION field is defined in 4.3.5.2 and table 611.

The PARAMETER LIST LENGTH field indicates the amount of data being sent to the access controls coordinator in the Data-Out Buffer. The format of the parameter list is specific to each service action.

The CONTROL field is defined in SAM-5.

8.3.3.2 MANAGE ACL service action

8.3.3.2.1 MANAGE ACL introduction

The ACCESS CONTROL OUT command with MANAGE ACL service action is used to authorize access or revoke access to a logical unit or logical units by initiator ports. The ACCESS CONTROL OUT command with MANAGE ACL service action adds, changes or removes an entry or multiple entries in the access controls coordinator's ACL (see 8.3.1.3). If the ACCESS CONTROL OUT command is implemented, the MANAGE ACL service action shall be implemented.

The format of the CDB for the ACCESS CONTROL OUT command with MANAGE ACL service action is shown in table 612 (see 8.3.3.1).

If the PARAMETER LIST LENGTH field in the CDB is zero, the access controls coordinator shall take no action and the command shall be completed with GOOD status.

If the value in the PARAMETER LIST LENGTH field is less than 20 or results in truncation of any ACE page (see table 614), 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 PARAMETER LIST LENGTH ERROR.

If the access controls coordinator is unable to complete the ACCESS CONTROL OUT command with MANAGE ACL service action because it has insufficient resources, then the access controls coordinator shall take no action and not change any of its state and the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INSUFFICIENT ACCESS CONTROL RESOURCES.

The format of the parameter data for the ACCESS CONTROL OUT command with MANAGE ACL service action is shown in table 613.

Table 613 — ACCESS CONTROL OUT with MANAGE ACL parameter data format

Bit / Byte	7	6	5	4	3	2	1	0
	Parameter list header							
0	Reserved							
•••								
3								
4	(MSB)	MANAGEMENT IDENTIFIER KEY						
•••								
11								(LSB)
12	(MSB)	NEW MANAGEMENT IDENTIFIER KEY						
•••								
19								(LSB)
20	Reserved							
21	FLUSH	Reserved						
22	Reserved							
23	Reserved							
24	(MSB)	DLGENERATION						
•••								
27								(LSB)
	ACE pages							
28	ACE page 0							
•••								
	⋮							
	ACE page x							
•••								
n								

If access controls are enabled and the contents of the MANAGEMENT IDENTIFIER KEY field do not match the current management identifier key (see 8.3.1.8) maintained by the access controls coordinator, then the access controls coordinator's state shall not be altered, the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to ACCESS DENIED - INVALID MGMT ID KEY. This event shall be recorded in the invalid keys portion of the access controls log (see 8.3.1.10).

If the contents of the MANAGEMENT IDENTIFIER KEY field match the current management identifier key maintained by the access controls coordinator, the access controls coordinator shall set its management identifier key to the value specified in the NEW MANAGEMENT IDENTIFIER KEY field and if access controls are disabled it shall enable them.

The FLUSH bit set to one instructs the access controls coordinator to place every initiator port in the enrolled state into the pending-enrolled state (see 8.3.1.5.1.4).

The DLGENERATION field specifies the DLgeneration value (see 8.3.1.4.4) associated with the default LUN values in the Grant/Revoke ACE pages in the parameter data.

The ACE pages that may follow the parameter list header provide additional changes to the ACL. Each ACE page describes one ACE in the ACL that is to be added, modified, or removed. The content and format of an ACE page is indicated by a page code (see table 614).

Table 614 — ACE page codes

Page Code	ACE Page Name	Reference
00h	Grant/Revoke	8.3.3.2.2
01h	Grant All	8.3.3.2.3
02h	Revoke Proxy Token	8.3.3.2.4
03h	Revoke All Proxy Tokens	8.3.3.2.5
04h to EFh	Reserved	
F0h to FFh	Vendor specific	

The following requirements apply to the processing of changes to the access control state:

a) no change to the access control state shall occur if the ACCESS CONTROL OUT command with MANAGE ACL service action completes with a status other than GOOD status; and
b) if the ACCESS CONTROL OUT command with MANAGE ACL service action completes with GOOD status, the following shall have been performed as a single indivisible event:
   1) changes resulting from the contents of fields in the parameter list header shall be processed; and
   2) changes resulting from the contents of ACE pages shall be processed;
      a) multiple ACE pages shall be processed sequentially;
      b) if an ACE page contains conflicting instructions in LUACD descriptors, the instructions in the last LUACD descriptor within the ACE page shall take precedence; and
      c) if an ACE containing an AccessID type access identifier (see 8.3.1.3.2) is replaced and the ACE page that caused the change has the NOCNCL bit (see 8.3.3.2.2) set to zero, then any initiator port in the enrolled state or pending-enrolled state under the AccessID in that ACE shall be placed in the not-enrolled state (see 8.3.1.5.1.2).

An ACE page contains conflicting instructions if either of the following is true:

a) two LUACD descriptors are present with the same LUN value and different default LUN values; or
b) two LUACD descriptors are present with different LUN values and the same default LUN value.

8.3.3.2.2 The Grant/Revoke ACE page

The Grant/Revoke ACE page (see table 615) is used to add, modify, or remove an ACE from the ACL (see 8.3.1.3).

Table 615 — Grant/Revoke ACE page format

Bit / Byte	7	6	5	4	3	2	1	0
0	PAGE CODE (00h)							
1	Reserved							
2	(MSB)	PAGE LENGTH (n-3)						
3								(LSB)
4	NOCNCL	Reserved						
5	ACCESS IDENTIFIER TYPE							
6	(MSB)	ACCESS IDENTIFIER LENGTH (m-7)						
7								(LSB)
8	ACCESS IDENTIFIER							
•••								
m								
	LUACD descriptors							
m+1	LUACD descriptor 0							
•••								
m+20								
	⋮							
n-19	LUACD descriptor x							
•••								
n								

The PAGE LENGTH field specifies the number of additional bytes present in this ACE page.

A no changes to current logical unit access (NOCNCL) bit set to one specifies that the application client is telling the access controls coordinator that this ACE page makes no changes to the existing logical unit access conditions in the ACL. A NOCNCL bit set to zero specifies that the ACE page may or may not change existing logical unit access conditions. If the ACCESS IDENTIFIER TYPE specifies a TransportID (see 8.3.2.2.2.2), the NOCNCL bit shall be ignored.

The ACCESS IDENTIFIER TYPE and ACCESS IDENTIFIER LENGTH fields are described in 8.3.2.2.2.2.

The ACCESS IDENTIFIER field contains the identifier that the access controls coordinator uses to select the ACE that is to be added, modified, or removed. The format of the ACCESS IDENTIFIER field is specified in table 584 (see 8.3.1.13).

Any of the following conditions in the parameter header or any Grant/Revoke ACE page or Grant All ACE page shall cause the access coordinator to not change its state and shall cause the command to 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:

a) the value in the DLGENERATION field in the parameter list header (see 8.3.3.2.1) does not match the current value of the DLgeneration counter (see 8.3.1.4.4) maintained by the access controls coordinator;
b) an ACCESS IDENTIFIER TYPE field specifies an unsupported value;
c) an ACCESS IDENTIFIER TYPE field contains 01h (see 8.3.1.3.2) with an ACCESS IDENTIFIER field that contains an invalid TransportID (see 8.3.1.3.2) as defined for the applicable protocol standard;
d) two ACE pages that have the same values in the ACCESS IDENTIFIER TYPE and ACCESS IDENTIFIER fields; or
e) changes in the ACL that result in an ACL LUN conflict (see 8.3.1.5.2).

NOTE 84 - The application client is responsible for obtaining the current association of default LUN values to logical units and the DLgeneration value for that association prior to issuing this service action. The ACCESS CONTROL IN command with REPORT LU DESCRIPTORS service action (see 8.3.2.3) returns the necessary information.

Each LUACD descriptor (see table 616) describes the access to be allowed to one logical unit based on the access identifier in the ACE page. An ACE page may contain zero or more LUACD descriptors.

Table 616 — ACE page LUACD descriptor format

Bit / Byte	7	6	5	4	3	2	1	0
0	ACCESS MODE							
1	Reserved							
2								
3								
4	LUN VALUE							
•••								
11								
12	DEFAULT LUN							
•••								
19								

The ACCESS MODE field is described in 8.3.2.2.2.2.

The LUN VALUE field specifies the LUN value an accessing application client uses to access the logical unit via the initiator port to which the LUACD descriptor applies.

The DEFAULT LUN field specifies the logical unit to which the value in the LUN VALUE allows access. The DEFAULT LUN field shall contain a default LUN value (see 8.3.1.4.3). The value in the DEFAULT LUN field shall be consistent with the DLGENERATION field value specified in the parameter list header (see 8.3.3.2.1). If the DEFAULT LUN field references a well known logical unit, the access controls coordinator's state shall not be modified and 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.

If the specified access mode is not supported or if the DEFAULT LUN field contains value that is not valid or the LUN VALUE field contains a value that the access controls coordinator does not support as a valid LUN, then the access controls coordinator's state shall not be modified and the command shall be terminated with CHECK CONDITION

status, with the sense key set to ILLEGAL REQUEST, the additional sense code set to ACCESS DENIED - INVALID LU IDENTIFIER, and the SENSE-KEY SPECIFIC field shall be set as described for the ILLEGAL REQUEST sense key in 4.5.2.4.2. If the error is an unsupported value in the LUN VALUE field, then the access controls coordinator should determine a suggested LUN value that is unlikely to produce an error while also minimizing the absolute value of the difference of the erroneous default LUN value and the suggested LUN value. If a suggested LUN value is determined, the first four bytes of the suggested LUN value shall be placed in the INFORMATION field and the last four bytes shall be placed in the COMMAND-SPECIFIC INFORMATION field of the sense data (see 4.5).

Based on the access identifier and the presence or absence of LUACD descriptors, the access controls coordinator shall add, modify, or remove an ACE in the ACL as shown in table 617.

Table 617 — Access Coordinator Grant/Revoke ACE page actions

		ACL already contains an ACE with the access identifier matching the one in the ACE page?	
		Yes	**No**
ACE page includes LUCAD descriptors?	**Yes**	Modify the existing ACE in the ACL.	Add a new ACE to the ACL.
	No	Remove the existing ACE from the ACL.	Take no action; this shall not be considered an error.

If the ACCESS IDENTIFIER TYPE indicates type AccessID, the enrollment state (see 8.3.1.5.1) of any initiator port that is enrolled under the specified AccessID, shall be affected as follows:

a) if the ACE containing the AccessID is removed, the initiator port shall be placed in the not-enrolled state; or
b) if the ACE containing the AccessID is modified by a Grant/Revoke ACE page or a Grant All ACE page, then;
   A) if the NOCNCL bit is set to zero in that ACE page, the initiator port shall be placed in the not-enrolled state; or
   B) if the NOCNCL bit is set to one in that ACE page, the enrollment state of the initiator port may be left unchanged or the initiator port may be placed in the not-enrolled state (see 8.3.1.5.1.2) based on vendor specific considerations.

8.3.3.2.3 The Grant All ACE page

The Grant All ACE page (see table 618) is used to add or modify an ACE in the ACL (see 8.3.1.3). An ACE added or modified using the Grant All ACE page allows initiator ports with the specified access identifier to access the SCSI target device as if access controls were disabled.

Table 618 — Grant All ACE page format

Bit / Byte	7	6	5	4	3	2	1	0
0	PAGE CODE (01h)							
1	Reserved							
2	(MSB)	PAGE LENGTH (n-3)						
3								(LSB)
4	NOCNCL	Reserved						
5	ACCESS IDENTIFIER TYPE							
6	(MSB)	ACCESS IDENTIFIER LENGTH (m-7)						
7								(LSB)
8	ACCESS IDENTIFIER							
•••								
n								

The PAGE LENGTH, NOCNCL, ACCESS IDENTIFIER TYPE, ACCESS IDENTIFIER LENGTH, and ACCESS IDENTIFIER fields are defined in 8.3.3.2.2.

The Grant All ACE page shall be processed as if it is a Grant/Revoke ACE page (see 8.3.3.2.2) with one LUACD descriptor for every logical unit managed by the access controls coordinator with the fields in each LUACD containing:

a) an access mode of 00h (see 8.3.2.2.2.2);
b) a LUN VALUE field whose contents match the contents of the DEFAULT LUN field; and
c) a DEFAULT LUN field whose contents reference the logical unit appropriate to the DLgeneration value (see 8.3.1.4.4).

8.3.3.2.4 The Revoke Proxy Token ACE page

The Revoke Proxy Token ACE page (see table 619) is used to revoke one or more proxy tokens (see 8.3.1.6.2).

Table 619 — Revoke Proxy Token ACE page format

Bit / Byte	7	6	5	4	3	2	1	0
0	PAGE CODE (02h)							
1	Reserved							
2	(MSB)	PAGE LENGTH (n-3)						
3								(LSB)
4	PROXY TOKEN 0							
•••								
11								
	⋮							
n-7	PROXY TOKEN x							
•••								
n								

The PAGE LENGTH field specifies the number of additional bytes present in this ACE page. If the page length is less than eight or not a multiple of eight, the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense be set to PARAMETER LIST LENGTH ERROR.

The PROXY TOKEN field(s) specify the proxy tokens to be revoked. The access controls coordinator shall revoke each proxy token listed in a PROXY TOKEN field. If the contents of a PROXY TOKEN field do not identify a valid proxy token the field shall be ignored and this shall not be considered an error.

Multiple Revoke Proxy Token ACE pages may be included in the parameter data.

8.3.3.2.5 The Revoke All Proxy Tokens ACE page

The Revoke All Proxy Tokens ACE page (see table 620) is used to revoke all currently valid proxy tokens (see 8.3.1.6.2).

Table 620 — Revoke All Proxy Tokens ACE page format

Bit / Byte	7	6	5	4	3	2	1	0
0	PAGE CODE (03h)							
1	Reserved							
2	(MSB)	PAGE LENGTH (0000h)						
3								(LSB)

Multiple Revoke All Proxy Tokens ACE pages may be included in the parameter data.

8.3.3.3 DISABLE ACCESS CONTROLS service action

The ACCESS CONTROL OUT command with DISABLE ACCESS CONTROLS service action is used to place the access controls coordinator in the access controls disabled state. If the ACCESS CONTROL OUT command is implemented, the DISABLE ACCESS CONTROLS service action shall be implemented.

The format of the CDB for the ACCESS CONTROL OUT command with DISABLE ACCESS CONTROLS service action is shown in table 612 (see 8.3.3.1).

If access controls are disabled or if the PARAMETER LIST LENGTH field in the CDB is zero, the access controls coordinator shall take no action and the command shall be completed with GOOD status.

If the value in the PARAMETER LIST LENGTH field is neither zero nor 12, the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to PARAMETER LIST LENGTH ERROR.

If the value in the PARAMETER LIST LENGTH field is 12, the parameter list shall have the format shown in table 621.

Table 621 — ACCESS CONTROL OUT with DISABLE ACCESS CONTROLS parameter data format

Bit / Byte	7	6	5	4	3	2	1	0
0	Reserved							
•••								
3								
4	(MSB)	MANAGEMENT IDENTIFIER KEY						
•••								
11								(LSB)

If access controls are enabled and the contents of the MANAGEMENT IDENTIFIER KEY field do not match the current management identifier key (see 8.3.1.8) maintained by the access controls coordinator, then the access controls coordinator's states shall not be altered, the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to ACCESS DENIED - INVALID MGMT ID KEY. This event shall be recorded in the invalid keys portion of the access controls log (see 8.3.1.10).

In response to an ACCESS CONTROL OUT command with DISABLE ACCESS CONTROLS service action with correct management identifier key value the access controls coordinator shall:

a) disable access controls;
b) clear the ACL (see 8.3.1.3);
c) place all initiator ports into the not-enrolled state (see 8.3.1.5.1);
d) set the management identifier key to zero (see 8.3.1.8);
e) set the override lockout timer to zero (see 8.3.1.8.2.2);
f) set the initial override lockout timer value to zero (see 8.3.1.8.2.2);
g) clear the access controls log, including resetting the events counters to zero, with the exception of the key overrides portion of the access controls log (see 8.3.1.10);
h) allow all initiator port's access to all logical units at their default LUN value;
i) optionally, set the DLgeneration counter to zero (see 8.3.1.4.4); and
j) establish a unit attention condition for the initiator port associated with every I_T nexus in each logical unit in the SCSI target device, with the additional sense code set to REPORTED LUNS DATA HAS CHANGED.

8.3.3.4 ACCESS ID ENROLL service action

The ACCESS ID ENROLL service action of the ACCESS CONTROL OUT command is used by an application client to enroll an AccessID for an initiator port with the access controls coordinator. If the ACCESS CONTROL OUT command is implemented, the ACCESS ID ENROLL service action shall be implemented.

The format of the CDB for the ACCESS CONTROL OUT command with ACCESS ID ENROLL service action is shown in table 612 (see 8.3.3.1).

If access controls are disabled or if the PARAMETER LIST LENGTH field in the CDB is zero, the access controls coordinator shall take no action and the command shall be completed with GOOD status.

If the value in the PARAMETER LIST LENGTH field is neither zero nor 24, the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to PARAMETER LIST LENGTH ERROR.

If the value in the PARAMETER LIST LENGTH field is 24, the parameter list shall have the format shown in table 622.

Table 622 — ACCESS CONTROL OUT with ACCESS ID ENROLL parameter data format

Bit / Byte	7	6	5	4	3	2	1	0
0								
•••				ACCESSID				
15								
16								
•••				Reserved				
23								

The ACCESSID field is described in 8.3.1.3.2.

If the initiator port is in the enrolled state or pending-enrolled state (see 8.3.1.5.1) under a specific AccessID and the ACCESSID field contains a different AccessID, then the access controls coordinator shall place the initiator port in the pending-enrolled state, the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to ACCESS DENIED - ENROLLMENT CONFLICT.

If the initiator port is in the enrolled state or pending-enrolled state under a specific AccessID and the ACCESSID field contains a matching AccessID, the access controls coordinator shall place the initiator port in the enrolled state and make no other changes.

If the initiator port is in the not-enrolled state and the ACCESSID field contents do not match the AccessID in any ACE in the ACL (see 8.3.1.3), then the initiator port shall remain in the not-enrolled state and the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to ACCESS DENIED - NO ACCESS RIGHTS.

If the initiator port is in the not-enrolled state and the ACCESSID field contents matches the AccessID in an ACE in the ACL, the actions taken depend on whether enrolling the initiator port would create an ACL LUN conflict (see 8.3.1.5.2). If there is no ACL LUN conflict, the initiator port shall be placed in the enrolled state (see 8.3.1.5.1.3). If there is an ACL LUN conflict, then the initiator port shall remain in the not-enrolled state and the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to ACCESS DENIED - ACL LUN CONFLICT. This event shall be recorded in the ACL LUN conflicts portion of the access controls log (see 8.3.1.10).

An application client that receives the ACCESS DENIED - ACL LUN CONFLICT additional sense code should remove any proxy access rights it has acquired using the ACCESS CONTROL OUT command with RELEASE PROXY LUN service action and retry the enrollment request. If the ACL LUN conflict resulted from proxy access, the retried enrollment succeeds. Otherwise, the mechanisms for resolving ACL LUN conflicts are outside the scope of this standard.

8.3.3.5 CANCEL ENROLLMENT service action

The ACCESS CONTROL OUT command with CANCEL ENROLLMENT service action is used to remove an initiator port's enrollment with the access controls coordinator (see 8.3.1.5). Successful completion of this command changes the state of the initiator port to the not-enrolled state. If the ACCESS CONTROL OUT command is implemented, the CANCEL ENROLLMENT service action shall be implemented.

The ACCESS CONTROL OUT command with CANCEL ENROLLMENT service action should be used by an application client prior to any period where use of its accessible logical units may be suspended for a lengthy period of time (e.g., when a host is preparing to shutdown). This allows the access controls coordinator to free any resources allocated to manage the enrollment for the initiator port.

The format of the CDB for the ACCESS CONTROL OUT command with CANCEL ENROLLMENT service action is shown in table 612 (see 8.3.3.1).

If access controls are disabled, the access controls coordinator shall take no action and the command shall be completed with GOOD status.

There is no parameter data for the ACCESS CONTROL OUT command with CANCEL ENROLLMENT service action. If the PARAMETER LIST LENGTH field in the CDB is not set to zero, the initiator port's enrollment shall not be changed and the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to PARAMETER LIST LENGTH ERROR.

If the PARAMETER LIST LENGTH field in the CDB is set to zero, the initiator port shall be placed in the not-enrolled state (see 8.3.1.5.1.2) Any subsequent commands addressed to the logical units no longer accessible are handled according to the requirements stated in 8.3.1.7.

8.3.3.6 CLEAR ACCESS CONTROLS LOG service action

The ACCESS CONTROL OUT command with CLEAR ACCESS CONTROLS LOG service action is used to instruct the access controls coordinator to reset a specific access control events counter to zero and to clear a portion of the access controls log (see 8.3.1.10). If the ACCESS CONTROL OUT command is implemented, the CLEAR ACCESS CONTROLS LOG service action shall be implemented.

The format of the CDB for the ACCESS CONTROL OUT command with CLEAR ACCESS CONTROLS LOG service action is shown in table 612 (see 8.3.3.1).

If access controls are disabled or if the PARAMETER LIST LENGTH field in the CDB is zero, the access controls coordinator shall take no action and the command shall be completed with GOOD status.

If the value in the PARAMETER LIST LENGTH field is neither zero nor 12, the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to PARAMETER LIST LENGTH ERROR.

If the value in the PARAMETER LIST LENGTH field is 12, the parameter list shall have the format shown in table 623.

Table 623 — ACCESS CONTROL OUT with CLEAR ACCESS CONTROLS LOG parameter data format

Bit / Byte	7	6	5	4	3	2	1	0
0	Reserved							
1								
2								
3	Reserved						LOG PORTION	
4	(MSB)	MANAGEMENT IDENTIFIER KEY						
•••								
11								(LSB)

The LOG PORTION field (see table 624) specifies the access controls log portion to be cleared.

Table 624 — CLEAR ACCESS CONTROLS LOG LOG PORTION field

Code	Description
00b	Reserved
01b	Invalid Keys portion
10b	ACL LUN Conflicts portion
11b	Reserved

If access controls are enabled and the contents of the MANAGEMENT IDENTIFIER KEY field do not match the current management identifier key (see 8.3.1.8) maintained by the access controls coordinator, then the access controls coordinator's states shall not be altered, the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to ACCESS DENIED - INVALID MGMT ID KEY. This event shall be recorded in the invalid keys portion of the access controls log (see 8.3.1.10).

In response to an ACCESS CONTROL OUT command with CLEAR ACCESS CONTROLS LOG service action with correct management identifier key value the access controls coordinator shall perform the following to clear the portion of the access controls log identified by the LOG PORTION field (see table 624) in the parameter data:

a) set the events counter for the specified log portion to zero; and
b) if the specified log portion contains log records, remove the log records from the specified log portion.

8.3.3.7 MANAGE OVERRIDE LOCKOUT TIMER service action

The ACCESS CONTROL OUT command with MANAGE OVERRIDE LOCKOUT TIMER service action is used to manage the override lockout timer (see 8.3.1.8.2.2). If the ACCESS CONTROL OUT command is implemented, the MANAGE OVERRIDE LOCKOUT TIMER service action shall be implemented.

If access controls are disabled, the access controls coordinator shall take no action and the command shall be completed with GOOD status.

The format of the CDB for the ACCESS CONTROL OUT command with MANAGE OVERRIDE LOCKOUT TIMER service action is shown in table 612 (see 8.3.3.1).

If the PARAMETER LIST LENGTH field in the CDB is zero, the access controls coordinator shall reset the override lockout timer to the current initial override lockout timer value maintained by the access controls coordinator.

If the value in the PARAMETER LIST LENGTH field is neither zero nor 12, the device server shall respond with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to PARAMETER LIST LENGTH ERROR.

If the value in the PARAMETER LIST LENGTH field is 12, the parameter list shall have the format shown in table 625.

Table 625 — ACCESS CONTROL OUT with MANAGE OVERRIDE LOCKOUT TIMER parameter data format

Bit / Byte	7	6	5	4	3	2	1	0
0	Reserved							
1								
2	(MSB)	NEW INITIAL OVERRIDE LOCKOUT TIMER						
3								(LSB)
4	(MSB)	MANAGEMENT IDENTIFIER KEY						
•••								
11								(LSB)

The NEW INITIAL OVERRIDE LOCKOUT TIMER field specifies the value that access controls coordinator shall maintain for initial override lockout timer if the specified management identifier key is correct.

If access controls are enabled and the contents of the MANAGEMENT IDENTIFIER KEY field do not match the current management identifier key (see 8.3.1.8) maintained by the access controls coordinator, then the access controls coordinator shall not change the initial override lockout timer value but shall set the override lockout timer to the unaltered current initial override lockout timer value. The command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to ACCESS DENIED - INVALID MGMT ID KEY. This event shall be recorded in the invalid keys portion of the access controls log (see 8.3.1.10).

In response to an ACCESS CONTROL OUT command with MANAGE OVERRIDE LOCKOUT TIMER service action with correct management identifier key value the access controls coordinator shall:

a) replace the currently saved initial override lockout timer with the value in the NEW INITIAL OVERRIDE LOCKOUT TIMER field; and
b) set the override lockout timer to the new initial value.

8.3.3.8 OVERRIDE MGMT ID KEY service action

The ACCESS CONTROL OUT command with OVERRIDE MGMT ID KEY service action is used to override the current management identifier key (see 8.3.1.4.2) maintained by the access controls coordinator. The ACCESS CONTROL OUT command with OVERRIDE MGMT ID KEY service action should be used in a failure situation where the application client no longer has access to its copy of the current management identifier key.

Successful use of the ACCESS CONTROL OUT command with OVERRIDE MGMT ID KEY service action is constrained by the override lockout timer (see 8.3.1.8.2.2).

If the ACCESS CONTROL OUT command is implemented, the OVERRIDE MGMT ID KEY service action shall be implemented.

The format of the CDB for the ACCESS CONTROL OUT command with OVERRIDE MGMT ID KEY service action is shown in table 612 (see 8.3.3.1).

If access controls are disabled or if the PARAMETER LIST LENGTH field in the CDB is zero, the access controls coordinator shall take no action and the command shall be completed with GOOD status.

If access controls are enabled, the access controls coordinator shall log every ACCESS CONTROL OUT command with OVERRIDE MGMT ID KEY service action processed whether successful or not in the access controls log as specified in 8.3.1.10.

If the value in the PARAMETER LIST LENGTH field is neither zero nor 12, the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to PARAMETER LIST LENGTH ERROR.

If the value in the PARAMETER LIST LENGTH field is 12, the parameter data shall have the format shown in table 626.

Table 626 — ACCESS CONTROL OUT with OVERRIDE MGMT ID KEY parameter data format

Bit / Byte	7	6	5	4	3	2	1	0
0	Reserved							
•••								
3								
4	(MSB)	NEW MANAGEMENT IDENTIFIER KEY						
•••								
11								(LSB)

The NEW MANAGEMENT IDENTIFIER KEY field specifies a new management identifier key.

If the override lockout timer managed by the access controls coordinator is not zero, the access controls coordinator's states shall not be altered, 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 override lockout timer managed by the access controls coordinator is zero, then the access controls coordinator shall replace the current management identifier key with the value in the to the NEW MANAGEMENT IDENTIFIER KEY field.

8.3.3.9 REVOKE PROXY TOKEN service action

An application client for an initiator port uses the ACCESS CONTROL OUT command with REVOKE PROXY TOKEN service action to cancel all proxy access rights to a logical unit that have been granted under the specified proxy token (see 8.3.1.6.2). If this service action is not supported, 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 format of the CDB for the ACCESS CONTROL OUT command with REVOKE PROXY TOKEN service action is shown in table 612 (see 8.3.3.1).

If access controls are disabled or if the PARAMETER LIST LENGTH field in the CDB is zero, the access controls coordinator shall take no action and the command shall be completed with GOOD status.

If the value in the PARAMETER LIST LENGTH field is neither zero nor eight, the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to PARAMETER LIST LENGTH ERROR.

If the value in the PARAMETER LIST LENGTH field is eight, the parameter data shall have the format shown in table 627.

Table 627 — ACCESS CONTROL OUT with REVOKE PROXY TOKEN parameter data format

Bit / Byte	7	6	5	4	3	2	1	0
0	PROXY TOKEN							
•••								
7								

If the PROXY TOKEN field does not contain a valid proxy token previously obtained via the initiator port, no action is taken by the access controls coordinator. This shall not be considered an error.

If the proxy token is valid, the access controls coordinator shall take the following actions:

a) invalidate the proxy token; and
b) deny access to the associated logical unit by any initiator port whose rights were granted under that proxy token via an ACCESS CONTROL OUT command with ASSIGN PROXY LUN service action (see 8.3.3.11) according to the requirements stated in 8.3.1.7.

8.3.3.10 REVOKE ALL PROXY TOKENS service action

An application client for an initiator port uses the ACCESS CONTROL OUT command with REVOKE ALL PROXY TOKENS service action to cancel all proxy access rights to a specified logical unit that it obtained with zero or more proxy tokens (see 8.3.1.6.2). If this service action is not supported, 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 format of the CDB for the ACCESS CONTROL OUT command with REVOKE ALL PROXY TOKENS service action is shown in table 612 (see 8.3.3.1).

If access controls are disabled or if the PARAMETER LIST LENGTH field in the CDB is zero, the access controls coordinator shall take no action and the command shall be completed with GOOD status.

If the value in the PARAMETER LIST LENGTH field is neither zero nor eight, the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to PARAMETER LIST LENGTH ERROR.

If the value in the PARAMETER LIST LENGTH field is eight, the parameter data shall have the format shown in table 628.

Table 628 — ACCESS CONTROL OUT with REVOKE ALL PROXY TOKENS parameter data format

Bit / Byte	7	6	5	4	3	2	1	0
0	LUN VALUE							
•••								
7								

If the LUN in the LUN VALUE field is not associated to a logical unit to which the requesting initiator port has non-proxy access rights based on the contents of an ACE (see 8.3.1.3) or if the LUN value is based on a proxy token (see 8.3.1.6.2), then no further action is taken by the access controls coordinator. This shall not be considered an error.

If the LUN value is associated to a logical unit to which the requesting initiator port has non-proxy access rights, the access controls coordinator shall take the following additional actions:

a) invalidate all proxy tokens for the initiator port for the logical unit specified by the LUN VALUE field;
b) deny access to that logical unit by any initiator port whose rights were granted under any of the invalidated proxy tokens via an ACCESS CONTROL OUT command with ASSIGN PROXY LUN service action (see 8.3.3.11) according to the requirements stated in 8.3.1.7.

8.3.3.11 ASSIGN PROXY LUN service action

The ACCESS CONTROL OUT command with ASSIGN PROXY LUN service action is used to request access to a logical unit under the rights of a proxy token (see 8.3.1.6.2) and to assign that logical unit a particular LUN value for addressing by the requesting initiator port. If this service action is not supported, 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 format of the CDB for the ACCESS CONTROL OUT command with ASSIGN PROXY LUN service action is shown in table 612 (see 8.3.3.1).

If the PARAMETER LIST LENGTH field in the CDB is zero, the access controls coordinator shall take no action and the command shall be completed with GOOD status.

If the value in the PARAMETER LIST LENGTH field is neither zero nor 16, the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to PARAMETER LIST LENGTH ERROR.

If the value in the PARAMETER LIST LENGTH field is 16, the parameter data shall have the format shown in table 629.

Table 629 — ACCESS CONTROL OUT with ASSIGN PROXY LUN parameter data format

Bit / Byte	7	6	5	4	3	2	1	0
0	PROXY TOKEN							
•••								
7								
8	LUN VALUE							
•••								
15								

The PROXY TOKEN field contains a proxy token. If the contents of the PROXY TOKEN field are not valid, 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 ACCESS DENIED - INVALID PROXY TOKEN.

NOTE 85 - If access controls are disabled, there are no valid proxy tokens and the device server always responds with the specified error information. This differs from the behavior of many other ACCESS CONTROL OUT service actions where the response is GOOD status when access controls are disabled. The difference in behavior is intended to inform the application client that its request for the new LUN assignment failed.

The LUN VALUE field specifies the LUN value the application client intends to use when accessing the logical unit described by the proxy token.

If the proxy token is valid but the access controls coordinator is unable to assign the requested LUN value to the associated logical unit (e.g., because the LUN value already is associated with a logical unit for the initiator port, or because the LUN value is not a supported logical unit address), then access rights shall not be granted, the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to ACCESS DENIED - INVALID LU IDENTIFIER, and the SENSE-KEY SPECIFIC field shall be set as described for the ILLEGAL REQUEST sense key in 4.5.2.4.2. The access controls coordinator should determine a suggested LUN value that is unlikely to produce an error while also minimizing the absolute value of the difference of the erroneous default LUN value and the suggested LUN value. If a suggested LUN value is determined, the first four bytes of the suggested LUN value shall be placed in the INFORMATION field and the last four bytes shall be placed in the COMMAND-SPECIFIC INFORMATION field of the sense data (see 4.5).

If the proxy token is valid but the access controls coordinator has insufficient resources to manage proxy logical unit access, the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INSUFFICIENT ACCESS CONTROL RESOURCES.

If the proxy token is valid and the access controls coordinator has sufficient resources, the initiator port shall be allowed proxy access to the referenced logical unit at the specified LUN value.

8.3.3.12 RELEASE PROXY LUN service action

The ACCESS CONTROL OUT command with RELEASE PROXY LUN service action is used to release proxy access to a logical unit created with a proxy token (see 8.3.1.6.2) and the ACCESS CONTROL OUT command with ASSIGN PROXY LUN service action (see 8.3.3.11). If this service action is not supported, 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 ACCESS CONTROL OUT command with RELEASE PROXY LUN service action should be used when an application client no longer requires the logical unit access rights granted to an initiator port under a proxy token (e.g., when a copy manager has completed a specific third party copy operation under a proxy token). This allows the access controls coordinator to free any resources allocated to manage the proxy access.

The format of the CDB for the ACCESS CONTROL OUT command with RELEASE PROXY LUN service action is shown in table 612 (see 8.3.3.1).

If the PARAMETER LIST LENGTH field in the CDB is zero, the access controls coordinator shall take no action and the command shall be completed with GOOD status.

If the value in the PARAMETER LIST LENGTH field is neither zero nor eight, the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to PARAMETER LIST LENGTH ERROR.

If the value in the PARAMETER LIST LENGTH field is eight, the parameter data shall have the format shown in table 630.

Table 630 — ACCESS CONTROL OUT with RELEASE PROXY LUN parameter data format

Bit / Byte	7	6	5	4	3	2	1	0
0	LUN VALUE							
•••								
7								

The LUN VALUE field specifies a LUN value that was associated with a logical unit based on a proxy token using an ACCESS CONTROL OUT command with ASSIGN PROXY LUN service action. If the LUN value was not assigned to a logical unit by an ACCESS CONTROL OUT command with ASSIGN PROXY LUN service action, 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.

NOTE 86 - If access controls are disabled, there are no valid proxy tokens and therefore no LUN value could be assigned to a logical unit by an ACCESS CONTROL OUT command with ASSIGN PROXY LUN service action so the device server always responds with the specified error information. This differs from the behavior of many other ACCESS CONTROL OUT service actions where the response is GOOD status when access controls are disabled. The difference in behavior is intended to inform the application client that the LUN value remains as a valid address for the logical unit.

If the LUN value was assigned to a logical unit by an ACCESS CONTROL OUT command with ASSIGN PROXY LUN service action, the access controls coordinator shall not allow access to the logical unit at the specified LUN value.

8.4 TARGET LOG PAGES well known logical unit

The TARGET LOG PAGES well known logical unit shall only process the commands listed in table 631. If a command is received by the TARGET LOG PAGES well know logical unit that is not listed in table 631, then the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID COMMAND OPERATION CODE.

Table 631 — Commands for the TARGET LOG PAGES well known logical unit

Command name	Operation code	Type	Reference
INQUIRY	12h	M	6.4
LOG SELECT	4Ch	M	6.5
LOG SENSE	4Dh	M	6.6
REQUEST SENSE	03h	M	6.29
TEST UNIT READY	00h	M	6.33
Key: M = Command implementation is mandatory.			

The TARGET LOG PAGES well known logical unit shall support the Protocol Specific Port log page (see 7.3.13) and may support other log pages with parameters that apply to the SCSI target device.

8.5 SECURITY PROTOCOL well known logical unit

The SECURITY PROTOCOL well known logical unit shall only process the commands listed in table 632. If a command is received by the SECURITY PROTOCOL well known logical unit that is not listed in table 632, then the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID COMMAND OPERATION CODE.

Table 632 — Commands for the SECURITY PROTOCOL well known logical unit

Command name	Operation code	Type	Reference
INQUIRY	12h	M	6.4
REQUEST SENSE	03h	M	6.29
SECURITY PROTOCOL IN	A2h	M	6.30
SECURITY PROTOCOL OUT	B5h	M	6.31
TEST UNIT READY	00h	M	6.37
Key: M = Command implementation is mandatory.			

8.6 MANAGEMENT PROTOCOL well known logical unit

The MANAGEMENT PROTOCOL well known logical unit shall only process the commands listed in table 633. If a command is received by the MANAGEMENT PROTOCOL well known logical unit that is not listed in table 633, then the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID COMMAND OPERATION CODE.

Table 633 — Commands for the MANAGEMENT PROTOCOL well known logical unit

Command name	Operation code	Type	Reference
INQUIRY	12h	M	6.4
MANAGEMENT PROTOCOL IN	A3h/10h [a]	M	6.7
MANAGEMENT PROTOCOL OUT	A4h/10h [a]	M	6.8
REQUEST SENSE	03h	M	6.29
TEST UNIT READY	00h	M	6.37
Key: M = Command implementation is mandatory.			
[a] 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.			

9 Security manager command set

A security manager shall only process the commands listed in table 634. If a command is received by the security manager that is not listed in table 634, then the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID COMMAND OPERATION CODE.

Table 634 — Commands for a security manager

Command name	Operation code	Type	Reference
INQUIRY	12h	M	6.4
RECEIVE CREDENTIAL	7Fh/1800 [a]	M	6.19
REPORT LUNS	A0h	M	6.23
REQUEST SENSE	03h	M	6.29
SECURITY PROTOCOL IN	A2h	M	6.30
SECURITY PROTOCOL OUT	B5h	M	6.31
TEST UNIT READY	00h	M	6.37
Type Key: M = Command implementation is mandatory.			

[a] 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.

Annex A
(informative)

Terminology mapping

The introduction of a model for SCSI devices with multiple ports resulted in changes in terminology between SPC-2 and this standard (see table A.1).

Table A.1 — This standard to SPC-2 terminology mapping

SPC-3 equivalent term	SPC-2 term
initiator port identifier	initiator identifier
queue	task set
SCSI initiator port	initiator
SCSI port	port
SCSI port identifier	device identifier
SCSI port identifier	SCSI identifier
SCSI target port	target
target port identifier	target identifier

Annex B
(Informative)

PERSISTENT RESERVE IN/OUT functionality for RESERVE/RELEASE replacement

B.1 Introduction

This annex specifies the PERSISTENT RESERVE OUT command features necessary to replace the reserve/release management method (see SPC-2) and provides guidance on how to perform a third party reservation using persistent reservations. The PERSISTENT RESERVE IN command is not used to replace any feature of the reserve/release management method.

B.2 Replacing the reserve/release method with the PERSISTENT RESERVE OUT COMMAND

The minimum PERSISTENT RESERVE OUT command (see 6.14) features necessary to replace the reserve/release management method (see SPC-2) are shown in table B.1.

Table B.1 — PERSISTENT RESERVE OUT command features

PERSISTENT RESERVE OUT command features		Replaces reserve/release
Service Action	REGISTER	Yes [b]
	RESERVE	Yes
	RELEASE	Yes
	CLEAR	Yes [c]
	PREEMPT	No
	PREEMPT AND ABORT	No
	REGISTER AND IGNORE EXISTING KEY	Yes [b]
	REGISTER AND MOVE	Yes [d]
Scope	LU_SCOPE	Yes
Type	Write Exclusive	No
	Exclusive Access	Yes
	Write Exclusive – Registrants Only	No
	Exclusive Access – Registrants Only	No
	Write Exclusive – All Registrants	No
	Exclusive Access – All Registrants	No

[b] An implementation uses either the REGISTER or REGISTER AND IGNORE EXISTING KEY service action.
[c] Necessary to clear the registration and reservation (e.g, a failed initiator).
[d] Necessary only for third party reservations.

B.3 Third party reservations

For some uses of the EXTENDED COPY command (see 6.3), the application client performs a locking function to maintain data integrity on the source and may also lock the destination device prior to starting the copy operation. The persistent reservation management method may be used to perform the locking function. Other methods (e.g., access controls, see 8.3) may also perform the locking function.

To accomplish a third party persistent reservation the following steps are recommended:

1) backup application uses the REGISTER service action to register an I_T nexus with a logical unit (e.g., a tape drive logical unit);
2) backup application uses the RESERVE service action to establish a persistent reservation with the Exclusive Access type;
3) backup application prepares the logical unit for access (e.g., medium is loaded and positioned);
4) backup application uses the REGISTER AND MOVE service action to register the I_T nexus that the copy manager is expected to use and to move the persistent reservation to that I_T nexus;
5) backup application sends the EXTENDED COPY command to the copy manager that includes a third party persistent reservations source I_T nexus segment descriptor (see 6.3.7.19);
6) copy manager processes all segment descriptors in the received EXTENDED COPY command except the third party persistent reservations source I_T nexus segment descriptor; and
7) copy manager issues a REGISTER AND MOVE service action, using the reservation key and I_T nexus specified in the third party persistent reservations source I_T nexus segment descriptor received from the backup application (see step 5), to move the persistent reservation back to the original I_T nexus.

Annex C
(Informative)

Variations between this standard and equivalent security protocols

C.1 IKEv2 protocol details and variations for IKEv2-SCSI

The IKEv2 protocol details and variations specified in RFC 4306 apply to IKEv2-SCSI (i.e., this standard) as follows:

a) any SECURITY PROTOCOL OUT command with a transfer length of up to 16 384 bytes is not terminated with an error due to the number of bytes transferred;
b) the timeout and retransmission mechanisms defined in RFC 4306 are not used by this standard, instead a new payload is defined to transfer appropriate timeout values to the device server;
c) each SCSI command used by this standard completes by conveying a status from the device server to the application client;
d) the IKEv2 header EXCHANGE TYPE field is reserved in this standard as a result of equivalent information being transferred in the SECURITY PROTOCOL OUT command and SECURITY PROTOCOL IN command CDBs;
e) the IKEv2 header VERSION bit is reserved in this standard;
f) this standard uses the pseudo-random functions (PRF) functions defined by RFC 4306;
g) the key derivation functions defined and used by this standard (see 5.16.3) are equivalent to the PRF+ found in RFC 4306;
h) the SA creation transactions (see 3.1.130) defined by this standard are not overlapped. If an application client attempts to start a second SA creation transaction before the first is completed, the offending command is terminated as described in 5.16.4.1, but this does not affect the SA creation transaction that is already in progress;
i) the NO_PROPOSAL_CHOSEN and INVALID_KE_PAYLOAD notify error types are replaced by the SA CREATION PARAMETER VALUE INVALID additional sense code (see 7.7.3.9) because IKEv2-SCSI has a different negotiation structure. As defined in RFC 4306, an IKEv2 initiator offers one or more proposals to a responder without knowing what is acceptable to the responder, and chooses a DH group without knowing whether it is acceptable to the responder. These two notify error types allow the responder to inform the initiator that one or more of its choices are not acceptable. In contrast, an IKEv2-SCSI application client obtains the device server capabilities in the Device Capabilities step (see 5.16.4.5) and selects algorithms from them in the Key Exchange step (see 5.16.4.6). An error only occurs if the application client has made an invalid selection, hence the SA CREATION PARAMETER VALUE INVALID description;
j) IKEv2 version numbers (see RFC 4306) are used by this standard (see 7.7.3.4), but the ability to respond to an unsupported version number with the highest version number to be used is not supported, and this standard does not include checks for version downgrade attacks;
k) IKEv2 cookies (see RFC 4306) are not used by this standard;
l) IKEv2 cryptographic algorithm negotiation (see RFC 4306) is replaced by the Device Server Capabilities step (see 5.16.4.5) and the Key Exchange step (see 5.16.4.6) (i.e., the IKEv2 proposal construct is not used by this standard);
m) in this standard an SA is rekeyed by replacing it with a new SA:
   A) CHILD_SAs are not used by this standard;
   B) the RFC 4306 discussion of CHILD_SAs does not apply to this standard;
   C) coexistence of the original SA and the new SA is achieved for rekeying purposes by restricting the device server's ability to delete SAs to the following cases:
      a) expiration of a timeout (see 7.7.3.5.15);
      b) processing of an IKEv2-SCSI Delete function (see 5.16.4.11); and
      c) responding to an initial contact notification (see 7.7.3.5.9);
      and

D) IKEv2 does not support rekeying notification for IKE_SAs, therefore this standard does not support rekeying notification;

n) the choice of authentication methods for both transfer directions is negotiated using the SA_AUTH_OUT IKEv2-SCSI cryptographic algorithm descriptor and SA_AUTH_IN IKEv2-SCSI cryptographic algorithm descriptor (see 7.7.3.6.6) during the Key Exchange step (see 5.16.4.6);

o) the usage of Certificate Encodings in the Certificate payload (see 7.7.3.5.5) and Certificate Request payload (see 7.7.3.5.6) are constrained as follows:
   A) in accordance with the recommendations in RFC 4718, Certificate Encoding values 01h-03h and 05h-0Ah are prohibited;
   B) this standard forbids the use of URL-based Certificate Encodings (i.e., Certificate Encodings values 0Ch and 0Dh); and
   C) certificate Encoding values that RFC 4306 defines as vendor specific are reserved in this standard;

p) deleting an SA requires knowing the SAIs (i.e., SPIs) in both directions and including both SAIs in the Delete payload. The RFC 4306 description of the Delete payload is vague enough to allow this. The requirement is consistent with the SCSI model for SAs;

q) the Vendor ID payload is not used by this standard;

r) Traffic Selectors (see RFC 4306) are not used by this standard;

s) the requirements in RFC 4306 on nonces are be followed for the random nonces (see 3.1.122) defined by this standard;

t) the RFC 4306 requirements on address and port agility are specific to the user datagram protocol and the IP protocol and do not apply to this standard;

u) keys for the Authentication step are generated as specified in RFC 4306;

v) this standard uses a slightly modified version of the authentication calculations in RFC 4306 (see 7.7.3.5.7);

w) the RFC 4306 sections that describe the following features are not used by this standard:
   A) extensible authentication protocol methods;
   B) generating keying Material for CHILD_SAs;
   C) rekeying an IKE SA using CREATE_CHILD_SA;
   D) requesting an internal address;
   E) requesting the peer's version;
   F) IPComp;
   G) NAT traversal; and
   H) explicit congestion notification;

x) IKEv2 Error Handling (see RFC 4306) is replaced by the use of CHECK CONDITION status and sense data by this standard. See 7.7.3.9 for details of how errors reported in the Notify payload are translated to sense data;

y) IETF standards omit the Integrity transform instead of using AUTH_COMBINED;

z) the command-response architecture of SCSI makes it difficult to protect the device server against denial of service attacks, and no such protection is defined by this standard. Protection against denial of service attacks against the application client is described in 7.7.3.8;

aa) IKEv2-SCSI requires Encrypted Payloads be padded to at least the 4 byte minimum alignment required by ESP-SCSI, whereas IKEv2 imposes no such requirement; and

ab) the critical (CRIT) bit is set to one in all IKEv2-SCSI payloads specified in this standard and these payloads are required to be recognized by all IKEv2-SCSI implementations. RFC 4306 sets the Critical (C) bit to zero in all IKEv2 payloads specified in RFC 4306, and requires that all IKEv2 implementations recognize all payloads specified in RFC 4306; and

ac) use of the Identification payloads is required by IKEv2-SCSI, whereas IKEv2 allows the Identification payloads to be omitted.

Where this standard uses IKE payload names (see 7.7.3.4) RFC 4306 uses the shorthand notation shown in table C.1.

Table C.1 — IKE payload names shorthand

IKE payload name in this standard [a]	RFC 4306 shorthand [b]
Security Association	SAi or SAr
Key Exchange	KEi or KEr
Identification – Application Client	IDi
Identification – Device Server	IDr
Certificate	CERTi or CERTr
Certificate Request	CERTREQi
Authentication	AUTHi or AUTHr
Nonce	NONCEi or NONCEr
Notify	N-ICi or N-ICr
Delete	Di
Vendor ID	Vi or Vr
Traffic Selector – Application Client	TSi
Traffic Selector – Device Server	TSr
Encrypted	Ei or Er
Configuration	CPi or CPr
Extensible Authentication	EAPi or EAPr

[a] To facilitate future enhancements, all IKE payloads are listed in this table, but not all entries in this table are used in this standard.

[b] In RFC 4306 the lowercase i indicates initiator and r indicates responder. In this standard, the initiator is the application client and all such IKE payloads (e.g., KEi) appear in a SECURITY PROTOCOL OUT parameter list. The responder is always the device server in this standard and all such IKE payloads (e.g., AUTHr) appear in SECURITY PROTOCOL IN parameter data.

C.2 ESP protocol details and variations for ESP-SCSI

The IKEv2 protocol details and variations specified in RFC 4303 apply to ESP-SCSI (i.e., this standard) as follows:

a) this standard requires an integrity check value (ICV field), whereas ESP allows support of confidentiality-only;
b) this standard does not support traffic flow confidentiality;
c) this standard does not support the TCP/IP aspects of ESP (e.g., IP addresses, multicast);
d) this standard requires anti-replay detection using the sequence number, whereas ESP makes this optional;
e) this standard does not support the Next Header field, but does reserve space for it in the MUST BE ZERO field (see table 81 in 5.16.7.3);
f) this standard requires verification of the padding bytes, when possible;
g) there is no provision in this standard for generating 'dummy packets'; and
h) this standard does not support out-of-order parameter data.

Annex D
(informative)

Numeric order codes

D.1 Numeric order codes introduction

This annex contains SCSI additional sense codes, operation codes, diagnostic page codes, log page codes, mode page codes, VPD page codes, version descriptor values, and T10 IEEE binary identifiers in numeric order as a reference. In the event of a conflict with between the codes or usage requirements in this annex and equivalent information in the body of this standard or in any command standard, the normative codes and usage information is correct.

The information in this annex was complete and accurate at the time of publication. However, the information is subject to change. Technical Committee T10 of INCITS maintains an electronic copy of this information on its world wide web site (http://www.t10.org/). In the event that the T10 world wide web site is no longer active, access may be possible via the INCITS world wide web site (http://www.incits.org), the ANSI world wide web site (http://www.ansi.org), the IEC site (http://www.iec.ch/), the ISO site (http://www.iso.ch/), or the ISO/IEC JTC 1 web site (http://www.jtc1.org/).

D.2 Additional sense codes

Table D.1 is a numerical order listing of the additional sense codes (i.e., the ADDITIONAL SENSE CODE field and ADDITIONAL SENSE CODE QUALIFIER field values returned in sense data).

Table D.1 — ASC and ASCQ assignments (part 1 of 18)

```
                D – Direct Access Block Device (SBC-3)                    Device Column key
                . T – Sequential Access Device (SSC-4)                    blank = code not used
                .   L – Printer Device (SSC)                              not blank = code used
                .     P – Processor Device (SPC-2)
                .     . W – Write Once Block Device (SBC)
                .     .   R – C/DVD Device (MMC-6)
                .     .     O – Optical Memory Block Device (SBC)
                .     .     . M – Media Changer Device (SMC-3)
                .     .     .   A – Storage Array Device (SCC-2)
                .     .     .     E – SCSI Enclosure Services device (SES-3)
                .     .     .     . B – Simplified Direct-Access (Reduced Block) device (RBC)
                .     .     .     .   K – Optical Card Reader/Writer device (OCRW)
                .     .     .     .     V – Automation/Device Interface device (ADC-3)
                .     .     .     .     . F – Object-based Storage Device (OSD-2)
                .     .     .     .     .
ASC  ASCQ       DTLPWROMAEBKVF  Description
00h  00h        DTLPWROMAEBKVF  NO ADDITIONAL SENSE INFORMATION
00h  01h        T               FILEMARK DETECTED
00h  02h        T               END-OF-PARTITION/MEDIUM DETECTED
00h  03h        T               SETMARK DETECTED
00h  04h        T               BEGINNING-OF-PARTITION/MEDIUM DETECTED
00h  05h        TL              END-OF-DATA DETECTED
00h  06h        DTLPWROMAEBKVF  I/O PROCESS TERMINATED
00h  07h        T               PROGRAMMABLE EARLY WARNING DETECTED
00h  11h                R       AUDIO PLAY OPERATION IN PROGRESS
00h  12h                R       AUDIO PLAY OPERATION PAUSED
00h  13h                R       AUDIO PLAY OPERATION SUCCESSFULLY COMPLETED
00h  14h                R       AUDIO PLAY OPERATION STOPPED DUE TO ERROR
```

Table D.1 — ASC and ASCQ assignments (part 2 of 18)

```
D – Direct Access Block Device (SBC-3)
. T – Sequential Access Device (SSC-4)
.   L – Printer Device (SSC)
.     P – Processor Device (SPC-2)
.     . W – Write Once Block Device (SBC)
.     .   R – C/DVD Device (MMC-6)
.     .     O – Optical Memory Block Device (SBC)
.     .     . M – Media Changer Device (SMC-3)
.     .     .   A – Storage Array Device (SCC-2)
.     .     .     E – SCSI Enclosure Services device (SES-3)
.     .     .     . B – Simplified Direct-Access (Reduced Block) device (RBC)
.     .     .     .   K – Optical Card Reader/Writer device (OCRW)
.     .     .     .     V – Automation/Device Interface device (ADC-3)
.     .     .     .     . F – Object-based Storage Device (OSD-2)
.     .     .     .     .
```

Device Column key
blank = code not used
not blank = code used

ASC	ASCQ	D	T	L	P	W	R	O	M	A	E	B	K	V	F	Description
00h	15h						R									NO CURRENT AUDIO STATUS TO RETURN
00h	16h	D	T	L	P	W	R	O	M	A	E	B	K	V	F	OPERATION IN PROGRESS
00h	17h	D	T	L		W	R	O	M	A	E	B	K	V	F	CLEANING REQUESTED
00h	18h		T													ERASE OPERATION IN PROGRESS
00h	19h		T													LOCATE OPERATION IN PROGRESS
00h	1Ah		T													REWIND OPERATION IN PROGRESS
00h	1Bh		T													SET CAPACITY OPERATION IN PROGRESS
00h	1Ch		T													VERIFY OPERATION IN PROGRESS
00h	1Dh	D	T									B				ATA PASS THROUGH INFORMATION AVAILABLE
00h	1Eh	D	T				R		M	A	E	B	K	V		CONFLICTING SA CREATION REQUEST
00h	1Fh	D	T									B				LOGICAL UNIT TRANSITIONING TO ANOTHER POWER CONDITION
01h	00h	D				W		O				B	K			NO INDEX/SECTOR SIGNAL
02h	00h	D				W	R	O				B	K			NO SEEK COMPLETE
03h	00h	D	T	L		W		O				B	K			PERIPHERAL DEVICE WRITE FAULT
03h	01h		T													NO WRITE CURRENT
03h	02h		T													EXCESSIVE WRITE ERRORS
04h	00h	D	T	L	P	W	R	O	M	A	E	B	K	V	F	LOGICAL UNIT NOT READY, CAUSE NOT REPORTABLE
04h	01h	D	T	L	P	W	R	O	M	A	E	B	K	V	F	LOGICAL UNIT IS IN PROCESS OF BECOMING READY
04h	02h	D	T	L	P	W	R	O	M	A	E	B	K	V	F	LOGICAL UNIT NOT READY, INITIALIZING COMMAND REQUIRED
04h	03h	D	T	L	P	W	R	O	M	A	E	B	K	V	F	LOGICAL UNIT NOT READY, MANUAL INTERVENTION REQUIRED
04h	04h	D	T	L			R	O				B				LOGICAL UNIT NOT READY, FORMAT IN PROGRESS
04h	05h	D	T			W		O		A		B	K		F	LOGICAL UNIT NOT READY, REBUILD IN PROGRESS
04h	06h	D	T			W		O		A		B	K			LOGICAL UNIT NOT READY, RECALCULATION IN PROGRESS
04h	07h	D	T	L	P	W	R	O	M	A	E	B	K	V	F	LOGICAL UNIT NOT READY, OPERATION IN PROGRESS
04h	08h						R									LOGICAL UNIT NOT READY, LONG WRITE IN PROGRESS
04h	09h	D	T	L	P	W	R	O	M	A	E	B	K	V	F	LOGICAL UNIT NOT READY, SELF-TEST IN PROGRESS
04h	0Ah	D	T	L	P	W	R	O	M	A	E	B	K	V	F	LOGICAL UNIT NOT ACCESSIBLE, ASYMMETRIC ACCESS STATE TRANSITION
04h	0Bh	D	T	L	P	W	R	O	M	A	E	B	K	V	F	LOGICAL UNIT NOT ACCESSIBLE, TARGET PORT IN STANDBY STATE
04h	0Ch	D	T	L	P	W	R	O	M	A	E	B	K	V	F	LOGICAL UNIT NOT ACCESSIBLE, TARGET PORT IN UNAVAILABLE STATE
04h	0Dh														F	LOGICAL UNIT NOT READY, STRUCTURE CHECK REQUIRED
04h	10h	D	T			W	R	O	M			B				LOGICAL UNIT NOT READY, AUXILIARY MEMORY NOT ACCESSIBLE
04h	11h	D	T			W	R	O		A	E	B		V	F	LOGICAL UNIT NOT READY, NOTIFY (ENABLE SPINUP) REQUIRED
04h	12h								M					V		LOGICAL UNIT NOT READY, OFFLINE
04h	13h	D	T				R		M	A	E	B	K	V		LOGICAL UNIT NOT READY, SA CREATION IN PROGRESS
04h	14h	D										B				LOGICAL UNIT NOT READY, SPACE ALLOCATION IN PROGRESS
04h	15h								M							LOGICAL UNIT NOT READY, ROBOTICS DISABLED
04h	16h								M							LOGICAL UNIT NOT READY, CONFIGURATION REQUIRED
04h	17h								M							LOGICAL UNIT NOT READY, CALIBRATION REQUIRED
04h	18h								M							LOGICAL UNIT NOT READY, A DOOR IS OPEN
04h	19h								M							LOGICAL UNIT NOT READY, OPERATING IN SEQUENTIAL MODE

Table D.1 — ASC and ASCQ assignments (part 3 of 18)

```
D – Direct Access Block Device (SBC-3)                                   Device Column key
. T – Sequential Access Device (SSC-4)                                   blank = code not used
.   L – Printer Device (SSC)                                             not blank = code used
.     P – Processor Device (SPC-2)
.     . W – Write Once Block Device (SBC)
.     .   R – C/DVD Device (MMC-6)
.     .     O – Optical Memory Block Device (SBC)
.     .     . M – Media Changer Device (SMC-3)
.     .     .   A – Storage Array Device (SCC-2)
.     .     .     E – SCSI Enclosure Services device (SES-3)
.     .     .     . B – Simplified Direct-Access (Reduced Block) device (RBC)
.     .     .     .   K – Optical Card Reader/Writer device (OCRW)
.     .     .     .     V – Automation/Device Interface device (ADC-3)
.     .     .     .     . F – Object-based Storage Device (OSD-2)
.     .     .     .     .
```

ASC	ASCQ	DTLPWROMAEBKVF	Description
04h	1Ah	DT B	LOGICAL UNIT NOT READY, START STOP UNIT COMMAND IN PROGRESS
04h	1Bh	D B	LOGICAL UNIT NOT READY, SANITIZE IN PROGRESS
04h	1Ch	DT MAEB	LOGICAL UNIT NOT READY, ADDITIONAL POWER USE NOT YET GRANTED
05h	00h	DTL WROMAEBKVF	LOGICAL UNIT DOES NOT RESPOND TO SELECTION
06h	00h	D WROM BK	NO REFERENCE POSITION FOUND
07h	00h	DTL WROM BK	MULTIPLE PERIPHERAL DEVICES SELECTED
08h	00h	DTL WROMAEBKVF	LOGICAL UNIT COMMUNICATION FAILURE
08h	01h	DTL WROMAEBKVF	LOGICAL UNIT COMMUNICATION TIME-OUT
08h	02h	DTL WROMAEBKVF	LOGICAL UNIT COMMUNICATION PARITY ERROR
08h	03h	DT ROM BK	LOGICAL UNIT COMMUNICATION CRC ERROR (ULTRA-DMA/32)
08h	04h	DTLPWRO K	UNREACHABLE COPY TARGET
09h	00h	DT WRO B	TRACK FOLLOWING ERROR
09h	01h	WRO K	TRACKING SERVO FAILURE
09h	02h	WRO K	FOCUS SERVO FAILURE
09h	03h	WRO	SPINDLE SERVO FAILURE
09h	04h	DT WRO B	HEAD SELECT FAULT
0Ah	00h	DTLPWROMAEBKVF	ERROR LOG OVERFLOW
0Bh	00h	DTLPWROMAEBKVF	WARNING
0Bh	01h	DTLPWROMAEBKVF	WARNING - SPECIFIED TEMPERATURE EXCEEDED
0Bh	02h	DTLPWROMAEBKVF	WARNING - ENCLOSURE DEGRADED
0Bh	03h	DTLPWROMAEBKVF	WARNING - BACKGROUND SELF-TEST FAILED
0Bh	04h	DTLPWRO AEBKVF	WARNING - BACKGROUND PRE-SCAN DETECTED MEDIUM ERROR
0Bh	05h	DTLPWRO AEBKVF	WARNING - BACKGROUND MEDIUM SCAN DETECTED MEDIUM ERROR
0Bh	06h	DTLPWROMAEBKVF	WARNING - NON-VOLATILE CACHE NOW VOLATILE
0Bh	07h	DTLPWROMAEBKVF	WARNING - DEGRADED POWER TO NON-VOLATILE CACHE
0Bh	08h	DTLPWROMAEBKVF	WARNING - POWER LOSS EXPECTED
0Ch	00h	T R	WRITE ERROR
0Ch	01h	K	WRITE ERROR - RECOVERED WITH AUTO REALLOCATION
0Ch	02h	D W O BK	WRITE ERROR - AUTO REALLOCATION FAILED
0Ch	03h	D W O BK	WRITE ERROR - RECOMMEND REASSIGNMENT
0Ch	04h	DT W O B	COMPRESSION CHECK MISCOMPARE ERROR
0Ch	05h	DT W O B	DATA EXPANSION OCCURRED DURING COMPRESSION
0Ch	06h	DT W O B	BLOCK NOT COMPRESSIBLE
0Ch	07h	R	WRITE ERROR - RECOVERY NEEDED
0Ch	08h	R	WRITE ERROR - RECOVERY FAILED
0Ch	09h	R	WRITE ERROR - LOSS OF STREAMING
0Ch	0Ah	R	WRITE ERROR - PADDING BLOCKS ADDED
0Ch	0Bh	DT WROM B	AUXILIARY MEMORY WRITE ERROR
0Ch	0Ch	DTLPWRO AEBKVF	WRITE ERROR - UNEXPECTED UNSOLICITED DATA
0Ch	0Dh	DTLPWRO AEBKVF	WRITE ERROR - NOT ENOUGH UNSOLICITED DATA
0Ch	0Fh	R	DEFECTS IN ERROR WINDOW

Table D.1 — ASC and ASCQ assignments (part 4 of 18)

```
D – Direct Access Block Device (SBC-3)
. T – Sequential Access Device (SSC-4)
.   L – Printer Device (SSC)
.     P – Processor Device (SPC-2)
.     . W – Write Once Block Device (SBC)
.     .   R – C/DVD Device (MMC-6)
.     .     O – Optical Memory Block Device (SBC)
.     .       . M – Media Changer Device (SMC-3)
.     .       .   A – Storage Array Device (SCC-2)
.     .       .     E – SCSI Enclosure Services device (SES-3)
.     .       .       . B – Simplified Direct-Access (Reduced Block) device (RBC)
.     .       .       .   K – Optical Card Reader/Writer device (OCRW)
.     .       .       .     V – Automation/Device Interface device (ADC-3)
.     .       .       .       . F – Object-based Storage Device (OSD-2)
.     .       .       .       .
```

Device Column key
blank = code not used
not blank = code used

ASC	ASCQ	D	T	L	P	W	R	O	M	A	E	B	K	V	F	Description
0Dh	00h	D	T	L	P	W	R	O		A			K			ERROR DETECTED BY THIRD PARTY TEMPORARY INITIATOR
0Dh	01h	D	T	L	P	W	R	O		A			K			THIRD PARTY DEVICE FAILURE
0Dh	02h	D	T	L	P	W	R	O		A			K			COPY TARGET DEVICE NOT REACHABLE
0Dh	03h	D	T	L	P	W	R	O		A			K			INCORRECT COPY TARGET DEVICE TYPE
0Dh	04h	D	T	L	P	W	R	O		A			K			COPY TARGET DEVICE DATA UNDERRUN
0Dh	05h	D	T	L	P	W	R	O		A			K			COPY TARGET DEVICE DATA OVERRUN
0Eh	00h	D	T		P	W	R	O	M	A	E	B	K		F	INVALID INFORMATION UNIT
0Eh	01h	D	T		P	W	R	O	M	A	E	B	K		F	INFORMATION UNIT TOO SHORT
0Eh	02h	D	T		P	W	R	O	M	A	E	B	K		F	INFORMATION UNIT TOO LONG
0Eh	03h	D	T		P		R		M	A	E	B	K		F	INVALID FIELD IN COMMAND INFORMATION UNIT
0Fh	00h															
10h	00h	D				W		O				B	K			ID CRC OR ECC ERROR
10h	01h	D	T			W		O								LOGICAL BLOCK GUARD CHECK FAILED
10h	02h	D	T			W		O								LOGICAL BLOCK APPLICATION TAG CHECK FAILED
10h	03h	D	T			W		O								LOGICAL BLOCK REFERENCE TAG CHECK FAILED
10h	04h		T													LOGICAL BLOCK PROTECTION ERROR ON RECOVER BUFFERED DATA
10h	05h		T													LOGICAL BLOCK PROTECTION METHOD ERROR
11h	00h	D	T			W	R	O				B	K			UNRECOVERED READ ERROR
11h	01h	D	T			W	R	O				B	K			READ RETRIES EXHAUSTED
11h	02h	D	T			W	R	O				B	K			ERROR TOO LONG TO CORRECT
11h	03h	D	T			W		O				B	K			MULTIPLE READ ERRORS
11h	04h	D				W		O				B	K			UNRECOVERED READ ERROR - AUTO REALLOCATE FAILED
11h	05h					W	R	O				B				L-EC UNCORRECTABLE ERROR
11h	06h					W	R	O				B				CIRC UNRECOVERED ERROR
11h	07h					W		O				B				DATA RE-SYNCHRONIZATION ERROR
11h	08h		T													INCOMPLETE BLOCK READ
11h	09h		T													NO GAP FOUND
11h	0Ah	D	T					O				B	K			MISCORRECTED ERROR
11h	0Bh	D				W		O				B	K			UNRECOVERED READ ERROR - RECOMMEND REASSIGNMENT
11h	0Ch	D				W		O				B	K			UNRECOVERED READ ERROR - RECOMMEND REWRITE THE DATA
11h	0Dh	D	T			W	R	O				B				DE-COMPRESSION CRC ERROR
11h	0Eh	D	T			W	R	O				B				CANNOT DECOMPRESS USING DECLARED ALGORITHM
11h	0Fh						R									ERROR READING UPC/EAN NUMBER
11h	10h						R									ERROR READING ISRC NUMBER
11h	11h						R									READ ERROR - LOSS OF STREAMING
11h	12h	D	T			W	R	O	M			B				AUXILIARY MEMORY READ ERROR
11h	13h	D	T	L	P	W	R	O		A	E	B	K	V	F	READ ERROR - FAILED RETRANSMISSION REQUEST
11h	14h	D														READ ERROR - LBA MARKED BAD BY APPLICATION CLIENT
12h	00h	D				W		O				B	K			ADDRESS MARK NOT FOUND FOR ID FIELD
13h	00h	D				W		O				B	K			ADDRESS MARK NOT FOUND FOR DATA FIELD
14h	00h	D	T	L		W	R	O				B	K			RECORDED ENTITY NOT FOUND

Table D.1 — ASC and ASCQ assignments (part 5 of 18)

```
D – Direct Access Block Device (SBC-3)
. T – Sequential Access Device (SSC-4)
.   L – Printer Device (SSC)
.     P – Processor Device (SPC-2)
.     . W – Write Once Block Device (SBC)
.     .   R – C/DVD Device (MMC-6)
.     .     O – Optical Memory Block Device (SBC)
.     .     . M – Media Changer Device (SMC-3)
.     .     .   A – Storage Array Device (SCC-2)
.     .     .     E – SCSI Enclosure Services device (SES-3)
.     .     .     . B – Simplified Direct-Access (Reduced Block) device (RBC)
.     .     .     .   K – Optical Card Reader/Writer device (OCRW)
.     .     .     .     V – Automation/Device Interface device (ADC-3)
.     .     .     .     . F – Object-based Storage Device (OSD-2)
.     .     .     .     .
```

Device Column key
blank = code not used
not blank = code used

ASC	ASCQ	DTLPWROMAEBKVF	Description
14h	01h	DT WRO BK	RECORD NOT FOUND
14h	02h	T	FILEMARK OR SETMARK NOT FOUND
14h	03h	T	END-OF-DATA NOT FOUND
14h	04h	T	BLOCK SEQUENCE ERROR
14h	05h	DT W O BK	RECORD NOT FOUND - RECOMMEND REASSIGNMENT
14h	06h	DT W O BK	RECORD NOT FOUND - DATA AUTO-REALLOCATED
14h	07h	T	LOCATE OPERATION FAILURE
15h	00h	DTL WROM BK	RANDOM POSITIONING ERROR
15h	01h	DTL WROM BK	MECHANICAL POSITIONING ERROR
15h	02h	DT WRO BK	POSITIONING ERROR DETECTED BY READ OF MEDIUM
16h	00h	D W O BK	DATA SYNCHRONIZATION MARK ERROR
16h	01h	D W O BK	DATA SYNC ERROR - DATA REWRITTEN
16h	02h	D W O BK	DATA SYNC ERROR - RECOMMEND REWRITE
16h	03h	D W O BK	DATA SYNC ERROR - DATA AUTO-REALLOCATED
16h	04h	D W O BK	DATA SYNC ERROR - RECOMMEND REASSIGNMENT
17h	00h	DT WRO BK	RECOVERED DATA WITH NO ERROR CORRECTION APPLIED
17h	01h	DT WRO BK	RECOVERED DATA WITH RETRIES
17h	02h	DT WRO BK	RECOVERED DATA WITH POSITIVE HEAD OFFSET
17h	03h	DT WRO BK	RECOVERED DATA WITH NEGATIVE HEAD OFFSET
17h	04h	WRO B	RECOVERED DATA WITH RETRIES AND/OR CIRC APPLIED
17h	05h	D WRO BK	RECOVERED DATA USING PREVIOUS SECTOR ID
17h	06h	D W O BK	RECOVERED DATA WITHOUT ECC - DATA AUTO-REALLOCATED
17h	07h	D WRO BK	RECOVERED DATA WITHOUT ECC - RECOMMEND REASSIGNMENT
17h	08h	D WRO BK	RECOVERED DATA WITHOUT ECC - RECOMMEND REWRITE
17h	09h	D WRO BK	RECOVERED DATA WITHOUT ECC - DATA REWRITTEN
18h	00h	DT WRO BK	RECOVERED DATA WITH ERROR CORRECTION APPLIED
18h	01h	D WRO BK	RECOVERED DATA WITH ERROR CORR. & RETRIES APPLIED
18h	02h	D WRO BK	RECOVERED DATA - DATA AUTO-REALLOCATED
18h	03h	R	RECOVERED DATA WITH CIRC
18h	04h	R	RECOVERED DATA WITH L-EC
18h	05h	D WRO BK	RECOVERED DATA - RECOMMEND REASSIGNMENT
18h	06h	D WRO BK	RECOVERED DATA - RECOMMEND REWRITE
18h	07h	D W O BK	RECOVERED DATA WITH ECC - DATA REWRITTEN
18h	08h	R	RECOVERED DATA WITH LINKING
19h	00h	D O K	DEFECT LIST ERROR
19h	01h	D O K	DEFECT LIST NOT AVAILABLE
19h	02h	D O K	DEFECT LIST ERROR IN PRIMARY LIST
19h	03h	D O K	DEFECT LIST ERROR IN GROWN LIST
1Ah	00h	DTLPWROMAEBKVF	PARAMETER LIST LENGTH ERROR
1Bh	00h	DTLPWROMAEBKVF	SYNCHRONOUS DATA TRANSFER ERROR
1Ch	00h	D O BK	DEFECT LIST NOT FOUND

Table D.1 — ASC and ASCQ assignments (part 6 of 18)

D – Direct Access Block Device (SBC-3)
. T – Sequential Access Device (SSC-4)
. L – Printer Device (SSC)
. P – Processor Device (SPC-2)
. . W – Write Once Block Device (SBC)
. . R – C/DVD Device (MMC-6)
. . O – Optical Memory Block Device (SBC)
. . . M – Media Changer Device (SMC-3)
. . . A – Storage Array Device (SCC-2)
. . . E – SCSI Enclosure Services device (SES-3)
. . . . B – Simplified Direct-Access (Reduced Block) device (RBC)
. . . . K – Optical Card Reader/Writer device (OCRW)
. . . . V – Automation/Device Interface device (ADC-3)
. . . . . F – Object-based Storage Device (OSD-2)

Device Column key
blank = code not used
not blank = code used

ASC	ASCQ	D	T	L	P	W	R	O	M	A	E	B	K	V	F	Description
1Ch	01h	D						O				B	K			PRIMARY DEFECT LIST NOT FOUND
1Ch	02h	D						O				B	K			GROWN DEFECT LIST NOT FOUND
1Dh	00h	D	T			W	R	O				B	K			MISCOMPARE DURING VERIFY OPERATION
1Dh	01h	D										B				MISCOMPARE VERIFY OF UNMAPPED LBA
1Eh	00h	D				W		O				B	K			RECOVERED ID WITH ECC CORRECTION
1Fh	00h	D						O					K			PARTIAL DEFECT LIST TRANSFER
20h	00h	D	T	L	P	W	R	O	M	A	E	B	K	V	F	INVALID COMMAND OPERATION CODE
20h	01h	D	T		P	W	R	O	M	A	E	B	K			ACCESS DENIED - INITIATOR PENDING-ENROLLED
20h	02h	D	T		P	W	R	O	M	A	E	B	K			ACCESS DENIED - NO ACCESS RIGHTS
20h	03h	D	T		P	W	R	O	M	A	E	B	K			ACCESS DENIED - INVALID MGMT ID KEY
20h	04h		T													ILLEGAL COMMAND WHILE IN WRITE CAPABLE STATE
20h	05h		T													Obsolete
20h	06h		T													ILLEGAL COMMAND WHILE IN EXPLICIT ADDRESS MODE
20h	07h		T													ILLEGAL COMMAND WHILE IN IMPLICIT ADDRESS MODE
20h	08h	D	T		P	W	R	O	M	A	E	B	K			ACCESS DENIED - ENROLLMENT CONFLICT
20h	09h	D	T		P	W	R	O	M	A	E	B	K			ACCESS DENIED - INVALID LU IDENTIFIER
20h	0Ah	D	T		P	W	R	O	M	A	E	B	K			ACCESS DENIED - INVALID PROXY TOKEN
20h	0Bh	D	T		P	W	R	O	M	A	E	B	K			ACCESS DENIED - ACL LUN CONFLICT
20h	0Ch		T													ILLEGAL COMMAND WHEN NOT IN APPEND-ONLY MODE
21h	00h	D	T			W	R	O				B	K			LOGICAL BLOCK ADDRESS OUT OF RANGE
21h	01h	D	T			W	R	O	M			B	K			INVALID ELEMENT ADDRESS
21h	02h						R									INVALID ADDRESS FOR WRITE
21h	03h						R									INVALID WRITE CROSSING LAYER JUMP
22h	00h	D														ILLEGAL FUNCTION (USE 20 00, 24 00, OR 26 00)
23h	00h	D	T		P		R					B				INVALID TOKEN OPERATION, CAUSE NOT REPORTABLE
24h	00h	D	T	L	P	W	R	O	M	A	E	B	K	V	F	INVALID FIELD IN CDB
24h	01h	D	T	L	P	W	R	O		A	E	B	K	V	F	CDB DECRYPTION ERROR
24h	02h		T													Obsolete
24h	03h		T													Obsolete
24h	04h														F	SECURITY AUDIT VALUE FROZEN
24h	05h														F	SECURITY WORKING KEY FROZEN
24h	06h														F	NONCE NOT UNIQUE
24h	07h														F	NONCE TIMESTAMP OUT OF RANGE
24h	08h	D	T				R		M	A	E	B	K	V		INVALID XCDB
25h	00h	D	T	L	P	W	R	O	M	A	E	B	K	V	F	LOGICAL UNIT NOT SUPPORTED
26h	00h	D	T	L	P	W	R	O	M	A	E	B	K	V	F	INVALID FIELD IN PARAMETER LIST
26h	01h	D	T	L	P	W	R	O	M	A	E	B	K	V	F	PARAMETER NOT SUPPORTED
26h	02h	D	T	L	P	W	R	O	M	A	E	B	K	V	F	PARAMETER VALUE INVALID
26h	03h	D	T	L	P	W	R	O	M	A	E		K			THRESHOLD PARAMETERS NOT SUPPORTED
26h	04h	D	T	L	P	W	R	O	M	A	E	B	K	V	F	INVALID RELEASE OF PERSISTENT RESERVATION
26h	05h	D	T	L	P	W	R	O		A		B	K			DATA DECRYPTION ERROR

Table D.1 — ASC and ASCQ assignments (part 7 of 18)

```
D – Direct Access Block Device (SBC-3)
. T – Sequential Access Device (SSC-4)
.   L – Printer Device (SSC)
.     P – Processor Device (SPC-2)
.     . W – Write Once Block Device (SBC)
.     .   R – C/DVD Device (MMC-6)
.     .     O – Optical Memory Block Device (SBC)
.     .     . M – Media Changer Device (SMC-3)
.     .     .   A – Storage Array Device (SCC-2)
.     .     .     E – SCSI Enclosure Services device (SES-3)
.     .     .     . B – Simplified Direct-Access (Reduced Block) device (RBC)
.     .     .     .   K – Optical Card Reader/Writer device (OCRW)
.     .     .     .     V – Automation/Device Interface device (ADC-3)
.     .     .     .     . F – Object-based Storage Device (OSD-2)
.     .     .     .     .
```

Device Column key
blank = code not used
not blank = code used

ASC	ASCQ	D	T	L	P	W	R	O	M	A	E	B	K	V	F	Description
26h	06h	D	T	L	P	W	R	O					K			TOO MANY TARGET DESCRIPTORS
26h	07h	D	T	L	P	W	R	O					K			UNSUPPORTED TARGET DESCRIPTOR TYPE CODE
26h	08h	D	T	L	P	W	R	O					K			TOO MANY SEGMENT DESCRIPTORS
26h	09h	D	T	L	P	W	R	O					K			UNSUPPORTED SEGMENT DESCRIPTOR TYPE CODE
26h	0Ah	D	T	L	P	W	R	O					K			UNEXPECTED INEXACT SEGMENT
26h	0Bh	D	T	L	P	W	R	O					K			INLINE DATA LENGTH EXCEEDED
26h	0Ch	D	T	L	P	W	R	O					K			INVALID OPERATION FOR COPY SOURCE OR DESTINATION
26h	0Dh	D	T	L	P	W	R	O					K			COPY SEGMENT GRANULARITY VIOLATION
26h	0Eh	D	T		P	W	R	O	M	A	E	B	K			INVALID PARAMETER WHILE PORT IS ENABLED
26h	0Fh														F	INVALID DATA-OUT BUFFER INTEGRITY CHECK VALUE
26h	10h		T													DATA DECRYPTION KEY FAIL LIMIT REACHED
26h	11h		T													INCOMPLETE KEY-ASSOCIATED DATA SET
26h	12h		T													VENDOR SPECIFIC KEY REFERENCE NOT FOUND
27h	00h	D	T			W	R	O				B	K			WRITE PROTECTED
27h	01h	D	T			W	R	O				B	K			HARDWARE WRITE PROTECTED
27h	02h	D	T			W	R	O				B	K			LOGICAL UNIT SOFTWARE WRITE PROTECTED
27h	03h		T				R									ASSOCIATED WRITE PROTECT
27h	04h		T				R									PERSISTENT WRITE PROTECT
27h	05h		T				R									PERMANENT WRITE PROTECT
27h	06h						R								F	CONDITIONAL WRITE PROTECT
27h	07h	D										B				SPACE ALLOCATION FAILED WRITE PROTECT
28h	00h	D	T	L	P	W	R	O	M	A	E	B	K	V	F	NOT READY TO READY CHANGE, MEDIUM MAY HAVE CHANGED
28h	01h	D	T			W	R	O	M			B				IMPORT OR EXPORT ELEMENT ACCESSED
28h	02h						R									FORMAT-LAYER MAY HAVE CHANGED
28h	03h								M							IMPORT/EXPORT ELEMENT ACCESSED, MEDIUM CHANGED
29h	00h	D	T	L	P	W	R	O	M	A	E	B	K	V	F	POWER ON, RESET, OR BUS DEVICE RESET OCCURRED
29h	01h	D	T	L	P	W	R	O	M	A	E	B	K	V	F	POWER ON OCCURRED
29h	02h	D	T	L	P	W	R	O	M	A	E	B	K	V	F	SCSI BUS RESET OCCURRED
29h	03h	D	T	L	P	W	R	O	M	A	E	B	K	V	F	BUS DEVICE RESET FUNCTION OCCURRED
29h	04h	D	T	L	P	W	R	O	M	A	E	B	K	V	F	DEVICE INTERNAL RESET
29h	05h	D	T	L	P	W	R	O	M	A	E	B	K	V	F	TRANSCEIVER MODE CHANGED TO SINGLE-ENDED
29h	06h	D	T	L	P	W	R	O	M	A	E	B	K	V	F	TRANSCEIVER MODE CHANGED TO LVD
29h	07h	D	T	L	P	W	R	O	M	A	E	B	K	V	F	I_T NEXUS LOSS OCCURRED
2Ah	00h	D	T	L		W	R	O	M	A	E	B	K	V	F	PARAMETERS CHANGED
2Ah	01h	D	T	L		W	R	O	M	A	E	B	K	V	F	MODE PARAMETERS CHANGED
2Ah	02h	D	T	L		W	R	O	M	A	E		K			LOG PARAMETERS CHANGED
2Ah	03h	D	T	L	P	W	R	O	M	A	E		K			RESERVATIONS PREEMPTED
2Ah	04h	D	T	L	P	W	R	O	M	A	E					RESERVATIONS RELEASED
2Ah	05h	D	T	L	P	W	R	O	M	A	E					REGISTRATIONS PREEMPTED
2Ah	06h	D	T	L	P	W	R	O	M	A	E	B	K	V	F	ASYMMETRIC ACCESS STATE CHANGED
2Ah	07h	D	T	L	P	W	R	O	M	A	E	B	K	V	F	IMPLICIT ASYMMETRIC ACCESS STATE TRANSITION FAILED

Table D.1 — ASC and ASCQ assignments (part 8 of 18)

```
                D – Direct Access Block Device (SBC-3)                    Device Column key
                . T – Sequential Access Device (SSC-4)                    blank = code not used
                .   L – Printer Device (SSC)                              not blank = code used
                .     P – Processor Device (SPC-2)
                .     . W – Write Once Block Device (SBC)
                .     .   R – C/DVD Device (MMC-6)
                .     .     O – Optical Memory Block Device (SBC)
                .     .     . M – Media Changer Device (SMC-3)
                .     .     .   A – Storage Array Device (SCC-2)
                .     .     .     E – SCSI Enclosure Services device (SES-3)
                .     .     .     . B – Simplified Direct-Access (Reduced Block) device (RBC)
                .     .     .     .   K – Optical Card Reader/Writer device (OCRW)
                .     .     .     .     V – Automation/Device Interface device (ADC-3)
                .     .     .     .     . F – Object-based Storage Device (OSD-2)
                .     .     .     .     .
ASC ASCQ        DTLPWROMAEBKVF  Description
2Ah 08h         DT  WROMAEBKVF  PRIORITY CHANGED
2Ah 09h         D               CAPACITY DATA HAS CHANGED
2Ah 0Ah         DT              ERROR HISTORY I_T NEXUS CLEARED
2Ah 0Bh         DT              ERROR HISTORY SNAPSHOT RELEASED
2Ah 0Ch                       F ERROR RECOVERY ATTRIBUTES HAVE CHANGED
2Ah 0Dh           T             DATA ENCRYPTION CAPABILITIES CHANGED
2Ah 10h         DT        M E V TIMESTAMP CHANGED
2Ah 11h           T             DATA ENCRYPTION PARAMETERS CHANGED BY ANOTHER I_T NEXUS
2Ah 12h           T             DATA ENCRYPTION PARAMETERS CHANGED BY VENDOR SPECIFIC EVENT
2Ah 13h           T             DATA ENCRYPTION KEY INSTANCE COUNTER HAS CHANGED
2Ah 14h         DT    R MAEBKV  SA CREATION CAPABILITIES DATA HAS CHANGED
2Bh 00h         DTLPWRO     K   COPY CANNOT EXECUTE SINCE HOST CANNOT DISCONNECT
2Ch 00h         DTLPWROMAEBKVF  COMMAND SEQUENCE ERROR
2Ch 01h                         TOO MANY WINDOWS SPECIFIED
2Ch 02h                         INVALID COMBINATION OF WINDOWS SPECIFIED
2Ch 03h             R           CURRENT PROGRAM AREA IS NOT EMPTY
2Ch 04h             R           CURRENT PROGRAM AREA IS EMPTY
2Ch 05h                   B     ILLEGAL POWER CONDITION REQUEST
2Ch 06h             R           PERSISTENT PREVENT CONFLICT
2Ch 07h         DTLPWROMAEBKVF  PREVIOUS BUSY STATUS
2Ch 08h         DTLPWROMAEBKVF  PREVIOUS TASK SET FULL STATUS
2Ch 09h         DTLPWROM  EBKVF PREVIOUS RESERVATION CONFLICT STATUS
2Ch 0Ah                       F PARTITION OR COLLECTION CONTAINS USER OBJECTS
2Ch 0Bh           T             NOT RESERVED
2Ch 0Ch         D               ORWRITE GENERATION DOES NOT MATCH
2Dh 00h           T             OVERWRITE ERROR ON UPDATE IN PLACE
2Eh 00h             R           INSUFFICIENT TIME FOR OPERATION
2Fh 00h         DTLPWROMAEBKVF  COMMANDS CLEARED BY ANOTHER INITIATOR
2Fh 01h         D               COMMANDS CLEARED BY POWER LOSS NOTIFICATION
2Fh 02h         DTLPWROMAEBKVF  COMMANDS CLEARED BY DEVICE SERVER
30h 00h         DT  WROM    BK  INCOMPATIBLE MEDIUM INSTALLED
30h 01h         DT  WRO     BK  CANNOT READ MEDIUM - UNKNOWN FORMAT
30h 02h         DT  WRO     BK  CANNOT READ MEDIUM - INCOMPATIBLE FORMAT
30h 03h         DT    R M     K CLEANING CARTRIDGE INSTALLED
30h 04h         DT  WRO     BK  CANNOT WRITE MEDIUM - UNKNOWN FORMAT
30h 05h         DT  WRO     BK  CANNOT WRITE MEDIUM - INCOMPATIBLE FORMAT
30h 06h         DT  WRO     B   CANNOT FORMAT MEDIUM - INCOMPATIBLE MEDIUM
30h 07h         DTL WROMAEBKVF  CLEANING FAILURE
30h 08h             R           CANNOT WRITE - APPLICATION CODE MISMATCH
30h 09h             R           CURRENT SESSION NOT FIXATED FOR APPEND
30h 0Ah         DT  WRO  AEBK   CLEANING REQUEST REJECTED
```

Table D.1 — ASC and ASCQ assignments (part 9 of 18)

```
                D – Direct Access Block Device (SBC-3)                    Device Column key
                . T – Sequential Access Device (SSC-4)                    blank = code not used
                .   L – Printer Device (SSC)                              not blank = code used
                .     P – Processor Device (SPC-2)
                .     . W – Write Once Block Device (SBC)
                .     .   R – C/DVD Device (MMC-6)
                .     .     O – Optical Memory Block Device (SBC)
                .     .     . M – Media Changer Device (SMC-3)
                .     .     .   A – Storage Array Device (SCC-2)
                .     .     .     E – SCSI Enclosure Services device (SES-3)
                .     .     .     . B – Simplified Direct-Access (Reduced Block) device (RBC)
                .     .     .     .   K – Optical Card Reader/Writer device (OCRW)
                .     .     .     .     V – Automation/Device Interface device (ADC-3)
                .     .     .     .     . F – Object-based Storage Device (OSD-2)
                .     .     .     .     .
ASC  ASCQ  DTLPWROMAEBKVF  Description
30h  0Ch     T             WORM MEDIUM - OVERWRITE ATTEMPTED
30h  0Dh     T             WORM MEDIUM - INTEGRITY CHECK
30h  10h         R         MEDIUM NOT FORMATTED
30h  11h             M     INCOMPATIBLE VOLUME TYPE
30h  12h             M     INCOMPATIBLE VOLUME QUALIFIER
30h  13h             M     CLEANING VOLUME EXPIRED
31h  00h  DT   WRO     BK  MEDIUM FORMAT CORRUPTED
31h  01h  D L   RO     B   FORMAT COMMAND FAILED
31h  02h         R         ZONED FORMATTING FAILED DUE TO SPARE LINKING
31h  03h  D            B   SANITIZE COMMAND FAILED
32h  00h  D     W O    BK  NO DEFECT SPARE LOCATION AVAILABLE
32h  01h  D     W O    BK  DEFECT LIST UPDATE FAILURE
33h  00h     T             TAPE LENGTH ERROR
34h  00h  DTLPWROMAEBKVF  ENCLOSURE FAILURE
35h  00h  DTLPWROMAEBKVF  ENCLOSURE SERVICES FAILURE
35h  01h  DTLPWROMAEBKVF  UNSUPPORTED ENCLOSURE FUNCTION
35h  02h  DTLPWROMAEBKVF  ENCLOSURE SERVICES UNAVAILABLE
35h  03h  DTLPWROMAEBKVF  ENCLOSURE SERVICES TRANSFER FAILURE
35h  04h  DTLPWROMAEBKVF  ENCLOSURE SERVICES TRANSFER REFUSED
35h  05h  DTL  WROMAEBKVF ENCLOSURE SERVICES CHECKSUM ERROR
36h  00h       L          RIBBON, INK, OR TONER FAILURE
37h  00h  DTL  WROMAEBKVF ROUNDED PARAMETER
38h  00h               B  EVENT STATUS NOTIFICATION
38h  02h               B  ESN - POWER MANAGEMENT CLASS EVENT
38h  04h               B  ESN - MEDIA CLASS EVENT
38h  06h               B  ESN - DEVICE BUSY CLASS EVENT
38h  07h  D               THIN PROVISIONING SOFT THRESHOLD REACHED
39h  00h  DTL  WROMAE  K  SAVING PARAMETERS NOT SUPPORTED
3Ah  00h  DTL  WROM    BK MEDIUM NOT PRESENT
3Ah  01h  DT   WROM    BK MEDIUM NOT PRESENT - TRAY CLOSED
3Ah  02h  DT   WROM    BK MEDIUM NOT PRESENT - TRAY OPEN
3Ah  03h  DT   WROM    B  MEDIUM NOT PRESENT - LOADABLE
3Ah  04h  DT   WRO     B  MEDIUM NOT PRESENT - MEDIUM AUXILIARY MEMORY ACCESSIBLE
3Bh  00h     TL            SEQUENTIAL POSITIONING ERROR
3Bh  01h     T             TAPE POSITION ERROR AT BEGINNING-OF-MEDIUM
3Bh  02h     T             TAPE POSITION ERROR AT END-OF-MEDIUM
3Bh  03h       L           TAPE OR ELECTRONIC VERTICAL FORMS UNIT NOT READY
3Bh  04h       L           SLEW FAILURE
3Bh  05h       L           PAPER JAM
3Bh  06h       L           FAILED TO SENSE TOP-OF-FORM
3Bh  07h       L           FAILED TO SENSE BOTTOM-OF-FORM
```

Table D.1 — ASC and ASCQ assignments (part 10 of 18)

```
                D – Direct Access Block Device (SBC-3)                    Device Column key
                . T – Sequential Access Device (SSC-4)                    blank = code not used
                .   L – Printer Device (SSC)                              not blank = code used
                .     P – Processor Device (SPC-2)
                .     . W – Write Once Block Device (SBC)
                .     .   R – C/DVD Device (MMC-6)
                .     .     O – Optical Memory Block Device (SBC)
                .     .     . M – Media Changer Device (SMC-3)
                .     .     .   A – Storage Array Device (SCC-2)
                .     .     .     E – SCSI Enclosure Services device (SES-3)
                .     .     .     . B – Simplified Direct-Access (Reduced Block) device (RBC)
                .     .     .     .   K – Optical Card Reader/Writer device (OCRW)
                .     .     .     .     V – Automation/Device Interface device (ADC-3)
                .     .     .     .     . F – Object-based Storage Device (OSD-2)
                .     .     .     .     .
ASC  ASCQ  DTLPWROMAEBKVF  Description
```

ASC	ASCQ	DTLPWROMAEBKVF	Description
3Bh	08h	T	REPOSITION ERROR
3Bh	09h		READ PAST END OF MEDIUM
3Bh	0Ah		READ PAST BEGINNING OF MEDIUM
3Bh	0Bh		POSITION PAST END OF MEDIUM
3Bh	0Ch	T	POSITION PAST BEGINNING OF MEDIUM
3Bh	0Dh	DT WROM BK	MEDIUM DESTINATION ELEMENT FULL
3Bh	0Eh	DT WROM BK	MEDIUM SOURCE ELEMENT EMPTY
3Bh	0Fh	R	END OF MEDIUM REACHED
3Bh	11h	DT WROM BK	MEDIUM MAGAZINE NOT ACCESSIBLE
3Bh	12h	DT WROM BK	MEDIUM MAGAZINE REMOVED
3Bh	13h	DT WROM BK	MEDIUM MAGAZINE INSERTED
3Bh	14h	DT WROM BK	MEDIUM MAGAZINE LOCKED
3Bh	15h	DT WROM BK	MEDIUM MAGAZINE UNLOCKED
3Bh	16h	R	MECHANICAL POSITIONING OR CHANGER ERROR
3Bh	17h	F	READ PAST END OF USER OBJECT
3Bh	18h	M	ELEMENT DISABLED
3Bh	19h	M	ELEMENT ENABLED
3Bh	1Ah	M	DATA TRANSFER DEVICE REMOVED
3Bh	1Bh	M	DATA TRANSFER DEVICE INSERTED
3Ch	00h		
3Dh	00h	DTLPWROMAE K	INVALID BITS IN IDENTIFY MESSAGE
3Eh	00h	DTLPWROMAEBKVF	LOGICAL UNIT HAS NOT SELF-CONFIGURED YET
3Eh	01h	DTLPWROMAEBKVF	LOGICAL UNIT FAILURE
3Eh	02h	DTLPWROMAEBKVF	TIMEOUT ON LOGICAL UNIT
3Eh	03h	DTLPWROMAEBKVF	LOGICAL UNIT FAILED SELF-TEST
3Eh	04h	DTLPWROMAEBKVF	LOGICAL UNIT UNABLE TO UPDATE SELF-TEST LOG
3Fh	00h	DTLPWROMAEBKVF	TARGET OPERATING CONDITIONS HAVE CHANGED
3Fh	01h	DTLPWROMAEBKVF	MICROCODE HAS BEEN CHANGED
3Fh	02h	DTLPWROM BK	CHANGED OPERATING DEFINITION
3Fh	03h	DTLPWROMAEBKVF	INQUIRY DATA HAS CHANGED
3Fh	04h	DT WROMAEBK	COMPONENT DEVICE ATTACHED
3Fh	05h	DT WROMAEBK	DEVICE IDENTIFIER CHANGED
3Fh	06h	DT WROMAEB	REDUNDANCY GROUP CREATED OR MODIFIED
3Fh	07h	DT WROMAEB	REDUNDANCY GROUP DELETED
3Fh	08h	DT WROMAEB	SPARE CREATED OR MODIFIED
3Fh	09h	DT WROMAEB	SPARE DELETED
3Fh	0Ah	DT WROMAEBK	VOLUME SET CREATED OR MODIFIED
3Fh	0Bh	DT WROMAEBK	VOLUME SET DELETED
3Fh	0Ch	DT WROMAEBK	VOLUME SET DEASSIGNED
3Fh	0Dh	DT WROMAEBK	VOLUME SET REASSIGNED
3Fh	0Eh	DTLPWROMAE	REPORTED LUNS DATA HAS CHANGED

Table D.1 — ASC and ASCQ assignments (part 11 of 18)

```
                D – Direct Access Block Device (SBC-3)                    Device Column key
                . T – Sequential Access Device (SSC-4)                    blank = code not used
                .   L – Printer Device (SSC)                              not blank = code used
                .     P – Processor Device (SPC-2)
                .     . W – Write Once Block Device (SBC)
                .     .   R – C/DVD Device (MMC-6)
                .     .     O – Optical Memory Block Device (SBC)
                .     .     . M – Media Changer Device (SMC-3)
                .     .     .   A – Storage Array Device (SCC-2)
                .     .     .     E – SCSI Enclosure Services device (SES-3)
                .     .     .     . B – Simplified Direct-Access (Reduced Block) device (RBC)
                .     .     .     .   K – Optical Card Reader/Writer device (OCRW)
                .     .     .     .     V – Automation/Device Interface device (ADC-3)
                .     .     .     .     . F – Object-based Storage Device (OSD-2)
                .     .     .     .     .
ASC  ASCQ       DTLPWROMAEBKVF  Description
3Fh  0Fh        DTLPWROMAEBKVF  ECHO BUFFER OVERWRITTEN
3Fh  10h        DT   WROM  B    MEDIUM LOADABLE
3Fh  11h        DT   WROM  B    MEDIUM AUXILIARY MEMORY ACCESSIBLE
3Fh  12h        DTLPWR MAEBK F  iSCSI IP ADDRESS ADDED
3Fh  13h        DTLPWR MAEBK F  iSCSI IP ADDRESS REMOVED
3Fh  14h        DTLPWR MAEBK F  iSCSI IP ADDRESS CHANGED
40h  00h        D               RAM FAILURE (SHOULD USE 40 NN)
40h  NNh        DTLPWROMAEBKVF  DIAGNOSTIC FAILURE ON COMPONENT NN (80h-FFh)
41h  00h        D               DATA PATH FAILURE (SHOULD USE 40 NN)
42h  00h        D               POWER-ON OR SELF-TEST FAILURE (SHOULD USE 40 NN)
43h  00h        DTLPWROMAEBKVF  MESSAGE ERROR
44h  00h        DTLPWROMAEBKVF  INTERNAL TARGET FAILURE
44h  71h        DT              ATA DEVICE FAILED SET FEATURES
45h  00h        DTLPWROMAEBKVF  SELECT OR RESELECT FAILURE
46h  00h        DTLPWROM  BK    UNSUCCESSFUL SOFT RESET
47h  00h        DTLPWROMAEBKVF  SCSI PARITY ERROR
47h  01h        DTLPWROMAEBKVF  DATA PHASE CRC ERROR DETECTED
47h  02h        DTLPWROMAEBKVF  SCSI PARITY ERROR DETECTED DURING ST DATA PHASE
47h  03h        DTLPWROMAEBKVF  INFORMATION UNIT iuCRC ERROR DETECTED
47h  04h        DTLPWROMAEBKVF  ASYNCHRONOUS INFORMATION PROTECTION ERROR DETECTED
47h  05h        DTLPWROMAEBKVF  PROTOCOL SERVICE CRC ERROR
47h  06h        DT        MAEBKVF  PHY TEST FUNCTION IN PROGRESS
47h  7Fh        DT  PWROMAEBK   SOME COMMANDS CLEARED BY ISCSI PROTOCOL EVENT
48h  00h        DTLPWROMAEBKVF  INITIATOR DETECTED ERROR MESSAGE RECEIVED
49h  00h        DTLPWROMAEBKVF  INVALID MESSAGE ERROR
4Ah  00h        DTLPWROMAEBKVF  COMMAND PHASE ERROR
4Bh  00h        DTLPWROMAEBKVF  DATA PHASE ERROR
4Bh  01h        DT  PWROMAEBK   INVALID TARGET PORT TRANSFER TAG RECEIVED
4Bh  02h        DT  PWROMAEBK   TOO MUCH WRITE DATA
4Bh  03h        DT  PWROMAEBK   ACK/NAK TIMEOUT
4Bh  04h        DT  PWROMAEBK   NAK RECEIVED
4Bh  05h        DT  PWROMAEBK   DATA OFFSET ERROR
4Bh  06h        DT  PWROMAEBK   INITIATOR RESPONSE TIMEOUT
4Bh  07h        DT  PWROMAEBK F CONNECTION LOST
4Ch  00h        DTLPWROMAEBKVF  LOGICAL UNIT FAILED SELF-CONFIGURATION
4Dh  NNh        DTLPWROMAEBKVF  TAGGED OVERLAPPED COMMANDS (NN = TASK TAG)
4Eh  00h        DTLPWROMAEBKVF  OVERLAPPED COMMANDS ATTEMPTED
4Fh  00h
50h  00h          T             WRITE APPEND ERROR
50h  01h          T             WRITE APPEND POSITION ERROR
50h  02h          T             POSITION ERROR RELATED TO TIMING
```

Table D.1 — ASC and ASCQ assignments (part 12 of 18)

```
                D – Direct Access Block Device (SBC-3)                    Device Column key
                . T – Sequential Access Device (SSC-4)                    blank = code not used
                .   L – Printer Device (SSC)                              not blank = code used
                .     P – Processor Device (SPC-2)
                .     . W – Write Once Block Device (SBC)
                .     .   R – C/DVD Device (MMC-6)
                .     .     O – Optical Memory Block Device (SBC)
                .     .     . M – Media Changer Device (SMC-3)
                .     .     .   A – Storage Array Device (SCC-2)
                .     .     .     E – SCSI Enclosure Services device (SES-3)
                .     .     .     . B – Simplified Direct-Access (Reduced Block) device (RBC)
                .     .     .     .   K – Optical Card Reader/Writer device (OCRW)
                .     .     .     .     V – Automation/Device Interface device (ADC-3)
                .     .     .     .     . F – Object-based Storage Device (OSD-2)
                .     .     .     .     .
ASC ASCQ        DTLPWROMAEBKVF  Description
```

ASC	ASCQ	DTLPWROMAEBKVF	Description
51h	00h	T RO	ERASE FAILURE
51h	01h	R	ERASE FAILURE - INCOMPLETE ERASE OPERATION DETECTED
52h	00h	T	CARTRIDGE FAULT
53h	00h	DTL WROM BK	MEDIA LOAD OR EJECT FAILED
53h	01h	T	UNLOAD TAPE FAILURE
53h	02h	DT WROM BK	MEDIUM REMOVAL PREVENTED
53h	03h	M	MEDIUM REMOVAL PREVENTED BY DATA TRANSFER ELEMENT
53h	04h	T	MEDIUM THREAD OR UNTHREAD FAILURE
53h	05h	M	VOLUME IDENTIFIER INVALID
53h	06h	M	VOLUME IDENTIFIER MISSING
53h	07h	M	DUPLICATE VOLUME IDENTIFIER
53h	08h	M	ELEMENT STATUS UNKNOWN
54h	00h	P	SCSI TO HOST SYSTEM INTERFACE FAILURE
55h	00h	P	SYSTEM RESOURCE FAILURE
55h	01h	D O BK	SYSTEM BUFFER FULL
55h	02h	DTLPWROMAE K	INSUFFICIENT RESERVATION RESOURCES
55h	03h	DTLPWROMAE K	INSUFFICIENT RESOURCES
55h	04h	DTLPWROMAE K	INSUFFICIENT REGISTRATION RESOURCES
55h	05h	DT PWROMAEBK	INSUFFICIENT ACCESS CONTROL RESOURCES
55h	06h	DT WROM B	AUXILIARY MEMORY OUT OF SPACE
55h	07h	F	QUOTA ERROR
55h	08h	T	MAXIMUM NUMBER OF SUPPLEMENTAL DECRYPTION KEYS EXCEEDED
55h	09h	M	MEDIUM AUXILIARY MEMORY NOT ACCESSIBLE
55h	0Ah	M	DATA CURRENTLY UNAVAILABLE
55h	0Bh	DTLPWROMAEBKVF	INSUFFICIENT POWER FOR OPERATION
56h	00h		
57h	00h	R	UNABLE TO RECOVER TABLE-OF-CONTENTS
58h	00h	O	GENERATION DOES NOT EXIST
59h	00h	O	UPDATED BLOCK READ
5Ah	00h	DTLPWRO BK	OPERATOR REQUEST OR STATE CHANGE INPUT
5Ah	01h	DT WROM BK	OPERATOR MEDIUM REMOVAL REQUEST
5Ah	02h	DT WRO A BK	OPERATOR SELECTED WRITE PROTECT
5Ah	03h	DT WRO A BK	OPERATOR SELECTED WRITE PERMIT
5Bh	00h	DTLPWROM K	LOG EXCEPTION
5Bh	01h	DTLPWROM K	THRESHOLD CONDITION MET
5Bh	02h	DTLPWROM K	LOG COUNTER AT MAXIMUM
5Bh	03h	DTLPWROM K	LOG LIST CODES EXHAUSTED
5Ch	00h	D O	RPL STATUS CHANGE
5Ch	01h	D O	SPINDLES SYNCHRONIZED
5Ch	02h	D O	SPINDLES NOT SYNCHRONIZED
5Dh	00h	DTLPWROMAEBKVF	FAILURE PREDICTION THRESHOLD EXCEEDED

Table D.1 — ASC and ASCQ assignments (part 13 of 18)

```
D – Direct Access Block Device (SBC-3)
. T – Sequential Access Device (SSC-4)
.   L – Printer Device (SSC)
.     P – Processor Device (SPC-2)
.     . W – Write Once Block Device (SBC)
.     .   R – C/DVD Device (MMC-6)
.     .     O – Optical Memory Block Device (SBC)
.     .     . M – Media Changer Device (SMC-3)
.     .     .   A – Storage Array Device (SCC-2)
.     .     .     E – SCSI Enclosure Services device (SES-3)
.     .     .     . B – Simplified Direct-Access (Reduced Block) device (RBC)
.     .     .     .   K – Optical Card Reader/Writer device (OCRW)
.     .     .     .     V – Automation/Device Interface device (ADC-3)
.     .     .     .     . F – Object-based Storage Device (OSD-2)
.     .     .     .     .
```

Device Column key
blank = code not used
not blank = code used

ASC	ASCQ	D	T	L	P	W	R	O	M	A	E	B	K	V	F	Description
5Dh	01h						R					B				MEDIA FAILURE PREDICTION THRESHOLD EXCEEDED
5Dh	02h						R									LOGICAL UNIT FAILURE PREDICTION THRESHOLD EXCEEDED
5Dh	03h						R									SPARE AREA EXHAUSTION PREDICTION THRESHOLD EXCEEDED
5Dh	10h	D										B				HARDWARE IMPENDING FAILURE GENERAL HARD DRIVE FAILURE
5Dh	11h	D										B				HARDWARE IMPENDING FAILURE DRIVE ERROR RATE TOO HIGH
5Dh	12h	D										B				HARDWARE IMPENDING FAILURE DATA ERROR RATE TOO HIGH
5Dh	13h	D										B				HARDWARE IMPENDING FAILURE SEEK ERROR RATE TOO HIGH
5Dh	14h	D										B				HARDWARE IMPENDING FAILURE TOO MANY BLOCK REASSIGNS
5Dh	15h	D										B				HARDWARE IMPENDING FAILURE ACCESS TIMES TOO HIGH
5Dh	16h	D										B				HARDWARE IMPENDING FAILURE START UNIT TIMES TOO HIGH
5Dh	17h	D										B				HARDWARE IMPENDING FAILURE CHANNEL PARAMETRICS
5Dh	18h	D										B				HARDWARE IMPENDING FAILURE CONTROLLER DETECTED
5Dh	19h	D										B				HARDWARE IMPENDING FAILURE THROUGHPUT PERFORMANCE
5Dh	1Ah	D										B				HARDWARE IMPENDING FAILURE SEEK TIME PERFORMANCE
5Dh	1Bh	D										B				HARDWARE IMPENDING FAILURE SPIN-UP RETRY COUNT
5Dh	1Ch	D										B				HARDWARE IMPENDING FAILURE DRIVE CALIBRATION RETRY COUNT
5Dh	20h	D										B				CONTROLLER IMPENDING FAILURE GENERAL HARD DRIVE FAILURE
5Dh	21h	D										B				CONTROLLER IMPENDING FAILURE DRIVE ERROR RATE TOO HIGH
5Dh	22h	D										B				CONTROLLER IMPENDING FAILURE DATA ERROR RATE TOO HIGH
5Dh	23h	D										B				CONTROLLER IMPENDING FAILURE SEEK ERROR RATE TOO HIGH
5Dh	24h	D										B				CONTROLLER IMPENDING FAILURE TOO MANY BLOCK REASSIGNS
5Dh	25h	D										B				CONTROLLER IMPENDING FAILURE ACCESS TIMES TOO HIGH
5Dh	26h	D										B				CONTROLLER IMPENDING FAILURE START UNIT TIMES TOO HIGH
5Dh	27h	D										B				CONTROLLER IMPENDING FAILURE CHANNEL PARAMETRICS
5Dh	28h	D										B				CONTROLLER IMPENDING FAILURE CONTROLLER DETECTED
5Dh	29h	D										B				CONTROLLER IMPENDING FAILURE THROUGHPUT PERFORMANCE
5Dh	2Ah	D										B				CONTROLLER IMPENDING FAILURE SEEK TIME PERFORMANCE
5Dh	2Bh	D										B				CONTROLLER IMPENDING FAILURE SPIN-UP RETRY COUNT
5Dh	2Ch	D										B				CONTROLLER IMPENDING FAILURE DRIVE CALIBRATION RETRY COUNT
5Dh	30h	D										B				DATA CHANNEL IMPENDING FAILURE GENERAL HARD DRIVE FAILURE
5Dh	31h	D										B				DATA CHANNEL IMPENDING FAILURE DRIVE ERROR RATE TOO HIGH
5Dh	32h	D										B				DATA CHANNEL IMPENDING FAILURE DATA ERROR RATE TOO HIGH
5Dh	33h	D										B				DATA CHANNEL IMPENDING FAILURE SEEK ERROR RATE TOO HIGH
5Dh	34h	D										B				DATA CHANNEL IMPENDING FAILURE TOO MANY BLOCK REASSIGNS
5Dh	35h	D										B				DATA CHANNEL IMPENDING FAILURE ACCESS TIMES TOO HIGH
5Dh	36h	D										B				DATA CHANNEL IMPENDING FAILURE START UNIT TIMES TOO HIGH
5Dh	37h	D										B				DATA CHANNEL IMPENDING FAILURE CHANNEL PARAMETRICS
5Dh	38h	D										B				DATA CHANNEL IMPENDING FAILURE CONTROLLER DETECTED
5Dh	39h	D										B				DATA CHANNEL IMPENDING FAILURE THROUGHPUT PERFORMANCE
5Dh	3Ah	D										B				DATA CHANNEL IMPENDING FAILURE SEEK TIME PERFORMANCE
5Dh	3Bh	D										B				DATA CHANNEL IMPENDING FAILURE SPIN-UP RETRY COUNT

Table D.1 — ASC and ASCQ assignments (part 14 of 18)

```
D – Direct Access Block Device (SBC-3)
. T – Sequential Access Device (SSC-4)
.   L – Printer Device (SSC)
.     P – Processor Device (SPC-2)
.     . W – Write Once Block Device (SBC)
.     .   R – C/DVD Device (MMC-6)
.     .     O – Optical Memory Block Device (SBC)
.     .     . M – Media Changer Device (SMC-3)
.     .     .   A – Storage Array Device (SCC-2)
.     .     .     E – SCSI Enclosure Services device (SES-3)
.     .     .     . B – Simplified Direct-Access (Reduced Block) device (RBC)
.     .     .     .   K – Optical Card Reader/Writer device (OCRW)
.     .     .     .     V – Automation/Device Interface device (ADC-3)
.     .     .     .     . F – Object-based Storage Device (OSD-2)
.     .     .     .     .
```

Device Column key
blank = code not used
not blank = code used

ASC	ASCQ	DTLPWROMAEBKVF	Description
5Dh	3Ch	D B	DATA CHANNEL IMPENDING FAILURE DRIVE CALIBRATION RETRY COUNT
5Dh	40h	D B	SERVO IMPENDING FAILURE GENERAL HARD DRIVE FAILURE
5Dh	41h	D B	SERVO IMPENDING FAILURE DRIVE ERROR RATE TOO HIGH
5Dh	42h	D B	SERVO IMPENDING FAILURE DATA ERROR RATE TOO HIGH
5Dh	43h	D B	SERVO IMPENDING FAILURE SEEK ERROR RATE TOO HIGH
5Dh	44h	D B	SERVO IMPENDING FAILURE TOO MANY BLOCK REASSIGNS
5Dh	45h	D B	SERVO IMPENDING FAILURE ACCESS TIMES TOO HIGH
5Dh	46h	D B	SERVO IMPENDING FAILURE START UNIT TIMES TOO HIGH
5Dh	47h	D B	SERVO IMPENDING FAILURE CHANNEL PARAMETRICS
5Dh	48h	D B	SERVO IMPENDING FAILURE CONTROLLER DETECTED
5Dh	49h	D B	SERVO IMPENDING FAILURE THROUGHPUT PERFORMANCE
5Dh	4Ah	D B	SERVO IMPENDING FAILURE SEEK TIME PERFORMANCE
5Dh	4Bh	D B	SERVO IMPENDING FAILURE SPIN-UP RETRY COUNT
5Dh	4Ch	D B	SERVO IMPENDING FAILURE DRIVE CALIBRATION RETRY COUNT
5Dh	50h	D B	SPINDLE IMPENDING FAILURE GENERAL HARD DRIVE FAILURE
5Dh	51h	D B	SPINDLE IMPENDING FAILURE DRIVE ERROR RATE TOO HIGH
5Dh	52h	D B	SPINDLE IMPENDING FAILURE DATA ERROR RATE TOO HIGH
5Dh	53h	D B	SPINDLE IMPENDING FAILURE SEEK ERROR RATE TOO HIGH
5Dh	54h	D B	SPINDLE IMPENDING FAILURE TOO MANY BLOCK REASSIGNS
5Dh	55h	D B	SPINDLE IMPENDING FAILURE ACCESS TIMES TOO HIGH
5Dh	56h	D B	SPINDLE IMPENDING FAILURE START UNIT TIMES TOO HIGH
5Dh	57h	D B	SPINDLE IMPENDING FAILURE CHANNEL PARAMETRICS
5Dh	58h	D B	SPINDLE IMPENDING FAILURE CONTROLLER DETECTED
5Dh	59h	D B	SPINDLE IMPENDING FAILURE THROUGHPUT PERFORMANCE
5Dh	5Ah	D B	SPINDLE IMPENDING FAILURE SEEK TIME PERFORMANCE
5Dh	5Bh	D B	SPINDLE IMPENDING FAILURE SPIN-UP RETRY COUNT
5Dh	5Ch	D B	SPINDLE IMPENDING FAILURE DRIVE CALIBRATION RETRY COUNT
5Dh	60h	D B	FIRMWARE IMPENDING FAILURE GENERAL HARD DRIVE FAILURE
5Dh	61h	D B	FIRMWARE IMPENDING FAILURE DRIVE ERROR RATE TOO HIGH
5Dh	62h	D B	FIRMWARE IMPENDING FAILURE DATA ERROR RATE TOO HIGH
5Dh	63h	D B	FIRMWARE IMPENDING FAILURE SEEK ERROR RATE TOO HIGH
5Dh	64h	D B	FIRMWARE IMPENDING FAILURE TOO MANY BLOCK REASSIGNS
5Dh	65h	D B	FIRMWARE IMPENDING FAILURE ACCESS TIMES TOO HIGH
5Dh	66h	D B	FIRMWARE IMPENDING FAILURE START UNIT TIMES TOO HIGH
5Dh	67h	D B	FIRMWARE IMPENDING FAILURE CHANNEL PARAMETRICS
5Dh	68h	D B	FIRMWARE IMPENDING FAILURE CONTROLLER DETECTED
5Dh	69h	D B	FIRMWARE IMPENDING FAILURE THROUGHPUT PERFORMANCE
5Dh	6Ah	D B	FIRMWARE IMPENDING FAILURE SEEK TIME PERFORMANCE
5Dh	6Bh	D B	FIRMWARE IMPENDING FAILURE SPIN-UP RETRY COUNT
5Dh	6Ch	D B	FIRMWARE IMPENDING FAILURE DRIVE CALIBRATION RETRY COUNT
5Dh	FFh	DTLPWROMAEBKVF	FAILURE PREDICTION THRESHOLD EXCEEDED (FALSE)

Table D.1 — ASC and ASCQ assignments (part 15 of 18)

```
                D – Direct Access Block Device (SBC-3)                    Device Column key
                . T – Sequential Access Device (SSC-4)                    blank = code not used
                .   L – Printer Device (SSC)                              not blank = code used
                .     P – Processor Device (SPC-2)
                .     . W – Write Once Block Device (SBC)
                .     .   R – C/DVD Device (MMC-6)
                .     .     O – Optical Memory Block Device (SBC)
                .     .     . M – Media Changer Device (SMC-3)
                .     .     .   A – Storage Array Device (SCC-2)
                .     .     .     E – SCSI Enclosure Services device (SES-3)
                .     .     .     . B – Simplified Direct-Access (Reduced Block) device (RBC)
                .     .     .     .   K – Optical Card Reader/Writer device (OCRW)
                .     .     .     .     V – Automation/Device Interface device (ADC-3)
                .     .     .     .     . F – Object-based Storage Device (OSD-2)
                .     .     .     .     .
ASC ASCQ        DTLPWROMAEBKVF  Description
```

ASC	ASCQ	DTLPWROMAEBKVF	Description
5Eh	00h	DTLPWRO A K	LOW POWER CONDITION ON
5Eh	01h	DTLPWRO A K	IDLE CONDITION ACTIVATED BY TIMER
5Eh	02h	DTLPWRO A K	STANDBY CONDITION ACTIVATED BY TIMER
5Eh	03h	DTLPWRO A K	IDLE CONDITION ACTIVATED BY COMMAND
5Eh	04h	DTLPWRO A K	STANDBY CONDITION ACTIVATED BY COMMAND
5Eh	05h	DTLPWRO A K	IDLE_B CONDITION ACTIVATED BY TIMER
5Eh	06h	DTLPWRO A K	IDLE_B CONDITION ACTIVATED BY COMMAND
5Eh	07h	DTLPWRO A K	IDLE_C CONDITION ACTIVATED BY TIMER
5Eh	08h	DTLPWRO A K	IDLE_C CONDITION ACTIVATED BY COMMAND
5Eh	09h	DTLPWRO A K	STANDBY_Y CONDITION ACTIVATED BY TIMER
5Eh	0Ah	DTLPWRO A K	STANDBY_Y CONDITION ACTIVATED BY COMMAND
5Eh	41h	B	POWER STATE CHANGE TO ACTIVE
5Eh	42h	B	POWER STATE CHANGE TO IDLE
5Eh	43h	B	POWER STATE CHANGE TO STANDBY
5Eh	45h	B	POWER STATE CHANGE TO SLEEP
5Eh	47h	BK	POWER STATE CHANGE TO DEVICE CONTROL
5Fh	00h		
60h	00h		LAMP FAILURE
61h	00h		VIDEO ACQUISITION ERROR
61h	01h		UNABLE TO ACQUIRE VIDEO
61h	02h		OUT OF FOCUS
62h	00h		SCAN HEAD POSITIONING ERROR
63h	00h	R	END OF USER AREA ENCOUNTERED ON THIS TRACK
63h	01h	R	PACKET DOES NOT FIT IN AVAILABLE SPACE
64h	00h	R	ILLEGAL MODE FOR THIS TRACK
64h	01h	R	INVALID PACKET SIZE
65h	00h	DTLPWROMAEBKVF	VOLTAGE FAULT
66h	00h		AUTOMATIC DOCUMENT FEEDER COVER UP
66h	01h		AUTOMATIC DOCUMENT FEEDER LIFT UP
66h	02h		DOCUMENT JAM IN AUTOMATIC DOCUMENT FEEDER
66h	03h		DOCUMENT MISS FEED AUTOMATIC IN DOCUMENT FEEDER
67h	00h	A	CONFIGURATION FAILURE
67h	01h	A	CONFIGURATION OF INCAPABLE LOGICAL UNITS FAILED
67h	02h	A	ADD LOGICAL UNIT FAILED
67h	03h	A	MODIFICATION OF LOGICAL UNIT FAILED
67h	04h	A	EXCHANGE OF LOGICAL UNIT FAILED
67h	05h	A	REMOVE OF LOGICAL UNIT FAILED
67h	06h	A	ATTACHMENT OF LOGICAL UNIT FAILED
67h	07h	A	CREATION OF LOGICAL UNIT FAILED
67h	08h	A	ASSIGN FAILURE OCCURRED
67h	09h	A	MULTIPLY ASSIGNED LOGICAL UNIT

Table D.1 — ASC and ASCQ assignments (part 16 of 18)

D – Direct Access Block Device (SBC-3)
. T – Sequential Access Device (SSC-4)
. L – Printer Device (SSC)
. P – Processor Device (SPC-2)
. . W – Write Once Block Device (SBC)
. . R – C/DVD Device (MMC-6)
. . O – Optical Memory Block Device (SBC)
. . . M – Media Changer Device (SMC-3)
. . . A – Storage Array Device (SCC-2)
. . . E – SCSI Enclosure Services device (SES-3)
. . . . B – Simplified Direct-Access (Reduced Block) device (RBC)
. . . . K – Optical Card Reader/Writer device (OCRW)
. . . . V – Automation/Device Interface device (ADC-3)
. . . . . F – Object-based Storage Device (OSD-2)

Device Column key
blank = code not used
not blank = code used

ASC	ASCQ	D	T	L	P	W	R	O	M	A	E	B	K	V	F	Description
67h	0Ah	D	T	L	P	W	R	O	M	A	E	B	K	V	F	SET TARGET PORT GROUPS COMMAND FAILED
67h	0Bh	D	T									B				ATA DEVICE FEATURE NOT ENABLED
68h	00h									A						LOGICAL UNIT NOT CONFIGURED
69h	00h									A						DATA LOSS ON LOGICAL UNIT
69h	01h									A						MULTIPLE LOGICAL UNIT FAILURES
69h	02h									A						PARITY/DATA MISMATCH
6Ah	00h									A						INFORMATIONAL, REFER TO LOG
6Bh	00h									A						STATE CHANGE HAS OCCURRED
6Bh	01h									A						REDUNDANCY LEVEL GOT BETTER
6Bh	02h									A						REDUNDANCY LEVEL GOT WORSE
6Ch	00h									A						REBUILD FAILURE OCCURRED
6Dh	00h									A						RECALCULATE FAILURE OCCURRED
6Eh	00h									A						COMMAND TO LOGICAL UNIT FAILED
6Fh	00h						R									COPY PROTECTION KEY EXCHANGE FAILURE - AUTHENTICATION FAILURE
6Fh	01h						R									COPY PROTECTION KEY EXCHANGE FAILURE - KEY NOT PRESENT
6Fh	02h						R									COPY PROTECTION KEY EXCHANGE FAILURE - KEY NOT ESTABLISHED
6Fh	03h						R									READ OF SCRAMBLED SECTOR WITHOUT AUTHENTICATION
6Fh	04h						R									MEDIA REGION CODE IS MISMATCHED TO LOGICAL UNIT REGION
6Fh	05h						R									DRIVE REGION MUST BE PERMANENT/REGION RESET COUNT ERROR
6Fh	06h						R									INSUFFICIENT BLOCK COUNT FOR BINDING NONCE RECORDING
6Fh	07h						R									CONFLICT IN BINDING NONCE RECORDING
70h	NNh		T													DECOMPRESSION EXCEPTION SHORT ALGORITHM ID OF NN
71h	00h		T													DECOMPRESSION EXCEPTION LONG ALGORITHM ID
72h	00h						R									SESSION FIXATION ERROR
72h	01h						R									SESSION FIXATION ERROR WRITING LEAD-IN
72h	02h						R									SESSION FIXATION ERROR WRITING LEAD-OUT
72h	03h						R									SESSION FIXATION ERROR - INCOMPLETE TRACK IN SESSION
72h	04h						R									EMPTY OR PARTIALLY WRITTEN RESERVED TRACK
72h	05h						R									NO MORE TRACK RESERVATIONS ALLOWED
72h	06h						R									RMZ EXTENSION IS NOT ALLOWED
72h	07h						R									NO MORE TEST ZONE EXTENSIONS ARE ALLOWED
73h	00h						R									CD CONTROL ERROR
73h	01h						R									POWER CALIBRATION AREA ALMOST FULL
73h	02h						R									POWER CALIBRATION AREA IS FULL
73h	03h						R									POWER CALIBRATION AREA ERROR
73h	04h						R									PROGRAM MEMORY AREA UPDATE FAILURE
73h	05h						R									PROGRAM MEMORY AREA IS FULL
73h	06h						R									RMA/PMA IS ALMOST FULL
73h	10h						R									CURRENT POWER CALIBRATION AREA ALMOST FULL
73h	11h						R									CURRENT POWER CALIBRATION AREA IS FULL
73h	17h						R									RDZ IS FULL

Table D.1 — ASC and ASCQ assignments (part 17 of 18)

D – Direct Access Block Device (SBC-3)
. T – Sequential Access Device (SSC-4)
. L – Printer Device (SSC)
. P – Processor Device (SPC-2)
. . W – Write Once Block Device (SBC)
. . R – C/DVD Device (MMC-6)
. . O – Optical Memory Block Device (SBC)
. . . M – Media Changer Device (SMC-3)
. . . A – Storage Array Device (SCC-2)
. . . E – SCSI Enclosure Services device (SES-3)
. . . . B – Simplified Direct-Access (Reduced Block) device (RBC)
. . . . K – Optical Card Reader/Writer device (OCRW)
. . . . V – Automation/Device Interface device (ADC-3)
. . . . . F – Object-based Storage Device (OSD-2)

Device Column key
blank = code not used
not blank = code used

ASC	ASCQ	D	T	L	P	W	R	O	M	A	E	B	K	V	F	Description
74h	00h		T													SECURITY ERROR
74h	01h		T													UNABLE TO DECRYPT DATA
74h	02h		T													UNENCRYPTED DATA ENCOUNTERED WHILE DECRYPTING
74h	03h		T													INCORRECT DATA ENCRYPTION KEY
74h	04h		T													CRYPTOGRAPHIC INTEGRITY VALIDATION FAILED
74h	05h		T													ERROR DECRYPTING DATA
74h	06h		T													UNKNOWN SIGNATURE VERIFICATION KEY
74h	07h		T													ENCRYPTION PARAMETERS NOT USEABLE
74h	08h	D	T				R		M		E			V	F	DIGITAL SIGNATURE VALIDATION FAILURE
74h	09h		T													ENCRYPTION MODE MISMATCH ON READ
74h	0Ah		T													ENCRYPTED BLOCK NOT RAW READ ENABLED
74h	0Bh		T													INCORRECT ENCRYPTION PARAMETERS
74h	0Ch	D	T				R		M	A	E	B	K	V		UNABLE TO DECRYPT PARAMETER LIST
74h	0Dh		T													ENCRYPTION ALGORITHM DISABLED
74h	10h	D	T				R		M	A	E	B	K	V		SA CREATION PARAMETER VALUE INVALID
74h	11h	D	T				R		M	A	E	B	K	V		SA CREATION PARAMETER VALUE REJECTED
74h	12h	D	T				R		M	A	E	B	K	V		INVALID SA USAGE
74h	21h		T													DATA ENCRYPTION CONFIGURATION PREVENTED
74h	30h	D	T				R		M	A	E	B	K	V		SA CREATION PARAMETER NOT SUPPORTED
74h	40h	D	T				R		M	A	E	B	K	V		AUTHENTICATION FAILED
74h	61h													V		EXTERNAL DATA ENCRYPTION KEY MANAGER ACCESS ERROR
74h	62h													V		EXTERNAL DATA ENCRYPTION KEY MANAGER ERROR
74h	63h													V		EXTERNAL DATA ENCRYPTION KEY NOT FOUND
74h	64h													V		EXTERNAL DATA ENCRYPTION REQUEST NOT AUTHORIZED
74h	6Eh		T													EXTERNAL DATA ENCRYPTION CONTROL TIMEOUT
74h	6Fh		T													EXTERNAL DATA ENCRYPTION CONTROL ERROR
74h	71h	D	T				R		M		E			V		LOGICAL UNIT ACCESS NOT AUTHORIZED
74h	79h	D														SECURITY CONFLICT IN TRANSLATED DEVICE
75h	00h															
76h	00h															
77h	00h															
78h	00h															
79h	00h															
7Ah	00h															
7Bh	00h															
7Ch	00h															
7Dh	00h															
7Eh	00h															

Table D.1 — ASC and ASCQ assignments (part 18 of 18)

```
                D – Direct Access Block Device (SBC-3)                                  Device Column key
                . T – Sequential Access Device (SSC-4)                                  blank = code not used
                .   L – Printer Device (SSC)                                            not blank = code used
                .     P – Processor Device (SPC-2)
                .     . W – Write Once Block Device (SBC)
                .     .   R – C/DVD Device (MMC-6)
                .     .     O – Optical Memory Block Device (SBC)
                .     .     . M – Media Changer Device (SMC-3)
                .     .     .   A – Storage Array Device (SCC-2)
                .     .     .     E – SCSI Enclosure Services device (SES-3)
                .     .     .     . B – Simplified Direct-Access (Reduced Block) device (RBC)
                .     .     .     .   K – Optical Card Reader/Writer device (OCRW)
                .     .     .     .     V – Automation/Device Interface device (ADC-3)
                .     .     .     .     . F – Object-based Storage Device (OSD-2)
                .     .     .     .     .
ASC  ASCQ   D T L P W R O M A E B K V F   Description
7Fh  00h

80h  xxh                \
  Through                 >               Vendor specific
FFh  xxh                /

xxh  80h                \
  Through                 >               Vendor specific qualification of standard ASC
xxh  FFh                /
                                          All codes not shown are reserved.
```

D.3 Operation codes

D.3.1 Operation codes

Table D.2 is a numerical order listing of the command operation codes.

Table D.2 — Operation codes (part 1 of 6)

```
        D – Direct Access Block Device (SBC-3)                          Device Column key
        . T – Sequential Access Device (SSC-4)                          M = Mandatory
        .   L – Printer Device (SSC)                                    O = Optional
        .     P – Processor Device (SPC-2)                              V = Vendor specific
        .     . W – Write Once Block Device (SBC)                       Z = Obsolete
        .     .   R – C/DVD Device (MMC-6)
        .     .     O – Optical Memory Block Device (SBC)
        .     .     . M – Media Changer Device (SMC-3)
        .     .     .   A – Storage Array Device (SCC-2)
        .     .     .     E – SCSI Enclosure Services device (SES-3)
        .     .     .     . B – Simplified Direct-Access (Reduced Block) device (RBC)
        .     .     .     .   K – Optical Card Reader/Writer device (OCRW)
        .     .     .     .     V – Automation/Device Interface device (ADC-3)
        .     .     .     .     . F – Object-based Storage Device (OSD-2)
OP      D T L PWROMA E B K V F  Description
00h     MMMMMMMMMMMMMM          TEST UNIT READY
01h       M                     REWIND
01h     Z   V   Z Z Z Z         REZERO UNIT
02h     V V V V V V   V
03h     MMMMMMMMMMOMMM          REQUEST SENSE
04h     M         OO            FORMAT UNIT
04h       O                     FORMAT MEDIUM
04h         O                   FORMAT
05h     VMV V V V   V           READ BLOCK LIMITS
06h     V V V V V V   V
07h     OVV   O   OV            REASSIGN BLOCKS
07h                 O           INITIALIZE ELEMENT STATUS
08h     MMV   O   OV            READ(6)
08h         O                   RECEIVE
08h                             GET MESSAGE(6)
09h     V V V V V V   V
0Ah     OO    O   OV            WRITE(6)
0Ah         M                   SEND(6)
0Ah                             SEND MESSAGE(6)
0Ah         M                   PRINT
0Bh     Z       ZOZV            SEEK(6)
0Bh       O                     SET CAPACITY
0Bh         O                   SLEW AND PRINT
0Ch     V V V V V V   V
0Dh     V V V V V V   V
0Eh     V V V V V V   V
0Fh     VOV V V V   V           READ REVERSE(6)
10h     VM  V V V               WRITE FILEMARKS(6)
10h         O                   SYNCHRONIZE BUFFER
11h     VMV V V V               SPACE(6)
12h     MMMMMMMMMMMMMM          INQUIRY
13h     V   V V V V
13h       O                     VERIFY(6)
14h     VOOV V V                RECOVER BUFFERED DATA
15h     OMO   O   OOOO   OO     MODE SELECT(6)
```

Table D.2 — Operation codes (part 2 of 6)

```
D – Direct Access Block Device (SBC-3)
. T – Sequential Access Device (SSC-4)
.   L – Printer Device (SSC)
.     P – Processor Device (SPC-2)
.     . W – Write Once Block Device (SBC)
.     .   R – C/DVD Device (MMC-6)
.     .     O – Optical Memory Block Device (SBC)
.     .     . M – Media Changer Device (SMC-3)
.     .     .   A – Storage Array Device (SCC-2)
.     .     .     E – SCSI Enclosure Services device (SES-3)
.     .     .     . B – Simplified Direct-Access (Reduced Block) device (RBC)
.     .     .     .   K – Optical Card Reader/Writer device (OCRW)
.     .     .     .     V – Automation/Device Interface device (ADC-3)
.     .     .     .     . F – Object-based Storage Device (OSD-2)
```

Device Column key
M = Mandatory
O = Optional
V = Vendor specific
Z = Obsolete

OP	D T L P W R O M A E B K V F	Description
16h	Z Z M Z O O O O Z O	RESERVE(6)
16h	Z	RESERVE ELEMENT(6)
17h	Z Z M Z O O O O Z O	RELEASE(6)
17h	Z	RELEASE ELEMENT(6)
18h	Z Z Z Z O Z O Z	COPY
19h	V M V V V V	ERASE(6)
1Ah	O M O O O O O O O O	MODE SENSE(6)
1Bh	O O O O O M O O	START STOP UNIT
1Bh	O M	LOAD UNLOAD
1Bh		SCAN
1Bh	O	STOP PRINT
1Bh	O	OPEN/CLOSE IMPORT/EXPORT ELEMENT
1Ch	O O O O O O O O M O O O	RECEIVE DIAGNOSTIC RESULTS
1Dh	M M M M M M M O M M M M	SEND DIAGNOSTIC
1Eh	O O O O O O O O	PREVENT ALLOW MEDIUM REMOVAL
1Fh		
20h	V V V V V	
21h	V V V V V	
22h	V V V V V	
23h	V V V V	
23h	O	READ FORMAT CAPACITIES
24h	V V V	SET WINDOW
25h	M M M M	READ CAPACITY(10)
25h	O	READ CAPACITY
25h	M	READ CARD CAPACITY
25h		GET WINDOW
26h	V V V	
27h	V V V	
28h	M M O M M M	READ(10)
28h		GET MESSAGE(10)
29h	V V V O	READ GENERATION
2Ah	O M O M M O	WRITE(10)
2Ah		SEND(10)
2Ah		SEND MESSAGE(10)
2Bh	Z O O O O	SEEK(10)
2Bh	M	LOCATE(10)
2Bh	O	POSITION TO ELEMENT
2Ch	V O O	ERASE(10)
2Dh	O	READ UPDATED BLOCK
2Dh	V	
2Eh	O O O O M O	WRITE AND VERIFY(10)
2Fh	O O O O	VERIFY(10)

Table D.2 — Operation codes (part 3 of 6)

D – Direct Access Block Device (SBC-3)
. T – Sequential Access Device (SSC-4)
. L – Printer Device (SSC)
. P – Processor Device (SPC-2)
. . W – Write Once Block Device (SBC)
. . R – C/DVD Device (MMC-6)
. . O – Optical Memory Block Device (SBC)
. . . M – Media Changer Device (SMC-3)
. . . A – Storage Array Device (SCC-2)
. . . E – SCSI Enclosure Services device (SES-3)
. . . . B – Simplified Direct-Access (Reduced Block) device (RBC)
. . . . K – Optical Card Reader/Writer device (OCRW)
. . . . V – Automation/Device Interface device (ADC-3)
. . . . . F – Object-based Storage Device (OSD-2)

Device Column key
M = Mandatory
O = Optional
V = Vendor specific
Z = Obsolete

OP	D	T	L	P	W	R	O	M	A	E	B	K	V	F	Description
30h	Z				Z	Z	Z								SEARCH DATA HIGH(10)
31h	Z				Z	Z	Z								SEARCH DATA EQUAL(10)
31h															OBJECT POSITION
32h	Z				Z	Z	Z								SEARCH DATA LOW(10)
33h	Z				O	Z	O								SET LIMITS(10)
34h	O				O		O					O			PRE-FETCH(10)
34h		M													READ POSITION
34h															GET DATA BUFFER STATUS
35h	O				O	O	O				M	O			SYNCHRONIZE CACHE(10)
36h	Z				O		O					O			LOCK UNLOCK CACHE(10)
37h	O						O								READ DEFECT DATA(10)
37h								O							INITIALIZE ELEMENT STATUS WITH RANGE
38h					O		O					O			MEDIUM SCAN
39h	Z	Z	Z	Z	O	Z	O					Z			COMPARE
3Ah	Z	Z	Z	Z	O	Z	O					Z			COPY AND VERIFY
3Bh	O	O	O	O	O	O	O	O	O	O	M	O	O	O	WRITE BUFFER
3Ch	O	O	O	O	O	O	O	O	O	O		O	O	O	READ BUFFER
3Dh							O								UPDATE BLOCK
3Eh	O				O		O								READ LONG(10)
3Fh	O				O		O								WRITE LONG(10)
40h	Z	Z	Z	Z	O	Z	O	Z							CHANGE DEFINITION
41h	O														WRITE SAME(10)
42h	O														UNMAP
42h						O									READ SUB-CHANNEL
43h						O									READ TOC/PMA/ATIP
44h		M											M		REPORT DENSITY SUPPORT
44h															READ HEADER
45h						O									PLAY AUDIO(10)
46h						M									GET CONFIGURATION
47h						O									PLAY AUDIO MSF
48h	O										O				SANITIZE
49h															
4Ah						M									GET EVENT STATUS NOTIFICATION
4Bh						O									PAUSE/RESUME
4Ch	O	O	O	O	O		O	O	O	O		O	O	O	LOG SELECT
4Dh	O	O	O	O	O		O	O	O	O		O	M	O	LOG SENSE
4Eh						O									STOP PLAY/SCAN
4Fh															
50h	O														XDWRITE(10)
51h	O														XPWRITE(10)
51h						O									READ DISC INFORMATION
52h	O														XDREAD(10)

Table D.2 — Operation codes (part 4 of 6)

```
D – Direct Access Block Device (SBC-3)
. T – Sequential Access Device (SSC-4)
.   L – Printer Device (SSC)
.     P – Processor Device (SPC-2)
.     . W – Write Once Block Device (SBC)
.     .   R – C/DVD Device (MMC-6)
.     .     O – Optical Memory Block Device (SBC)
.     .     . M – Media Changer Device (SMC-3)
.     .     .   A – Storage Array Device (SCC-2)
.     .     .     E – SCSI Enclosure Services device (SES-3)
.     .     .     . B – Simplified Direct-Access (Reduced Block) device (RBC)
.     .     .     .   K – Optical Card Reader/Writer device (OCRW)
.     .     .     .     V – Automation/Device Interface device (ADC-3)
.     .     .     .     . F – Object-based Storage Device (OSD-2)
```

Device Column key
M = Mandatory
O = Optional
V = Vendor specific
Z = Obsolete

OP	D T L P W R O M A E B K V F	Description
52h	O	READ TRACK INFORMATION
53h	O	XDWRITEREAD(10)
53h	O	RESERVE TRACK
54h	O	SEND OPC INFORMATION
55h	O O O O M O O O O O M O M O	MODE SELECT(10)
56h	Z Z M Z O O O O Z	RESERVE(10)
56h	Z	RESERVE ELEMENT(10)
57h	Z Z M Z O O O O Z	RELEASE(10)
57h	Z	RELEASE ELEMENT(10)
58h	O	REPAIR TRACK
59h		
5Ah	O O O O M O O O O O M O M O	MODE SENSE(10)
5Bh	O	CLOSE TRACK/SESSION
5Ch	O	READ BUFFER CAPACITY
5Dh	O	SEND CUE SHEET
5Eh	O M O O O O O O O M	PERSISTENT RESERVE IN
5Fh	O M O O O O O O O M	PERSISTENT RESERVE OUT
7Eh	O O O O O O O O O	extended CDB
7Fh	O M	variable length CDB (more than 16 bytes)
80h	Z	XDWRITE EXTENDED(16)
80h	M	WRITE FILEMARKS(16)
81h	Z	REBUILD(16)
81h	O	READ REVERSE(16)
82h	Z	REGENERATE(16)
82h	O	ALLOW OVERWRITE
83h	O O O O O O O O O	EXTENDED COPY
84h	O O O O O O O O O	RECEIVE COPY RESULTS
85h	O O O	ATA PASS THROUGH(16)
86h	O O O O O O O O O O	ACCESS CONTROL IN
87h	O O O O O O O O O O	ACCESS CONTROL OUT
88h	M O O O O	READ(16)
89h	O	COMPARE AND WRITE
8Ah	O O O O O	WRITE(16)
8Bh	O	ORWRITE
8Ch	O O O O O O M	READ ATTRIBUTE
8Dh	O O O O O O O	WRITE ATTRIBUTE
8Eh	O O O O	WRITE AND VERIFY(16)
8Fh	O O O O O	VERIFY(16)
90h	O O O O	PRE-FETCH(16)
91h	O O O O	SYNCHRONIZE CACHE(16)
91h	O	SPACE(16)
92h	Z O O	LOCK UNLOCK CACHE(16)

Table D.2 — Operation codes (part 5 of 6)

```
D – Direct Access Block Device (SBC-3)
. T – Sequential Access Device (SSC-4)
.   L – Printer Device (SSC)
.     P – Processor Device (SPC-2)
.     . W – Write Once Block Device (SBC)
.     .   R – C/DVD Device (MMC-6)
.     .     O – Optical Memory Block Device (SBC)
.     .     . M – Media Changer Device (SMC-3)
.     .     .   A – Storage Array Device (SCC-2)
.     .     .     E – SCSI Enclosure Services device (SES-3)
.     .     .     . B – Simplified Direct-Access (Reduced Block) device (RBC)
.     .     .     .   K – Optical Card Reader/Writer device (OCRW)
.     .     .     .     V – Automation/Device Interface device (ADC-3)
.     .     .     .     . F – Object-based Storage Device (OSD-2)
```

Device Column key
M = Mandatory
O = Optional
V = Vendor specific
Z = Obsolete

OP	D T L P W R O M A E B K V F	Description
92h	M	LOCATE(16)
93h	O	WRITE SAME(16)
93h	M	ERASE(16)
94h		[usage proposed by SCSI Socket Services project]
95h		[usage proposed by SCSI Socket Services project]
96h		[usage proposed by SCSI Socket Services project]
97h		[usage proposed by SCSI Socket Services project]
98h		
99h		
9Ah		
9Bh		
9Ch		
9Dh		
9Eh		SERVICE ACTION IN(16)
9Fh	M	SERVICE ACTION OUT(16)
A0h	MMOOO OMMM OMO	REPORT LUNS
A1h	O	BLANK
A1h	O O	ATA PASS THROUGH(12)
A2h	OO O O	SECURITY PROTOCOL IN
A3h	OOO O OOMOOOM	MAINTENANCE (IN)
A3h	O	SEND KEY
A4h	OOO O OOOOOOO	MAINTENANCE (OUT)
A4h	O	REPORT KEY
A5h	Z O OM	MOVE MEDIUM
A5h	O	PLAY AUDIO(12)
A6h	O	EXCHANGE MEDIUM
A6h	O	LOAD/UNLOAD C/DVD
A7h	Z Z O O	MOVE MEDIUM ATTACHED
A7h	O	SET READ AHEAD
A8h	O OOO	READ(12)
A8h		GET MESSAGE(12)
A9h	O	SERVICE ACTION OUT(12)
AAh	O OOO	WRITE(12)
AAh		SEND MESSAGE(12)
ABh	O O	SERVICE ACTION IN(12)
ACh	O	ERASE(12)
ACh	O	GET PERFORMANCE
ADh	O	READ DVD STRUCTURE
AEh	O O O	WRITE AND VERIFY(12)
AFh	O O O	VERIFY(12)
B0h	Z Z Z	SEARCH DATA HIGH(12)
B1h	Z Z Z	SEARCH DATA EQUAL(12)

Table D.2 — Operation codes (part 6 of 6)

D – Direct Access Block Device (SBC-3)
. T – Sequential Access Device (SSC-4)
. L – Printer Device (SSC)
. P – Processor Device (SPC-2)
. . W – Write Once Block Device (SBC)
. . R – C/DVD Device (MMC-6)
. . O – Optical Memory Block Device (SBC)
. . . M – Media Changer Device (SMC-3)
. . . . A – Storage Array Device (SCC-2)
. . . . E – SCSI Enclosure Services device (SES-3)
. . . . . B – Simplified Direct-Access (Reduced Block) device (RBC)
. . . . . K – Optical Card Reader/Writer device (OCRW)
. . . . . V – Automation/Device Interface device (ADC-3)
. . . . . F – Object-based Storage Device (OSD-2)

Device Column key
M = Mandatory
O = Optional
V = Vendor specific
Z = Obsolete

OP	D	T	L	P	W	R	O	M	A	E	B	K	V	F	Description
B2h					Z	Z	Z								SEARCH DATA LOW(12)
B3h	Z				O	Z	O								SET LIMITS(12)
B4h	Z	Z			O	Z	O								READ ELEMENT STATUS ATTACHED
B5h	O	O				O								O	SECURITY PROTOCOL OUT
B5h									O						REQUEST VOLUME ELEMENT ADDRESS
B6h									O						SEND VOLUME TAG
B6h						O									SET STREAMING
B7h	O						O								READ DEFECT DATA(12)
B8h		Z			O	Z	O	M							READ ELEMENT STATUS
B9h						O									READ CD MSF
BAh	O				O		O	O	M	O					REDUNDANCY GROUP (IN)
BAh						O									SCAN
BBh	O				O		O	O	O	O					REDUNDANCY GROUP (OUT)
BBh						O									SET CD SPEED
BCh	O				O		O	O	M	O					SPARE (IN)
BDh	O				O		O	O	O	O					SPARE (OUT)
BDh						O									MECHANISM STATUS
BEh	O				O		O	O	M	O					VOLUME SET (IN)
BEh						O									READ CD
BFh	O				O		O	O	O	O					VOLUME SET (OUT)
BFh						O									SEND DVD STRUCTURE

D.3.2 Additional operation codes for devices with the EncServ bit set to one

Table D.3 is a numerical order listing of the additional command operation codes used by devices that have the EncServ bit set to one in their standard INQUIRY data. The operation codes listed in table D.3 are in addition to the operation codes listed in D.3.1 for the device type indicated by the standard INQUIRY data having the EncServ bit set to one.

Table D.3 — Additional operation codes for devices with the EncServ bit set to one

	D – Direct Access Block Device (SBC-3)	Device Column key
	. T – Sequential Access Device (SSC-4)	M = Mandatory
	. L – Printer Device (SSC)	O = Optional
	. P – Processor Device (SPC-2)	V = Vendor specific
	. . W – Write Once Block Device (SBC)	Z = Obsolete
	. . R – C/DVD Device (MMC-6)	
	. . O – Optical Memory Block Device (SBC)	
	. . . M – Media Changer Device (SMC-3)	
	. . . A – Storage Array Device (SCC-2)	
	. . . E – SCSI Enclosure Services device (SES-3)	
	. . . . B – Simplified Direct-Access (Reduced Block) device (RBC)	
	. . . . K – Optical Card Reader/Writer device (OCRW)	
	. . . . V – Automation/Device Interface device (ADC-3)	
	. . . . . F – Object-based Storage Device (OSD-2)	
OP	D T L P W R O M A E B K V F	Description
1C	M M M M M M M M M M M M M	RECEIVE DIAGNOSTIC RESULTS
1D	M M M M M M M M M M M M M	SEND DIAGNOSTIC

D.3.3 MAINTENANCE (IN) and MAINTENANCE (OUT) service actions

The assignment of service action codes for the MAINTENANCE (IN) and MAINTENANCE (OUT) operation codes by this standard is shown in table D.4. The MAINTENANCE (IN) and MAINTENANCE (OUT) service actions that may be assigned by other command standards are noted as restricted but their specific usage is not described.

Table D.4 — MAINTENANCE (IN) and MAINTENANCE (OUT) service actions

Service Action	Description
MAINTENANCE (IN) [operation code A3h]	
00h to 04h	Restricted (see SCC-2)
05h	REPORT IDENTIFYING INFORMATION
06h to 09h	Restricted (see SCC-2)
0Ah	REPORT TARGET PORT GROUPS
0Bh	REPORT ALIASES
0Ch	REPORT SUPPORTED OPERATION CODES
0Dh	REPORT SUPPORTED TASK MANAGEMENT FUNCTIONS
0Eh	REPORT PRIORITY
0Fh	REPORT TIMESTAMP
10h	MANAGEMENT PROTOCOL IN
11h to 1Dh	Reserved
1Eh	Restricted (see ADC-3)
1Fh	Vendor specific
MAINTENANCE (OUT) [operation code A4h]	
00h to 05h	Restricted (see SCC-2)
06h	SET IDENTIFYING INFORMATION
07h to 09h	Restricted (see SCC-2)
0Ah	SET TARGET PORT GROUPS
0Bh	CHANGE ALIASES
0Ch to 0Dh	Reserved
0Eh	SET PRIORITY
0Fh	SET TIMESTAMP
10h	MANAGEMENT PROTOCOL OUT
11h to 1Dh	Reserved
1Eh	Restricted (see ADC-3, and SMC-3)
1Fh	Vendor specific

D.3.4 SERVICE ACTION IN and SERVICE ACTION OUT service actions

The assignment of service action codes for the SERVICE ACTION IN(12) and SERVICE ACTION OUT(12) operation codes by this standard is shown in table D.5. The SERVICE ACTION IN(12) and SERVICE ACTION OUT(12) service actions that may be assigned by other command standards are noted as restricted but their specific usage is not described.

Table D.5 — SERVICE ACTION IN(12) and SERVICE ACTION OUT(12) service actions

Service Action	Description
SERVICE ACTION IN(12) [operation code ABh]	
00h	Reserved
01h	READ MEDIA SERIAL NUMBER
02h to 1Fh	Reserved
SERVICE ACTION OUT(12) [operation code A9h]	
00h to 1Eh	Reserved
1Fh	Restricted

The assignment of service action codes for the SERVICE ACTION IN(16) and SERVICE ACTION OUT(16) operation codes by this standard is shown in table D.6. The SERVICE ACTION IN(16) and SERVICE ACTION OUT(16) service actions that may be assigned by other command standards are noted as restricted but their specific usage is not described.

Table D.6 — SERVICE ACTION IN(16) and SERVICE ACTION OUT(16) service actions

Service Action	Description
SERVICE ACTION IN(16) [operation code 9Eh]	
00h to 0Fh	Reserved
10h to 1Fh	Restricted
SERVICE ACTION OUT(16) [operation code 9Fh]	
00h to 0Fh	Reserved
10h to 1Fh	Restricted

D.3.5 Variable length CDB service action codes

Only one operation code is assigned to the variable length CDB (see 4.3.3). Therefore, the service action code is effectively the operation code for variable length CDB uses. To allow command standards to assign uses of the variable length CDB without consulting this standard, ranges of service action codes are assigned to command sets as shown in table D.7.

Table D.7 — Variable Length CDB Service Action Code Ranges

Service Action Code Range	Doc.	Description
0000h to 07FFh	SBC-3	Direct access block device (e.g., magnetic disk)
0800h to 0FFFh	SSC-4	Sequential-access device (e.g., magnetic tape)
1000h to 17FFh	SSC	Printer device
1800h to 1FFFh	this standard	Commands for all device types (see table D.8)
2000h to 27FFh		Reserved
2800h to 2FFFh	MMC-6	CD-ROM device
3800h to 3FFFh		Reserved
4000h to 47FFh	SMC-3	Media changer device (e.g., jukeboxes)
5000h to 5FFFh		Defined by ASC IT8 (Graphic arts pre-press devices)
6000h to 67FFh	SCC-2	Storage array controller device (e.g., RAID)
7000h to 77FFh	RBC	Simplified direct-access device (e.g., magnetic disk)
7800h to 7FFFh	OCRW	Optical card reader/writer device
8800h to 8FFFh	OSD-2	Object-based Storage Device
3000h to 37FFh		Reserved
4800h to 4FFFh		Reserved
6800h to 6FFFh		Reserved
8000h to 87FFh		Reserved
9000h to F7FFh		Reserved
F800h to FFFFh		Vendor specific

The variable length CDB service action codes assigned by this standard are shown in table D.8.

Table D.8 — Variable Length CDB Service Action Codes Used by All Device Types

Service Action Code	Description
1800h	RECEIVE CREDENTIAL command
1801h to 1FF7h	Reserved
1FF8h to 1FFFh	Restricted (see FC-SB-4)

D.4 Diagnostic page codes

Table D.9 is a numerical order listing of the diagnostic page codes.

Table D.9 — Diagnostic page codes

D – Direct Access Block Device (SBC-3)
. T – Sequential Access Device (SSC-4)
. L – Printer Device (SSC)
. P – Processor Device (SPC-2)
. . W – Write Once Block Device (SBC)
. . R – C/DVD Device (MMC-6)
. . O – Optical Memory Block Device (SBC)
. . . M – Media Changer Device (SMC-3)
. . . A – Storage Array Device (SCC-2)
. . . E – SCSI Enclosure Services device (SES-3)
. . . . B – Simplified Direct-Access (Reduced Block) device (RBC)
. . . . K – Optical Card Reader/Writer device (OCRW)
. . . . V – Automation/Device Interface device (ADC-3)
. . . . . F – Object-based Storage Device (OSD-2)

Device Column key
blank = code not used
not blank = code used

Diagnostic Page Code	D	T	L	P	W	R	O	M	A	E	B	K	V	F	Diagnostic Page Name
00h	D	T	L	P	W	R	O	M	A	E		K	V	F	Supported Diagnostic Pages
01h										E					Configuration
02h										E					Enclosure Status/Control
03h										E					Help Text
04h										E					String In/Out
05h										E					Threshold In/Out
06h										E					Obsolete
07h										E					Element Descriptor
08h										E					Short Enclosure Status
09h										E					Enclosure Busy
0Ah										E					Additional Element Status
0Bh										E					Subenclosure Help Text
0Ch										E					Subenclosure String In/Out
0Dh										E					Supported SES Diagnostic Pages
0Eh										E					Download Microcode Status/Control
0Fh										E					Subenclosure Nickname Status/Control
10h to 1Fh															SES vendor specific
3Fh															Protocol Specific diagnostic page
40h	D				W		O								Translate Address In/Out
41h	D				W		O								Device Status In/Out
80h to FFh															Vendor specific

All codes not shown are restricted or reserved.

D.5 Log page codes

Table D.10 is a numerical order listing of the log page codes.

Table D.10 — Log page codes (part 1 of 2)

Device Column key
blank = code not used
not blank = code used

Log Page Code	Log Subpage Code	D .T .L .P .W .R .O .M .A .E .B .K .V .F	Log Page Name
00h	00h	DTLPWROMAE KVF	Supported Log Pages
00h	FFh	DTLPWROMAE KVF	Supported Log Pages and Subpages
01h to 3Fh	FFh	DTLPWROMAE KVF	Supported Subpages
01h	00h	DTLPWRO A KVF	Buffer Over-Run/Under-Run
02h	00h	DT WRO KV	Write Error Counter
03h	00h	DT WRO KV	Read Error Counter
04h	00h	T V	Read Reverse Error Counter
05h	00h	DT WRO KV	Verify Error Counter
06h	00h	DTLPWROMAE KVF	Non-Medium Error
07h	00h	DTLPWROMAE KVF	Last n Error Events
08h	00h	D W O V	Format Status
09h	00h to FEh	O	Reserved to the MS59 Std. (contact AIIM C21 comm.)
0Ah	00h to FEh	O	Reserved to the MS59 Std. (contact AIIM C21 comm.)
0Bh	00h	DTLPWROMAE VF	Last n Deferred Error or Asynchronous Events
0Ch	00h	D	Logical Block Provisioning
0Ch	00h	T V	Sequential-Access Device
0Dh	00h	DTLPWROMAE V	Temperature
0Eh	00h	DTLPWROMAE V	Start-Stop Cycle Counter
0Fh	00h	DTLPWROMAE V	Application Client
10h	00h	DTLPWROMAE V	Self-Test Results
11h	00h	D	Solid State Media
11h	00h	T V	DT Device Status
12h	00h	V	TapeAlert Response
13h	00h	V	Requested Recovery
14h	00h	T V	Device Statistics
15h	00h	D	Background Scan Results
15h	00h	V	Service Buffers Information
16h	00h	D	ATA PASS-THROUGH Results
16h	00h	T V	Tape Diagnostic Data
17h	00h	D	Non-Volatile Cache
17h	00h	T	Volume Statistics
18h	xxh	DTLPWROMAE KV	Protocol Specific Port (see table D.11)
19h	00h	D	General Statistics and Performance
19h	01h	D	Group Statistics and Performance (1)
19h	02h	D	Group Statistics and Performance (2)
19h	03h	D	Group Statistics and Performance (3)

Table D.10 — Log page codes (part 2 of 2)

D – Direct Access Block Device (SBC-3)
. T – Sequential Access Device (SSC-4)
. L – Printer Device (SSC)
. P – Processor Device (SPC-2)
. . W – Write Once Block Device (SBC)
. . R – C/DVD Device (MMC-6)
. . O – Optical Memory Block Device (SBC)
. . . M – Media Changer Device (SMC-3)
. . . A – Storage Array Device (SCC-2)
. . . E – SCSI Enclosure Services device (SES-3)
. . . . B – Simplified Direct-Access (Reduced Block) device (RBC)
. . . . K – Optical Card Reader/Writer device (OCRW)
. . . . V – Automation/Device Interface device (ADC-3)
. . . . . F – Object-based Storage Device (OSD-2)

Device Column key
blank = code not used
not blank = code used

Log Page Code	Log Subpage Code	D T L P W R O M A E B K V F	Log Page Name
19h	04h	D	Group Statistics and Performance (4)
19h	05h	D	Group Statistics and Performance (5)
19h	06h	D	Group Statistics and Performance (6)
19h	07h	D	Group Statistics and Performance (7)
19h	08h	D	Group Statistics and Performance (8)
19h	09h	D	Group Statistics and Performance (9)
19h	0Ah	D	Group Statistics and Performance (10)
19h	0Bh	D	Group Statistics and Performance (11)
19h	0Ch	D	Group Statistics and Performance (12)
19h	0Dh	D	Group Statistics and Performance (13)
19h	0Eh	D	Group Statistics and Performance (14)
19h	0Fh	D	Group Statistics and Performance (15)
19h	10h	D	Group Statistics and Performance (16)
19h	11h	D	Group Statistics and Performance (17)
19h	12h	D	Group Statistics and Performance (18)
19h	13h	D	Group Statistics and Performance (19)
19h	14h	D	Group Statistics and Performance (20)
19h	15h	D	Group Statistics and Performance (21)
19h	16h	D	Group Statistics and Performance (22)
19h	17h	D	Group Statistics and Performance (23)
19h	18h	D	Group Statistics and Performance (24)
19h	19h	D	Group Statistics and Performance (25)
19h	1Ah	D	Group Statistics and Performance (26)
19h	1Bh	D	Group Statistics and Performance (27)
19h	1Ch	D	Group Statistics and Performance (28)
19h	1Dh	D	Group Statistics and Performance (29)
19h	1Eh	D	Group Statistics and Performance (30)
19h	1Fh	D	Group Statistics and Performance (31)
1Ah	00h	D T L P W R O A K	Power Condition Transitions
1Bh	00h	T	Data Compression
2Dh	00h	T	Current Service Information
2Eh	00h	T M	TapeAlert
2Fh	00h	D T L P W R O M A E K V	Informational Exceptions
30h to 3Eh	00h to FEh		Vendor specific
3Fh	00h to FEh		Reserved
			All codes not shown here or in table D.11 are restricted or reserved.

Table D.11 is a numerical order listing of the log page codes used by SCSI transport protocols.

Table D.11 — Transport protocol specific log page codes

F – Fibre Channel Protocol for SCSI (FCP-4)
. S – SAS Protocol Layer (SPL-2)
. V – Automation/Device Interface device (ADT-2)
. U – USB Attached SCSI (UAS-2)

Device Column key
blank = code not used
not blank = code used

Log Page Code	Log Subpage Code	F	S	V	U	Log Page Name
18h	00h		S			Protocol Specific Port
						See table D.10 for information on the codes not shown here.

D.6 Mode page codes

Table D.12 is a numerical order listing of the mode page codes.

Table D.12 — Mode page codes (part 1 of 2)

D – Direct Access Block Device (SBC-3)
. T – Sequential Access Device (SSC-4)
. L – Printer Device (SSC)
. P – Processor Device (SPC-2)
. . W – Write Once Block Device (SBC)
. . R – C/DVD Device (MMC-6)
. . O – Optical Memory Block Device (SBC)
. . . M – Media Changer Device (SMC-3)
. . . A – Storage Array Device (SCC-2)
. . . E – SCSI Enclosure Services device (SES-3)
. . . . B – Simplified Direct-Access (Reduced Block) device (RBC)
. . . . K – Optical Card Reader/Writer device (OCRW)
. . . . V – Automation/Device Interface device (ADC-3)
. . . . . F – Object-based Storage Device (OSD-2)

Device Column key
blank = code not used
not blank = code used

Mode Page Code	Mode Subpage Code	D	T	L	P	W	R	O	M	A	E	B	K	V	F	Mode Page Name
01h	00h	D	T			W	R	O					K			Read-Write Error Recovery
02h	00h	D	T	L		W	R	O	M	A	E		K	V	F	Disconnect-Reconnect
03h	00h	D														Format Device
03h	00h			L												Parallel Printer Interface
03h	00h						R									MRW CD-RW
04h	00h	D														Rigid Disk Geometry
04h	00h			L												Serial Printer Interface
05h	00h	D														Flexible Disk
05h	00h			L												Printer Options
05h	00h						R									Write Parameters
06h	00h					W		O								Optical Memory
06h	00h											B				RBC Device Parameters
07h	00h	D				W		O					K			Verify Error Recovery
08h	00h	D				W	R	O					K			Caching
09h	00h	D	T	L		W	R	O		A	E		K			obsolete
0Ah	00h	D	T	L		W	R	O	M	A	E		K	V	F	Control
0Ah	01h	D	T	L		W	R	O	M	A	E		K	V	F	Control Extension
0Ah	02h	D														Application Tag
0Ah	F0h		T													Control Data Protection
0Ah	F1h	D														Parallel ATA Control
0Bh	00h	D				W		O					K			Medium Types Supported
0Ch	00h	D														Notch and Partition
0Dh	00h	D														obsolete
0Dh	00h						R									CD Device Parameters
0Eh	00h						R									CD Audio Control
0Eh	01h													V		Target Device
0Eh	02h													V		DT Device Primary Port
0Eh	03h													V		Logical Unit
0Eh	04h													V		Target Device Serial Number
0Fh	00h		T													Data Compression
10h	00h	D														XOR Control
10h	00h		T													Device Configuration
11h	00h		T													Medium Partition (1)
12h																
13h																
14h	00h	D	T		P	W	R	O	M	A	E	B	K	V	F	Enclosure Services Management

Table D.12 — Mode page codes (part 2 of 2)

D – Direct Access Block Device (SBC-3)
. T – Sequential Access Device (SSC-4)
. L – Printer Device (SSC)
. P – Processor Device (SPC-2)
. . W – Write Once Block Device (SBC)
. . R – C/DVD Device (MMC-6)
. . O – Optical Memory Block Device (SBC)
. . . M – Media Changer Device (SMC-3)
. . . A – Storage Array Device (SCC-2)
. . . E – SCSI Enclosure Services device (SES-3)
. . . . B – Simplified Direct-Access (Reduced Block) device (RBC)
. . . . K – Optical Card Reader/Writer device (OCRW)
. . . . V – Automation/Device Interface device (ADC-3)
. . . . . F – Object-based Storage Device (OSD-2)

Device Column key
blank = code not used
not blank = code used

Mode Page Code	Mode Subpage Code	D	T	L	P	W	R	O	M	A	E	B	K	V	F	Mode Page Name
15h																Extended
16h																Extended Device-Type Specific
17h																
18h	xxh	D	T	L		W	R	O	M	A	E			V	F	Protocol Specific Logical Unit (see table D.13)
19h	xxh	D	T	L		W	R	O	M	A	E			V	F	Protocol Specific Port (see table D.13)
1Ah	00h	D	T	L		W	R	O	M	A				V		Power Condition
1Ah	F1h	D										B				ATA Power Condition
1Bh	00h									A						LUN Mapping
1Ch	00h	D								A	E					Informational Exceptions Control
1Ch	00h		T						M					V		Informational Exceptions Control (tapes-specific format)
1Ch	00h						R									Fault/Failure Reporting
1Ch	00h			L												Informational Exceptions Control (see SSC)
1Ch	00h					W		O								Informational Exceptions Control (see SBC)
1Ch	01h	D														Background Control
1Ch	02h	D														Logical Block Provisioning
1Dh	00h		T													Medium Configuration
1Dh	00h						R									C/DVD Time-Out and Protect
1Dh	00h								M							Element Address Assignments
1Eh	00h								M							Transport Geometry Parameters
1Fh	00h								M							Device Capabilities
00h																Vendor specific (does not require page format)
20h to 29h																Device-type specific (vendor specific in common usage)
2Ah		D	T	L		W		O	M	A	E	B	K	V	F	Device-type specific (vendor specific in common usage)
2Ah	00h						R									CD Capabilities and Mechanical Status
2Bh to 3Eh																Device-type specific (vendor specific in common usage)
3Fh	xxh															Return all pages and/or subpages (MODE SENSE only)
xxh	FFh															Return all subpages (MODE SENSE only)
																All codes not shown here or in table D.13 are restricted or reserved.

Table D.13 is a numerical order listing of the mode page codes used by SCSI transport protocols.

Table D.13 — Transport protocol specific mode page codes

F – Fibre Channel Protocol for SCSI (FCP-4)
. S – SAS Protocol Layer (SPL-2)
. V – Automation/Device Interface device (ADT-2)
. U – USB Attached SCSI (UAS-2)

Device Column key
blank = code not used
not blank = code used

Mode Page Code	Mode Subpage Code	F	S	V	U	Mode Page Name
18h	00h	F	S			Protocol Specific Logical Unit
19h	00h	F	S			Protocol Specific Port
19h	01h		S			Phy Control And Discover
19h	02h		S			Shared Port Control

See table D.12 for information on the codes not shown here.

D.7 VPD page codes

Table D.14 is a numerical order listing of the VPD page codes.

Table D.14 — VPD page codes

D – Direct Access Block Device (SBC-3)
. T – Sequential Access Device (SSC-4)
. L – Printer Device (SSC)
. P – Processor Device (SPC-2)
. . W – Write Once Block Device (SBC)
. . R – C/DVD Device (MMC-6)
. . O – Optical Memory Block Device (SBC)
. . . M – Media Changer Device (SMC-3)
. . . . A – Storage Array Device (SCC-2)
. . . . E – SCSI Enclosure Services device (SES-3)
. . . . . B – Simplified Direct-Access (Reduced Block) device (RBC)
. . . . . K – Optical Card Reader/Writer device (OCRW)
. . . . . V – Automation/Device Interface device (ADC-3)
. . . . . F – Object-based Storage Device (OSD-2)
. . . . . .

Device Column key
blank = code not used
not blank = code used

VPD Page Code	D T L P W R O M A E B K V F	VPD Page Name
00h	D T L P W R O M A E B K V F	Supported VPD Pages
01h to 7Fh	D T L P W R O M A E B K V F	ASCII Information
80h	D T L P W R O M A E B K V F	Unit Serial Number
81h	D T L P W R O M A E B K V F	Obsolete
82h	D T L P W R O M A E B K V F	Obsolete
83h	D T L P W R O M A E B K V F	Device Identification
84h	D T L P W R O M A E B K V F	Software Interface Identification
85h	D T L P W R O M A E B K V F	Management Network Addresses
86h	D T L P W R O M A E B K V F	Extended INQUIRY Data
87h	D T L P W R O M A E B K V F	Mode Page Policy
88h	D T L P W R O M A E B K V F	SCSI Ports
89h	D	ATA Information
8Ah	D T L P W R O A K	Power Condition
8Bh	T M	Device Constituents
8Ch	D B	CFA Profile Information
90h	D T L W R O M A E V F	Protocol Specific Logical Unit Information
91h	D T L W R O M A E V F	Protocol Specific Port Information
B0h	D	Block Limits
B0h	T	Sequential-access Device Capabilities
B0h	F	OSD information
B1h	D	Block Device Characteristics
B1h	T V	Manufacturer-assigned Serial Number
B1h	F	Security Token
B2h	D	Logical Block Provisioning
B2h	T	TapeAlert Supported Flags
B3h	D	Referrals
B3h	T	Automation Device Serial Number
C0h to FFh		Vendor specific

All codes not shown here or in table D.15 are restricted or reserved.

Table D.13 is a numerical order listing of the VPD page codes used by SCSI transport protocols.

Table D.15 — Transport protocol specific VPD page codes

F – Fibre Channel Protocol for SCSI (FCP-4)
. S – SAS Protocol Layer (SPL-2)
. V – Automation/Device Interface device (ADT-2)
. U – USB Attached SCSI (UAS-2)

Device Column key
blank = code not used
not blank = code used

VPD Page Code	F S V U	VPD Page Name
90h	S	Protocol Specific Logical Unit Information
91h	S	Protocol Specific Port Information

See table D.14 for information on the codes not shown here.

D.8 Version descriptor values

Table D.16 is a numerical order listing of the version descriptor values used in the standard INQUIRY data. Each version descriptor value is computed from a coded value identifying the standard and a coded value representing the revision of the standard. The formula is ((standard×32)+revison). Table D.16 shows all three code values and the associated standard name. The version descriptor code is shown in both decimal and hexadecimal.

Table D.16 — Version descriptor assignments (part 1 of 14)

Standard Code	Revision Code	Version Descriptor Code		Standard
		decimal	hex	
0	0	0	0000h	Version Descriptor Not Supported or No Standard Identified
1	0	32	0020h	SAM (no version claimed)
1	27	59	003Bh	SAM T10/0994-D revision 18
1	28	60	003Ch	SAM ANSI INCITS 270-1996
2	0	64	0040h	SAM-2 (no version claimed)
2	20	84	0054h	SAM-2 T10/1157-D revision 23
2	21	85	0055h	SAM-2 T10/1157-D revision 24
2	28	92	005Ch	SAM-2 ANSI INCITS 366-2003
2	30	94	005Eh	SAM-2 ISO/IEC 14776-412
3	0	96	0060h	SAM-3 (no version claimed)
3	2	98	0062h	SAM-3 T10/1561-D revision 7
3	21	117	0075h	SAM-3 T10/1561-D revision 13
3	22	118	0076h	SAM-3 T10/1561-D revision 14
3	23	119	0077h	SAM-3 ANSI INCITS 402-2005
4	0	128	0080h	SAM-4 (no version claimed)
4	7	135	0087h	SAM-4 T10/1683-D revision 13
4	11	139	008Bh	SAM-4 T10/1683-D revision 14
4	16	144	0090h	SAM-4 ANSI INCITS 447-2008
4	18	146	0092h	SAM-4 ISO/IEC 14776-414
5	0	160	00A0h	SAM-5 (no version claimed)
5	2	162	00A2h	SAM-5 T10/2104-D revision 4
9	0	288	0120h	SPC (no version claimed)
9	27	315	013Bh	SPC T10/0995-D revision 11a
9	28	316	013Ch	SPC ANSI INCITS 301-1997

Table D.16 — Version descriptor assignments (part 2 of 14)

Standard Code	Revision Code	Version Descriptor Code		Standard
		decimal	hex	
10	0	320	0140h	MMC (no version claimed)
10	27	347	015Bh	MMC T10/1048-D revision 10a
10	28	348	015Ch	MMC ANSI INCITS 304-1997
11	0	352	0160h	SCC (no version claimed)
11	27	379	017Bh	SCC T10/1047-D revision 06c
11	28	380	017Ch	SCC ANSI INCITS 276-1997
12	0	384	0180h	SBC (no version claimed)
12	27	411	019Bh	SBC T10/0996-D revision 08c
12	28	412	019Ch	SBC ANSI INCITS 306-1998
13	0	416	01A0h	SMC (no version claimed)
13	27	443	01BBh	SMC T10/0999-D revision 10a
13	28	444	01BCh	SMC ANSI INCITS 314-1998
13	30	446	01BEh	SMC ISO/IEC 14776-351
14	0	448	01C0h	SES (no version claimed)
14	27	475	01DBh	SES T10/1212-D revision 08b
14	28	476	01DCh	SES ANSI INCITS 305-1998
14	29	477	01DDh	SES T10/1212 revision 08b w/ Amendment ANSI INCITS.305/ AM1-2000
14	30	478	01DEh	SES ANSI INCITS 305-1998 w/ Amendment ANSI INCITS.305/ AM1-2000
15	0	480	01E0h	SCC-2 (no version claimed)
15	27	507	01FBh	SCC-2 T10/1125-D revision 04
15	28	508	01FCh	SCC-2 ANSI INCITS 318-1998
16	0	512	0200h	SSC (no version claimed)
16	1	513	0201h	SSC T10/0997-D revision 17
16	7	519	0207h	SSC T10/0997-D revision 22
16	28	540	021Ch	SSC ANSI INCITS 335-2000
17	0	544	0220h	RBC (no version claimed)
17	24	568	0238h	RBC T10/1240-D revision 10a
17	28	572	023Ch	RBC ANSI INCITS 330-2000

Table D.16 — Version descriptor assignments (part 3 of 14)

Standard Code	Revision Code	Version Descriptor Code		Standard
		decimal	hex	
18	0	576	0240h	MMC-2 (no version claimed)
18	21	597	0255h	MMC-2 T10/1228-D revision 11
18	27	603	025Bh	MMC-2 T10/1228-D revision 11a
18	28	604	025Ch	MMC-2 ANSI INCITS 333-2000
19	0	608	0260h	SPC-2 (no version claimed)
19	7	615	0267h	SPC-2 T10/1236-D revision 12
19	9	617	0269h	SPC-2 T10/1236-D revision 18
19	21	629	0275h	SPC-2 T10/1236-D revision 19
19	22	630	0276h	SPC-2 T10/1236-D revision 20
19	23	631	0277h	SPC-2 ANSI INCITS 351-2001
19	24	632	0278h	SPC-2 ISO/IEC 14776-452
20	0	640	0280h	OCRW (no version claimed)
20	30	670	029Eh	OCRW ISO/IEC 14776-381
21	0	672	02A0h	MMC-3 (no version claimed)
21	21	693	02B5h	MMC-3 T10/1363-D revision 9
21	22	694	02B6h	MMC-3 T10/1363-D revision 10g
21	24	696	02B8h	MMC-3 ANSI INCITS 360-2002
23	0	736	02E0h	SMC-2 (no version claimed)
23	21	757	02F5h	SMC-2 T10/1383-D revision 5
23	28	764	02FCh	SMC-2 T10/1383-D revision 6
23	29	765	02FDh	SMC-2 T10/1383-D revision 7
23	30	766	02FEh	SMC-2 ANSI INCITS 382-2004
24	0	768	0300h	SPC-3 (no version claimed)
24	1	769	0301h	SPC-3 T10/1416-D revision 7
24	7	775	0307h	SPC-3 T10/1416-D revision 21
24	15	783	030Fh	SPC-3 T10/1416-D revision 22
24	18	786	0312h	SPC-3 T10/1416-D revision 23
24	20	788	0314h	SPC-3 ANSI INCITS 408-2005
24	22	790	0316h	SPC-3 ISO/IEC 14776-453

Table D.16 — Version descriptor assignments (part 4 of 14)

Standard Code	Revision Code	Version Descriptor Code		Standard
		decimal	hex	
25	0	800	0320h	SBC-2 (no version claimed)
25	2	802	0322h	SBC-2 T10/1417-D revision 5a
25	4	804	0324h	SBC-2 T10/1417-D revision 15
25	27	827	033Bh	SBC-2 T10/1417-D revision 16
25	29	829	033Dh	SBC-2 ANSI INCITS 405-2005
25	30	830	033Eh	SBC-2 ISO/IEC 14776-322
26	0	832	0340h	OSD (no version claimed)
26	1	833	0341h	OSD T10/1355-D revision 0
26	2	834	0342h	OSD T10/1355-D revision 7a
26	3	835	0343h	OSD T10/1355-D revision 8
26	4	836	0344h	OSD T10/1355-D revision 9
26	21	853	0355h	OSD T10/1355-D revision 10
26	22	854	0356h	OSD ANSI INCITS 400-2004
27	0	864	0360h	SSC-2 (no version claimed)
27	20	884	0374h	SSC-2 T10/1434-D revision 7
27	21	885	0375h	SSC-2 T10/1434-D revision 9
27	29	893	037Dh	SSC-2 ANSI INCITS 380-2003
28	0	896	0380h	BCC (no version claimed)
29	0	928	03A0h	MMC-4 (no version claimed)
29	16	944	03B0h	MMC-4 T10/1545-D revision 5
29	17	945	03B1h	MMC-4 T10/1545-D revision 5a
29	29	957	03BDh	MMC-4 T10/1545-D revision 3
29	30	958	03BEh	MMC-4 T10/1545-D revision 3d
29	31	959	03BFh	MMC-4 ANSI INCITS 401-2005
30	0	960	03C0h	ADC (no version claimed)
30	21	981	03D5h	ADC T10/1558-D revision 6
30	22	982	03D6h	ADC T10/1558-D revision 7
30	23	983	03D7h	ADC ANSI INCITS 403-2005
31	0	992	03E0h	SES-2 (no version claimed)

Table D.16 — Version descriptor assignments (part 5 of 14)

Standard Code	Revision Code	Version Descriptor Code		Standard
		decimal	hex	
31	1	993	03E1h	SES-2 T10/1559-D revision 16
31	7	999	03E7h	SES-2 T10/1559-D revision 19
31	11	1003	03EBh	SES-2 T10/1559-D revision 20
31	16	1008	03F0h	SES-2 ANSI INCITS 448-2008
32	0	1024	0400h	SSC-3 (no version claimed)
32	3	1027	0403h	SSC-3 T10/1611-D revision 04a
32	7	1031	0407h	SSC-3 T10/1611-D revision 05
33	0	1056	0420h	MMC-5 (no version claimed)
33	15	1071	042Fh	MMC-5 T10/1675-D revision 03
33	17	1073	0431h	MMC-5 T10/1675-D revision 03b
33	18	1074	0432h	MMC-5 T10/1675-D revision 04
33	20	1076	0434h	MMC-5 ANSI INCITS 430-2007
34	0	1088	0440h	OSD-2 (no version claimed)
34	4	1092	0444h	OSD-2 T10/1729-D revision 4
34	6	1094	0446h	OSD-2 T10/1729-D revision 5
34	8	1096	0448h	OSD-2 ANSI INCITS 458-2011
35	0	1120	0460h	SPC-4 (no version claimed)
35	1	1121	0461h	SPC-4 T10/1731-D revision 16
35	2	1122	0462h	SPC-4 T10/1731-D revision 18
35	3	1123	0463h	SPC-4 T10/1731-D revision 23
36	0	1152	0480h	SMC-3 (no version claimed)
36	2	1154	0482h	SMC-3 T10/1730-D revision 15
37	0	1184	04A0h	ADC-2 (no version claimed)
37	7	1191	04A7h	ADC-2 T10/1741-D revision 7
37	10	1194	04AAh	ADC-2 T10/1741-D revision 8
37	12	1196	04ACh	ADC-2 ANSI INCITS 441-2008
38	0	1216	04C0h	SBC-3 (no version claimed)
39	0	1248	04E0h	MMC-6 (no version claimed)
39	3	1251	04E3h	MMC-6 T10/1836-D revision 02b

Table D.16 — Version descriptor assignments (part 6 of 14)

Standard Code	Revision Code	Version Descriptor Code		Standard
		decimal	hex	
39	5	1253	04E5h	MMC-6 T10/1836-D revision 02g
39	6	1254	04E6h	MMC-6 ANSI INCITS 468-2010
40	0	1280	0500h	ADC-3 (no version claimed)
40	2	1282	0502h	ADC-3 T10/1895-D revision 04
41	0	1312	0520h	SSC-4 (no version claimed)
43	0	1376	0560h	OSD-3 (no version claimed)
44	0	1408	0580h	SES-3 (no version claimed)
65	0	2080	0820h	SSA-TL2 (no version claimed)
65	27	2107	083Bh	SSA-TL2 T10.1/1147-D revision 05b
65	28	2108	083Ch	SSA-TL2 ANSI INCITS 308-1998
66	0	2112	0840h	SSA-TL1 (no version claimed)
66	27	2139	085Bh	SSA-TL1 T10.1/0989-D revision 10b
66	28	2140	085Ch	SSA-TL1 ANSI INCITS 295-1996
67	0	2144	0860h	SSA-S3P (no version claimed)
67	27	2171	087Bh	SSA-S3P T10.1/1051-D revision 05b
67	28	2172	087Ch	SSA-S3P ANSI INCITS 309-1998
68	0	2176	0880h	SSA-S2P (no version claimed)
68	27	2203	089Bh	SSA-S2P T10.1/1121-D revision 07b
68	28	2204	089Ch	SSA-S2P ANSI INCITS 294-1996
69	0	2208	08A0h	SIP (no version claimed)
69	27	2235	08BBh	SIP T10/0856-D revision 10
69	28	2236	08BCh	SIP ANSI INCITS 292-1997
70	0	2240	08C0h	FCP (no version claimed)
70	27	2267	08DBh	FCP T10/0993-D revision 12
70	28	2268	08DCh	FCP ANSI INCITS 269-1996
71	0	2272	08E0h	SBP-2 (no version claimed)
71	27	2299	08FBh	SBP-2 T10/1155-D revision 04
71	28	2300	08FCh	SBP-2 ANSI INCITS 325-1998
72	0	2304	0900h	FCP-2 (no version claimed)

Table D.16 — Version descriptor assignments (part 7 of 14)

Standard Code	Revision Code	Version Descriptor Code		Standard
		decimal	hex	
72	1	2305	0901h	FCP-2 T10/1144-D revision 4
72	21	2325	0915h	FCP-2 T10/1144-D revision 7
72	22	2326	0916h	FCP-2 T10/1144-D revision 7a
72	23	2327	0917h	FCP-2 ANSI INCITS 350-2003
72	24	2328	0918h	FCP-2 T10/1144-D revision 8
73	0	2336	0920h	SST (no version claimed)
73	21	2357	0935h	SST T10/1380-D revision 8b
74	0	2368	0940h	SRP (no version claimed)
74	20	2388	0954h	SRP T10/1415-D revision 10
74	21	2389	0955h	SRP T10/1415-D revision 16a
74	28	2396	095Ch	SRP ANSI INCITS 365-2002
75	0	2400	0960h	iSCSI (no version claimed)
76	0	2432	0980h	SBP-3 (no version claimed)
76	2	2434	0982h	SBP-3 T10/1467-D revision 1f
76	20	2452	0994h	SBP-3 T10/1467-D revision 3
76	26	2458	099Ah	SBP-3 T10/1467-D revision 4
76	27	2459	099Bh	SBP-3 T10/1467-D revision 5
76	28	2460	099Ch	SBP-3 ANSI INCITS 375-2004
78	0	2496	09C0h	ADP (no version claimed)
79	0	2528	09E0h	ADT (no version claimed)
79	25	2553	09F9h	ADT T10/1557-D revision 11
79	26	2554	09FAh	ADT T10/1557-D revision 14
79	29	2557	09FDh	ADT ANSI INCITS 406-2005
80	0	2560	0A00h	FCP-3 (no version claimed)
80	7	2567	0A07h	FCP-3 T10/1560-D revision 3f
80	15	2575	0A0Fh	FCP-3 T10/1560-D revision 4
80	17	2577	0A11h	FCP-3 ANSI INCITS 416-2006
80	28	2588	0A1Ch	FCP-3 ISO/IEC 14776-223
81	0	2592	0A20h	ADT-2 (no version claimed)

Table D.16 — Version descriptor assignments (part 8 of 14)

Standard Code	Revision Code	Version Descriptor Code		Standard
		decimal	hex	
81	2	2594	0A22h	ADT-2 T10/1742-D revision 06
81	7	2599	0A27h	ADT-2 T10/1742-D revision 08
81	8	2600	0A28h	ADT-2 T10/1742-D revision 09
82	0	2624	0A40h	FCP-4 (no version claimed)
82	2	2626	0A42h	FCP-4 T10/1828-D revision 01
82	4	2628	0A44h	FCP-4 T10/1828-D revision 02
82	5	2629	0A45h	FCP-4 T10/1828-D revision 02b
85	0	2720	0AA0h	SPI (no version claimed)
85	25	2745	0AB9h	SPI T10/0855-D revision 15a
85	26	2746	0ABAh	SPI ANSI INCITS 253-1995
85	27	2747	0ABBh	SPI T10/0855-D revision 15a with SPI Amnd revision 3a
85	28	2748	0ABCh	SPI ANSI INCITS 253-1995 with SPI Amnd ANSI INCITS 253/ AM1-1998
86	0	2752	0AC0h	Fast-20 (no version claimed)
86	27	2779	0ADBh	Fast-20 T10/1071 revision 06
86	28	2780	0ADCh	Fast-20 ANSI INCITS 277-1996
87	0	2784	0AE0h	SPI-2 (no version claimed)
87	27	2811	0AFBh	SPI-2 T10/1142-D revision 20b
87	28	2812	0AFCh	SPI-2 ANSI INCITS 302-1999
88	0	2816	0B00h	SPI-3 (no version claimed)
88	24	2840	0B18h	SPI-3 T10/1302-D revision 10
88	25	2841	0B19h	SPI-3 T10/1302-D revision 13a
88	26	2842	0B1Ah	SPI-3 T10/1302-D revision 14
88	28	2844	0B1Ch	SPI-3 ANSI INCITS 336-2000
89	0	2848	0B20h	EPI (no version claimed)
89	27	2875	0B3Bh	EPI T10/1134 revision 16
89	28	2876	0B3Ch	EPI ANSI INCITS TR-23 1999
90	0	2880	0B40h	SPI-4 (no version claimed)
90	20	2900	0B54h	SPI-4 T10/1365-D revision 7

Table D.16 — Version descriptor assignments (part 9 of 14)

Standard Code	Revision Code	Version Descriptor Code		Standard
		decimal	hex	
90	21	2901	0B55h	SPI-4 T10/1365-D revision 9
90	22	2902	0B56h	SPI-4 ANSI INCITS 362-2002
90	25	2905	0B59h	SPI-4 T10/1365-D revision 10
91	0	2912	0B60h	SPI-5 (no version claimed)
91	25	2937	0B79h	SPI-5 T10/1525-D revision 3
91	26	2938	0B7Ah	SPI-5 T10/1525-D revision 5
91	27	2939	0B7Bh	SPI-5 T10/1525-D revision 6
91	28	2940	0B7Ch	SPI-5 ANSI INCITS 367-2003
95	0	3040	0BE0h	SAS (no version claimed)
95	1	3041	0BE1h	SAS T10/1562-D revision 01
95	21	3061	0BF5h	SAS T10/1562-D revision 03
95	26	3066	0BFAh	SAS T10/1562-D revision 04
95	27	3067	0BFBh	SAS T10/1562-D revision 04
95	28	3068	0BFCh	SAS T10/1562-D revision 05
95	29	3069	0BFDh	SAS ANSI INCITS 376-2003
96	0	3072	0C00h	SAS-1.1 (no version claimed)
96	7	3079	0C07h	SAS-1.1 T10/1601-D revision 9
96	15	3087	0C0Fh	SAS-1.1 T10/1601-D revision 10
96	17	3089	0C11h	SAS-1.1 ANSI INCITS 417-2006
97	0	3104	0C20h	SAS-2 (no version claimed)
97	3	3107	0C23h	SAS-2 T10/1760-D revision 14
97	7	3111	0C27h	SAS-2 T10/1760-D revision 15
97	8	3112	0C28h	SAS-2 T10/1760-D revision 16
97	10	3114	0C2Ah	SAS-2 ANSI INCITS 457-2010
98	0	3136	0C40h	SAS-2.1 (no version claimed)
98	8	3144	0C48h	SAS-2.1 T10/2125-D revision 04
98	10	3146	0C4Ah	SAS-2.1 T10/2125-D revision 06
98	11	3147	0C4Bh	SAS-2.1 T10/2125-D revision 07
99	0	3168	0C60h	SAS-3 (no version claimed)

Table D.16 — Version descriptor assignments (part 10 of 14)

Standard Code	Revision Code	Version Descriptor Code		Standard
		decimal	hex	
105	0	3360	0D20h	FC-PH (no version claimed)
105	27	3387	0D3Bh	FC-PH ANSI INCITS 230-1994
105	28	3388	0D3Ch	FC-PH ANSI INCITS 230-1994 with Amnd 1 ANSI INCITS 230/ AM1-1996
106	0	3392	0D40h	FC-AL (no version claimed)
106	28	3420	0D5Ch	FC-AL ANSI INCITS 272-1996
107	0	3424	0D60h	FC-AL-2 (no version claimed)
107	1	3425	0D61h	FC-AL-2 T11/1133-D revision 7.0
107	3	3427	0D63h	FC-AL-2 ANSI INCITS 332-1999 with AM1-2003 & AM2-2006
107	4	3428	0D64h	FC-AL-2 ANSI INCITS 332-1999 with Amnd 2 AM2-2006
107	5	3429	0D65h	FC-AL-2 ISO/IEC 14165-122 with AM1 & AM2
107	28	3452	0D7Ch	FC-AL-2 ANSI INCITS 332-1999
107	29	3453	0D7Dh	FC-AL-2 ANSI INCITS 332-1999 with Amnd 1 AM1-2003
108	0	3456	0D80h	FC-PH-3 (no version claimed)
108	28	3484	0D9Ch	FC-PH-3 ANSI INCITS 303-1998
109	0	3488	0DA0h	FC-FS (no version claimed)
109	23	3511	0DB7h	FC-FS T11/1331-D revision 1.2
109	24	3512	0DB8h	FC-FS T11/1331-D revision 1.7
109	28	3516	0DBCh	FC-FS ANSI INCITS 373-2003
109	29	3517	0DBDh	FC-FS ISO/IEC 14165-251
110	0	3520	0DC0h	FC-PI (no version claimed)
110	28	3548	0DDCh	FC-PI ANSI INCITS 352-2002
111	0	3552	0DE0h	FC-PI-2 (no version claimed)
111	2	3554	0DE2h	FC-PI-2 T11/1506-D revision 5.0
111	4	3556	0DE4h	FC-PI-2 ANSI INCITS 404-2006
112	0	3584	0E00h	FC-FS-2 (no version claimed)
112	2	3586	0E02h	FC-FS-2 ANSI INCITS 242-2007
112	3	3587	0E03h	FC-FS-2 ANSI INCITS 242-2007 with AM1 ANSI INCITS 242/ AM1-2007
113	0	3616	0E20h	FC-LS (no version claimed)

Table D.16 — Version descriptor assignments (part 11 of 14)

Standard Code	Revision Code	Version Descriptor Code		Standard
		decimal	hex	
113	1	3617	0E21h	FC-LS T11/1620-D revision 1.62
113	9	3625	0E29h	FC-LS ANSI INCITS 433-2007
114	0	3648	0E40h	FC-SP (no version claimed)
114	2	3650	0E42h	FC-SP T11/1570-D revision 1.6
114	5	3653	0E45h	FC-SP ANSI INCITS 426-2007
115	0	3680	0E60h	FC-PI-3 (no version claimed)
115	2	3682	0E62h	FC-PI-3 T11/1625-D revision 2.0
115	8	3688	0E68h	FC-PI-3 T11/1625-D revision 2.1
115	10	3690	0E6Ah	FC-PI-3 T11/1625-D revision 4.0
116	0	3712	0E80h	FC-PI-4 (no version claimed)
116	2	3714	0E82h	FC-PI-4 T11/1647-D revision 8.0
116	8	3720	0E88h	FC-PI-4 ANSI INCITS 450-2009
117	0	3744	0EA0h	FC 10GFC (no version claimed)
117	2	3746	0EA2h	FC 10GFC ANSI INCITS 364-2003
117	3	3747	0EA3h	FC 10GFC ISO/IEC 14165-116
117	5	3749	0EA5h	FC 10GFC ISO/IEC 14165-116 with AM1
117	6	3750	0EA6h	FC 10GFC ANSI INCITS 364-2003 with AM1 ANSI INCITS 364/ AM1-2007
118	0	3776	0EC0h	FC-SP-2 (no version claimed)
119	0	3808	0EE0h	FC-FS-3 (no version claimed)
119	2	3810	0EE2h	FC-FS-3 T11/1861-D revision 0.9
119	7	3815	0EE7h	FC-FS-3 T11/1861-D revision 1.0
119	9	3817	0EE9h	FC-FS-3 T11/1861-D revision 1.10
120	0	3840	0F00h	FC-LS-2 (no version claimed)
120	3	3843	0F03h	FC-LS-2 T11/2103-D revision 2.11
120	5	3845	0F05h	FC-LS-2 T11/2103-D revision 2.21
121	0	3872	0F20h	FC-PI-5 (no version claimed)
121	7	3879	0F27h	FC-PI-5 T11/2118-D revision 2.00
121	8	3880	0F28h	FC-PI-5 T11/2118-D revision 3.00

Table D.16 — Version descriptor assignments (part 12 of 14)

Standard Code	Revision Code	Version Descriptor Code		Standard
		decimal	hex	
121	10	3882	0F2Ah	FC-PI-5 T11/2118-D revision 6.00
121	11	3883	0F2Bh	FC-PI-5 T11/2118-D revision 6.10
122	0	3904	0F40h	FC-PI-6 (no version claimed)
123	0	3936	0F60h	FC-FS-4 (no version claimed)
124	0	3968	0F80h	FC-LS-3 (no version claimed)
149	0	4768	12A0h	FC-SCM (no version claimed)
149	3	4771	12A3h	FC-SCM T11/1824DT revision 1.0
149	5	4773	12A5h	FC-SCM T11/1824DT revision 1.1
149	7	4775	12A7h	FC-SCM T11/1824DT revision 1.4
150	0	4800	12C0h	FC-DA-2 (no version claimed)
150	3	4803	12C3h	FC-DA-2 T11/1870DT revision 1.04
150	5	4805	12C5h	FC-DA-2 T11/1870DT revision 1.06
151	0	4832	12E0h	FC-DA (no version claimed)
151	2	4834	12E2h	FC-DA T11/1513-DT revision 3.1
151	8	4840	12E8h	FC-DA ANSI INCITS TR-36 2004
151	9	4841	12E9h	FC-DA ISO/IEC 14165-341
152	0	4864	1300h	FC-Tape (no version claimed)
152	1	4865	1301h	FC-Tape T11/1315 revision 1.16
152	27	4891	131Bh	FC-Tape T11/1315 revision 1.17
152	28	4892	131Ch	FC-Tape ANSI INCITS TR-24 1999
153	0	4896	1320h	FC-FLA (no version claimed)
153	27	4923	133Bh	FC-FLA T11/1235 revision 7
153	28	4924	133Ch	FC-FLA ANSI INCITS TR-20 1998
154	0	4928	1340h	FC-PLDA (no version claimed)
154	27	4955	135Bh	FC-PLDA T11/1162 revision 2.1
154	28	4956	135Ch	FC-PLDA ANSI INCITS TR-19 1998
155	0	4960	1360h	SSA-PH2 (no version claimed)
155	27	4987	137Bh	SSA-PH2 T10.1/1145-D revision 09c
155	28	4988	137Ch	SSA-PH2 ANSI INCITS 293-1996

Table D.16 — Version descriptor assignments (part 13 of 14)

Standard Code	Revision Code	Version Descriptor Code		Standard
		decimal	hex	
156	0	4992	1380h	SSA-PH3 (no version claimed)
156	27	5019	139Bh	SSA-PH3 T10.1/1146-D revision 05b
156	28	5020	139Ch	SSA-PH3 ANSI INCITS 307-1998
165	0	5280	14A0h	IEEE 1394 (no version claimed)
165	29	5309	14BDh	ANSI IEEE 1394-1995
166	0	5312	14C0h	IEEE 1394a (no version claimed)
167	0	5344	14E0h	IEEE 1394b (no version claimed)
175	0	5600	15E0h	ATA/ATAPI-6 (no version claimed)
175	29	5629	15FDh	ATA/ATAPI-6 ANSI INCITS 361-2002
176	0	5632	1600h	ATA/ATAPI-7 (no version claimed)
176	2	5634	1602h	ATA/ATAPI-7 T13/1532-D revision 3
176	28	5660	161Ch	ATA/ATAPI-7 ANSI INCITS 397-2005
177	0	5664	1620h	ATA/ATAPI-8 ATA8-AAM (no version claimed)
177	1	5665	1621h	ATA/ATAPI-8 ATA8-APT Parallel Transport (no version claimed)
177	2	5666	1622h	ATA/ATAPI-8 ATA8-AST Serial Transport (no version claimed)
177	3	5667	1623h	ATA/ATAPI-8 ATA8-ACS ATA/ATAPI Command Set (no version claimed)
177	8	5672	1628h	ATA/ATAPI-8 ATA8-AAM ANSI INCITS 451-2008
177	10	5674	162Ah	ATA/ATAPI-8 ATA8-ACS ANSI INCITS 452-2009 w/ Amendment 1
185	8	5928	1728h	Universal Serial Bus Specification, Revision 1.1
185	9	5929	1729h	Universal Serial Bus Specification, Revision 2.0
185	16	5936	1730h	USB Mass Storage Class Bulk-Only Transport, Revision 1.0
186	0	5952	1740h	UAS (no version claimed)
186	3	5955	1743h	UAS T10/2095-D revision 02
186	7	5959	1747h	UAS T10/2095-D revision 04
186	8	5960	1748h	UAS ANSI INCITS 471-2010
187	1	5985	1761h	ACS-2 (no version claimed)
188	0	6016	1780h	UAS-2 (no version claimed)
245	0	7840	1EA0h	SAT (no version claimed)

Table D.16 — Version descriptor assignments (part 14 of 14)

Standard Code	Revision Code	Version Descriptor Code		Standard
		decimal	hex	
245	7	7847	1EA7h	SAT T10/1711-D revision 8
245	11	7851	1EABh	SAT T10/1711-D revision 9
245	13	7853	1EADh	SAT ANSI INCITS 431-2007
246	0	7872	1EC0h	SAT-2 (no version claimed)
246	4	7876	1EC4h	SAT-2 T10/1826-D revision 06
246	8	7880	1EC8h	SAT-2 T10/1826-D revision 09
246	10	7882	1ECAh	SAT-2 ANSI INCITS 465-2010
247	0	7904	1EE0h	SAT-3 (no version claimed)
261	0	8352	20A0h	SPL (no version claimed)
261	3	8355	20A3h	SPL T10/2124-D revision 6a
261	5	8357	20A5h	SPL T10/2124-D revision 7
262	0	8384	20C0h	SPL-2 (no version claimed)
271	0	8672	21E0h	SOP (no version claimed)
272	0	8704	2200h	PQI (no version claimed)
2046	0	65472	FFC0h	IEEE 1667 (no version claimed)
2046	1	65473	FFC1h	IEEE 1667-2006
2046	2	65474	FFC2h	IEEE 1667-2009

Table D.17 shows the guidelines used by T10 when selecting a coded value for a standard.

Table D.17 — Standard code value guidelines (part 1 of 6)

Standard Code	Standard or standards family
0	Version Descriptor Not Supported
1 to 8	Architecture Model
1	SAM
2	SAM-2
3	SAM-3
4	SAM-4
5	SAM-5

Table D.17 — Standard code value guidelines (part 2 of 6)

Standard Code	Standard or standards family
9 to 64	Command Set
9	SPC
10	MMC
11	SCC
12	SBC
13	SMC
14	SES
15	SCC-2
16	SSC
17	RBC
18	MMC-2
19	SPC-2
20	OCRW
21	MMC-3
22	obsolete
23	SMC-2
24	SPC-3
25	SBC-2
26	OSD
27	SSC-2
28	obsolete
29	MMC-4
30	ADC
31	SES-2
32	SSC-3
33	MMC-5
34	OSD-2
35	SPC-4
36	SMC-3
37	ADC-2
38	SBC-3

Table D.17 — Standard code value guidelines (part 3 of 6)

Standard Code	Standard or standards family
39	MMC-6
40	ADC-3
41	SSC-4
43	OSD-3
44	SES-3
65 to 84	Physical Mapping Protocol
65	SSA-TL2
66	SSA-TL1
67	SSA-S3P
68	SSA-S2P
69	SIP
70	FCP
71	SBP-2
72	FCP-2
73	SST
74	SRP
75	iSCSI
76	SBP-3
77	obsolete
78	ADP
79	ADT
80	FCP-3
81	ADT-2
82	FCP-4
85 to 94	Parallel SCSI Physical
85	SPI and SPI Amendment
86	Fast-20
87	SPI-2
88	SPI-3
89	EPI
90	SPI-4

Table D.17 — Standard code value guidelines (part 4 of 6)

Standard Code	Standard or standards family
91	SPI-5
95 to 104	Serial Attached SCSI
95	SAS
96	SAS-1.1
97	SAS-2
98	SAS-2.1
99	SAS-3
105 to 154	Fibre Channel
105	FC-PH and FC-PH Amendment
106	FC-AL
107	FC-AL-2
108	FC-PH-3
109	FC-FS
110	FC-PI
111	FC-PI-2
112	FC-FS-2
113	FC-LS
114	FC-SP
115	FC-PI-3
116	FC-PI-4
117	FC 10GFC
118	FC-SP-2
119	FC-FS-3
120	FC-LS-2
121	FC-PI-5
122	FC-PI-6
123	FC-FS-4
124	FC-LS-3
149	FC-SCM
150	FC-DA-2
151	FC-DA

Table D.17 — Standard code value guidelines (part 5 of 6)

Standard Code	Standard or standards family
152	FC-Tape
153	FC-FLA
154	FC-PLDA
155 to 164	SSA
155	SSA-PH2
156	SSA-PH3
165 to 174	IEEE 1394
165	IEEE 1394:1995
166	IEEE 1394a
167	IEEE 1394b
175 to 200	ATAPI & USB
175	ATA/ATAPI-6
176	ATA/ATAPI-7
177	ATA/ATAPI-8
185	USB
186	UAS
187	ACS-x
188	UAS-2
201 to 224	Networking
225 to 244	ATM
245 to 260	Translators
245	SAT
246	SAT-2
247	SAT-3
261 to 270	SAS Transport Protocols
261	SPL
262	SPL-2

Table D.17 — Standard code value guidelines (part 6 of 6)

Standard Code	Standard or standards family
271 to 290	SCSI over PCI Express Transport Protocols
271	SOP
272	PQI
291 to 2045	Reserved for Expansion
2046	IEEE 1667
2047	Reserved

D.9 T10 IEEE binary identifiers

The IEEE binary identifiers assigned to T10 standards are shown in table D.18.

Table D.18 — IEEE binary identifiers assigned by T10

IEEE Binary Identifier	T10 Standard
0060 9E01 03E0h	SBP (obsolete)
0060 9E01 0483h	SBP-2
0060 9E01 04D8h	SPC-2
0060 9E01 05BBh	SBP-3
0060 9E01 0800h	SRP revision 10
0060 9E01 0801h	ANSI SRP

Annex E
(informative)

T10 vendor identification

This annex contains the list of T10 vendor identifications (see table E.1) as of the date of this document. The purpose of this list is to help avoid redundant usage of T10 vendor identifications. Technical Committee T10 of Accredited Standards Committee INCITS maintains an informal list of T10 vendor identifications currently in use. The T10 web site, http://www.t10.org, provides a convenient means to request an identification code. If problems are encountered using the T10 web site, please contact the chairman of T10 prior to using a new T10 vendor identification to avoid conflicts.

The information in this annex was complete and accurate at the time of publication. However, the information is subject to change. Technical Committee T10 of INCITS maintains an electronic copy of this information on its world wide web site (http://www.t10.org/). In the event that the T10 world wide web site is no longer active, access may be possible via the INCITS world wide web site (http://www.incits.org), the ANSI world wide web site (http://www.ansi.org), the IEC site (http://www.iec.ch/), the ISO site (http://www.iso.ch/), or the ISO/IEC JTC 1 web site (http://www.jtc1.org/).

Table E.1 — T10 vendor identification list (part 1 of 15)

ID	Organization
0B4C	MOOSIK Ltd.
2AI	2AI (Automatisme et Avenir Informatique)
3M	3M Company
3nhtech	3NH Technologies
3PARdata	3PARdata, Inc. (now HP)
A-Max	A-Max Technology Co., Ltd
ABSOLUTE	Absolute Analysis
ACARD	ACARD Technology Corp.
Accusys	Accusys INC.
Acer	Acer, Inc.
ACL	Automated Cartridge Librarys, Inc.
Acuid	Acuid Corporation Ltd.
AcuLab	AcuLab, Inc. (Tulsa, OK)
ADAPTEC	Adaptec (now PMC-Sierra)
ADIC	Advanced Digital Information Corporation
ADSI	Adaptive Data Systems, Inc. (a Western Digital subsidiary)
ADTX	ADTX Co., Ltd.
ADVA	ADVA Optical Networking AG
AEM	AEM Performance Electronics
AERONICS	Aeronics, Inc.
AGFA	AGFA
AIC	Advanced Industrial Computer, Inc.
AIPTEK	AIPTEK International Inc.
ALCOR	Alcor Micro, Corp.
AMCC	Applied Micro Circuits Corporation
AMCODYNE	Amcodyne
Amgeon	Amgeon LLC
AMI	American Megatrends, Inc.

Table E.1 — T10 vendor identification list (part 2 of 15)

ID	Organization
Amphenol	Amphenol
Amtl	Tenlon Technology Co.,Ltd
ANAMATIC	Anamartic Limited (England)
Ancor	Ancor Communications, Inc.
ANCOT	ANCOT Corp.
ANDATACO	Andataco (now nStor)
andiamo	Andiamo Systems, Inc.
ANOBIT	Anobit
ANRITSU	Anritsu Corporation
ANTONIO	Antonio Precise Products Manufactory Ltd.
AoT	Art of Technology AG
APPLE	Apple Computer, Inc.
ARCHIVE	Archive
ARDENCE	Ardence Inc
Areca	Areca Technology Corporation
Arena	MaxTronic International Co., Ltd.
ARIO	Ario Data Networks, Inc.
ARISTOS	Aristos Logic Corp. (now part of Adaptec)
ARK	ARK Research Corporation
ARTECON	Artecon Inc. (Obs. - now Dot Hill)
Artistic	Artistic Licence (UK) Ltd
ARTON	Arton Int.
ASACA	ASACA Corp.
ASC	Advanced Storage Concepts, Inc.
ASPEN	Aspen Peripherals
AST	AST Research
ASTEK	Astek Corporation
ASTK	Alcatel STK A/S
ASTUTE	Astute Networks, Inc.
AT&T	AT&T
ATA	SCSI / ATA Translator Software (Organization Not Specified)
ATARI	Atari Corporation
ATG CYG	ATG Cygnet Inc.
ATL	Quantum|ATL Products
ATTO	ATTO Technology Inc.
ATX	Alphatronix
AURASEN	Aurasen Limited
AVC	AVC Technology Ltd
AVIDVIDR	AVID Technologies, Inc.
AVR	Advanced Vision Research
AXSTOR	AXSTOR
BALLARD	Ballard Synergy Corp.
Barco	Barco
BAROMTEC	Barom Technologies Co., Ltd.
Bassett	Bassett Electronic Systems Ltd
BDT	BDT AG
BECEEM	Beceem Communications, Inc

Table E.1 — T10 vendor identification list (part 3 of 15)

ID	Organization
BENQ	BENQ Corporation.
BERGSWD	Berg Software Design
BEZIER	Bezier Systems, Inc.
BHTi	Breece Hill Technologies
BIOS	BIOS Corporation
BIR	Bio-Imaging Research, Inc.
BiT	BiT Microsystems (obsolete, new ID: BITMICRO)
BITMICRO	BiT Microsystems, Inc.
BLOOMBAS	Bloombase Technologies Limited
BlueArc	BlueArc Corporation
BNCHMARK	Benchmark Tape Systems Corporation
Bosch	Robert Bosch GmbH
BoxHill	Box Hill Systems Corporation (Obs. - now Dot Hill)
BRDGWRKS	Bridgeworks Ltd.
BREA	BREA Technologies, Inc.
BREECE	Breece Hill LLC
Broadcom	Broadcom Corporation
BROCADE	Brocade Communications Systems, Incorporated
BUFFALO	BUFFALO INC.
BULL	Bull Peripherals Corp.
BUSLOGIC	BusLogic Inc.
BVIRTUAL	B-Virtual N.V.
CalComp	CalComp, A Lockheed Company
CALCULEX	CALCULEX, Inc.
CALIPER	Caliper (California Peripheral Corp.)
CAMBEX	Cambex Corporation
CAMEOSYS	Cameo Systems Inc.
CANDERA	Candera Inc.
CAPTION	CAPTION BANK
CAST	Advanced Storage Tech
CATALYST	Catalyst Enterprises
CCDISK	iSCSI Cake
CDC	Control Data or MPI
CDP	Columbia Data Products
Celsia	A M Bromley Limited
CenData	Central Data Corporation
Cereva	Cereva Networks Inc.
CERTANCE	Certance
CHEROKEE	Cherokee Data Systems
CHINON	Chinon
CHRISTMA	Christmann Informationstechnik + Medien GmbH & Co KG
CIE&YED	YE Data, C.Itoh Electric Corp.
CIPHER	Cipher Data Products
Ciprico	Ciprico, Inc.
CIRRUSL	Cirrus Logic Inc.
CISCO	Cisco Systems, Inc.
CLOVERLF	Cloverleaf Communications, Inc

Table E.1 — T10 vendor identification list (part 4 of 15)

ID	Organization
CLS	Celestica
CMD	CMD Technology Inc.
CMTechno	CMTech
CNGR SFW	Congruent Software, Inc.
CNSi	Chaparral Network Storage, Inc.
CNT	Computer Network Technology
COBY	Coby Electronics Corporation, USA
COGITO	Cogito
COMPAQ	Compaq Computer Corporation (now HP)
COMPELNT	Compellent Technologies, Inc. (now Dell)
COMPORT	Comport Corp.
COMPSIG	Computer Signal Corporation
COMPTEX	Comptex Pty Limited
CONNER	Conner Peripherals
COPANSYS	COPAN SYSTEMS INC
CORAID	Coraid, Inc
CORE	Core International, Inc.
COVOTE	Covote GmbH & Co KG
COWON	COWON SYSTEMS, Inc.
CPL	Cross Products Ltd
CPU TECH	CPU Technology, Inc.
CREO	Creo Products Inc.
CROSFLD	Crosfield Electronics (now FujiFilm Electonic Imaging Ltd)
CROSSRDS	Crossroads Systems, Inc.
crosswlk	Crosswalk, Inc.
CSCOVRTS	Cisco - Veritas
CSM, INC	Computer SM, Inc.
Cunuqui	CUNUQUI SLU
CYBERNET	Cybernetics
Cygnal	Dekimo
D Bit	Digby’s Bitpile, Inc. DBA D Bit
DALSEMI	Dallas Semiconductor
DANEELEC	Dane-Elec
DANGER	Danger Inc.
DAT-MG	DAT Manufacturers Group
Data Com	Data Com Information Systems Pty. Ltd.
DATABOOK	Databook, Inc.
DATACOPY	Datacopy Corp.
DataCore	DataCore Software Corporation
DATAPT	Datapoint Corp.
DATARAM	Dataram Corporation
DAVIS	Daviscomms (S) Pte Ltd
DDN	DataDirect Networks, Inc.
DDRDRIVE	DDRdrive LLC
DEC	Digital Equipment Corporation (now HP)
DEI	Digital Engineering, Inc.
DELL	Dell, Inc.

Table E.1 — T10 vendor identification list (part 5 of 15)

ID	Organization
DELPHI	Delphi Data Div. of Sparks Industries, Inc.
DENON	Denon/Nippon Columbia
DenOptix	DenOptix, Inc.
DEST	DEST Corp.
DFC	DavioFranke.com
DGC	Data General Corp.
DIGIDATA	Digi-Data Corporation
DigiIntl	Digi International
Digital	Digital Equipment Corporation (now HP)
DILOG	Distributed Logic Corp.
DISC	Document Imaging Systems Corp.
DLNET	Driveline
DNS	Data and Network Security
DotHill	Dot Hill Systems Corp.
DP	Dell, Inc.
DPT	Distributed Processing Technology
DROBO	Data Robotics, Inc.
DSC	DigitalStream Corporation
DSI	Data Spectrum, Inc.
DSM	Deterner Steuerungs- und Maschinenbau GmbH & Co.
DSNET	Cleversafe, Inc.
DT	Double-Take Software, INC.
DTC QUME	Data Technology Qume
DXIMAGIN	DX Imaging
EARTHLAB	EarthLabs
ECCS	ECCS, Inc.
ECMA	European Computer Manufacturers Association
ELE Intl	ELE International
ELEGANT	Elegant Invention, LLC
elipsan	Elipsan UK Ltd.
Elms	Elms Systems Corporation
ELSE	ELSE Ltd.
EMASS	EMASS, Inc.
EMC	EMC Corp.
EMiT	EMiT Conception Eletronique
EMTEC	EMTEC Magnetics
EMULEX	Emulex
ENERGY-B	Energybeam Corporation
ENGENIO	Engenio Information Technologies, Inc.
EPOS	EPOS Technologies Ltd.
EPSON	Epson
EQLOGIC	EqualLogic
Eris/RSI	RSI Systems, Inc.
ETERNE	EterneData Technology Co.,Ltd.(China PRC.)
EuroLogc	Eurologic Systems Limited (now part of Adaptec)
evolve	Evolution Technologies, Inc
EXABYTE	Exabyte Corp.

Table E.1 — T10 vendor identification list (part 6 of 15)

ID	Organization
EXATEL	Exatelecom Co., Ltd.
EXAVIO	Exavio, Inc.
Exsequi	Exsequi Ltd
Exxotest	Annecy Electronique
FALCON	FalconStor, Inc.
FFEILTD	FujiFilm Electonic Imaging Ltd
Fibxn	Fiberxon, Inc.
FID	First International Digital, Inc.
FILENET	FileNet Corp.
FirmFact	Firmware Factory Ltd
FLYFISH	Flyfish Technologies
FOXCONN	Foxconn Technology Group
FRAMDRV	FRAMEDRIVE Corp.
FREECION	Nable Communications, Inc.
FSC	Fujitsu Siemens Computers
FTPL	Frontline Technologies Pte Ltd
FUJI	Fuji Electric Co., Ltd. (Japan)
FUJIFILM	Fuji Photo Film, Co., Ltd.
FUJITSU	Fujitsu
FUNAI	Funai Electric Co., Ltd.
FUTURED	Future Domain Corp.
G&D	Giesecke & Devrient GmbH
Gadzoox	Gadzoox Networks, Inc. (now part of Broadcom)
Gammaflx	Gammaflux L.P.
GDI	Generic Distribution International
GEMALTO	gemalto
Gen_Dyn	General Dynamics
Generic	Generic Technology Co., Ltd.
GENSIG	General Signal Networks
GIGATAPE	GIGATAPE GmbH
GIGATRND	GigaTrend Incorporated
Global	Global Memory Test Consortium
Gnutek	Gnutek Ltd.
Goidelic	Goidelic Precision, Inc.
GoldStar	LG Electronics Inc.
GORDIUS	Gordius
GOULD	Gould
HAGIWARA	Hagiwara Sys-Com Co., Ltd.
HAPP3	Inventec Multimedia and Telecom co., ltd
HDS	Horizon Data Systems, Inc.
Heydays	Mazo Technology Co., Ltd.
HI-TECH	HI-TECH Software Pty. Ltd.
HITACHI	Hitachi America Ltd or Nissei Sangyo America Ltd
HL-DT-ST	Hitachi-LG Data Storage, Inc.
HONEYWEL	Honeywell Inc.
Hoptroff	HexWax Ltd
HORIZONT	Horizontigo Software

Table E.1 — T10 vendor identification list (part 7 of 15)

ID	Organization
HP	Hewlett Packard
HPQ	Hewlett Packard
HUASY	Huawei Symantec Technologies Co., Ltd.
HYUNWON	HYUNWON inc
i-cubed	i-cubed ltd.
IBM	International Business Machines
Icefield	Icefield Tools Corporation
ICL	ICL
ICP	ICP vortex Computersysteme GmbH
IDE	International Data Engineering, Inc.
IDG	Interface Design Group
IET	ISCSI ENTERPRISE TARGET
IFT	Infortrend Technology, Inc.
IGR	Intergraph Corp.
IMATION	Imation
IMPLTD	Integrated Micro Products Ltd.
IMPRIMIS	Imprimis Technology Inc.
INCIPNT	Incipient Technologies Inc.
INCITS	InterNational Committee for Information Technology
INDCOMP	Industrial Computing Limited
Indigita	Indigita Corporation
INITIO	Initio Corporation
INRANGE	INRANGE Technologies Corporation
Insight	L-3 Insight Technology Inc
INSITE	Insite Peripherals
integrix	Integrix, Inc.
INTEL	Intel Corporation
Intransa	Intransa, Inc.
IOC	I/O Concepts, Inc.
iofy	iofy Corporation
IOMEGA	Iomega
IOT	IO Turbine, Inc.
iqstor	iQstor Networks, Inc.
iQue	iQue
ISi	Information Storage inc.
Isilon	Isilon Systems, Inc.
ISO	International Standards Organization
iStor	iStor Networks, Inc.
ITC	International Tapetronics Corporation
iTwin	iTwin Pte Ltd
IVIVITY	iVivity, Inc.
IVMMLTD	InnoVISION Multimedia Ltd.
JABIL001	Jabil Circuit
JETWAY	Jetway Information Co., Ltd
JOLLYLOG	Jolly Logic
JPC Inc.	JPC Inc.
JSCSI	jSCSI Project

Table E.1 — T10 vendor identification list (part 8 of 15)

ID	Organization
Juniper	Juniper Networks
JVC	JVC Information Products Co.
KASHYA	Kashya, Inc.
KENNEDY	Kennedy Company
KENWOOD	KENWOOD Corporation
KEWL	Shanghai KEWL Imp&Exp Co., Ltd.
Key Tech	Key Technologies, Inc
KMNRIO	Kaminario Technologies Ltd.
KODAK	Eastman Kodak
KONAN	Konan
koncepts	koncepts International Ltd.
KONICA	Konica Japan
KOVE	KOVE
KSCOM	KSCOM Co. Ltd.,
KUDELSKI	Nagravision SA - Kudelski Group
Kyocera	Kyocera Corporation
Lapida	Gonmalo Electronics
LAPINE	Lapine Technology
LASERDRV	LaserDrive Limited
LASERGR	Lasergraphics, Inc.
LeapFrog	LeapFrog Enterprises, Inc.
LEFTHAND	LeftHand Networks (now HP)
Leica	Leica Camera AG
Lexar	Lexar Media, Inc.
LEYIO	LEYIO
LG	LG Electronics Inc.
LGE	LG Electronics Inc.
LION	Lion Optics Corporation
LMS	Laser Magnetic Storage International Company
LoupTech	Loup Technologies, Inc.
LSI	LSI Corp. (was LSI Logic Corp.)
LSILOGIC	LSI Logic Storage Systems, Inc.
LTO-CVE	Linear Tape - Open, Compliance Verification Entity
LUXPRO	Luxpro Corporation
Malakite	Malachite Technologies (New VID is: Sandial)
MarcBoon	marcboon.com
Marner	Marner Storage Technologies, Inc.
MARVELL	Marvell Semiconductor, Inc.
MATSHITA	Matsushita
MAXELL	Hitachi Maxell, Ltd.
MAXIM-IC	Maxim Integrated Products
MaxOptix	Maxoptix Corp.
MAXSTRAT	Maximum Strategy, Inc.
MAXTOR	Maxtor Corp.
MaXXan	MaXXan Systems, Inc.
MAYCOM	maycom Co., Ltd.
MBEAT	K-WON C&C Co.,Ltd

Table E.1 — T10 vendor identification list (part 9 of 15)

ID	Organization
MCC	Measurement Computing Corporation
McDATA	McDATA Corporation
MCUBE	Mcube Technology Co., Ltd.
MDI	Micro Design International, Inc.
MEADE	Meade Instruments Corporation
mediamat	mediamatic
MEII	Mountain Engineering II, Inc.
MELA	Mitsubishi Electronics America
MELCO	Mitsubishi Electric (Japan)
mellanox	Mellanox Technologies Ltd.
MEMOREX	Memorex Telex Japan Ltd.
MEMREL	Memrel Corporation
MEMTECH	MemTech Technology
MendoCno	Mendocino Software
MERIDATA	Oy Meridata Finland Ltd
METHODEI	Methode Electronics India pvt ltd
METRUM	Metrum, Inc.
MHTL	Matsunichi Hi-Tech Limited
MICROBTX	Microbotics Inc.
Microchp	Microchip Technology, Inc.
MICROLIT	Microlite Corporation
MICRON	Micron Technology, Inc.
MICROP	Micropolis
MICROTEK	Microtek Storage Corp
Minitech	Minitech (UK) Limited
Minolta	Minolta Corporation
MINSCRIB	Miniscribe
MiraLink	MiraLink Corporation
Mirifica	Mirifica s.r.l.
MITSUMI	Mitsumi Electric Co., Ltd.
MKM	Mitsubishi Kagaku Media Co., LTD.
Mobii	Mobii Systems (Pty.) Ltd.
MOL	Petrosoft Sdn. Bhd.
MOSAID	Mosaid Technologies Inc.
MOTOROLA	Motorola
MP-400	Daiwa Manufacturing Limited
MPC	MPC Corporation
MPCCORP	MPC Computers
MPEYE	Touchstone Technology Co., Ltd
MPIO	DKT Co.,Ltd
MPM	Mitsubishi Paper Mills, Ltd.
MPMan	MPMan.com, Inc.
MSFT	Microsoft Corporation
MSI	Micro-Star International Corp.
MST	Morning Star Technologies, Inc.
MSystems	M-Systems Flash Disk Pioneers
MTI	MTI Technology Corporation

Table E.1 — T10 vendor identification list (part 10 of 15)

ID	Organization
MTNGATE	MountainGate Data Systems
MXI	Memory Experts International
nac	nac Image Technology Inc.
NAGRA	Nagravision SA - Kudelski Group
NAI	North Atlantic Industries
NAKAMICH	Nakamichi Corporation
NatInst	National Instruments
NatSemi	National Semiconductor Corp.
NCITS	InterNational Committee for Information Technology Standards (INCITS)
NCL	NCL America
NCR	NCR Corporation
Neartek	Neartek, Inc.
NEC	NEC
NETAPP	NetApp, Inc. (was Network Appliance)
NetBSD	The NetBSD Foundation
Netcom	Netcom Storage
NETENGIN	NetEngine, Inc.
NEWISYS	Newisys Data Storage
Newtech	Newtech Co., Ltd.
NEXSAN	Nexsan Technologies, Ltd.
NFINIDAT	Infinidat Ltd.
NHR	NH Research, Inc.
Nike	Nike, Inc.
Nimble	Nimble Storage
NISCA	NISCA Inc.
NISHAN	Nishan Systems Inc.
NKK	NKK Corp.
NRC	Nakamichi Research Corporation
NSD	Nippon Systems Development Co.,Ltd.
NSM	NSM Jukebox GmbH
nStor	nStor Technologies, Inc.
NT	Northern Telecom
NUCONNEX	NuConnex
NUSPEED	NuSpeed, Inc.
NVIDIA	NVIDIA Corporation
NVMe	NVM Express Working Group
OAI	Optical Access International
OCE	Oce Graphics
OHDEN	Ohden Co., Ltd.
OKI	OKI Electric Industry Co.,Ltd (Japan)
Olidata	Olidata S.p.A.
OMI	Optical Media International
OMNIFI	Rockford Corporation - Omnifi Media
OMNIS	OMNIS Company (FRANCE)
Ophidian	Ophidian Designs
opslag	Tyrone Systems
Optelec	Optelec BV

Table E.1 — T10 vendor identification list (part 11 of 15)

ID	Organization
Optiarc	Sony Optiarc Inc.
OPTIMEM	Cipher/Optimem
OPTOTECH	Optotech
ORACLE	Oracle Corporation
ORANGE	Orange Micro, Inc.
ORCA	Orca Technology
OSI	Optical Storage International
OSNEXUS	OS NEXUS, Inc.
OTL	OTL Engineering
OVERLAND	Overland Storage Inc.
pacdigit	Pacific Digital Corp
Packard	Parkard Bell
Panasas	Panasas, Inc.
PARALAN	Paralan Corporation
PASCOsci	Pasco Scientific
PATHLGHT	Pathlight Technology, Inc.
PerStor	Perstor
PERTEC	Pertec Peripherals Corporation
PFTI	Performance Technology Inc.
PFU	PFU Limited
PHILIPS	Philips Electronics
PICO	Packard Instrument Company
PIK	TECHNILIENT & MCS
Pillar	Pillar Data Systems
PIONEER	Pioneer Electronic Corp.
Pirus	Pirus Networks
PIVOT3	Pivot3, Inc.
PLASMON	Plasmon Data
Pliant	Pliant Technology, Inc.
PMCSIERA	PMC-Sierra
POKEN	Poken SA
POLYTRON	PT. HARTONO ISTANA TEKNOLOGI
PRAIRIE	PrairieTek
PREPRESS	PrePRESS Solutions
PRESOFT	PreSoft Architects
PRESTON	Preston Scientific
PRIAM	Priam
PRIMAGFX	Primagraphics Ltd
PROCOM	Procom Technology
PROMISE	PROMISE TECHNOLOGY, Inc
PROSTOR	ProStor Systems, Inc.
PROWARE	Proware Technology Corp.
PTI	Peripheral Technology Inc.
PTICO	Pacific Technology International
PURE	PURE Storage
QIC	Quarter-Inch Cartridge Drive Standards, Inc.
QLogic	QLogic Corporation

Table E.1 — T10 vendor identification list (part 12 of 15)

ID	Organization
Qsan	QSAN Technology, Inc.
QUALSTAR	Qualstar
QUANTEL	Quantel Ltd.
QUANTUM	Quantum Corp.
QUIX	Quix Computerware AG
R-BYTE	R-Byte, Inc.
RACALREC	Racal Recorders
RADITEC	Radikal Technologies Deutschland GmbH
RADSTONE	Radstone Technology
RASSYS	Rasilient Systems Inc.
RASVIA	Rasvia Systems, Inc.
rave-mp	Go Video
Readboy	Readboy Ltd Co.
Realm	Realm Systems
realtek	Realtek Semiconductor Corp.
RELDATA	RELDATA Inc
RENAGmbH	RENA GmbH
Revivio	Revivio, Inc.
RGI	Raster Graphics, Inc.
RHAPSODY	Rhapsody Networks, Inc.
RHS	Racal-Heim Systems GmbH
RICOH	Ricoh
RODIME	Rodime
Royaltek	RoyalTek company Ltd.
RPS	RPS
RTI	Reference Technology
S-D	Sauer-Danfoss
S-flex	Storageflex Inc
S-SYSTEM	S-SYSTEM
SAMSUNG	Samsung Electronics Co., Ltd.
SAN	Storage Area Networks, Ltd.
Sandial	Sandial Systems, Inc.
SanDisk	SanDisk Corporation
SANKYO	Sankyo Seiki
SANRAD	SANRAD Inc.
SANYO	SANYO Electric Co., Ltd.
SC.Net	StorageConnections.Net
SCALE	Scale Computing, Inc.
SCIENTEK	SCIENTEK CORP
SCInc.	Storage Concepts, Inc.
SCREEN	Dainippon Screen Mfg. Co., Ltd.
SDI	Storage Dimensions, Inc.
SDS	Solid Data Systems
SEAC	SeaChange International, Inc.
SEAGATE	Seagate
SEAGRAND	SEAGRAND In Japan
Seanodes	Seanodes

Table E.1 — T10 vendor identification list (part 13 of 15)

ID	Organization
Sec. Key	SecureKey Technologies Inc.
SEQUOIA	Sequoia Advanced Technologies, Inc.
Shinko	Shinko Electric Co., Ltd.
SIEMENS	Siemens
SigmaTel	SigmaTel, Inc.
SII	Seiko Instruments Inc.
SIMPLE	SimpleTech, Inc. (Obs - now STEC, Inc.)
SLCNSTOR	SiliconStor, Inc.
SLI	Sierra Logic, Inc.
SMS	Scientific Micro Systems/OMTI
SMSC	SMSC Storage, Inc.
SMX	Smartronix, Inc.
SNYSIDE	Sunnyside Computing Inc.
SoftLock	Softlock Digital Security Provider
SolidFir	SolidFire, Inc.
SONIC	Sonic Solutions
SoniqCas	SoniqCast
SONY	Sony Corporation Japan
SPD	Storage Products Distribution, Inc.
SPECIAL	Special Computing Co.
SPECTRA	Spectra Logic, a Division of Western Automation Labs, Inc.
SPERRY	Sperry (now Unisys Corp.)
Spintso	Spintso International AB
STARWIND	StarWind Software, Inc.
STEC	STEC, Inc.
Sterling	Sterling Diagnostic Imaging, Inc.
STK	Storage Technology Corporation
STNWOOD	Stonewood Group
STONEFLY	StoneFly Networks, Inc.
STOR	StorageNetworks, Inc.
STORAPP	StorageApps, Inc. (now HP)
STORCOMP	Storage Computer Corporation
STORM	Storm Technology, Inc.
StorMagc	StorMagic
Stratus	Stratus Technologies
StrmLgc	StreamLogic Corp.
SUMITOMO	Sumitomo Electric Industries, Ltd.
SUN	Sun Microsystems, Inc.
SUNCORP	SunCorporation
suntx	Suntx System Co., Ltd
Swinxs	Swinxs BV
SYMANTEC	Symantec Corporation
SYMBIOS	Symbios Logic Inc.
SYMWAVE	Symwave, Inc.
SYNCSORT	Syncsort Incorporated
SYNERWAY	Synerway
SyQuest	SyQuest Technology, Inc.

Table E.1 — T10 vendor identification list (part 14 of 15)

ID	Organization
SYSGEN	Sysgen
T-MITTON	Transmitton England
T-MOBILE	T-Mobile USA, Inc.
T11	INCITS Technical Committee T11
TALARIS	Talaris Systems, Inc.
TALLGRAS	Tallgrass Technologies
TANDBERG	Tandberg Data A/S
TANDEM	Tandem (now HP)
TANDON	Tandon
TCL	TCL Shenzhen ASIC MIcro-electronics Ltd
TDK	TDK Corporation
TEAC	TEAC Japan
TECOLOTE	Tecolote Designs
TEGRA	Tegra Varityper
Tek	Tektronix
TELLERT	Tellert Elektronik GmbH
TENTIME	Laura Technologies, Inc.
TFDATACO	TimeForge
TGEGROUP	TGE Group Co.,LTD.
Thecus	Thecus Technology Corp.
TI-DSG	Texas Instruments
TiGi	TiGi Corporation
TILDESGN	Tildesign bv
Tite	Tite Technology Limited
TMS	Texas Memory Systems, Inc.
TMS100	TechnoVas
TOLISGRP	The TOLIS Group
TOSHIBA	Toshiba Japan
TRIOFLEX	Trioflex Oy
TRIPACE	Tripace
TRLogger	TrueLogger Ltd.
TROIKA	Troika Networks, Inc.
TRULY	TRULY Electronics MFG. LTD.
TRUSTED	Trusted Data Corporation
TSSTcorp	Toshiba Samsung Storage Technology Corporation
TZM	TZ Medical
UDIGITAL	United Digital Limited
UIT	United Infomation Technology
ULTRA	UltraStor Corporation
UNISYS	Unisys
USCORE	Underscore, Inc.
USDC	US Design Corp.
VASCO	Vasco Data Security
VDS	Victor Data Systems Co., Ltd.
Verari	Verari Systems, Inc.
VERBATIM	Verbatim Corporation
Vercet	Vercet LLC

Table E.1 — T10 vendor identification list (part 15 of 15)

ID	Organization
VERITAS	VERITAS Software Corporation
VEXCEL	VEXCEL IMAGING GmbH
VicomSys	Vicom Systems, Inc.
VIDEXINC	Videx, Inc.
VIOLIN	Violin Memory, Inc.
VITESSE	Vitesse Semiconductor Corporation
VIXEL	Vixel Corporation (now part of Emulex)
VLS	Van Lent Systems BV
VMAX	VMAX Technologies Corp.
VMware	VMware Inc.
Vobis	Vobis Microcomputer AG
VOLTAIRE	Voltaire Ltd.
VRC	Vermont Research Corp.
VRugged	Vanguard Rugged Storage
Waitec	Waitec NV
WangDAT	WangDAT
WANGTEK	Wangtek
Wasabi	Wasabi Systems
WAVECOM	Wavecom
WD	Western Digital Corporation
WDC	Western Digital Corporation
WDIGTL	Western Digital
WEARNES	Wearnes Technology Corporation
WeeraRes	Weera Research Pte Ltd
Wildflwr	Wildflower Technologies, Inc.
WSC0001	Wisecom, Inc.
X3	InterNational Committee for Information Technology Standards (INCITS)
XEBEC	Xebec Corporation
XENSRC	XenSource, Inc.
Xerox	Xerox Corporation
XIOtech	XIOtech Corporation
XIRANET	Xiranet Communications GmbH
XIV	XIV (now IBM)
XtremIO	XtremIO
XYRATEX	Xyratex
YINHE	NUDT Computer Co.
YIXUN	Yixun Electronic Co.,Ltd.
YOTTA	YottaYotta, Inc.
Zarva	Zarva Digital Technology Co., Ltd.
ZETTA	Zetta Systems, Inc.