Working Draft American National Standard

Project T10/1417-D

Revision 15
22 July 2004

Information technology - SCSI Block Commands - 2 (SBC-2)

T10 Technical Editor: Robert C Elliott
Hewlett-Packard Corporation
MC 140801
PO Box 692000
Houston, TX 77269-2000
USA

Telephone: 281-518-5037
Email: elliott@hp.com

Reference number
ISO/IEC 14776-322:200x
ANSI INCITS.*:200x**

Printed 1:42 PM Thursday 22 July 2004

Points of contact

International Committee for Information Technology Standards (INCITS) T10 Technical Committee

T10 Chair
John B. Lohmeyer
LSI Logic
4420 Arrows West Drive
Colorado Springs, CO 80907-3444
USA

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

T10 Vice-Chair
George O. Penokie
IBM Corporation
MS: 2C6
3605 Highway 52 N
Rochester, MN 55901
USA

Telephone: (507) 253-5208
Email: gop@us.ibm.com

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

T10 E-mail reflector:
Server: majordomo@t10.org
To subscribe send e-mail with 'subscribe' in message body
To unsubscribe send e-mail with 'unsubscribe' in message body

INCITS Secretariat
Suite 200
1250 Eye Street, NW
Washington, DC 20005
USA

Telephone: 202-737-8888
Web site: http://www.incits.org
Email: incits@itic.org

Information Technology Industry Council
Web site: http://www.itic.org

Document Distribution
INCITS Online Store
managed by Techstreet
1327 Jones Drive
Ann Arbor, MI 48105
USA

Web site: http://www.techstreet.com/incits.html
Telephone: (734) 302-7801 or (800) 699-9277

Global Engineering Documents, an IHS Company
15 Inverness Way East
Englewood, CO 80112-5704
USA

Web site: http://global.ihs.com
Telephone: (303) 397-7956 or (303) 792-2181 or (800) 854-7179

American National Standard
for Information Technology

SCSI Block Commands - 2 (SBC-2)

Secretariat
Information Technology Industry Council

Approved mm.dd.yy

American National Standards Institute, Inc.

ABSTRACT

This standard specifies the functional requirements for the SCSI Block Commands - 2 (SBC-2) command set. SBC-2 permits SCSI block logical units such as rigid disks to attach to computers and provides the definition for their use.

This standard maintains a high degree of compatibility with the SCSI Block Commands (SBC) command set, NCITS.306:1998, and while providing additional functions, is not intended to require changes to presently installed devices or existing software.

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 towards their resolution.

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

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

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

CAUTION: The developers of this standard have requested that holders of patents that may be required for the implementation of the standard, disclose such patents to the publisher. However, neither the developers nor the publisher have undertaken a patent search in order to identify which, if any, patents may apply to this standard. As of the date of publication of this standard, following calls for the identification of patents that may be required for the implementation of the standard, no such claims have been made. 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 W. 42nd Street, New York, New York 10036

Printed in the United States of America

Dedication

This standard is dedicated to the memory of Gene E. Milligan, who was the original editor.

Mr. Milligan was a dedicated and energetic participant on several NCITS Technical Committees, including T10, T11, T12, and T13. He chaired both T12 and T13 and was the International Representative for T10, T11, and T12.

Mr. Milligan graduated in 1959 from UCLA with a degree in Electrical Engineering and was employed by Seagate Technology for over 30 years. His interests included flying, water and snow skiing, and tinkering with anything that needed repair. He was also an avid sports fan.

Memorial gifts may be made to Habitat for Humanity.

Revision Information

R.1 Revision 0 (5 July 2000)

a) Converted to ISO/IEC style.
b) Incorporated the following proposals:
   A) 98-202r1 - Obsolete Extent Reservations
   B) 98-202r1 - Obsolete Change Definition
   C) 98-203r9 - Persistent Reservation Changes
   D) 99-189r0 - ECC correction span spec for 255
   E) 99-259r4 - 2 Terabyte Changes. Note that the proposal had *, **, or *** on several commands but no explanation of the *, **, or ***.
   F) 00-125r0 - Large LBA address using variable length CDB structure

R.2 Revision 1 (27 August 2000)

a) Incorporated the following proposal:
   A) 99-258r2 - List lengths that exceed the maximum with wording changes approved at the 7/2000 meeting.
b) Added note after reservation conflict tables per prior editor's note and 7/2000 meeting.
c) Added "may not" to key words per 7/2000 meeting.

R.3 Revision 2 (4 October 2000)

a) Added missing operation code for XDREAD
b) Incorporated the following proposals:
   A) 00-248r2 - SBC-2 issues - Item 6 - Initialization Pattern using the least significant 4 bytes of the LBA
   B) 00-333r0 - SCSI is a functional standard
   C) 00-248r2 - SBC-2 issues - Editorial changes

R.4 Revision 3 (17 May 2001)

a) Added GEM dedication page.
b) Corrected spelling, cross reference, and formatting errors throughout the document.
c) Deleted about 12,000 extraneous spaces.
d) Reformatted tables to follow SPI-4 style.
e) Generated PDF with bookmarks enabled.
f) Changed Times-Roman font to Arial in the few places it was used.
g) Made small-caps use more consistent.
h) Updated front material based on SPI-3 revision 14.
i) Added hierarchy to annexes.
j) Fixed sense key/additional sense code mixup in Logical blocks section 4.2.1.3.
k) Fixed VERIFY (16) and WRITE SAME (16) (they only had 10 CDB bytes).
l) Moved all READ, VERIFY, WRITE, WRITE AND VERIFY commands into section 5.1. Added text explaining that the BLKVFY or EBP bits are considered reserved for direct-access devices. This addresses Gene's editorial note 1 asking whether optical VERIFY (16) and WRITE AND VERIFY (16) should point to optical-memory or direct-access versions of the CDBs.
m) Added notes in WRITE (6) and WRITE (10) about their different handling of transfer length of 0.
n) Added READ (16) and LOCK UNLOCK CACHE (16) to list of supported commands for optical and write-once devices, since the write commands were already added. The preface to 99-259r4 only requested new large LBA commands for direct-access devices, but provided reservation tables for all 3 types. SBC-2 revision 2 included most of them; it seems appropriate to allow the rest as well.
o) Changed PRE-FETCH to PRE-FETCH (10) since there is now a PRE-FETCH (16) too.
p) Merged all the reservation tables into one to avoid duplication of most of the rows (and potential conflicts).
q) Reconciled with SPC-2 revision 19:
   A) In variable length CDBs, changed ENCRYPTION IDENTIFICATION to Reserved to match SPC-2 revision 19.

B) Removed CHANGE DEFINITION, COMPARE, COPY, and COPY AND VERIFY references, since they are obsolete in SPC-2.
C) Removed the power condition mode page. SPC-2 marks page code 0Dh as obsolete, since it incorporated the entire page under code 1Ah per these proposals:

- 95-222r2 Power condition mode page code
- 95-265r1 T10 plenary minutes July 1995 item 10.6 Power Condition mode page

r) Incorporated the following proposals:
A) 00-315r1 - Bidirectional XDWRITEREAD command for SBC-2
B) 00-395r1 - Increased defect list lengths for SBC-2
C) 00-375r1 - November 2000 T10 plenary minutes - motion 10.4.4: “READ (16) and WRITE (16) [shall] be made mandatory for the direct-access device type”. Noted that WRITE (16) is only mandatory if any WRITE command is implemented.

R.5 Revision 4 (28 July 2001)

a) Obsoleted 0Dh and moved Power Condition mode page to 1Ah in optical drives table 122.
b) Corrected opcodes in READ (16) and WRITE SAME (16)
c) Incorporated the following proposals:
A) 00-425r4 Long Identifiers in SPC-3, SAM-2, SBC-2 and other XOR issues
B) 01-134r2 WAKEUP and RESET cleanup
C) 01-210r0 Reassign Blocks 2 TB support

R.6 Revision 5 (23 February 2002)

a) 01-246 Long LBA PMI support for Read Capacity
b) 01-199 Sense Data INFORMATION field for long LBAs and bidirectional commands
c) New text about security initialize for FORMAT UNIT per email from Eugene Zilberman of M-Systems

R.7 Revision 5a (2 March 2002)

a) made mode page, log page, and diagnostic page references use consistent naming convention and capitalization
b) upgraded references to other standards
c) definition and keyword sections reformatted
d) keywords and conventions section rewritten
e) updated log page and mode page list to match SPC-3
f) no change bars

R.8 Revision 6 (1 May 2002)

a) fixed hanging paragraphs in 5.1 and some table formats
b) 02-130r0 SBC-2, SSC-2, SMC-2 & OSD Support for All Registrants Persistent Reservations
c) Updated supported commands tables to match SPC-3 revision 6 opcode table. Added SCC-2 commands to each device type; corrected SMC-2 command names; added new SPC-2/SPC-3 commands to each device type; included obsolete command names in notes.

R.9 Revision 7 (28 June 2002)

a) changed footnotes to use superscript letters
b) 02-277r1 obsolete RESERVE and RELEASE (Dave Peterson)[02-273 7/2002 T10 Plenary]
c) 02-260r1 Mandatory REPORT LUNS support (Dave Peterson)[02-273 7/2002 T10 Plenary]

R.10 Revision 8 (30 September 2002)

Incorporated the following proposals:

a) removed LONGLBA bit from READ CAPACITY (10) table
b) 02-232r2 Clearing effects of I_T nexus loss (Rob Elliott) [02-344 9/2002 T10 Plenary]
c) Added table summarizing SERVICE ACTION IN service actions (READ CAPACITY (16).

d) Corrected Type column entries in FORMAT UNIT defect descriptor formats table from 000b to O (optional) to match SCSI-2.
e) Per 02-346r1 and spc3r09, added REPORT SUPPORTED TASK MANAGEMENT FUNCTIONS as an optional command
f) Per 02-189r1 and spc3r09, expanded range of diagnostic page codes for SES to 00-1Fh from 00-0Fh.

R.11 Revision 9 (31 May 2003)

Incorporated the following changes:

a) miscellaneous editorial corrections from George Penokie (IBM) and Jim Hafner (IBM)
b) converted from Microsoft Word into Adobe FrameMaker. Redrew all figures in Microsoft Visio.
c) 02-464r3 SAM-3 SPC-3 SBC-2 Power conditions updates (Mark Evans) [03-179 5/2003 T10 Plenary]
d) 03-028r1 SBC-2 Block Limits mode page [03-048 1/2003 T10 Plenary]
e) Marked certain fields “restricted for MMC-4” in WRITE (10), VERIFY (10), and the Read-Write Error Recovery mode page (byte 7 bits 1:0) based on MMC-4 revision 2. MMC-4 adds some streaming bits in those locations in its versions of those commands. They might be useful for disk drives in the future.

R.12 Revision 10 (13 September 2003)

Incorporated the following changes:

a) Corrected header levels for power conditions model section
b) 03-243r2 Optional INQUIRY processing before entering task set (George Penokie, IBM)
c) Correct byte numbers in REBUILD (32), XDWRITE EXTENDED (64), and XPWRITE (32)

R.13 Revision 11 (8 December 2003)

Incorporated the following changes per the November CAP WG (03-370), November T10 plenary (03-372), and November editor’s meeting:

a) 03-335r1 SBC-2 SPC-3 Obsolete block device linked commands (Rob Elliott, HP)
b) 03-365r1 End-to-end data protection (George Penokie, IBM) (really r11 of 03-176)
c) 03-362r0 SBC-2 Obsolete SEEK (10) (Rob Elliott, HP)
d) 03-383r1 SBC-2 new READ LONG (16) and WRITE LONG (16) commands (Mark Evans, Maxtor)
e) Miscellaneous changes from November editors meeting
f) Changed “block size” to “block length”
g) Changed “data block” and “block” to “logical block” or “physical block” as appropriate. Mention that logical block consists of both user data and protection information many places.
h) Changed “direct access” to “direct-access”, “write once” to “write-once”, and “optical memory” to “optical-memory”
i) Changed “direct-access block device” to “direct-access device”, “write-once block device” to “write-once device”, and “optical-memory block device” to “optical-memory device”
j) Added some definitions from SSC-2 (e.g. application client, additional sense code, byte, ...)
k) Changed “logical block address” to “LBA” except where used as a field name
l) Changed “command descriptor block” to “CDB”
m) Changed “sense data to ILLEGAL REQUEST” constructs to “sense key to ILLEGAL REQUEST”
n) Restructured the FORMAT UNIT defect descriptor format and requirements table
o) Made the defect descriptor names consistent: short block, long block, bytes from index, physical sector
p) Mention all the commands that use defect descriptors in the defect descriptor definition
q) Changed some specfies/indicates usage to follow T10 standard (application client specifies, device server indicates)
r) Cleaned up footer in optical-memory density codes table

R.14 Revision 12 (25 January 2004)

Incorporated the following changes per the January CAP WG (04-038r0) and plenary (04-039r0):

a) 03-387r2 SBC-2 Data protection usage detection (Rob Elliott, HP)
b) 04-016r2 SBC-2 SPC-3 Obsolete Third Party XOR Commands (Jim Coomes, Seagate)
c) 04-013r1 SBC-2 List of functions to obsolete (George Penokie, IBM). Obsoleted all the functional fields in the Device Input Status diagnostic page (synchronization, rotational position locking, etc.), just leaving the vendor-specific fields. Deleted the direct-access medium type codes table (for the mode parameter header), just requiring the medium type field be set to 00h.
d) 04-024r1 SBC-2 Data protection information for XPWRITE (Jim Coomes, Seagate)
e) 04-025r0 SPC-3 and SBC-2 power conditions clarifications (Mark Evans, Maxtor)
f) Changed PLIST, GLIST, DLIST, and CLIST acronyms from small caps to all caps. Renamed PLIST and GLIST fields in READ DEFECT DATA CDBs to REQ_PLIST and REQ_LIST. Renamed PLIST and GLIST fields in the read defect data to PLISTV and GLISTV.
g) Made specifies/indicates usage consistent throughout chapter 5.
h) Lots of changes from the January editor's meeting.

R.15 Revision 13 (20 March 2004)

Incorporated the following changes per the March CAP WG (04-084r0) and plenary (04-085r0):

a) 04-012r1 SBC-2 WRITE SAME corrections (George Penokie, IBM)
b) 03-348r2 SBC-2 4-byte LBA commands on 8-byte LBA capable drives (Rob Elliott, HP)
a) 04-076r0 SBC-2 PRE-FETCH and errors (Rob Elliott, HP)
b) Changed block defect descriptor paragraphs which were made more broken, not less, in revision 12 (please review)
c) Miscellaneous editorial changes

R.16 Revision 14 (11 May 2004)

Incorporated the following changes per the May CAP WG (04-134r0) and plenary (04-135r0):

a) 03-361r4 Command classification field (George Penokie, IBM)
b) 03-388r3 SBC-2 Nonvolatile caches (Rob Elliott, HP)
c) 03-307r7 SBC-2 32-byte commands for end-to-end data protection (Jim Coomes, Seagate)
d) 04-075r2 SBC-2 Obsolete more features (Rob Elliott, HP)
e) 04-082r1 SBC-2 Obsolete Notch and Partition mode page (Rob Elliott, HP)
f) 04-111r0 SBC-2 Protection information checking within service delivery subsystem (George Penokie, IBM)
g) 04-114r0 SBC-2 Option to check only the Logical Block Guard (Keith Holt, LSI Logic)
h) Moved the short LBA direct-access block descriptor from SPC-3 into SBC-2 (04-011 will do this in more detail and also move the long LBA block descriptor)
i) Incorporated the proposed paragraph from Jim Coomes (Seagate) in REASSIGN BLOCKS that has been sitting as an editor's note for the past few revisions
j) Marked the BLKVFY bit in VERIFY (12) and VERIFY (16) as reserved (which it has always been for direct-access devices)

R.17 Revision 15 (22 July 2004)

Incorporated the following changes per the July CAP WG (04-215r0) and plenary (04-216r0):

a) Mark log page codes 09h and 0Ah as restricted per SPC-3
b) 04-011r5 SBC-2 Changing logical block sizes (George Penokie, IBM)
c) 04-031r3 SPC-3 SES-2 SBC-2 Miscellaneous diagnostic page topics (Rob Elliott, HP)
d) 04-164r1 SBC-2 Defect descriptor wording corrections (Rob Elliott, HP)
e) 04-169r3 SBC-2 Protection information fixes (George Penokie, IBM)
f) 04-171r1 SBC-2 Make FORMAT UNIT operating modes optional (Rob Elliott, HP)
g) 04-176r0 SBC-2 New CRC figure and example C code (Rob Elliott, HP)
h) 04-184r1 SBC-2 Read Long clarification (Gerry Houlder, Seagate)

i) 04-192r0 SBC-2 Commands during format operation (Rob Elliott, HP)
j) 04-215r0 CAP WG motion: Change the Non-volatile Cache log page code to 17h (Ralph Weber, ENDL)
k) Changed references to APP_TAG_OWN bit in the Control mode page to ATO bit
l) Updated command, log page, and mode page tables to match SPC-3 revision 19

This revision is being submitted to T10 letter ballot in July 2004.

Contents

Tables

Page

Figures

Page

Foreword (This foreword is not part of this standard)

Requests for interpretation, suggestions for improvement and addenda, or defect reports are welcome. They should be sent to the INCITS Secretariat, International Committee for Information Technology Standards, Information Technology Institute, 1250 Eye Street, NW, Suite 200, 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 it approved this standard, INCITS had the following members:

Karen Higginbottom, Chair

David Michael, Vice-Chair

INCITS Technical Committee T10 on Lower Level Interfaces, which developed and reviewed this standard, had the following members:

John B. Lohmeyer, Chair

George O. Penokie, Vice-Chair

Ralph O. Weber, Secretary

Introduction

The standard is organized as follows:

Clause 1 (Scope) describes the relationship of this standard to the SCSI family of standards.
Clause 2 (Normative References) provides references to other standards and documents.
Clause 3 (Definitions, symbols, abbreviations, keywords, and conventions) defines terms and conventions used throughout this standard.
Clause 4 (Models) provides an overview of the block device class and the command set.
Clause 5 (Commands for direct-access block devices) defines commands specific to block devices.
Clause 6 (Parameters for direct-access block devices) defines diagnostic pages, mode parameters and pages, log pages, and VPD pages specific to block devices.

Informative Annex A (XOR command examples) provides examples of XOR command usage.
Informative Annex B (CRC example in C) provides example C code for the data protection CRC.

American National Standard
for Information Technology -

SCSI Block Commands - 2 (SBC-2)

1 Scope

This standard defines the command set extensions to facilitate operation of SCSI block devices. The clauses of this standard pertaining to the SCSI block device class, implemented in conjunction with the applicable clauses of SPC-3, fully specify the standard command set for SCSI block devices.

The objective of this standard is to provide the following:

a) Permit an application client to communicate over a SCSI service delivery subsystem with a logical unit that declares itself to be a direct-access device in the PERIPHERAL DEVICE TYPE field of the standard INQUIRY data (see SPC-3);
b) Define commands unique to the type of SCSI block device;
c) Define commands to manage the operation of SCSI block devices; and
d) Define the differences between types of SCSI block devices.

Figure 1 shows the relationship of this standard to the other standards and related projects in the SCSI family of standards.

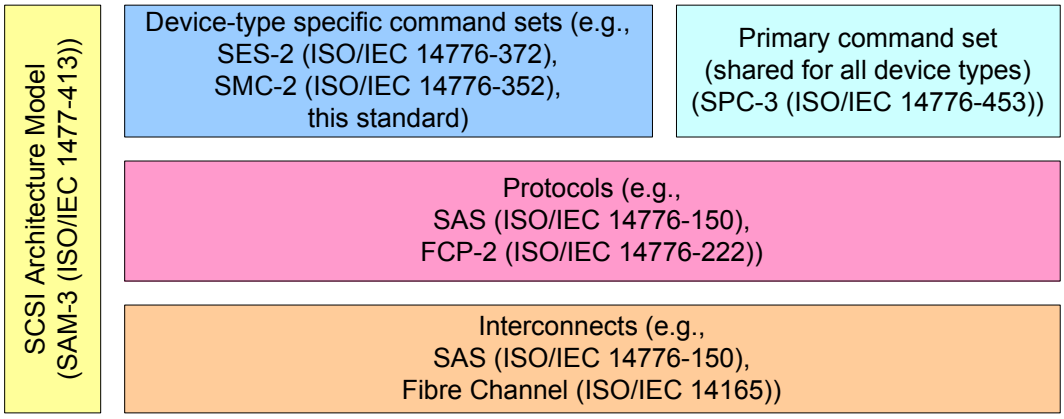


Figure 1 — SCSI document relationships

Figure 1 is intended to show the general relationship of the documents to one another, and is not intended to imply a relationship such as a hierarchy, protocol stack or system architecture. It indicates the applicability of a standard to the implementation of a given transport.

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 makes obsolete the following concepts from previous standards:

a) relative addressing (including the RELADR bit in many CDBs) and the SET LIMITS commands;
b) the CHANGE DEFINITION, COMPARE, COPY, COPY AND VERIFY, RESERVE, RELEASE, REZERO UNIT, SEEK, SEARCH DATA HIGH, SEARCH DATA EQUAL, and SEARCH DATA LOW commands;
c) third-party and hybrid XOR commands (REBUILD, REGENERATE, and XDWRITE EXTENDED);
d) the optical-memory device type, model, commands (ERASE, MEDIUM SCAN, READ GENERATION, READ UPDATED BLOCK, and UPDATE BLOCK), and parameters (Optical-Memory mode page);
e) the erase by-pass (EBP) bit in the WRITE and WRITE AND VERIFY commands (this bit was formerly reserved for direct-access device types, so is just marked reserved in this standard);

f) the write-once device type, model, commands, and parameters;
g) the Flexible Disk mode page;
h) the Format Device mode page;
i) the Medium Types Supported mode page and all medium types in the mode parameter header
j) the Rigid Disk Geometry mode page;
k) these Read-Write Errory Recovery mode page fields: CORRECTION SPAN, HEAD OFFSET COUNT, and DATA STROBE OFFSET COUNT;
l) this Verify Error Recovery mode page field: VERIFY CORRECTION SPAN;
m) rotational position locking model and Device Input Status diagnostic page fields relating to synchronization and rotational position locking;
n) the sequential media model;
o) the INTERLEAVE field in the FORMAT UNIT command;
p) the DISABLE SAVING PARAMETERS (DSP) bit in the FORMAT UNIT parameter list;
q) the Device Status Output and Device Status Input diagnostic pages; and
r) the Notch and Partition mode page.

2 Normative References

2.1 Normative references overview

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

Copies of the following documents may be obtained from ANSI:

a) approved ANSI standards;
b) approved and draft international and regional standards (ISO, IEC, CEN/CENELEC, ITUT); and
c) approved and draft foreign standards (including BSI, JIS, and DIN).

For further information, contact ANSI Customer Service Department at 212-642-4900 (phone), 212-302-1286 (fax) or via the World Wide Web at http://www.ansi.org.

Additional availability contact information is provided below as needed.

Table 1 lists standards bodies and their web sites.

Table 1 — Standards bodies

Abbreviation	Standards body	Web site
ANSI	American National Standards Institute	http://www.ansi.org
BSI	British Standards Institution	http://www.bsi-global.com
CEN	European Committee for Standardization	http://www.cenorm.be
CENELEC	European Committee for Electrotechnical Standardization	http://www.cenelec.org
DIN	German Institute for Standardization	http://www.din.de
IEC	International Engineering Consortium	http://www.iec.ch
IEEE	Institute of Electrical and Electronics Engineers	http://www.ieee.org
INCITS	International Committee for Information Technology Standards	http://www.incits.org
ISO	International Standards Organization	http://www.iso.ch
ITI	Information Technology Industry Council	http://www.itic.org
ITUT	International Telecommunications Union Telecommunications Standardization Sector	http://www.itu.int
JIS	Japanese Industrial Standards Committee	http://www.jisc.org
T10	INCITS T10 Committee - SCSI storage interfaces	http://www.t10.org
T11	INCITS T11 Committee - Fibre Channel interfaces	http://www.t11.org
T13	INCITS T13 Committee - ATA storage interface	http://www.t13.org

2.2 Approved references

At the time of publication, the following referenced standards were approved.

ISO/IEC 14776-342 Information technology - SCSI-3 Controller Commands - 2 (SCC-2)(ANSI NCITS.318:1998)

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 documents, or regarding availability, contact the relevant standards body as indicated.

ISO/IEC 14776-413, SCSI Architecture Model - 3 (SAM-3) standard (T10/1561-D)
ISO/IEC 14776-453, SCSI Primary Commands - 3 (SPC-3) standard (T10/1416-D)
ISO/IEC 14776-352, SCSI Media Changer Commands - 2 (SMC-2) standard (T10/1383-D)
ISO/IEC 14776-364, SCSI Multimedia Commands - 4 (MMC-4) standard (T10/1545-D)
ISO/IEC 14776-372, SCSI Enclosure Services - 2 (SES-2) standard (T10/1559-D)

NOTE 1 - For more information on the current status of the document, contact the INCITS Secretariat at 202-737-8888 (telephone), 202-638-4922 (fax) or via Email at incits@itic.org. To obtain copies of this document, contact Global Engineering at 15 Inverness Way East Englewood, CO 80112-5704 at 800-854-7179 (telephone), 303-792-2181 (telephone), or 303-792-2192 (fax).

3 Definitions, symbols, abbreviations, keywords, and conventions

3.1 Definitions

3.1.1 additional sense code: A combination of the ADDITIONAL SENSE CODE and ADDITIONAL SENSE CODE QUALIFIER fields in the sense data. See SPC-3.

3.1.2 application client: An object that is the source of SCSI commands. See SAM-3.

3.1.3 block device: A device that is capable of containing data stored in blocks that each have a unique logical block address.

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

3.1.5 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.

3.1.6 check data: Information contained within a redundancy group that allows lost or destroyed user data to be recreated.

3.1.7 command: A request describing a unit of work to be performed by a device server. See SAM-3.

3.1.8 command descriptor block (CDB): The structure used to communicate commands from an application client to a device server.See SPC-3.

3.1.9 data defect list (DLIST): A list of defects sent by the application client to the device server during a FORMAT UNIT command. See 4.8.

3.1.10 data-in buffer: The buffer identified by the application client to receive data from the device server during the processing of a command.

3.1.11 data-out buffer: The buffer identified by the application client to supply data that is sent from the application client to the device server during the processing of a command.

3.1.12 default protection information: Values placed into protection information fields if an application client does not specify specific protection information values.

3.1.13 device server: An object within a logical unit that processes SCSI tasks according to the rules of task management. See SAM-3.

3.1.14 device type: The type of device (or device model) implemented by the device server.

3.1.15 domain: An I/O system consisting of a set of SCSI devices that interact with one another by means of a service delivery subsystem.

3.1.16 exclusive-or (XOR): A logical function that combines two logical inputs producing a logical output true state if one but not both inputs are true. This function is used in error correction algorithms. In this standard the term encompasses the entire algorithm but does not define the specific polynomial. The exclusive-or operation may be performed by the storage array controller or by the storage device.

3.1.17 extent: A set of continuous logical blocks on a single logical unit.

3.1.18 field: A group of one or more contiguous bits, a part of a larger structure such as a CDB (see 3.1.8) or sense data (see SPC-3).

3.1.19 grown defect list (GLIST): All defects sent by the application client to the device server. See 4.8.

3.1.20 hard reset: A target action in response to a reset event in which the target port performs the operations described in SAM-3.

3.1.21 logical block: A set of data bytes accessed and referenced as a unit.

3.1.22 logical block address (LBA): The value used to reference a logical block.

3.1.23 logical unit certification list (CLIST): Defects detected by the device server during an optional certification process performed during the FORMAT UNIT command. See 4.8.

3.1.24 logical unit reset: A logical unit action in response to a logical unit reset event in which the logical unit performs the operations described in SAM-3.

3.1.25 logical unit reset event: An event that triggers a logical unit reset from a logical unit as described in SAM-3.

3.1.26 media: Plural of medium.

3.1.27 non-volatile cache memory: Cache memory that retains data through power cycles.

3.1.28 non-volatile medium: A physical storage medium that retains data written to it for a subsequent read operation through power off/on cycles. An example of this is a disk within a device that stores data as magnetic field changes that do not require device power to exist.

3.1.29 power cycle: Power being removed followed by power on.

3.1.30 power on: Power being applied.

3.1.31 primary defect list (PLIST): The list of defects that are considered permanent defects. See 4.8.

3.1.32 protection information: Fields appended to each logical block that contain a cyclic redundancy check (CRC), an application tag, and a reference tag.

3.1.33 read-only medium: Medium that is not capable of being changed. The medium contains data prepared in a manner not defined by this standard.

3.1.34 redundancy group: A grouping of protected space and associated check data into a single type of data redundancy (see SCC-2). This standard only supports the exclusive-or type of redundancy.

3.1.35 reset event: An event that triggers a hard reset from a SCSI device as described in the protocol standard. Reset events include power on and other protocol-specific events.

3.1.36 sense data: Data describing an error or exceptional condition that a device server delivers to an application client. See SPC-3.

3.1.37 sense key: The contents of the SENSE KEY field in the sense data. See SPC-3.

3.1.38 status: One byte of response information sent from a device server to an application client upon completion of each command. See SAM-3.

3.1.39 storage array controller: Any combination of an initiator and application clients (see SAM-3) that originates SCSI command descriptor blocks and performs the services of a SACL. A storage array controller organizes a group of storage devices into various objects (e.g., redundancy groups and volume sets). See SCC-2.

3.1.40 storage array conversion layer (SACL): Converts input logical unit numbers to output logical unit numbers and may convert input LBAs to output LBAs. See SCC-2.

3.1.41 update: To write new data to a logical block without destroying the previous data. After a logical block has been updated, a normal read returns the most recent generation of the data. Earlier generations are still available after the update.

3.1.42 user data: Data contained in logical blocks that is not protection information.

3.1.43 volatile cache memory: Cache memory that does not retain data through power cycles.

3.1.44 volatile medium: Medium that does not retain data written to it for a subsequent read operation through power cycles. An example of this is a silicon memory device that loses data written to it if device power is lost.

3.2 Symbols and abbreviations

See table 1 for abbreviations of standards bodies (e.g., ISO). Additional symbols and abbreviations used in this standard include:

Abbreviation	Meaning
CDB	command descriptor block (see 3.1.8)
CLIST	logical unit certification list (see 3.1.23)
DLIST	data defect list (see 3.1.9)
ECC	error correcting code
GLIST	grown defect list (see 3.1.19)
ID	identifier
I/O	input/output
kbit	kilobit (10^3 bits)
LBA	logical block address (see 3.1.22)
LSB	least significant bit
LUN	logical unit number
Mbit	megabit (10^6 bits)
MMC-4	SCSI Multimedia Commands - 4 standard
MSB	most significant bit
PLIST	primary defect list (see 3.1.31)
SACL	storage array conversion layer (see 3.1.40)
SAM-3	SCSI Architecture Model - 3 standard
SCSI	Small Computer System Interface family of standards
SCC-2	SCSI-3 Controller Commands - 2 standard
SES-2	SCSI Enclosure Services - 2 standard
SMC-2	SCSI Medium Changer Commands - 2 standard
SPC-3	SCSI Primary Commands - 3 standard
XOR	exclusive logical OR (see 3.1.16)

3.3 Keywords

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

3.3.2 ignored: A keyword used to describe an unused bit, byte, word, field or code value. The contents or value of an ignored bit, byte, word, field or code value shall not be examined by the receiving SCSI device and may be set to any value by the transmitting SCSI device.

3.3.3 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.

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

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

3.3.6 may not: Keywords that indicate flexibility of choice with no implied preference (equivalent to “may or may not”).

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

3.3.8 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.

3.3.9 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 error.

3.3.10 restricted: 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.

3.3.11 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.

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

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 this clause or in the text where they first appear.

Names of commands, status codes, sense keys, and additional sense codes are in all uppercase (e.g., REQUEST SENSE).

Names of fields and state variables are in small uppercase (e.g. NAME). When a field or state variable name contains acronyms, uppercase letters may be used for readability. Normal case is used when the contents of a field or state variable are being discussed. Fields or state variables containing only one bit are usually referred to as the NAME bit instead of the NAME field.

Normal case is used for words having the normal English meaning.

The ISO convention of numbering is used (i.e., the thousands and higher multiples are separated by a space and a comma is used as the decimal point). Table 2 shows a comparison of the ISO and American numbering conventions.

Table 2 — ISO and American numbering conventions

ISO	American
0,6	0.6
3,141 592 65	3.14159265
1 000	1,000
1 323 462,95	1,323,462.95

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

Numbers immediately followed by lower-case b (e.g., 0101b) are binary values. Underscores may be included in binary values to increase readability or delineate field boundaries (e.g., 0101_1010b).

A sequence of numbers or upper case letters ‘A’ through ‘F’ immediately followed by lower-case h (e.g., FA23h) are hexadecimal values. Underscores may be included in hexadecimal values to increase readability or delineate field boundaries (e.g., FD8C_FA23h).

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

If a conflict arises between text, tables or figures, the order of precedence to resolve the conflicts is text, then tables, and finally figures. Not all tables or figures are fully described in the text. Tables show data format and values.

Notes do not constitute any requirements for implementers.

4 Models

4.1 General

SCSI devices that conform to this standard are referred to as SCSI block devices. This includes the category of logical units commonly referred to as rigid disks and removable rigid disks. MMC-4 is typically used by CD-ROM devices.

The common attribute of block devices is that they are block addressable (i.e., the data are addressed on the block device in groups referred to as logical blocks). The number of bytes of user data contained in a single logical block is the block length. The block length is almost always greater than one byte and may be a multiple of 512 bytes. In addition, a logical block length is not required to bear any relation to the physical block length of the storage medium.

Each logical block has a block length associated with it. This means that the block length for the medium can change from logical block to logical block. However, for simplicity the block length typically remains constant over the entire capacity of the medium. The block length does not include the length of protection information and additional information, if any, that are associated with the logical block.

This standard is intended to be used in conjunction with SAM-3, SPC-3, SCC-2, SES-2, and SMC-2.

4.2 Direct-access device type model overview

Direct-access devices store user data for later retrieval in logical blocks. Logical blocks contain user data and may contain protection information. Each block of user data is stored at a unique logical block address (LBA), which is either four bytes (i.e., a short LBA) or eight bytes (i.e., a long LBA). An application client uses write operations (e.g. WRITE commands) to store user data and read operations (e.g. READ commands) to retrieve user data. Other commands issued by the application client may also cause write and read operations to occur. A write operation causes one or more logical blocks to be written on the medium. A read operation causes one or more logical blocks to be read from the medium. A verify operation confirms that one or more logical blocks were correctly written and can be read without error from the medium.

Logical blocks are stored by a process that causes localized changes or transitions within the medium. The changes made to the medium to store the logical blocks may be volatile (i.e., not retained through power cycles) or non-volatile (i.e., retained through power cycles). The medium may be divided in parts that are used for user data and protection information, parts that are reserved for defect management, and parts that are reserved for use by the controller for the management of the block device.

4.3 Removable medium

4.3.1 Removable medium overview

The medium may be removable or non-removable. The removable medium may be contained within a cartridge (or jacket) to prevent damage to the recording surfaces.

A removable medium has an attribute of being mounted or unmounted on a suitable transport mechanism in a block device. A removable medium is mounted when the block device is capable of performing write or read operations to the medium. A removable medium is unmounted at any other time (e.g., during loading, unloading, or storage).

An application client may check whether a removable medium is mounted by issuing a TEST UNIT READY command. A block device containing a removable medium may need to receive a START STOP UNIT command to become accessible for write or read operations.

The PREVENT ALLOW MEDIUM REMOVAL command allows an application client to restrict the unmounting of the removable medium. This is useful in maintaining system integrity. If the block device implements cache memory, either volatile or non-volatile, it ensures that all logical blocks of the medium contain the most recent user data and protection information, if any, prior to permitting unmounting of the removable medium. If the application client issues a START STOP UNIT command to eject the removable medium, and the block device is prevented from unmounting by the PREVENT ALLOW MEDIUM REMOVAL command, the START STOP UNIT command is rejected by the device server.

4.3.2 Removable medium with an attached medium changer

When a block device is served by a medium changer, control over a media transport element may be done using media changer commands (see SMC-2) sent to the logical unit.

The block device indicates its ability to support these commands by setting the MCHNGR bit to one in its standard INQUIRY data (see SPC-3). A MCHNGR bit set to one indicates that the MOVE MEDIUM ATTACHED and READ ELEMENT STATUS ATTACHED commands (see SMC-2) are supported. Only one medium transport element is permitted (element 0) and only one data transfer element is permitted.

4.4 Logical blocks

Logical blocks are stored on the medium along with additional information that the medium controller uses to manage the storage and retrieval. The format of the additional information is defined by other standards or is vendor-specific and is hidden from the application client during normal read or write operations. This additional information may be used to identify the physical location of the blocks of data and the address of the logical block, and to provide protection against the loss of user data (e.g., ECC bytes).

The address of the first logical block is zero. The address of the last logical block is [n-1], where [n] is the number of logical blocks available to the application client on the medium. A READ CAPACITY command may be issued to determine the value of [n-1].

Logical block addresses are no larger than 8 bytes. Some commands support only 4 byte (i.e., short) LOGICAL BLOCK ADDRESS fields (e.g., READ CAPACITY (10), READ (10), and WRITE (10)). The READ CAPACITY (10) command returns a capacity of FFFFFFFFh if the capacity exceeds that accessible with short LBAs, indicating that:

a) the application client should enable descriptor format sense data (see SPC-3) in the Control mode page (see SPC-3) and in any REQUEST SENSE commands (see SPC-3) it sends; and
b) the application client should use commands with 8-byte LOGICAL BLOCK ADDRESS fields (e.g., READ CAPACITY (16), READ (16), and WRITE (16)).

NOTE 2 - If a command with a 4-byte LOGICAL BLOCK ADDRESS field accesses logical blocks beyond logical block address FFFFFFFFh and fixed format sense data is used, there is no field in the sense data large enough to report the logical block address of an error (see 4.11).

If a command is issued that references or attempts to access a logical block not within the capacity of the medium, the command is terminated with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST an the additional sense code set to LOGICAL BLOCK ADDRESS OUT OF RANGE. The command may be terminated before processing or after the device server has transferred some or all of the data.

The number of bytes of user data contained in a logical block is the block length. Each logical block has a block length associated with it. The READ CAPACITY data (see 5.14) describes the block lengths that are used on the medium. The mode parameter block descriptor (see 6.3.2) is used to change the block length of block devices that support variable block lengths. The block length does not include the length of protection information and additional information, if any.

The location of a logical block on the medium is not required to have a relationship to the location of any other logical block. However, in a typical block device, the time to access a logical block at address [x+1] after accessing logical block [x] is often less than the time to access some other logical block. The time to access the logical block at address [x] and then the logical block at address [x+1] need not be less than time to access [x] and then [x+100]. The READ CAPACITY command issued with a PMI bit set to one may be useful in determining where longer access times occur.

4.5 Ready state

A block device is ready when medium access commands can be processed. A block device using removable media is not ready until a volume is mounted. Such a block device, with a volume not mounted, shall terminate medium access commands with CHECK CONDITION status and the sense key shall be set to NOT READY with the appropriate additional sense code for the condition.

Some block devices may be switched from being ready to being not ready by using the START STOP UNIT command. An application client may need to issue a START STOP UNIT command with a START bit set to bring a block device ready.

4.6 Initialization

Block devices may require initialization prior to write or read operations. This initialization is performed by a FORMAT UNIT command (see 5.4). Parameters related to the geometry and performance characteristics may be set with the MODE SELECT command prior to the format operation. Some block devices are initialized by means not specified in this standard. The time when the initialization occurs is vendor-specific.

Block devices using a non-volatile medium may save the parameters and only need to be initialized once. However, some mode parameters may need to be initialized after each logical unit reset. A catastrophic failure of the block device may require the FORMAT UNIT command to be reissued.

Block devices that use a volatile medium may need to be initialized after each logical unit reset prior to the processing of read or write operations. Mode parameters may also need initialization after logical unit resets.

NOTE 3 - After changing the mode parameter block descriptor with MODE SELECT, the new values do not become effective until FORMAT UNIT command completes. Mode parameter block descriptors read with the MODE SENSE command before a FORMAT UNIT completes return information that may not reflect the true state of the medium.

4.7 Implicit HEAD OF QUEUE command processing

The following commands specific to this command set may be processed by the task manager as if they have a task attribute of HEAD OF QUEUE (see SAM-3) if they are received with a SIMPLE task attribute, an ORDERED task attribute, or no task attribute:

a) READ CAPACITY (10); and
b) READ CAPACITY (16).

See SPC-3 for additional commands subject to implicit HEAD OF QUEUE command processing.

4.8 Medium defects

Any medium has the potential for defects that can cause user data to be lost. Therefore, each logical block may contain additional information that allows the detection of changes to the user data caused by defects in the medium or other phenomena, and may also allow the data to be reconstructed following the detection of such a change (e.g., ECC bytes). Some block devices provide the application client control through use of the mode parameters. Some block devices allow the application client to examine and modify the additional information by using the READ LONG and WRITE LONG commands.

Defects may also be detected and managed during processing of the FORMAT UNIT command (see 5.4). The FORMAT UNIT command defines four sources of defect information: the PLIST, CLIST, DLIST, and GLIST. These defects may be reassigned or avoided during the initialization process so that they do not appear in a logical block.

The algorithm may be controlled by the application client, using options in the FORMAT UNIT command. Four sources of defect location information (i.e., defects) are defined as follows:

a) Primary defect list (PLIST). This is the list of defects, which may be supplied by the original manufacturer of the device or medium, that are considered permanent defects. The PLIST is located outside of the application client-accessible logical block space. The PLIST is accessible by the device server (to reference while formatting), but it is not accessible by the application client except through the READ DEFECT DATA command (see 5.16). Once created, the original PLIST shall not be subject to change;
b) Logical unit certification list (CLIST). This list includes defects detected by the device server during an optional certification process performed during the FORMAT UNIT command. This list shall be added to the GLIST;

c) Data defect list (DLIST). This list of defects may be supplied by the application client to the device server during the the FORMAT UNIT command. This list shall be added to the GLIST. If the DEFECT LIST LENGTH field in the parameter list header is set to zero, there is no DLIST; and
d) Grown defect list (GLIST). The GLIST includes all defects sent by the application client (i.e., the DLIST) or detected by the device server (i.e., the CLIST). The GLIST does not include the PLIST. If the CMPLST bit is set to zero, the GLIST shall include DLISTs provided to the device server during theprevious and the current FORMAT UNIT commands. The GLIST shall also include:
   A) defects detected by the format operation during medium certification;
   B) defects previously identified with a REASSIGN BLOCKS command (see 5.20); and
   C) defects previously detected by the device server and automatically reallocated.

The block device may automatically reassign defects if allowed by the Read-Write Error Recovery mode page (see 6.3.4).

Defects may also occur after initialization. The application client issues a REASSIGN BLOCKS command (see 5.20) to request that the specified logical block address be reassigned to a different part of the medium. This operation may be repeated if a new defect appears at a later time. The total number of defects that may be handled in this manner can be specified in the mode parameters.

Defect management on block devices is vendor-specific. Block devices not using a removable medium may optimize the defect management for capacity or performance or both. Some block devices that use a removable medium do not support defect management or use defect management that does not impede the ability to interchange the medium.

4.9 Cache memory

Some block devices implement cache memory. A cache memory is usually an area of temporary storage in the block device with a fast access time that is used to enhance performance. It exists separately from the stored logical blocks and is not directly accessible by the application client. Use of cache memory for write or read operations may reduce the access time to a logical block and can increase the overall data throughput.

During read operations, the block device uses the cache memory to store logical blocks that the application client may request at some future time. The algorithm used to manage the cache memory is not part of this standard. However, parameters are provided to advise the device server about future requests, or to restrict the use of cache memory for a particular request.

Cache memory stores user data and protection information, if any.

Cache memory may be volatile or non-volatile. Volatile caches do not retain data through power cycles. Non-volatile caches retain data through power cycles. There may be a limit on the amount of time a non-volatile cache is able to retain data.

During write operations, the block device uses the cache memory to store data that is written to the medium at a later time. This is called write-back caching. The command may complete prior to logical blocks being written to the medium. As a result of using a write-back caching there is a period of time when the data may be lost if power to the device is lost and a volatile cache is being used or a hardware failure occurs. There is also the possibility of an error occurring during the subsequent write operation. If an error occurred during the write, it may be reported as a deferred error on a later command. The application client may request that write-back caching be disabled with the Caching mode page (see 6.3.3) to prevent detected write errors from being reported by deferred errors. Even with write-back caching disabled, undetected write errors may occur. In order to detect these errors, the VERIFY and WRITE AND VERIFY commands are provided.

When the cache memory fills up with logical blocks that are being kept for possible future access, new logical blocks that are to be kept replace those currently in cache memory. The disable page out (DPO) bit in the CDB of the read, write, and verify commands allows the application client to influence the replacement of logical blocks in the cache. For write operations, setting this bit to one advises the device server to not replace existing logical blocks in the cache memory with the write data. For read and verify operations, setting this bit to one causes logical blocks that are being read to not replace existing ones in the cache memory.

Sometimes the application client may want to have the logical blocks read from the medium instead of from the cache memory. The force unit access (FUA) bit in the CDB of the read and write commands is used to specify that the device server shall access the medium. For a write operation, setting the FUA bit to one causes

the device server to complete the data write to the medium before completing the command. For a read operation, setting the FUA bit to one causes the logical blocks to be retrieved from the medium. The FUA_NV bit specifies allows the device server to access a non-volatile cache rather than the medium.

When the DPO and FUA bits are both set to one, write and read operations effectively bypass the cache memory.

When a VERIFY command or WRITE AND VERIFY is processed, a forced unit access is implied, since the logical blocks stored on the medium are being verified. Furthermore, a synchronize cache operation is also implied to write unwritten logical blocks still in the cache memory. These logical blocks are stored on the medium before the verify operation begins. The DPO bit is provided since the VERIFY command may cause the replacement of logical blocks in the cache.

Commands may be implemented by the device server that allow the application client to control other behavior of the cache memory:

a) the LOCK UNLOCK CACHE command (see 5.5) controls whether certain logical blocks shall be held in the data cache for future use. Locking a logical block prevents its replacement by a future access. Unlocking a logical block exposes it to possible replacement by a future access;
b) the PRE-FETCH command (see 5.7) causes a set of logical blocks requested by the application client to be read into the data cache for possible future access. The logical blocks fetched are subject to later replacement unless they are locked;
c) the SYNCHRONIZE CACHE command (see 5.22) forces any pending write data in the requested set of logical blocks to be stored in the medium. This command may be used to ensure that the data was written and any detected errors reported;
d) the Caching mode page (see 6.3.3) writeable by the MODE SELECT command allows control of cache behavior and handles certain basic elements of cache replacement algorithms.

4.10 Reservations

The access enabled or access disabled condition determines when an application client may store or retrieve user data on all or part of the medium. Access may be restricted for read operations, write operations, or both. This attribute may be controlled by an external mechanism or by persistent reservations (see SPC-3).

An application client uses reservations to gain a level of exclusivity in access to all or part of the medium for itself or another application client. It is expected that the reservation is retained until released. The device server ensures that the application client with the reservation is able to access the reserved medium within the operating parameters established by that application client.

Reservation restrictions are placed on commands as a result of access qualifiers associated with the type of reservation. The details of commands that are allowed under what types of reservations are described in table 3.

Commands from initiators holding a reservation should complete normally. The behavior of commands from registered initiators when a registrants only or all registrants persistent reservation is present is specified in table 3. A command that does not explicitly write the medium shall be checked for reservation conflicts before the command enters the current task state for the first time. Once the command has entered the current task state, it shall not be terminated with a RESERVATION CONFLICT due to a subsequent reservation. A command that explicitly writes the medium shall be checked for reservation conflicts before the device server modifies the medium or cache as a result of the command. Once the command has modified the medium, it shall not be terminated with a RESERVATION CONFLICT due to a subsequent reservation.

For each command, this standard or SPC-3 defines the conditions that result in RESERVATION CONFLICT.

Extent reservations and RESERVE/RELEASE reservations have been made obsolete in SPC-3 and in this standard.

NOTE 4 - When a system is integrated with more than one application client, agreement is required between the application clients as to how media is reserved and released during operations, otherwise, an application client may be locked out of access to a logical unit in the middle of an operation.

Table 3 — SBC-2 commands that are allowed in the presence of various reservations (part 1 of 2)

Command	Addressed LU has this type of persistent reservation held by another initiator [B]				
	From any initiator		From registered initiator (RR all types)	From initiator not registered	
	Write Excl	Excl Access		Write Excl - RR	Exclusive Access - RR
ERASE (10)/(12)	Conflict	Conflict	Allowed	Conflict	Conflict
FORMAT UNIT	Conflict	Conflict	Allowed	Conflict	Conflict
LOCK UNLOCK CACHE (10)/(16)	Conflict	Conflict	Allowed	Conflict	Conflict
MEDIUM SCAN	Allowed	Conflict	Allowed	Allowed	Conflict
PRE-FETCH (10)/(16)	Allowed	Conflict	Allowed	Allowed	Conflict
READ (6)/(10)/(12)/(16)/(32)	Allowed	Conflict	Allowed	Allowed	Conflict
READ CAPACITY (10)/(16)	Allowed	Allowed	Allowed	Allowed	Allowed
READ DEFECT DATA (10)/(12)	Conflict	Conflict	Allowed	Conflict	Conflict
READ GENERATION	Allowed	Conflict	Allowed	Allowed	Conflict
READ LONG (10)/(16)	Conflict	Conflict	Allowed	Conflict	Conflict
READ UPDATED BLOCK	Allowed	Conflict	Allowed	Allowed	Conflict
REASSIGN BLOCKS	Conflict	Conflict	Allowed	Conflict	Conflict
START STOP UNIT with START=1 and POWER CONDITION=0	Allowed	Allowed	Allowed	Allowed	Allowed
START STOP UNIT with START=0 or POWER CONDITION<>0	Conflict	Conflict	Allowed	Conflict	Conflict
SYNCHRONIZE CACHE (10)/(16)	Conflict	Conflict	Allowed	Conflict	Conflict
UPDATE BLOCK	Conflict	Conflict	Allowed	Conflict	Conflict
VERIFY (10)/(12)/(16)/(32)	Allowed	Conflict	Allowed	Allowed	Conflict
WRITE (6)/(10)/(12)/(16)/(32)	Conflict	Conflict	Allowed	Conflict	Conflict
WRITE AND VERIFY (10)/(12)/(16)/(32)	Conflict	Conflict	Allowed	Conflict	Conflict
WRITE LONG (10)/(16)	Conflict	Conflict	Allowed	Conflict	Conflict
WRITE SAME (10)/(16)/(32)	Conflict	Conflict	Allowed	Conflict	Conflict
XDREAD (10)/(32)	Allowed	Conflict	Allowed	Allowed	Conflict
XDWRITE (10)/(32)	Conflict	Conflict	Allowed	Conflict	Conflict

Table 3 — SBC-2 commands that are allowed in the presence of various reservations (part 2 of 2)

Command	Addressed LU has this type of persistent reservation held by another initiator [B]				
	From any initiator		From registered initiator (RR all types)	From initiator not registered	
	Write Excl	Excl Access		Write Excl - RR	Exclusive Access - RR
XDWRITEREAD (10)/(32)	Conflict	Conflict	Allowed	Conflict	Conflict
XPWRITE (10)/(32)	Conflict	Conflict	Allowed	Conflict	Conflict

Key: [B] = Persistent Reservations LU = Logical Unit, Excl = Exclusive, RR = Registrants Only or All Registrants, <> Not Equal
Allowed: Commands issued by initiators not holding the reservation or by initiators not registered when a registrants only or all registrants persistent reservation is present should complete normally.
Conflict: Commands issued by initiators not holding the reservation or by initiators not registered when a registrants only or all registrants persistent reservation is present shall not be performed and the device server shall terminate the command with a RESERVATION CONFLICT status.

4.11 Error reporting

If any of the conditions in table 4 occur during the processing of a command, the command shall be terminated with CHECK CONDITION status and the sense key shall be set to the appropriate sense key with the appropriate additional sense code for the condition. Some errors may occur after the completion status has already been reported. For such errors, SPC-3 defines a deferred error reporting mechanism. Table 4 lists some error conditions and the applicable sense keys. The list does not provide an exhaustive enumeration of all conditions that may cause the CHECK CONDITION status.

Table 4 — Example error conditions

Condition	Sense key
Invalid LBA	ILLEGAL REQUEST
Unsupported option requested	ILLEGAL REQUEST
Logical unit reset, I_T nexus loss, or medium change since last command from this application client	UNIT ATTENTION
Self diagnostic failed	HARDWARE ERROR
Unrecovered read error	MEDIUM ERROR or HARDWARE ERROR
Recovered read error	RECOVERED ERROR
Over-run or other error that might be resolved by repeating the command	ABORTED COMMAND
Attempt to write on write protected medium	DATA PROTECT

When an invalid LBA is encountered, the first invalid LBA shall be returned in the INFORMATION field of the sense data. When a recovered read error is reported, the INFORMATION field of the sense data shall contain the LBA of the last recovered error during the transfer. When an unrecovered read error is reported, the INFORMATION field of the sense data shall contain the LBA of the unrecovered logical block.

The sense data INCORRECT LENGTH INDICATION (ILI) bit indicates that the requested data length in a READ LONG or WRITE LONG command did not match the length of the data on the medium.

Direct-access devices compliant with this standard shall support both the fixed and descriptor formats of sense data (see SPC-3). If fixed format sense data is used but the sense data contains an INFORMATION field value or COMMAND-SPECIFIC INFORMATION field value too large for the fixed format sense data (e.g., an 8-byte LBA), the VALID bit shall be set to zero.

Table 5 summarizes use of the sense data fields.

Table 5 — Sense data field usage for direct-access devices

Field	Usage	Reference
VALID bit and INFORMATION field	READ LONG	5.18
	REASSIGN BLOCKS	5.20
	WRITE LONG	5.37
	Any command that accesses the medium, based on the Read-Write Error Recovery mode page	6.3.4
COMMAND-SPECIFIC INFORMATION field	EXTENDED COPY	SPC-3
	REASSIGN BLOCKS	5.20
ILI bit	READ LONG	5.18
	WRITE LONG	5.37

4.12 Examples

4.12.1 Examples overview

The following examples show some typical variations of the direct-access device. Other variations are possible.

4.12.2 Rotating media

The typical application of a direct-access device is a disk device. The medium is a disk coated with a material that allows flux changes to be induced. The disk device allows direct and random access to the medium. This is done with an actuator that positions the read-write head, and a rotating disk. Data is stored and retrieved through the interaction of the read-write head and the disk.

The disk(s) may be divided into cylinders. Each cylinder may be divided into tracks. Each track may be divided into sectors. A cylinder is a set of tracks that can be accessed without movement of the actuator. A track is a recording path that the read-write head travels over during one rotation of the disk. A sector is a part of a track that contains the stored data.

A logical block is stored in one or more sectors, or a sector may store more than one logical block. A sector may be made up of a header, data, and a trailer. The header, if any, may contain a preamble used to synchronize read circuits to the data, an address field to identify the sector, flags to use for defect management, and a checksum that validates or corrects the header. The data field begins with a synchronizing field and a data area that contains user data. The trailer may contain checksum or error correction information (e.g., ECC bytes). The checksum or the error correction information allows the correction of data for medium defects.

A disk device is ready when the disks are rotating at the correct speed and the read-write circuitry is powered and ready to access the data. Some disks, particularly removable disks, require the user to issue load or start commands to bring the disk device to the ready state.

A disk device may have to be formatted prior to the initial access. Exceptions to this are devices that are formatted at the factory. A disk device format may create headers for each sector and initialize the data field. The MODE SELECT command is often used prior to formatting to establish the geometry (e.g., logical block length) and defect management scheme. Disk devices are usually non-volatile.

The defect management scheme of a disk device may not be discernible by the user through the interface, though some aspects can be evaluated and controlled by the application client. The device server may reserve some sectors and tracks for recording defect lists and for reassigning defective blocks. The READ LONG and WRITE LONG commands may access the user data and checksum portions of the data field so that defects may be induced by the application client to test the defect detection logic of the device server. WRITE LONG commands may also be used to emulate unrecoverable logical blocks when generating “mirror copies.”

4.12.3 Memory media

Memory media includes logical units that are traditionally used for primary storage within computer systems, such as solid state static or dynamic random access memories (e.g., SRAM, DRAM, or Flash).

These logical units may be non-mechanical, and therefore the entire medium may be accessed in virtually the same access time. The data may be accessed as a bit or byte and this also speeds access time. Memory block devices may store less data than disks or tapes, and may be volatile.

4.13 Model for XOR commands

4.13.1 Model for XOR commands overview

In storage arrays, a storage array controller organizes a group of storage devices into objects. The type of object used by this model is the redundancy group. Some areas within the address space of the storage array are used for check data. The check data is generated by performing a cumulative exclusive-or (XOR) operation with the data from other areas within the address space of the storage array known as protected data. The XOR operation may be performed by the storage array controller or by the storage device.

Performing the XOR operation in the storage device may result in a reduced number of data transfers across the interconnect. For example, when the XOR operation is done within the storage array controller four data transfer operations are needed for a typical update write sequence:

a) a read transfer from the device containing protected data;
b) a write transfer to the device containing protected data;
c) a read transfer from the device containing check data; and
d) a write transfer to the device containing check data.

The storage array controller also does two internal XOR operations in this sequence.

In contrast, during storage array controller supervised XOR operations (see 4.13.1.1) only three data transfer operations are needed:

a) a write transfer to the device containing protected data;
b) a read transfer from the device containing protected data; and
c) a write transfer to the device containing check data.

A storage array controller supervises three basic operations that require XOR functionality. These are the update write, regenerate, and rebuild operations. A command sequence for each of these operations is defined for the following operating modes. The command sequences use the device server to perform the XOR functions needed for the major operations.

4.13.1.1 Storage array controller supervised XOR operations

4.13.1.1.1 Storage array controller supervised XOR operations overview

Three XOR commands are needed to implement storage array controller supervised XOR operations: XDWRITE, XPWRITE, and XDREAD. The XDWRITEREAD command may be used in place of a sequence of XDWRITE followed by XDREAD. The storage array controller also uses READ and WRITE commands for certain operations. The XOR functionality may be used when all of the devices are in the same domain, when all devices are in separate domains, or any combination thereof, as long as the domains are accessible by the storage array controller.

4.13.1.1.2 Update write operation (storage array controller supervised)

The update write operation writes user data to a device containing protected user data and updates the parity information on the device containing check data. The sequence is:

1) An XDWRITE command is sent to the device containing protected user data. This transfers the user write data to that device. The device reads the old user data, performs an XOR operation using the old user data and the received user data, retains the intermediate XOR result, and writes the received user data to the medium;
2) An XDREAD command is sent to the device containing protected user data. This command transfers the intermediate XOR data from the XOR device to the storage array controller; and
3) An XPWRITE command is sent to the device containing check data. This transfers the intermediate XOR data (received in the previous XDREAD command) to the device containing check data. The device reads the old XOR data, performs an XOR operation using the old XOR data and the intermediate XOR data, and writes the new XOR result to the medium.

In place of steps 1) and 2), a single XDWRITEREAD command may be sent to the device containing protected data.

4.13.1.1.3 Regenerate operation (storage array controller supervised)

The regenerate operation is used to recreate a logical block, including user data and protection information, if any, that has an error. This is done by reading the associated logical block from each of the other devices within the redundancy group and performing an XOR operation with each of these logical blocks. The last XOR result is the data that should have been present on the unreadable device. The number of steps is dependent on the number of devices in the redundancy group, but the sequence is as follows:

1) A READ command is sent to the first device. This transfers the data from the device to the storage array controller;
2) An XDWRITE command with the DISABLE WRITE bit set is sent to the next device. This transfers the data from the previous read operation to the device. The device reads its data, performs an XOR operation on the received data and its data, and retains the intermediate XOR result;
3) An XDREAD command is sent to the same device as in step 2. This transfers the intermediate XOR data from the device to the storage array controller; and
4) Steps 2 and 3 are repeated until all devices (except the failed device) in the redundancy group have been accessed. The intermediate XOR data returned by the last XDREAD command is the regenerated user data for the failed device.

In place of steps 2) and 3), a single XDWRITEREAD command may be sent to the device.

4.13.1.1.4 Rebuild operation (storage array controller supervised)

The rebuild operation is similar to the regenerate operation, except that the last XOR result is written to the replacement device. This function is used when a failed device is replaced and the storage array controller is writing the rebuilt data to the replacement device. The sequence is as follows:

1) A READ command is sent to the first device. This transfers the data from the device to the storage array controller;
2) An XDWRITE command with the DISABLE WRITE bit set to one is sent to the next device. This transfers the data from the previous read operation to the device. The device reads its data, performs an XOR operation using the received data and its data, and retains the intermediate XOR result;
3) An XDREAD command is sent to the same device as in step 2. This transfers the intermediate XOR data from the device to the storage array controller;
4) Steps 2 and 3 are repeated until all devices (except the replacement device) in the redundancy group have been accessed. The intermediate XOR data returned by the last XDREAD command is the regenerated user data for the replacement device; and
5) A WRITE command is sent to the replacement device. This transfers the regenerated user data from step 4 to the replacement device. The replacement device writes the regenerated user data to the medium.

In place of steps 2) and 3), a single XDWRITEREAD command may be sent to the device.

4.13.1.2 Additional array subsystem considerations

4.13.1.2.1 Additional array subsystem considerations overview

This subclause lists considerations that apply to any array subsystem, but describes how use of the XOR commands may affect handling of those situations.

4.13.1.2.2 Buffer full status handling

When the storage array controller sends an XDWRITE command to a device, the device has an obligation to retain the resulting XOR data until the storage array controller issues a matching XDREAD command to retrieve the data. This locks up part or all (depending on the size of the device's buffer and the size of the XOR data) of the device's buffer space. When all of the device's buffer is allocated for XOR data, it may not be able to accept new medium access commands other than valid XDREAD commands and it may not be able to begin processing of commands that are already in the task set.

When the device is not able to accept a new command because there is not enough space in the buffer, the device shall terminate that command with a CHECK CONDITION status and the sense key shall be set to ILLEGAL REQUEST with the additional sense code set to BUFFER FULL.

When a storage array controller receives this status, it may issue any matching XDREAD commands needed to satisfy any previous XDWRITE commands. This results in buffer space being freed for other commands. If it is a multi-initiator system and the storage array controller has no XDREAD commands to send, the storage array controller may assume the buffer space has been allocated to another initiator. The storage array controller may retry the command in the same manner that a command ending with TASK SET FULL status would be retried including not retrying the command too frequently.

The bidirectional XDWRITEREAD command avoids the buffer full condition. The storage array controller may issue multiple XDWRITEREAD commands, since the device controls when it accepts more write data and provides read data.

4.13.1.2.3 Access to an inconsistent stripe

A stripe is a set of corresponding strips of consecutively addressed storage from two or more block devices. A strip is an equal division of the storage capacity in a set of consecutively addressed LBAs on a single block device. When the storage array controller issues an update write to a device, the data in the device has been updated when successful status is returned for the command. Until the device containing check data has been updated, however, the associated stripe in the redundancy group is not consistent (e.g., performing an XOR operation on the protected data does not produce the check data). The storage array controller shall keep track of this window of inconsistency and make sure that a regenerate or rebuild operation for any data extent within the stripe is not attempted until after the device containing check data has been updated (making the stripe consistent again). For multi-initiator systems, tracking the updates may be more complex because each storage array controller needs to ensure that a second storage array controller is not writing to a stripe that the first storage array controller is regenerating or rebuilding. The coordination between storage array controllers is system specific and is beyond the scope of this standard. A storage array controller needs to prevent data corruption due to a temporarily inconsistent stripe in one case. When an XDWRITE or XDWRITEREAD command has been issued and completed, the device containing protected data has been updated but the device containing check data has not. The stripe is inconsistent until the XPWRITE command to the device containing check data returns completion status.

4.13.1.3 Error handling considerations

4.13.1.3.1 Error handling considerations overview

If any of the XOR commands end with CHECK CONDITION status and an unrecovered error is indicated, an inconsistent stripe may result. It is the storage array controller's responsibility to identify the failing device and the extent of the failure, then limit access to the inconsistent stripe. The recovery procedures that the storage array controller implements are not addressed by this standard.

4.13.1.3.2 Primary errors - errors resulting directly from the primary command

The first class of errors consists of exception conditions that are detected by the device that received the primary command (primary target) and are not due to the failure of a resulting secondary command. These conditions include, but are not limited to, invalid parameters in the primary command, inability of the primary target to continue operating, and parity errors while transferring the primary command, data, or status byte. In the event of such an exception condition, the primary target shall:

1) terminate the primary command with CHECK CONDITION status; and
2) build sense data according to the exception condition.

4.13.1.4 XOR data retention requirements

The target shall retain XOR data while awaiting retrieval by an XDREAD command until one of the following events occurs:

a) a matching XDREAD command;
b) logical unit reset;
c) I_T nexus loss involving the initiator which sent the XDWRITE command;
d) CLEAR TASK SET;
e) ABORT TASK if the task matches the pending XDREAD; or
f) ABORT TASK SET.

4.14 START STOP UNIT and power conditions

4.14.1 START STOP UNIT and power conditions overview

The START STOP UNIT command (see 5.21) allows an application client to control the power condition of a logical unit. This method includes specifying that the logical unit transition to a power condition.

In addition to the START STOP UNIT command, the power condition of a logical unit may be controlled by the Power Condition mode page (see SPC-3). If both the START STOP UNIT command and the Power Condition mode page methods are being used to control the power condition of the same logical unit, then any START STOP UNIT command's power condition specification shall override the Power Condition mode page's power control.

There shall be no notification to the initiator that a logical unit has transitioned from one power condition to another. An application client may determine the current power condition of a logical unit by issuing a REQUEST SENSE command (see SPC-3).

No power condition shall affect the supply of any power required for proper operation of the service delivery subsystem.

4.14.2 START STOP UNIT and power conditions state machine

4.14.2.1 START STOP UNIT and power conditions state machine overview

The SSU_PC (start stop unit power condition) state machine for logical units implementing the START STOP UNIT command describes the logical unit power states and transitions resulting from settings by the START STOP UNIT command and settings in the Power Condition mode page (see SPC-3).

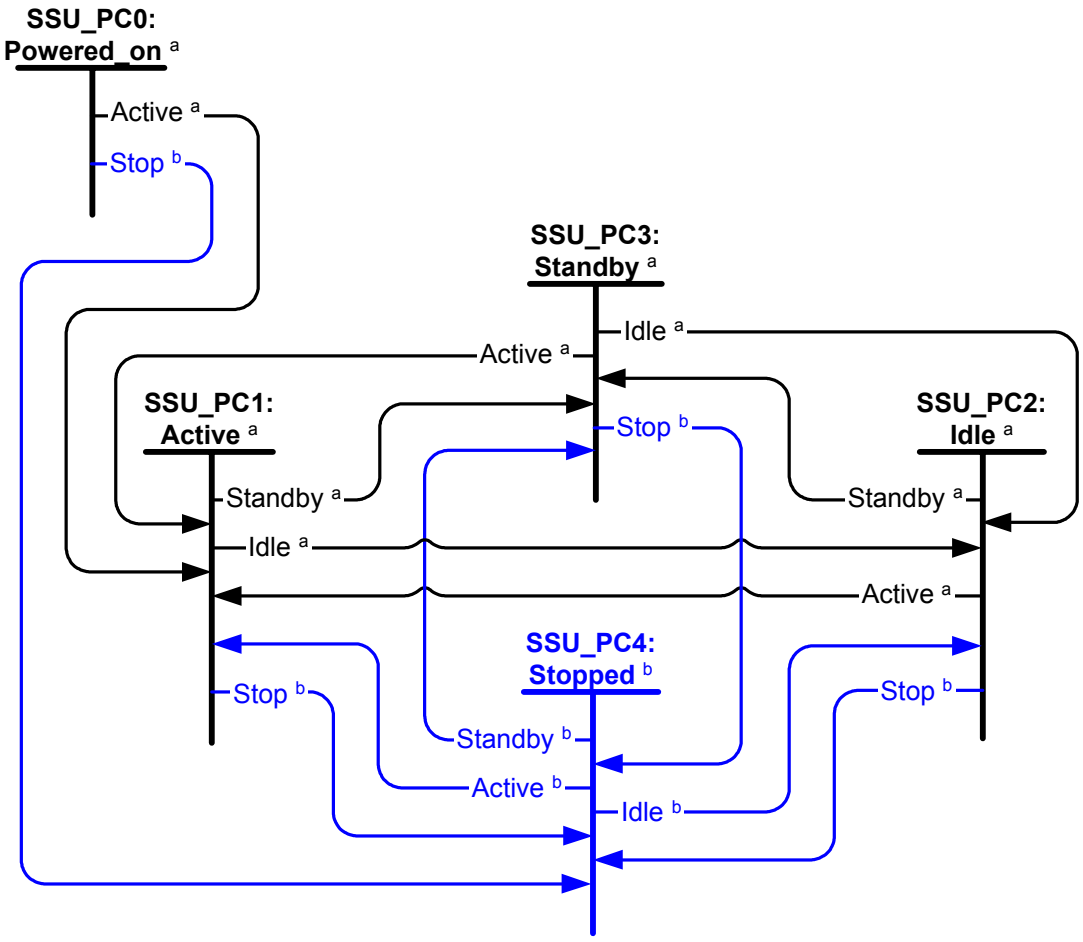
The SSU_PC states are as follows:

a) SSU_PC0:Powered_on (see 4.14.2.2) (initial state);
b) SSU_PC1:Active (see 4.14.2.3);
c) SSU_PC2:Idle (see 4.14.2.4);
d) SSU_PC3:Standby (see 4.14.2.5); and
e) SSU_PC4:Stopped (see 4.14.2.6).

The SSU_PC state machine shall start in the SSU_PC0:Powered_on state after power on.

NOTE 5 - The SSU_PC state machine is an enhanced version of the Power Condition state machine described in SPC-3.

Figure 2 describes the SSU_PC state machine.



Notes:
a This state or transition is also described in SPC-3, but may have additional characteristics unique to this standard.
b This state or transition is described in this standard.

Figure 2 — Power condition state machine for logical units implementing the START STOP UNIT command

4.14.2.2 SSU_PC0:Powered_on state

4.14.2.2.1 SSU_PC0:Powered_on state description

This logical unit shall enter this state upon power on. This state consumes zero time.

4.14.2.2.2 Transition SSU_PC0:Powered_on to SSU_PC1:Active

This transition shall occur if the logical unit has been configured to transition to the SSU_PC1:Active state.

4.14.2.2.3 Transition SSU_PC0:Powered_on to SSU_PC4:Stopped

This transition shall occur if the logical unit has been configured to transition to the SSU_PC4:Stopped state.

4.14.2.3 SSU_PC1:Active state

4.14.2.3.1 SSU_PC1:Active state description

While in this state, if power on initialization is not complete, then the logical unit completes its power on initialization.

While in this state, after power on initialization is complete, then:

a) a logical unit is in the active power condition (see SPC-3);
b) if the idle condition timer is active (see SPC-3) and not disabled (see 5.21), then the idle condition timer is running; and
c) if the standby condition timer is active (see SPC-3) and not disabled (see 5.21), then the standby condition timer is running.

4.14.2.3.2 Transition SSU_PC1:Active to SSU_PC2:Idle

This transition shall occur after:

a) the device server receives a START STOP UNIT command with the POWER CONDITION field set to IDLE;
b) the device server receives a START STOP UNIT command with the POWER CONDITION field set to FORCE_IDLE_0; or
c) the idle condition timer is active (see SPC-3), enabled (see 5.21), and zero.

4.14.2.3.3 Transition SSU_PC1:Active to SSU_PC3:Standby

This transition shall occur after:

a) the device server receives a START STOP UNIT command with the POWER CONDITION field set to STANDBY);
b) the device server receives a START STOP UNIT command with the POWER CONDITION field set to FORCE_STANDBY_0; or
c) the standby condition timer is active (see SPC-3), enabled (see 5.21), and zero.

4.14.2.3.4 Transition SSU_PC1:Active to SSU_PC4:Stopped

This transition shall occur after the device server receives a START STOP UNIT command with the START bit set to zero and the POWER CONDITION field set to START_VALID.

4.14.2.4 SSU_PC2:Idle state

4.14.2.4.1 SSU_PC2:Idle state description

While in this state:

a) a logical unit is in the idle power condition (see SPC-3); and
b) if the standby condition timer is active (see SPC-3) and not disabled (see 5.21), then the standby condition timer is running.

4.14.2.4.2 Transition SSU_PC2:Idle to SSU_PC1:Active

This transition shall occur after:

a) the device server receives a START STOP UNIT command with the START bit set to one;
b) the device server receives a START STOP UNIT command with the POWER CONDITION field set to ACTIVE; or
c) the device server receives a command that requires the logical unit to be in the SSU_PC1:Active state to process the command.

4.14.2.4.3 Transition SSU_PC2:Idle to SSU_PC3:Standby

This transition shall occur after:

a) the device server receives a START STOP UNIT command with the POWER CONDITION field set to STANDBY;
b) the device server receives a START STOP UNIT command with the POWER CONDITION field set to FORCE_STANDBY_0; or
c) the standby condition timer is active (see SPC-3), enabled (see 5.21), and expired.

4.14.2.4.4 Transition SSU_PC2:Idle to SSU_PC4:Stopped

This transition shall occur after the device server receives a START STOP UNIT command with the START bit set to zero.

4.14.2.5 SSU_PC3:Standby state

4.14.2.5.1 SSU_PC3:Standby state description

While in this state a logical unit is in the standby power condition (see SPC-3).

4.14.2.5.2 Transition SSU_PC3:Standby to SSU_PC1:Active

This transition shall occur after:

a) the device server receives a START STOP UNIT command with the START bit set to one;
b) the device server receives a START STOP UNIT command with the POWER CONDITION field set to ACTIVE; or
c) the device server receives a command that requires the logical unit to be in the SSU_PC1:Active state to process the command.

4.14.2.5.3 Transition SSU_PC3:Standby to SSU_PC2:Idle

This transition shall occur after:

a) the device server receives a START STOP UNIT command with the POWER CONDITION field set to IDLE;
b) the device server receives a START STOP UNIT command with the POWER CONDITION field set to FORCE_IDLE_0; or
c) the device server receives a command that requires the logical unit to be in the SSU_PC2:Idle state to process the command.

4.14.2.5.4 Transition SSU_PC3:Standby to SSU_PC4:Stopped

This transition shall occur after the device server receives a START STOP UNIT command with the START bit set to zero.

4.14.2.6 SSU_PC4:Stopped state

4.14.2.6.1 SSU_PC4:Stopped state description

While in this state:

a) the device server is not capable of processing medium access commands. Any medium access commands received while in this state shall cause the device server to terminate the command with a CHECK CONDITION status with a sense key of NOT READY and an additional sense code of LOGICAL UNIT NOT READY, INITIALIZING COMMAND REQUIRED; and
b) the logical unit may consume less power than when in the SSU_PC1:Active, SSU_PC2:Idle, or SSU_PC3:Standby states.

4.14.2.6.2 Transition SSU_PC4:Stopped to SSU_PC1:Active

This transition shall occur after:

a) the device server receives a START STOP UNIT command with the START bit set to one; or
b) the device server receives a START STOP UNIT command with the POWER CONDITION field set to ACTIVE.

4.14.2.6.3 Transition SSU_PC4:Stopped to SSU_PC2:Idle

This transition shall occur after:

a) the device server receives a START STOP UNIT command with the POWER CONDITION field set to IDLE; or
b) the device server receives a START STOP UNIT command with the POWER CONDITION field set to FORCE_IDLE_0.

4.14.2.6.4 Transition SSU_PC4:Stopped to SSU_PC3:Standby

This transition shall occur after:

a) the device server receives a START STOP UNIT command with the POWER CONDITION field set to STANDBY; or
b) the device server receives a START STOP UNIT command with the POWER CONDITION field set to FORCE_STANDBY_0.

4.15 Protection information model

4.15.1 Protection information overview

This data protection model provides for protection of the data while it is being transferred between a sender and a receiver. Protection information is generated at the application layer and may be checked by any object along the I_T_L nexus. Once received, protection information is retained (e.g., write to medium, store in non-volatile memory, recalculate on read back) by the device server until overwritten (e.g., power loss, hard reset, logical unit reset, and I_T nexus loss have no effect on the retention of protection information).

Support for protection information shall be indicated in the PROTECT bit in the standard INQUIRY data (see SPC-3).

For commands that are using protection information, the application client buffer shall consist of logical blocks with both user data and protection information. For commands that are not using protection information, the application client buffer shall consist of logical blocks with only user data.

If the logical unit is formatted with protection information and the EMDP bit is set to one in the Disconnect-Reconnect mode page (see SPC-3), then checking of the logical block reference tag within the service delivery subsystem may cause false errors because logical blocks may be transmitted out of order.

4.15.2 Protection information format

Table 6 defines the placement of protection information in a logical block.

Table 6 — User data and protection information format

Byte\Bit	7	6	5	4	3	2	1	0
0	USER DATA							
n - 1								
n	(MSB)	LOGICAL BLOCK GUARD						
n + 1								(LSB)
n + 2	(MSB)	LOGICAL BLOCK APPLICATION TAG						
n + 3								(LSB)
n + 4	(MSB)	LOGICAL BLOCK REFERENCE TAG						
n + 7								(LSB)

The USER DATA field shall contain user data. The contents of the USER DATA field shall be used to generate and check the CRC contained in the LOGICAL BLOCK GUARD field.

The LOGICAL BLOCK GUARD field contains the CRC (see 4.15.3) of the contents of the USER DATA field.

The LOGICAL BLOCK APPLICATION TAG field is set by the application client. The contents of the logical block application tag are not defined by this standard. The LOGICAL BLOCK APPLICATION TAG field may be modified by a device server if the ATO bit is set to zero in the Control mode page (see SPC-3). The contents of the LOGICAL BLOCK APPLICATION TAG field shall not be used to generate or check the CRC contained in the LOGICAL BLOCK GUARD field.

The LOGICAL BLOCK REFERENCE TAG field is an incrementing value associated with the logical block. For commands that do not include an INITIAL LOGICAL BLOCK REFERENCE TAG field (e.g., READ (16)), the first logical block in the application client data buffer shall contain the least significant four bytes of the LBA contained in the LOGICAL BLOCK ADDRESS field of the command associated with the logical block. For commands that do include an INITIAL LOGICAL BLOCK REFERENCE TAG field (e.g., READ (32)), the first logical block shall contain a LOGICAL BLOCK REFERENCE TAG field equal to the value in the command. Each logical block in the application client data buffer contains a LOGICAL BLOCK REFERENCE TAG field with the logical block reference tag of the previous logical block plus one. The contents of the LOGICAL BLOCK REFERENCE TAG field shall not be used to generate or check the CRC contained in the LOGICAL BLOCK GUARD field.

4.15.3 Logical block guard

4.15.3.1 Logical block guard overview

If data protection is enabled, the logical block guard shall contain a CRC that is generated from the contents of the USER DATA field.

Table 7 defines the CRC polynomials.

Table 7 — CRC polynomials

Function	Definition
F(x)	A polynomial of degree k-1 that is used to represent the k bits of the user data covered by the CRC. For the purposes of the CRC, the coefficient of the highest order term shall be byte zero bit seven of the USER DATA field.
G(x)	The generator polynomial: $G(x) = x^{16} + x^{15} + x^{11} + x^9 + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$ (i.e., G(x) = 18BB7h)
R(x)	The remainder polynomial, which is of degree less than 16.
P(x)	The remainder polynomial on the receive checking side, which is of degree less than 16. P(x) = 0 indicates no error was detected.
Q(x)	The greatest multiple of G(x) in $(x^{16} \times F(x)) + (x^k \times L(x))$
Q'(x)	$x^{16} \times Q(x)$
M(x)	The sequence that is transmitted.
M'(x)	The sequence that is received.

4.15.3.2 CRC generation

The equations that are used to generate the CRC from F(x) are as follows. All arithmetic is modulo 2.

CRC value in logical block = R(x)

The CRC is calculated by the following equation:

$$\frac{(x^{16} \times F(x))}{G(x)} = Q(x) + \frac{R(x)}{G(x)}$$

The following equation specifies that the CRC is appended to the end of F(x):

$M(x) = x^{16} \times F(x) + CRC$

The CRC generator processs the USER DATA field one word (i.e., two bytes) at a time, starting with byte 0. In the first word, byte 0 bit seven is the MSB and byte 1 bit 0 is the LSB. Figure 3 shows an example CRC generator.

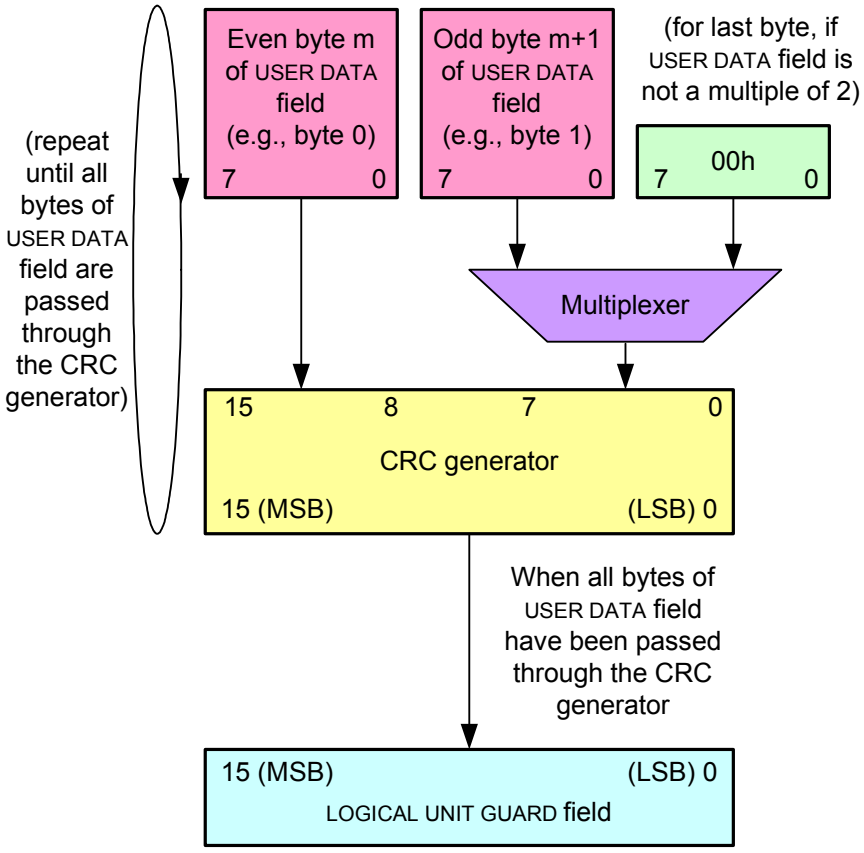


Figure 3 — CRC generator example

4.15.3.3 CRC checking

The received sequence M'(x) may differ from the transmitted sequence M(x) if there are transmission errors. The process of checking the sequence for validity involves dividing the received sequence by G(x) and testing the remainder. Mathematically, the received checking is shown by the following equation:

$$x^{16} \times \frac{M'(x)}{G(x)} = Q'(x) + \frac{P(x)}{G(x)}$$

In the absence of errors remainder P(x) is zero.

The bit order of F(x) presented to the CRC checking function is the same order as the CRC generation bit order (see figure 3).

4.15.3.4 CRC test cases

Several test cases are shown in table 8.

Table 8 — CRC test cases

Pattern	CRC
32 bytes each set to 00h	0000h
32 bytes each set to FFh	A293h
32 bytes of an incrementing pattern from 00h to 1Fh	0224h
2 bytes each set to FFh followed by 30 bytes set to 00h	21B8h
32 bytes of a decrementing pattern from FFh to E0h	A0B7h

4.15.4 Application of protected data

Before an application client transmits or receives protected data it shall:

1) Determine if a logical unit supports protected data using the INQUIRY command (see the PROTECT bit in the standard INQUIRY data in SPC-3);
2) If protected data is supported then determine if the logical unit is formatted to accept protected information using the READ CAPACITY (16) command (see the PROT_EN bit in 5.15);
3) If the logical unit supports protected information and is not formatted to accept protected information then format the logical unit with protected information usage enabled; and
4) If the logical unit supports protected information and is formatted to accept protected information then the read commands that support protected information may be used and write commands that support protected information should be used.

4.15.5 Protected data commands

The enabling of protection information enables fields in some commands that instruct the device server on the handling of protection information. The detailed description of each command's protection information fields are defined in the individual command descriptions.

The commands that are affected when protection information is enabled are listed in table 9.

Commands that result in the return of the length in bytes of each logical block (e.g., MODE SENSE, READ CAPACITY) shall return the length of the user data and shall not include the length of the protection information (e.g., if the user data plus the protection information is equal to 520 bytes then 512 is returned).

4.16 Grouping function

A grouping function is a function that collects information about attributes associated with commands (i.e., information about commands with the same group value are collected into the specified group). The definition of the attributes and the groups is outside the scope of this standard. Groups are identified with the GROUP NUMBER field in the CDB of certain commands (e.g., the PRE-FETCH (10) command (see 5.7)).

The collection of this information is outside the scope of this standard (i.e., the information may not be transmitted using any SCSI protocols).

NOTE 6 - An example of how grouping could be used would be if two applications use a subsystem; one application primarily streams data and another primarily accesses data randomly. If the streaming application groups all of its commands with one value (e.g., x), and the random application groups all of its commands with another value (e.g., y), then a group x defined to hold performance metrics collects all the performance metrics for the streamed commands together and a group y defined to also hold performance metrics collects all the performance metrics for the random commands together. The result is two sets of performance metrics (i.e., x and y). A management application then reads the performance metrics and determines if the performance of a specific group is acceptable.

5 Commands for direct-access block devices

5.1 Commands for direct-access devices overview

The commands for direct-access devices are listed in table 9. Commands with CDB or parameter data fields that support protection information are indicated by the “Protection information” column.

Table 9 — Commands for direct-access devices (part 1 of 4)

Command name	Operation code [a]	Type [b]	Protection information	Reference
ACCESS CONTROL IN	86h	O	no	SPC-3
ACCESS CONTROL OUT	87h	O	no	SPC-3
CHANGE ALIASES	A4h/0Bh	O	no	SPC-3
EXTENDED COPY	83h	O	no	SPC-3
FORMAT UNIT	04h	M	yes	5.4
INQUIRY	12h	M	no	SPC-3
LOCK UNLOCK CACHE (10)	36h	O	no	5.5
LOCK UNLOCK CACHE (16)	92h	O	no	5.6
LOG SELECT	4Ch	O	no	SPC-3
LOG SENSE	4Dh	O	no	SPC-3
MAINTENANCE IN	A3h/00h - 04h A3h/06h - 09h	X [e]	no	SCC-2
MAINTENANCE OUT	A4h/00h - 05h A4h/07h - 09h	X [e]	no	SCC-2
MODE SELECT (6)	15h	O	no	SPC-3
MODE SELECT (10)	55h	O	no	SPC-3
MODE SENSE (6)	1Ah	O	no	SPC-3
MODE SENSE (10)	5Ah	O	no	SPC-3
MOVE MEDIUM ATTACHED	A7h	X [f]	no	SMC-2
PERSISTENT RESERVE IN	5Eh	O [c]	no	SPC-3
PERSISTENT RESERVE OUT	5Fh	O [c]	no	SPC-3
PRE-FETCH (10)	34h	O	no	5.7
PRE-FETCH (16)	90h	O	no	5.8
PREVENT ALLOW MEDIUM REMOVAL	1Eh	O	no	SPC-3
READ (6)	08h	M	yes	5.9
READ (10)	28h	M	yes	5.10
READ (12)	A8h	O	yes	5.11
READ (16)	88h	M	yes	5.12
READ (32)	7Fh/0009h	O	yes	5.13
READ ATTRIBUTE	8Ch	O	no	SPC-3
READ BUFFER	3Ch	O	no	SPC-3

Table 9 — Commands for direct-access devices (part 2 of 4)

Command name	Operation code [a]	Type [b]	Protection information	Reference
READ CAPACITY (10)	25h	M	no	5.14
READ CAPACITY (16)	9Eh/10h	M	no	5.15
READ DEFECT DATA (10)	37h	O	no	5.16
READ DEFECT DATA (12)	B7h	O	no	5.17
READ ELEMENT STATUS ATTACHED	B4h	X [f]	no	SMC-2
READ LONG (10)	3Eh	O	yes	5.18
READ LONG (16)	9Eh/11h	O	yes	5.19
REASSIGN BLOCKS	07h	O	no	5.20
RECEIVE COPY RESULTS	84h	O	no	SPC-3
RECEIVE DIAGNOSTIC RESULTS	1Ch	O/M [g]	no	SPC-3
REDUNDANCY GROUP IN	BAh	X [e]	no	SCC-2
REDUNDANCY GROUP OUT	BBh	X [e]	no	SCC-2
REPORT ALIASES	A3h/0Bh	O	no	SPC-3
REPORT DEVICE IDENTIFIER	A3h/05h	O	no	SPC-3
REPORT LUNS	A0h	X	no	SPC-3
REPORT PRIORITY	A3h/0Eh	O	no	SPC-3
REPORT SUPPORTED OPERATION CODES	A3h/0Ch	O	no	SPC-3
REPORT SUPPORTED TASK MANAGEMENT FUNCTIONS	A3h/0Dh	O	no	SPC-3
REPORT TARGET PORT GROUPS	A3h/0Ah	O	no	SPC-3
REQUEST SENSE	03h	M	no	SPC-3
SEND DIAGNOSTIC	1Dh	M	no	SPC-3
SET DEVICE IDENTIFIER	A4h/06h	O	no	SPC-3
SET PRIORITY	A4h/0Eh	O	no	SPC-3
SET TARGET PORT GROUPS	A4h/0Ah	O	no	SPC-3
SPARE IN	BCh	X [e]	no	SCC-2
SPARE OUT	BDh	X [e]	no	SCC-2
START STOP UNIT	1Bh	O	no	5.21
SYNCHRONIZE CACHE (10)	35h	O	yes	5.22
SYNCHRONIZE CACHE (16)	91h	O	yes	5.23
TEST UNIT READY	00h	M	no	SPC-3
VERIFY (10)	2Fh	O	yes	5.24
VERIFY (12)	AFh	O	yes	5.25
VERIFY (16)	8Fh	O	yes	5.26

Table 9 — Commands for direct-access devices (part 3 of 4)

Command name	Operation code [a]	Type [b]	Protection information	Reference
VERIFY (32)	7Fh/000Ah	O	yes	5.27
VOLUME SET IN	BEh	X [e]	no	SCC-2
VOLUME SET OUT	BFh	X [e]	no	SCC-2
WRITE (6)	0Ah	O	yes	5.28
WRITE (10)	2Ah	O	yes	5.29
WRITE (12)	AAh	O	yes	5.30
WRITE (16)	8Ah	O [d]	yes	5.31
WRITE (32)	7Fh/000Bh	O	yes	5.32
WRITE AND VERIFY (10)	2Eh	O	yes	5.33
WRITE AND VERIFY (12)	AEh	O	yes	5.34
WRITE AND VERIFY (16)	8Eh	O	yes	5.35
WRITE AND VERIFY (32)	7Fh/000Ch	O	yes	5.36
WRITE ATTRIBUTE	8Dh	O	no	SPC-3
WRITE BUFFER	3Bh	O	no	SPC-3
WRITE LONG (10)	3Fh	O	yes	5.37
WRITE LONG (16)	9Fh/11h	O	yes	5.38
WRITE SAME (10)	41h	O	yes	5.39
WRITE SAME (16)	93h	O	yes	5.40
WRITE SAME (32)	7Fh/000Dh	O	yes	5.41
XDREAD (10)	52h	O	yes	5.42
XDREAD (32)	7Fh/0003h	O	yes	5.43
XDWRITE (10)	50h	O	yes	5.44
XDWRITE (32)	7Fh/0004h	O	yes	5.45
XDWRITEREAD (10)	53h	O	yes	5.46
XDWRITEREAD (32)	7Fh/0007h	O	yes	5.47

Table 9 — Commands for direct-access devices (part 4 of 4)

Command name	Operation code [a]	Type [b]	Protection information	Reference
XPWRITE (10)	51h	O	yes	5.48
XPWRITE (32)	7Fh/0006h	O	yes	5.49

Notes:
[a] Some commands are 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] M = command implementation is mandatory. O = command implementation is optional. X = Command implementation requirements detailed in the reference.
[c] If either PERSISTENT RESERVE IN or PERSISTENT RESERVE OUT is implemented, both shall be implemented.
[d] If any of WRITE (6)/(10)/(12) is implemented, WRITE (16) shall also be implemented.
[e] Specified SCC-2 commands are supported of and only if the SCCS bit is set to one in the standard INQUIRY data (see SPC-3).
[f] Specified SMC-2 commands are supported if and only if the MCHGR bit is set to one in the standard INQUIRY data (see SPC-3).
[g] This command shall be supported if the ENCSERV bit is set to one in the standard INQUIRY data (see SPC-3) and may be supported otherwise.

The following operation codes are obsolete:
01h (REZERO UNIT), 0Bh (SEEK (6)),
16h (RESERVE (6)), 17h (RELEASE (6)), 18h (COPY), 2Bh (SEEK (10)),
30h (SEARCH DATA HIGH (10)), 31h (SEARCH DATA EQUAL (10)), 32h (SEARCH DATA LOW (10)),
33h (SET LIMITS (10)), 39h (COMPARE), 3Ah (COPY AND VERIFY),
40h (CHANGE DEFINITION), 56h (RESERVE (10)), 57h (RELEASE (10h)),
80h (XDWRITE EXTENDED (16)), 81h (REBUILD (16)), 82h (REGENERATE (16)), and
B3h (SET LIMITS (12)).

The following operation codes are vendor-specific:
02h, 05h, 06h, 09h, 0Ch, 0Dh, 0Eh, 0Fh,
10h, 11h, 13h, 14h, 19h,
20h, 21h, 22h, 23h, 24h, 26h, 27h, 29h, 2Ch, 2Dh, and
C0h through FFh.

All remaining operation codes for direct-access devices are reserved for future standardization.

5.2 Variable length CDBs

Some commands in table 9 use the variable length command format defined in SPC-3. These commands use operation code 7Fh and are differentiated by service action codes as described in table 10.

Table 10 — Variable length command service action code assignments

Operation code/service action code	Description	Reference
7Fh/0000h	Reserved for direct-access devices	
7Fh/0001h	Obsolete (REBUILD (32))	
7Fh/0002h	Obsolete (REGENERATE (32))	
7Fh/0003h	XDREAD (32)	5.43
7Fh/0004h	XDWRITE (32)	5.45
7Fh/0005h	Obsolete (XDWRITE EXTENDED (32))	
7Fh/0006h	XPWRITE (32)	5.49
7Fh/0007h	XDWRITEREAD (32)	5.47
7Fh/0008h	Obsolete (XDWRITE EXTENDED (64))	
7Fh/0009h	READ (32)	5.13
7Fh/000Ah	VERIFY (32)	5.27
7Fh/000Bh	WRITE (32)	5.32
7Fh/000Ch	WRITE AND VERIFY (32)	5.36
7Fh/000Dh	WRITE SAME (32)	5.41
7Fh/000Eh - 07FFh	Reserved for direct-access devices	

5.3 Service action CDBs

Some commands in table 9 are implemented as device-type specific service actions for the SERVICE ACTION IN (16) operation code 9Eh defined in SPC-3. These commands are differentiated by service action codes as described in table 11.

Table 11 — SERVICE ACTION IN (16) service actions

Operation code/service action code	Description	Reference
9Eh/00h - 0Fh	Reserved for commands applicable to all device types	SPC-3
9Eh/10h	READ CAPACITY (16)	5.15
9Eh/11h	READ LONG (16)	5.19
9Eh/12h - 1Fh	Reserved for direct-access devices	

Some commands in table 9 are implemented as device-type specific service actions for the SERVICE ACTION OUT (16) operation code 9Fh defined in SPC-3. These commands are differentiated by service action codes as described in table 12.

Table 12 — SERVICE ACTION OUT (16) service actions

Operation code/service action code	Description	Reference
9Fh/00h - 0Fh	Reserved for commands applicable to all device types	SPC-3
9Fh/10h	Reserved for direct-access devices	
9Fh/11h	WRITE LONG (16)	5.38
9Fh/12h - 1Fh	Reserved for direct-access devices	

5.4 FORMAT UNIT command

5.4.1 FORMAT UNIT command overview

The FORMAT UNIT command (see table 13) formats the medium into application client addressable logical blocks as specified in the number of blocks and block length values received in the last mode parameter block descriptor (see 6.3.2) in a MODE SELECT command (see SPC-3). In addition, the medium may be certified and control structures may be created for the management of the medium and defects. The degree that the medium is altered by this command is vendor-specific.

If a device server receives a FORMAT UNIT command before receiving a MODE SELECT command with a mode parameter block descriptor the device server shall use the number of blocks and block length at which the logical unit is currently formatted (i.e., no change is made to the number of blocks and the block length of the logical unit during the format operation).

Table 13 — FORMAT UNIT command

Byte\Bit	7	6	5	4	3	2	1	0
0	OPERATION CODE (04h)							
1	FMTPINFO	RTO_REQ	LONGLIST	FMTDATA	CMPLIST	DEFECT LIST FORMAT		
2	Vendor specific							
3	Obsolete							
4								
5	CONTROL							

The simplest form of the FORMAT UNIT command (i.e., a FORMAT UNIT command with no parameter data) accomplishes medium formatting with little application client control over defect management. The device server implementation determines the degree of defect management that is to be performed. Additional forms of this command increase the application client's control over defect management. The application client may specify:

a) defect list(s) to be used;
b) defect locations;
c) that logical unit certification be enabled; and
d) exception handling in the event that defect lists are not accessible.

While performing a format operation, the device server shall respond to all new commands except INQUIRY, REPORT LUNS, and REQUEST SENSE with a CHECK CONDITION status, a sense key of NOT READY and

an additional sense code of LOGICAL UNIT NOT READY, FORMAT IN PROGRESS. Handling of commands already in the task set is vendor-specific.

The PROGRESS INDICATION field returned in response to a REQUEST SENSE command (see SPC-3) may be used by the application client at any time during a format operation to poll the logical unit's progress. While a format operation is in progress unless an error has occurred, a device server shall respond to a REQUEST SENSE command by returning a sense key of NOT READY and an additional sense code of LOGICAL UNIT NOT READY, FORMAT IN PROGRESS with the sense key specific bytes set for progress indication (see SPC-3).

NOTE 7 - The MODE SELECT parameters, if any, should be set prior to issuing the FORMAT UNIT command.

A format protection information (FMTPINFO) bit set to zero specifies that the device server shall format the medium to the block length specified in the mode parameter block descriptor of the mode parameter header (see SPC-3). A FMTPINFO bit set to one specifies that the device server shall format the medium to the block length specified in the mode parameter block descriptor of the mode parameter header plus eight (e.g., if the block length is 512, then the formatted block length is 520). Following a successful format, the PROT_EN bit in the long read capacity data (see 5.15) indicates whether protection information (see 4.15) is enabled.

If the FMTPINFO bit is set to zero, the device server shall ignore the reference tag own request (RTO_REQ) bit. If the FMTPINFO bit is set to one and the RTO_REQ bit is set to one, the device server shall enable application client ownership of the LOGICAL BLOCK REFERENCE TAG field in protection information (see 4.15). If the FMTPINFO bit set to one and the RTO_REQ bit is set to zero, the device server shall disable application client ownership of the LOGICAL BLOCK REFERENCE TAG field. Following a successful format, the RTO_EN bit in the long read capacity data (see 5.15) indicates whether application client ownership of the LOGICAL BLOCK REFERENCE TAG field is enabled.

When protection information is written during a FORMAT UNIT command (i.e., the FMTPINFO bit is set to one) protection information shall be written to a default value of FFFFFFFF FFFFFFFFh.

A LONGLIST bit set to zero specifies that the parameter list, if any, contains a short parameter list header as defined in table 16. A LONGLIST bit set to one specifies that the parameter list, if any, contains a long parameter list header as defined in table 17.

A format data (FMTDATA) bit set to zero specifies that no parameter list be transferred from the application client data-out buffer. The source of defect information is not specified.

A FMTDATA bit set to one specifies that the FORMAT UNIT parameter list (see table 15) shall be transferred from the application client data-out buffer. The parameter list consists of a parameter list header, followed by an optional initialization pattern descriptor, followed by an optional defect list.

A complete list (CMPLST) bit set to zero specifies that the defect list sent by the application client shall be used in an addition to the existing list of defects. As a result, the device server shall construct a new GLIST (see 4.8) that contains:

a) the existing GLIST;
b) the DLIST, if it is sent by the application client; and
c) the CLIST, if certification is enabled (i.e., the device server may add any defects it detects during the format operation).

A CMPLST bit set to one specifies that the defect list sent by the application client is a complete list of defects. Any existing defect list except the PLIST shall be ignored by the device server. As a result, the device server shall construct a new GLIST (see 4.8) that contains:

a) the DLIST, if it is sent by the application client; and
b) the CLIST, if certification is enabled (i.e., the device server may add any defects it detects during the format operation).

The DEFECT LIST FORMAT field specifies the format of the address descriptors in the defect list if the FMTDATA bit is set to one (see table 14).

Table 14 defines the address descriptor usage for the FORMAT UNIT command.

Table 14 — FORMAT UNIT address descriptor usage

Field in the FORMAT UNIT CDB			DEFECT LIST LENGTH field in the parameter list header	Type [a]	Comments
FMTDATA	CMPLST	DEFECT LIST FORMAT			
0	0	000b	N/A	M	Vendor-specific defect information
1	0	000b (short block)	Zero	O	See notes [b] and [d]
1	1			O	See notes [b] and [e]
1	0		> 0	O	See notes [c] and [d]
1	1			O	See notes [b] and [e]
		011b (long block)	Zero	O	See notes [b] and [d]
				O	See notes [b] and [e]
1	0		> 0	O	See notes [c] and [d]
1	1			O	See notes [c] and [e]
1	0	100b (bytes from index)	Zero	O	See notes [b] and [d]
1	1			O	See notes [b] and [e]
1	0		> 0	O	See notes [c] and [d]
1	1			O	See notes [c] and [e]
1	0	101b (physical sector)	Zero	O	See notes [b] and [d]
1	1			O	See notes [b] and [e]
1	0		> 0	O	See notes [c] and [d]
1	1			O	See notes [c] and [e]
1	0	110b (vendor specific)	Vendor specific	O	
1	1			O	
All others				All remaining codes are reserved.	

Notes:
[a] M = implementation is mandatory. O = implementation is optional.
[b] No DLIST is included in the parameter list.
[c] A DLIST is included in the parameter list. The device server shall add the DLIST defects to the new GLIST.
[d] The device server shall add existing GLIST defects to the new GLIST (i.e., use the existing GLIST).
[e] The device server shall not add existing GLIST defects to the new GLIST (i.e., discard the existing GLIST).

All the options described in this table cause a new GLIST to be created during processing of the FORMAT UNIT command as described in the text.

5.4.2 FORMAT UNIT parameter list

5.4.2.1 FORMAT UNIT parameter list overview

Table 15 defines the FORMAT UNIT parameter list.

Table 15 — FORMAT UNIT parameter list

Byte\Bit	7	6	5	4	3	2	1	0
0 to 3 or 0 to 7	Parameter list header							
	Initialization pattern descriptor (if any)							
	Defect list (if any)							

The parameter list header is defined in 5.4.2.2.

The initialization pattern descriptor, if any, is defined in 5.4.2.3.

The defect list, if any, contains address descriptors (see 5.4.2.4) each specifying a location on the medium that the device server shall map out of the area accessible by application clients.

5.4.2.2 Parameter list header

The parameter list headers (see table 16 and table 17) provide several optional format control bits. Device servers that implement these bits provide the application client additional control over the use of the four defect sources, and the formatting operation. If the application client attempts to select any function not implemented by the device server, the device server shall terminate the command with CHECK CONDITION status and the sense key shall be set to ILLEGAL REQUEST with the additional sense code set to INVALID FIELD IN PARAMETER LIST.

Table 16 shows the short parameter list header, which is used if the LONGLIST bit is set to zero in the FORMAT UNIT CDB.

Table 16 — Short parameter list header

Byte\Bit	7	6	5	4	3	2	1	0
0	Reserved							
1	FOV	DPRY	DCRT	STPF	IP	Obsolete	IMMED	Vendor specific
2	(MSB)	DEFECT LIST LENGTH						
3								(LSB)

Table 17 shows the long parameter list header, which is used if the LONGLIST bit is set to one in the FORMAT UNIT CDB.

Table 17 — Long parameter list header

Byte\Bit	7	6	5	4	3	2	1	0
0	Reserved							
1	FOV	DPRY	DCRT	STPF	IP	Obsolete	IMMED	Vendor specific
2	Reserved							
3	Reserved							
4	(MSB)	DEFECT LIST LENGTH						
7								(LSB)

A format options valid (FOV) bit set to zero specifies that the device server shall use its default settings for the DPRY, DCRT, STPF, and IP bits. If the FOV bit is set to zero, the application client shall set these bits to zero. If the FOV bit is set to zero and any of the other bits in this paragraph are not set to zero, the device server shall terminate the command with CHECK CONDITION status and the sense key shall be set to ILLEGAL REQUEST with the additional sense code set to INVALID FIELD IN PARAMETER LIST.

A FOV bit set to one specifies that the device server shall examine the setting of the DPRY, DCRT, STPF, and IP bits. When the FOV bit is set to one, the DPRY, DCRT, STPF, and IP bits are defined as follows.

A disable primary (DPRY) bit set to zero specifies that the device server shall not use portions of the medium identified as defective in the PLIST for application client addressable logical blocks. If the device server is not able to locate the PLIST or it is not able to determine whether a PLIST exists, it shall perform the action specified by the STPF bit. A DPRY bit set to one specifies that the device server shall not use the PLIST to identify defective areas of the medium. The PLIST is not deleted.

A disable certification (DCRT) bit set to zero specifies that the device server shall perform a vendor-specific medium certification operation to generate a CLIST. A DCRT bit set to one specifies that the device server shall not perform any vendor-specific medium certification process or format verification operation while processing the FORMAT UNIT command.

The stop format (STPF) bit controls the behavior of the device server if one of the following events occurs:

a) The device server has been requested to use the PLIST (i.e., the DPRY bit is set to zero) or the GLIST (i.e., the CMPLST bit is set to zero) and the device server is not able to locate the list nor determine whether the list exists; or
b) The device server has been requested to use the PLIST (i.e., the DPRY bit is set to zero) or the GLIST (i.e., the CMPLST bit is set to zero), and the device server encounters an error while accessing the defect list.

A STPF bit set to zero specifies that, if one or both of these events occurs, the device server shall continue to process the FORMAT UNIT command. The device server shall return CHECK CONDITION status at the completion of the FORMAT UNIT command and the sense key shall be set to RECOVERED ERROR with the additional sense code set to either DEFECT LIST NOT FOUND if the first condition occurred, or DEFECT LIST ERROR if the second condition occurred.

A STPF bit set to one specifies that, if one or both of these events occurs, the device server shall terminate the FORMAT UNIT command with CHECK CONDITION status and the sense key shall be set to MEDIUM ERROR with the additional sense code set to either DEFECT LIST NOT FOUND if the first condition occurred, or DEFECT LIST ERROR if the second condition occurred.

NOTE 8 - The use of the FMTDATA bit, the CMPLST bit, and the parameter list header allow the application client to control the source of the defect lists used by the FORMAT UNIT command. Setting the DEFECT LIST

LENGTH field to zero allows the application client to control the use of PLIST and CLIST without having to specify a DLIST.

An initialization pattern (IP) bit set to zero specifies that an initialization pattern descriptor is not included and that the device server shall use its default initialization pattern. An IP bit set to one specifies that an initialization pattern descriptor (see 5.4.2.3) is included in the FORMAT UNIT parameter list following the parameter list header.

An immediate (IMMED) bit set to zero specifies that status shall be returned after the format operation has completed. An IMMED bit value set to one specifies that the device server shall return status after the entire parameter list has been transferred.

The DEFECT LIST LENGTH field in the parameter list header specifies the total length in bytes of the defect list (i.e., the address descriptors) that follow and does not include the initialization pattern descriptor, if any. The length of the defect list varies with the format of the address descriptors. The formats for the address descriptor(s) are shown in 5.4.2.4.

Short block format address descriptors and long block format address descriptors should be in ascending order. Bytes from index format address descriptors and physical sector format address descriptors shall be in ascending order. More than one physical or logical block may be affected by each address descriptor. If the address descriptors are not in the required order, the device server shall return CHECK CONDITION status with a sense key of ILLEGAL REQUEST and an additional sense code of INVALID FIELD IN PARAMETER LIST.

5.4.2.3 Initialization pattern descriptor

The initialization pattern descriptor specifies that the device server initialize logical blocks to a specified pattern. The initialization pattern descriptor (see table 18) is sent to the device server as part of the FORMAT UNIT parameter list.

Table 18 — Initialization pattern descriptor

Byte\Bit	7	6	5	4	3	2	1	0
0	IP MODIFIER		SI	Reserved				
1	INITIALIZATION PATTERN TYPE							
2	(MSB)							
3	INITIALIZATION PATTERN LENGTH (n - 3)							(LSB)
4								
n	INITIALIZATION PATTERN							

The IP MODIFIER field specifies the type and location of a header that modifies the initialization pattern (see table 19).

Table 19 — Initialization pattern modifier

IP modifier	Description
00b	No header. The device server shall not modify the initialization pattern.
01b	The device server shall overwrite the initialization pattern to write the LBA in the first four bytes of each logical block. The LBA shall be written with the most significant byte first. If the LBA is larger than four bytes, the least significant four bytes shall be written ending with the least significant byte.
10b	The device server shall overwrite the initialization pattern to write the LBA in the first four bytes of each physical block contained within the logical block. The lowest numbered logical block or part thereof that occurs within the physical block is used. The LBA shall be written with the most significant byte first. If the LBA is larger than four bytes the least significant four bytes shall be written ending with the least significant byte.
11b	Reserved.

A security initialize (SI) bit set to one specifies that the device server shall attempt to write the initialization pattern to all areas of the medium including those that may have been reassigned (i.e. are in a defect list). An SI bit set to one shall take precedence over any other FORMAT UNIT CDB field. The initialization pattern shall be written using a security erasure write technique. Application clients may choose to use this command multiple times to fully erase the previous data. Such security erasure write technique procedures are outside the scope of this standard. The exact requirements placed on the security erasure write technique are vendor-specific. The intent of the security erasure write is to render any previous user data unrecoverable by any analog or digital technique.

An SI bit set to zero specifies that the device server shall initialize the application client accessible area of the medium. The device server is not required to initialize other areas of the medium. However, the device server shall format the medium as defined in the FORMAT UNIT command.

When the SI bit is set to one, the device server need not write the initialization pattern over the header and other header and other parts of the medium not previously accessible to the application client. If the device server is unable to write over any area of the medium that is currently accessible to the application client or may be made accessible to the application client in the future (e.g., by clearing the defect list), it shall terminate the command with CHECK CONDITION status and the sense key shall be set to MEDIUM ERROR with the appropriate additional sense code for the condition. The device server shall attempt to rewrite all remaining parts of the medium even if some portions are not able to be rewritten.

The INITIALIZATION PATTERN TYPE field (see table 20) specifies the type of pattern the device server shall use to initialize each logical block within the application client accessible portion of the medium. All bytes within a

logical block shall be written with the initialization pattern. The initialization pattern is modified by the IP MODIFIER field as described in table 20.

Table 20 — Initialization pattern type

Initialization pattern type	Description
00h	Use default pattern [a]
01h	Repeat the initialization pattern as required to fill the logical block [b]
02h - 7Fh	Reserved
80h - FFh	Vendor-specific

Notes:
[a] If the initialization pattern length is not zero the device server shall terminate the command with CHECK CONDITION status and the sense key shall be set to ILLEGAL REQUEST with the additional sense code set to INVALID FIELD IN PARAMETER LIST.
[b] If the initialization pattern length is zero the device server shall terminate the command with CHECK CONDITION status and the sense key shall be set to ILLEGAL REQUEST with the additional sense code set to INVALID FIELD IN PARAMETER LIST.

The INITIALIZATION PATTERN LENGTH field specifies the number of bytes contained in the initialization pattern. If the length exceeds the current block length the device server shall terminate the command with CHECK CONDITION status and the sense key shall be set to ILLEGAL REQUEST with the additional sense code set to INVALID FIELD IN PARAMETER LIST. The pattern is modified by the IP MODIFIER field.

The INITIALIZATION PATTERN field specifies the initialization pattern.

5.4.2.4 Address descriptor formats

5.4.2.4.1 Address descriptor formats overview

This subclause describes the address descriptor formats used in the FORMAT UNIT command, the READ DEFECT DATA commands (see 5.16 and 5.17), and the Translate Address diagnostic pages (see 6.1.2 and 6.1.3) of the SEND DIAGNOSTIC and RECEIVE DIAGNOSTIC RESULTS commands.

The format type of an address descriptor is specified with:

a) the DEFECT LIST FORMAT field in the CDB, for the FORMAT UNIT and READ DEFECT DATA commands;
b) the SUPPLIED FORMAT field, for the Translate Address diagnostic pages; or
c) the TRANSLATE FORMAT field, for the Translate Address diagnostic pages.

Table 21 defines the types of address descriptors.

Table 21 — Address descriptor formats

Format type	Description	Reference
000b	Short block format address descriptor	5.4.2.4.2
011b	Long block format address descriptor	5.4.2.4.3
100b	Bytes from index format address descriptor	5.4.2.4.4
101b	Physical sector format address descriptor	5.4.2.4.5
110b	Vendor-specific	
All others	Reserved	

5.4.2.4.2 Short block format address descriptor

A format type of 000b specifies the short block format address descriptor defined in table 22.

Table 22 — Short block format address descriptor (000b)

Byte\Bit	7	6	5	4	3	2	1	0
0	(MSB)							
				SHORT BLOCK ADDRESS				
3								(LSB)

For the FORMAT UNIT command, the SHORT BLOCK ADDRESS field contains the four-byte LBA of a defect. For the READ DEFECT DATA command, the SHORT BLOCK ADDRESS field contains a vendor-specific four-byte value. For the Translate Address diagnostic pages, the SHORT BLOCK ADDRESS field contains a four-byte LBA or a vendor-specific four byte value above the capacity of the medium.

5.4.2.4.3 Long block format address descriptor

A format type of 011b specifies the long block format address descriptor defined in table 23.

Table 23 — Long block format address descriptor (011b)

Byte\Bit	7	6	5	4	3	2	1	0
0	(MSB)							
				LONG BLOCK ADDRESS				
7								(LSB)

For the FORMAT UNIT command, the LONG BLOCK ADDRESS field contains the eight-byte logical block address of a defect. For the READ DEFECT DATA command, the LONG BLOCK ADDRESS field contains a vendor-specific eight-byte value. For the Translate Address diagnostic pages, the LONG BLOCK ADDRESS field contains a four-byte LBA or a vendor-specific four byte value above the capacity of the medium.

5.4.2.4.4 Bytes from index format address descriptor

A format type of 100b specifies the bytes from index address descriptor defined in table 24. For the FORMAT UNIT command and READ DEFECT DATA command, this descriptor specifies the location of a defect that is either the length of one track or is no more than eight bytes long. For the Translate Address diagnostic pages, this descriptor specifies the location of a track or the first byte or last byte of an area.

Table 24 — Bytes from index format address descriptor (100b)

Byte\Bit	7	6	5	4	3	2	1	0
0	(MSB)							
				CYLINDER NUMBER				
2								(LSB)
3	HEAD NUMBER							
4	(MSB)							
				BYTES FROM INDEX				
7								(LSB)

The CYLINDER NUMBER field contains the cylinder number.

The HEAD NUMBER field contains the head number.

The BYTES FROM INDEX field contains the number of bytes from the index (e.g., from the start of the track) to the location being described. A BYTES FROM INDEX field set to FFFFFFFFh specfiies that the entire track is being described by this descriptor.

For sorting bytes from index format address descriptors, the cylinder number is the most significant part of the address and the bytes from index is the least significant part of the address. More than one logical block may be described by this descriptor.

5.4.2.4.5 Physical sector format address descriptor

A format type of 101b specifies the physical sector address descriptor defined in table 25. For the FORMAT UNIT command and READ DEFECT DATA command, this descriptor specifies the location of a defect that is either the length of one track or the length of a sector. For the Translate Address diagnostic pages, this descriptor specifies the location of a track or sector.

Table 25 — Physical sector format address descriptor (101b)

Byte\Bit	7	6	5	4	3	2	1	0
0	(MSB)							
				CYLINDER NUMBER				
2								(LSB)
3				HEAD NUMBER				
4	(MSB)							
				SECTOR NUMBER				
7								(LSB)

The CYLINDER NUMBER field contains the cylinder number.

The HEAD NUMBER field contains the head number.

The SECTOR NUMBER field contains the sector number. A SECTOR NUMBER field set to FFFFFFFFh specifies that the entire track is being described.

For sorting physical sector format address descriptors, the cylinder number is the most significant part of the address and the sector number is the least significant part of the address. More than one logical block may be described by this descriptor.

5.5 LOCK UNLOCK CACHE (10) command

The LOCK UNLOCK CACHE (10) command (see table 26) requests that the device server disallow or allow logical blocks within the specified range to be removed from the volatile cache memory and/or the non-volatile cache memory by the device server's cache replacement algorithm. Locked logical blocks may be written to the medium when modified, but a copy of the modified logical block shall remain in the cache memory.

Table 26 — LOCK UNLOCK CACHE (10) command

Byte\Bit	7	6	5	4	3	2	1	0
0	OPERATION CODE (36h)							
1	Reserved						LOCK	Obsolete
2	(MSB)							
				LOGICAL BLOCK ADDRESS				
5								(LSB)
6	Reserved							
7	(MSB)							
				NUMBER OF BLOCKS				
8								(LSB)
9	CONTROL							

A LOCK bit set to zero specifies that all logical blocks in the specified range that are currently locked into the cache memory shall be unlocked and may or may not be removed. A LOCK bit set to one specifies that any logical block in the specified range that is currently present in the cache memory shall be locked into cache memory. Only logical blocks that are already present in the cache memory are actually locked.

The LOGICAL BLOCK ADDRESS field specifies the first logical block of the range of logical blocks for this command. If the logical block address exceeds the capacity of the medium the device server shall return CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to LOGICAL BLOCK ADDRESS OUT OF RANGE.

The NUMBER OF BLOCKS field specifies the number of contiguous logical blocks within the range. A NUMBER OF BLOCKS field set to zero specifies that the range contains all remaining logical blocks on the medium. Any other value specifies the number of logical blocks within the range. If the logical block address plus the number of blocks exceeds the capacity of the medium, the device server shall return CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to LOGICAL BLOCK ADDRESS OUT OF RANGE.

Multiple locks may be in effect from more than one initiator port. Locks from different initiator ports may overlap. An unlock of an overlapped area does not release the lock of another initiator port.

5.6 LOCK UNLOCK CACHE (16) command

The LOCK UNLOCK CACHE (16) command (see table 27) requests that the device server disallow or allow logical blocks within the specified range to be removed from the volatile cache memory and/or non-volatile cache memory by the device server's cache replacement algorithm.

Table 27 — LOCK UNLOCK CACHE (16) command

Byte\Bit	7	6	5	4	3	2	1	0
0	OPERATION CODE (92h)							
1	Reserved						LOCK	Reserved
2	(MSB)							
9				LOGICAL BLOCK ADDRESS				(LSB)
10	(MSB)							
13				NUMBER OF BLOCKS				(LSB)
14	Reserved							
15	CONTROL							

See the LOCK UNLOCK CACHE (10) command (see 5.5) for a description of the fields in this command.

5.7 PRE-FETCH (10) command

The PRE-FETCH (10) command (see table 28) requests that the device server transfer the specified logical blocks from the medium to the volatile cache memory and/or non-volatile cache memory. No data shall be transferred to the application client.

Table 28 — PRE-FETCH (10) command

Byte\Bit	7	6	5	4	3	2	1	0
0	OPERATION CODE (34h)							
1	Reserved						IMMED	Obsolete
2	(MSB)							
5	LOGICAL BLOCK ADDRESS							(LSB)
6	Reserved			GROUP NUMBER				
7	(MSB)							
8	PREFETCH LENGTH							(LSB)
9	CONTROL							

An immediate (IMMED) bit set to zero specifies that status shall be returned after the operation is complete. An IMMED bit set to one specifies that status shall be returned as soon as the CDB has been validated.

See the LOCK UNLOCK CACHE (10) command (see 5.5) for a definition of the LOGICAL BLOCK ADDRESS field.

The GROUP NUMBER field specifies the group into which attributes associated with the command should be collected (see 4.16). A group number value of zero specifies that any attributes associated with the command shall not be collected into any group.

The PREFETCH LENGTH field specifies the number of contiguous logical blocks of data that shall be transferred to the block device's cache memory from the medium. A PREFETCH LENGTH field set to zero specifies that all remaining logical blocks shall be transferred to the cache memory. Any other value specifies the number of logical blocks that shall be transferred. If the logical block address plus the prefetch length exceeds the capacity of the medium, the device server shall return CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to LOGICAL BLOCK ADDRESS OUT OF RANGE.The device server is not required to transfer logical blocks that already are contained in the cache memory.

If the IMMED bit is set to zero and the specified logical blocks were successfully transferred to the cache memory, the device server shall return:

a) CONDITION MET status if the LINK bit is set to zero in the CONTROL byte (see SPC-3); or
b) INTERMEDIATE-CONDITION MET status if the LINK bit is set to one.

If the IMMED bit is set to zero and the unlocked cache memory does not have sufficient capacity to accept all of the specified logical blocks, the device server shall transfer to the cache memory as many of the specified logical blocks that fit. If these logical blocks are transferred successfully it shall return:

a) GOOD status if the LINK bit is set to zero in the CONTROL byte (see SPC-3); or
b) INTERMEDIATE status if the LINK bit is set to one.

If the IMMED bit is set to one and the unlocked cache memory has sufficient capacity to accept all of the specified logical blocks, the device server shall return:

a) CONDITION MET status if the LINK bit is set to zero in the CONTROL byte (see SPC-3); or
b) INTERMEDIATE-CONDITION MET status if the LINK bit is set to ine.

If the IMMED bit is set to one and the unlocked cache memory does not have sufficient capacity to accept all of the specified logical blocks, the device server shall return:

a) GOOD status if the LINK bit is set to zero in the CONTROL byte (see SPC-3); or
b) INTERMEDIATE status if the LINK bit is set to one.

If the IMMED bit is set to zero and one or more of the specified logical blocks were not successfully transferred to the cache memory for reasons other than lack of cache memory capacity, the device server shall return CHECK CONDITION status with the appropriate sense key and additional sense code. If the IMMED bit is set to one and one or more of the specified logical blocks were not successfully transferred to the cache memory for reasons other than lack of cache memory capacity, the device server shall report a deferred error (see SPC-3).

5.8 PRE-FETCH (16) command

The PRE-FETCH (16) command (see table 29) requests that the device server transfer the specified logical blocks from the medium to the volatile cache memory and/or non-volatile cache memory. No data shall be transferred to the application client.

Table 29 — PRE-FETCH (16) command

Byte\Bit	7	6	5	4	3	2	1	0
0	OPERATION CODE (90h)							
1	Reserved						IMMED	Reserved
2	(MSB)							
9	LOGICAL BLOCK ADDRESS							(LSB)
10	(MSB)							
13	PREFETCH LENGTH							(LSB)
14	Reserved			GROUP NUMBER				
15	CONTROL							

See the PRE-FETCH (10) command (see 5.7) for a description of the fields in this command.

5.9 READ (6) command

The READ (6) command (see table 30) requests that the device server transfer data, including user data but not including protection information, to the application client. The most recent data value written, or to be written if cached, in the addressed logical block shall be returned.

Table 30 — READ (6) command

Byte\Bit	7	6	5	4	3	2	1	0
0	OPERATION CODE (08h)							
1	Reserved			(MSB)				
2	LOGICAL BLOCK ADDRESS							
3								(LSB)
4	TRANSFER LENGTH							
5	CONTROL							

The cache control bits (see 5.10) are not provided for this command. Block devices with cache memory may have values for the cache control bits that affect the READ (6) command; however, no default values are defined by this standard. If explicit control is required, the READ (10) command should be used.

The LOGICAL BLOCK ADDRESS field specifies the logical block where the read operation shall begin.

The TRANSFER LENGTH field specifies the number of contiguous logical blocks of data to be transferred. A TRANSFER LENGTH field set to zero specifies that 256 logical blocks shall be transferred. Any other value specifies the number of logical blocks that shall be transferred. If the logical block address plus the transfer length exceeds the capacity of the medium, the device server shall return CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to LOGICAL BLOCK ADDRESS OUT OF RANGE. The TRANSFER LENGTH field is constrained by the MAXIMUM TRANSFER LENGTH field in the Block Limits VPD page (see 6.4.2).

NOTE 9 - Although the READ (6) command is limited to directly addressing logical blocks up to a capacity of 2 Gigabytes, for block lengths of 512 bytes, this command has been maintained as mandatory since some system initialization routines require that the READ (6) command be used. Application clients should migrate from the READ (6) command to the READ (10) command which may address 2 Terabytes with block lengths of 512 bytes, or the READ (16) command to address more than 2 Terabytes.

NOTE 10 - For the READ (10) command, READ (12) command, READ (16) command, and READ (32) command, a TRANSFER LENGTH field set to zero specifies that no logical blocks are transferred.

The device server shall check the protection information read from the medium before returning status for the command as described in table 31.

Table 31 — Protection information checking for READ (6)

Logical unit formatted with protection information	Shall device server transmit protection information?	Field in protection information [e]	Extended INQUIRY Data VPD page bit value [d]	If check fails [b] [c], additional sense code
Yes	No	LOGICAL BLOCK GUARD	GRD_CHK = 1	LOGICAL BLOCK GUARD CHECK FAILED
			GRD_CHK = 0	No check performed
		LOGICAL BLOCK APPLICATION TAG	APP_CHK = 1 [a]	LOGICAL BLOCK APPLICATION TAG CHECK FAILED
			APP_CHK = 0	No check performed
		LOGICAL BLOCK REFERENCE TAG	REF_CHK = 1 [f]	LOGICAL BLOCK REFERENCE TAG CHECK FAILED
			REF_CHK = 0	No check performed
No		No protection information available to check		

[a] The device server checks the logical block application tag only if it has knowledge of the contents of the LOGICAL BLOCK APPLICATION TAG field. The method for acquiring this knowledge is not defined by this standard.

[b] If an error is reported the sense key shall be set to ABORTED COMMAND.

[c] If multiple errors occur, the selection of which error to report is not defined by this standard.

[d] See the Extended INQUIRY Data VPD page (see SPC-3) for a description of the GRD_CHK, APP_CHK, and REF_CHK bits.

[e] If the device server detects a LOGICAL BLOCK APPLICATION TAG field set to FFFFh, it shall not check any protection information in the associated logical block.

[f] If the RTO_EN bit is set to zero in the long read capacity data (see 5.15), the device server checks the logical block reference tag by comparing it to the lower 4 bytes of the LBA associated with the logical block. If the RTO_EN bit is set to one, the device server checks the logical block reference tag only if it has knowledge of the contents of the LOGICAL BLOCK REFERENCE TAG field. The method for acquiring this knowledge is not defined by this standard.

5.10 READ (10) command

The READ (10) command (see table 32) requests that the device server transfer data to the application client. Data includes user data and protection information, if any. The most recent data value written in the addressed logical block shall be returned.

Table 32 — READ (10) command

Byte\Bit	7	6	5	4	3	2	1	0
0	OPERATION CODE (28h)							
1	RDPROTECT			DPO	FUA	Reserved	FUA_NV	Obsolete
2	(MSB)							
5	LOGICAL BLOCK ADDRESS							(LSB)
6	Reserved			GROUP NUMBER				
7	(MSB)							
8	TRANSFER LENGTH							(LSB)
9	CONTROL							

See the LOCK UNLOCK CACHE (10) command (see 5.5) for a definition of the LOGICAL BLOCK ADDRESS field.

See the PRE-FETCH (10) command (see 5.7) and 4.16 for a description of the GROUP NUMBER field.

The device server shall check the protection information read from the medium before returning status for the command based on the RDPROTECT field as described in table 33.

Table 33 — RDPROTECT **field** (part 1 of 3)

Value	Logical unit formatted with protection information	Shall device server transmit protection information?	Field in protection information [h]	Extended INQUIRY Data VPD page bit value [g]	If check fails [d] [f], additional sense code
000b [i]	Yes	No	LOGICAL BLOCK GUARD	GRD_CHK = 1	LOGICAL BLOCK GUARD CHECK FAILED
				GRD_CHK = 0	No check performed
			LOGICAL BLOCK APPLICATION TAG	APP_CHK = 1 [c]	LOGICAL BLOCK APPLICATION TAG CHECK FAILED
				APP_CHK = 0	No check performed
			LOGICAL BLOCK REFERENCE TAG	REF_CHK = 1 [j]	LOGICAL BLOCK REFERENCE TAG CHECK FAILED
				REF_CHK = 0	No check performed
	No		No protection information available to check		

Table 33 — RDPROTECT **field** (part 2 of 3)

Value	Logical unit formatted with protection information	Shall device server transmit protection information?	Field in protection information [h]	Extended INQUIRY Data VPD page bit value [g]	If check fails [d] [f], additional sense code
001b [b] [i]	Yes	Yes [e]	LOGICAL BLOCK GUARD	GRD_CHK = 1	LOGICAL BLOCK GUARD CHECK FAILED
				GRD_CHK = 0	No check performed
			LOGICAL BLOCK APPLICATION TAG	APP_CHK = 1 [c]	LOGICAL BLOCK APPLICATION TAG CHECK FAILED
				APP_CHK = 0	No check performed
			LOGICAL BLOCK REFERENCE TAG	REF_CHK = 1	LOGICAL BLOCK REFERENCE TAG CHECK FAILED
				REF_CHK = 0	No check performed
	No [a]	No protection information available to transmit to the application client or for checking			
010b [b] [i]	Yes	Yes [e]	LOGICAL BLOCK GUARD	Shall not	No check performed
			LOGICAL BLOCK APPLICATION TAG	APP_CHK = 1 [c]	LOGICAL BLOCK APPLICATION TAG CHECK FAILED
				APP_CHK = 0	No check performed
			LOGICAL BLOCK REFERENCE TAG	REF_CHK = 1	LOGICAL BLOCK REFERENCE TAG CHECK FAILED
				REF_CHK = 0	No check performed
	No [a]	No protection information available to transmit to the application client or for checking			
011b [b] [i]	Yes	Yes [e]	LOGICAL BLOCK GUARD	Shall not	No check performed
			LOGICAL BLOCK APPLICATION TAG	Shall not	No check performed
			LOGICAL BLOCK REFERENCE TAG	Shall not	No check performed
	No [a]	No protection information available to transmit to the application client or for checking			

Table 33 — RDPROTECT **field** (part 3 of 3)

Value	Logical unit formatted with protection information	Shall device server transmit protection information?	Field in protection information [h]	Extended INQUIRY Data VPD page bit value [g]	If check fails [d] [f], additional sense code
100b [b] [i]	Yes	Yes [e]	LOGICAL BLOCK GUARD	GRD_CHK = 1	LOGICAL BLOCK GUARD CHECK FAILED
				GRD_CHK = 0	No check performed
			LOGICAL BLOCK APPLICATION TAG	Shall not	No check performed
			LOGICAL BLOCK REFERENCE TAG	Shall not	No check performed
	No [a]	No protection information available to transmit to the application client or for checking			
101b - 111b	Reserved				

[a] A read operation to a logical unit that supports protection information (see 4.15) and has not been formatted with protection information shall fail with a CHECK CONDITION status. The sense key shall be set to ILLEGAL REQUEST with the additional sense code set to INVALID FIELD IN CDB.

[b] If the logical unit does not support protection information the requested command should fail with CHECK CONDITION status with a sense key of ILLEGAL REQUEST and an additional sense code of INVALID FIELD IN CDB.

[c] The device server checks the logical block application tag only if it has knowledge of the contents of the LOGICAL BLOCK APPLICATION TAG field. This knowledge may be obtained by use of the READ (32) command (see 5.13) or by a method not defined by this standard.

[d] If an error is reported the sense key shall be set to ABORTED COMMAND.

[e] Transmit protection information to the application client.

[f] If multiple errors occur, the selection of which error to report is not defined by this standard.

[g] See the Extended INQUIRY Data VPD page (see SPC-3) for a description of the GRD_CHK, APP_CHK, and REF_CHK bits.

[h] If the application client or device server detects a LOGICAL BLOCK APPLICATION TAG field set to FFFFh, the checking of all protection information in the associated logical block shall be disabled.

[i] If the RTO_EN bit is set to zero in the long read capacity data (see 5.15), the device server may process the command. If the RTO_EN bit is set to one, READ (10), READ (12),and READ (16) commands with the RDPROTECT field set to 000b may be processed by the device server. If the RTO_EN bit is set to one, the device server shall terminate READ (10), READ (12), and READ (16) commands with the RDPROTECT field not set to 000b with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and an additional sense code of INVALID COMMAND OPERATION CODE.

[j] If the RTO_EN bit is set to zero in the long read capacity data (see 5.15), the device server checks the logical block reference tag by comparing it to the lower 4 bytes of the LBA associated with the logical block. If the RTO_EN bit is set to one, the device server checks the logical block reference tag only if it has knowledge of the contents of the LOGICAL BLOCK REFERENCE TAG field. The method for acquiring this knowledge is not defined by this standard.

A disable page out (DPO) bit set to zero specifies that the retention priority shall be determined by the RETENTION PRIORITY fields in the Caching mode page (see 6.3.3). A DPO bit set to one specifies that the device server shall assign the logical blocks accessed by this command the lowest retention priority for being fetched into or retained by the cache. A DPO bit set to one overrides any retention priority specified in the Caching

mode page. All other aspects of the algorithm implementing the cache memory replacement strategy are not defined by this standard.

NOTE 11 - The DPO bit is used to control replacement of logical blocks in the cache memory when the application client has information on the future usage of the logical blocks. If the DPO bit is set to one, the application client is specifying that the logical blocks accessed by the command are not likely to be accessed again in the near future and should not be put in the cache memory nor retained by the cache memory. If the DPO bit is set to zero, the application client is specifying that the logical blocks accessed by this command are likely to be accessed again in the near future.

The force unit access (FUA) and force unit access nonvolatile cache (FUA_NV) bits are defined in table 34.

Table 34 — Force unit access for reads

FUA	FUA_NV	**Description**
0	0	The device server may read the logical blocks from volatile cache, non-volatile cache, and/or the medium.
0	1	If the NV_SUP bit is set to one in the Extended INQUIRY Data VPD page (see SPC-3), the device server shall read the logical blocks from non-volatile cache or the medium. If a non-volatile cache is present and a volatile cache contains a more recent version of a logical block, the device server shall first write the logical block to: a) non-volatile cache; and/or b) the medium, before reading it. If the NV_SUP bit is set to zero in the Extended INQUIRY Data VPD page (see SPC-3), the device server may read the logical blocks from volatile cache, non-volatile cache, and/or the medium.
1	0 or 1	The device server shall read the logical blocks from the medium. If a cache contains a more recent version of a logical block, the device server shall first write the logical block to the medium before reading it.

The TRANSFER LENGTH field specifies the number of contiguous logical blocks of data that shall be transferred. A TRANSFER LENGTH field set to zero specifies that no logical blocks shall be transferred. This condition shall not be considered an error. Any other value specifies the number of logical blocks that shall be transferred. If the logical block address plus the transfer length exceeds the capacity of the medium, the device server shall return CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to LOGICAL BLOCK ADDRESS OUT OF RANGE. The TRANSFER LENGTH field is constrained by the MAXIMUM TRANSFER LENGTH field in the Block Limits VPD page (see 6.4.2).

NOTE 12 - For the READ (6) command, a TRANSFER LENGTH field set to zero specifies that 256 logical blocks are transferred.

5.11 READ (12) command

The READ (12) command (see table 35) requests that the device server transfer data to the application client from the medium. Data includes user data and protection information, if any.

Table 35 — READ (12) command

Byte\Bit	7	6	5	4	3	2	1	0
0	OPERATION CODE (A8h)							
1	RDPROTECT			DPO	FUA	Reserved	FUA_NV	Obsolete
2	(MSB)							
5	LOGICAL BLOCK ADDRESS							(LSB)
6	(MSB)							
9	TRANSFER LENGTH							(LSB)
10	Restricted for MMC-4	Reserved		GROUP NUMBER				
11	CONTROL							

See the READ (10) command (see 5.10) for a description of the fields in this command.

5.12 READ (16) command

The READ (16) command (see table 36) requests that the device server transfer data to the application client from the medium. Data includes user data and protection information, if any.

Table 36 — READ (16) command

Byte\Bit	7	6	5	4	3	2	1	0
0	OPERATION CODE (88h)							
1	RDPROTECT			DPO	FUA	Reserved	FUA_NV	Reserved
2	(MSB)							
9	LOGICAL BLOCK ADDRESS							(LSB)
10	(MSB)							
13	TRANSFER LENGTH							(LSB)
14	Restricted for MMC-4	Reserved		GROUP NUMBER				
15	CONTROL							

See the READ (10) command (see 5.10) for a description of the fields in this command.

5.13 READ (32) command

The READ (32) command (see table 37) requests that the device server transfer data to the application client from the medium. Data includes user data and protection information, if any.

Table 37 — READ (32) command

Byte\Bit	7	6	5	4	3	2	1	0
0	OPERATION CODE (7Fh)							
1	CONTROL							
2	Reserved							
5								
6	Reserved			GROUP NUMBER				
7	ADDITIONAL CDB LENGTH (18h)							
8	(MSB)							
9	SERVICE ACTION (0009h)							(LSB)
10	RDPROTECT			DPO	FUA	Reserved	FUA_NV	Reserved
12	(MSB)							
19	LOGICAL BLOCK ADDRESS							(LSB)
20	(MSB)							
23	INITIAL LOGICAL BLOCK REFERENCE TAG							(LSB)
24	(MSB)							
25	EXPECTED LOGICAL BLOCK APPLICATION TAG							(LSB)
26	(MSB)							
27	LOGICAL BLOCK APPLICATION TAG MASK							(LSB)
28	(MSB)							
31	TRANSFER LENGTH							(LSB)

See the READ (10) command (see 5.10) for a description of the GROUP NUMBER field, RDPROTECT field, DPO bit, FUA bit, FUA_NV bit, LOGICAL BLOCK ADDRESS field, and TRANSFER LENGTH field.

If the RTO_EN bit is set to zero in the long read capacity data (see 5.15), the device server shall terminate the command with CHECK CONDITION status with a sense key of ILLEGAL REQUEST and an additional sense code set to INVALID COMMAND OPERATION CODE. If the RTO_EN bit is set to one, the device server may process the command.

The INITIAL LOGICAL BLOCK REFERENCE TAG field contains the value of the LOGICAL BLOCK REFERENCE TAG expected on the first logical block of the range of logical blocks for this command. The checking enables and requirements are controlled by the RDPROTECT field. See the READ (10) command (see 5.10) for a definition description of the RDPROTECT field checking enables and requirements.

When checking of the LOGICAL BLOCK APPLICATION TAG is enabled (see table 33 in 5.10), the EXPECTED LOGICAL BLOCK APPLICATION TAG field contains a value that is expected in the LOGICAL BLOCK APPLICATION TAG with the LOGICAL BLOCK APPLICATION TAG MASK applied in the protection information of logical blocks for this command.

When checking of the LOGICAL BLOCK APPLICATION TAG is enabled (see table 33 in 5.10), the LOGICAL BLOCK APPLICATION MASK field contains a value that is a bit mask for enabling the checking of the LOGICAL BLOCK APPLICATION TAG in the protection information for each logical block of the range of logical blocks for this

command. A LOGICAL BLOCK APPLICATION TAG MASK bit set to one enables the checking of the corresponding bit in the EXPECTED LOGICAL BLOCK APPLICATION TAG field with the LOGICAL BLOCK APPLICATION TAG.

5.14 READ CAPACITY (10) command

The READ CAPACITY (10) command (see table 38) provides a means for the application client to request information regarding the capacity of the block device. This command may be processed as if it has a HEAD OF QUEUE task attribute (see 4.7). If the logical unit supports protection information (see 4.15), the READ CAPACITY (16) command should be used instead of READ CAPACITY (10).

Table 38 — READ CAPACITY (10) command

Byte\Bit	7	6	5	4	3	2	1	0
0	OPERATION CODE (25h)							
1	Reserved							Obsolete
2	(MSB)							
5	LOGICAL BLOCK ADDRESS							(LSB)
6								
7	Reserved							
8	Reserved							PMI
9	CONTROL							

See the LOCK UNLOCK CACHE (10) command (see 5.5) for a definition of the LOGICAL BLOCK ADDRESS field.

The LOGICAL BLOCK ADDRESS field shall be zero if the PMI bit is set to zero. If the PMI bit is set to zero and the LOGICAL BLOCK ADDRESS field is not set to zero, the device server shall return a CHECK CONDITION status and the sense key shall be set to ILLEGAL REQUEST with the additional sense code set to INVALID FIELD IN CDB.

A partial medium indicator (PMI) bit set to zero specifies that the device server return information on the last logical block on the block device.

A PMI bit set to one specifies that the device server return information on the last logical block after that specified in the LOGICAL BLOCK ADDRESS field before a substantial vendor-specific delay in data transfer may be encountered.

NOTE 13 - This function is intended to assist storage management software in determining whether there is sufficient space starting with the logical block address specified in the CDB to contain a frequently accessed data structure (e.g., a file directory or file index) without incurring an extra delay.

The short read capacity data is defined in table 39.

Table 39 — Short read capacity data

Byte\Bit	7	6	5	4	3	2	1	0
0	(MSB)							
3	RETURNED LOGICAL BLOCK ADDRESS							(LSB)
4	(MSB)							
7	BLOCK LENGTH IN BYTES							(LSB)

If the number of logical blocks exceeds the maximum value that may be specified in the RETURNED LOGICAL BLOCK ADDRESS field, the device server shall set the RETURNED LOGICAL BLOCK ADDRESS field to FFFFFFFFh. The initiator should then issue a READ CAPACITY (16) command.

If the PMI bit is set to zero, the device server shall set the RETURNED LOGICAL BLOCK ADDRESS field to the lower of:

a) the LBA of the last logical block on the block device; or
b) FFFFFFFFh.

If the PMI bit is set to one, the device server shall set the RETURNED LOGICAL BLOCK ADDRESS field to the lower of:

a) the last LBA after that specified in the LOGICAL BLOCK ADDRESS field of the CDB before a substantial vendor-specific delay in data transfer may be encountered; or
b) FFFFFFFFh.

The RETURNED LOGICAL BLOCK ADDRESS shall be greater than or equal to that specified by the LOGICAL BLOCK ADDRESS field in the CDB.

The BLOCK LENGTH IN BYTES field contains the number of bytes of user data in the logical block indicated by the RETURNED LOGICAL BLOCK ADDRESS field. This value does not include protection information or additional information (e.g., ECC bytes) recorded on the medium.

5.15 READ CAPACITY (16) command

The READ CAPACITY (16) command (see table 40) provides a means for the application client to request information regarding the capacity of the block device. This command is implemented as a service action of the SERVICE ACTION IN operation code. This command may be processed as if it has a HEAD OF QUEUE task attribute (see 4.7)

Table 40 — READ CAPACITY (16) command

Byte\Bit	7	6	5	4	3	2	1	0
0	OPERATION CODE (9Eh)							
1	Reserved			SERVICE ACTION (10h)				
2	(MSB)	LOGICAL BLOCK ADDRESS						
9								(LSB)
10	(MSB)	ALLOCATION LENGTH						
13								(LSB)
14	Reserved							PMI
15	CONTROL							

See the LOCK UNLOCK CACHE (10) command (see 5.5) for a definition of the LOGICAL BLOCK ADDRESS field. See the READ CAPACITY (10) command (see 5.14) for a description of the other fields in this command.

The long read capacity data is defined in table 41.

Table 41 — Long read capacity data

Byte\Bit	7	6	5	4	3	2	1	0
0	(MSB)							
7		RETURNED LOGICAL BLOCK ADDRESS						(LSB)
8	(MSB)							
11		BLOCK LENGTH IN BYTES						(LSB)
12	Reserved						RTO_EN	PROT_EN
13								
31		Reserved						

The RETURNED LOGICAL BLOCK ADDRESS field and BLOCK LENGTH IN BYTES field of the long read capacity data are the same as the in the short read capacity data described in the READ CAPACITY (10) command (see 5.14). The maximum value that shall be returned in the RETURNED LOGICAL BLOCK ADDRESS field is FFFFFFFF FFFFFFFEh.

A reference tag own enable (RTO_EN) bit set to one indicates that application client ownership of the LOGICAL BLOCK REFERENCE TAG field in protection information is enabled (i.e., the medium was formatted with protection information (see 4.15) enabled and the RTO_REQ bit was set to one). An RTO_EN bit set to zero indicates that application client ownership of the LOGICAL BLOCK REFERENCE TAG field in protection information is disabled.

A PROT_EN bit set to one indicates that the medium was formatted with protection information (see 4.15) enabled. A PROT_EN bit set to zero indicates that the medium was not formatted with protection information enabled.

5.16 READ DEFECT DATA (10) command

The READ DEFECT DATA (10) command (see table 42) requests that the device server transfer the medium defect data to the application client.

Table 42 — READ DEFECT DATA (10) command

Byte\Bit	7	6	5	4	3	2	1	0
0	OPERATION CODE (37h)							
1	Reserved							
2	Reserved			REQ_PLIST	REQ_GLIST	DEFECT LIST FORMAT		
3								
6		Reserved						
7	(MSB)							
8		ALLOCATION LENGTH						(LSB)
9	CONTROL							

If the device server is unable to access the medium defect data, it shall terminate the command with CHECK CONDITION status. The sense key shall be set to either MEDIUM ERROR, if a medium error occurred, or NO SENSE, if the list does not exist. The additional sense code set to DEFECT LIST NOT FOUND.

NOTE 14 - Some device servers may not be able to return medium defect data until after a FORMAT UNIT command (see 5.4) has been completed successfully.

A request primary defect list (REQ_PLIST) bit set to zero specifies that the device server shall not return the PLIST. A REQ_PLIST bit set to one specifies that the device server shall return the PLIST, if any.

A request grown defect list (REQ_GLIST) bit set to zero specifies that the device server shall not return the GLIST. A REQ_GLIST bit set to one specifies that the device server shall return the GLIST, if any.

A REQ_PLIST bit set to zero and a REQ_GLIST bit set to zero specifies that the device server shall return only the defect list header (i.e., the first four bytes of the defect list).

A REQ_PLIST bit set to one and a REQ_GLIST bit set to one specifies that the device server shall return both the PLIST and GLIST, if any. The order the lists are returned in is vendor-specific. Whether the lists are merged or not is vendor-specific.

The DEFECT LIST FORMAT field is used by the application client to specify the preferred format for the defect list. This field is intended for those device servers capable of returning more than one format, as defined in the FORMAT UNIT command (see 5.4.2.4). A device server unable to return the requested format shall return the defect list in its default format (see the DEFECT LIST FORMAT field in the defect list header).

If the requested defect list format and the returned defect list format are not the same, the device server shall transfer the defect data and then terminate the command with CHECK CONDITION status and the sense key shall be set to RECOVERED ERROR with the additional sense code set to DEFECT LIST NOT FOUND.

The READ DEFECT DATA (10) defect list (see table 43) contains a four-byte header, followed by zero or more address descriptors.

Table 43 — READ DEFECT DATA (10) defect list

Byte\Bit	7	6	5	4	3	2	1	0
0	Reserved							
1	Reserved			PLISTV	GLISTV	DEFECT LIST FORMAT		
2	(MSB)							
3	DEFECT LIST LENGTH (n - 3)							(LSB)
	Defect list (if any)							
4	Address descriptor(s) (if any)							
n								

A PLIST valid (PLISTV) bit set to zero indicates that the data returned does not contain the PLIST. A PLISTV bit set to one indicates that the data returned contains the PLIST.

A GLIST valid (GLISTV) bit set to zero indicates that the data returned does not contain the GLIST. A GLISTV bit set to one indicates that the data returned contains the GLIST.

The DEFECT LIST FORMAT field indicates the format of the address descriptors returned by the device server. This field is defined in the FORMAT UNIT command (see 5.4.2.4).

Short block format address descriptors and long block format address descriptors returned with this command are vendor-specific. Physical sector format address descriptors may or may not include defects in areas not accessible to the application client. Bytes from index format address descriptors shall comprise a complete list of the defects. A complete list of the defects may include defects in areas not within the capacity returned in the READ CAPACITY command.

NOTE 15 - The use of the short block format or the long block format is not recommended for this command. There is no standard model that defines the meaning of the block address of a defect. In the usual case, a defect that has been reassigned no longer has an LBA.

The DEFECT LIST LENGTH field indicates the length in bytes of the address descriptors that follow. The DEFECT LIST LENGTH is equal to four or eight times the number of the address descriptors, depending on the format of the returned address descriptors (see 5.4.2.4).

If the number of address descriptors the SCSI device has assigned does not exceed the capability of the ALLOCATION LENGTH field size but contains a value that is insufficient to transfer all of the address descriptors the defect list length shall not be adjusted to reflect the truncation and the device server shall not create a CHECK CONDITION status. The application client is responsible for comparing the defect list length and the allocation length to determine that a partial list was received. If the number of address descriptors the SCSI device has assigned exceeds the capability of the ALLOCATION LENGTH field size the device server shall transfer no data and return a CHECK CONDITION status and the sense key shall be set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.

NOTE 16 - The application client may determine the length of the defect list by sending the READ DEFECT DATA (10) command with an ALLOCATION LENGTH of four. The device server returns the defect list header that contains the length of the defect list.

The address descriptors may or may not be sent in ascending order. The application client may determine the exact number of the defects by dividing the DEFECT LIST LENGTH by the length of a single address descriptor for the returned format.

5.17 READ DEFECT DATA (12) command

The READ DEFECT DATA (12) command (see table 44) requests that the device server transfer the medium defect data to the application client.

Table 44 — READ DEFECT DATA (12) command

Byte\Bit	7	6	5	4	3	2	1	0
0	OPERATION CODE (B7h)							
1	Reserved			REQ_PLIST	REQ_GLIST	DEFECT LIST FORMAT		
2	Reserved							
5								
6	(MSB)							
9	ALLOCATION LENGTH							(LSB)
10	Reserved							
11	CONTROL							

See the READ DEFECT DATA (10) command (5.15) for a description of the fields in this command.

The READ DEFECT DATA (12) defect list (see table 45) contains an eight byte header, followed by zero or more address descriptors.

Table 45 — READ DEFECT DATA (12) defect list

Byte\Bit	7	6	5	4	3	2	1	0
0	Reserved							
1	Reserved			PLISTV	GLISTV	DEFECT LIST FORMAT		
2	Reserved							
3	Reserved							
4	(MSB)							
7	DEFECT LIST LENGTH (n - 7)							(LSB)
Defect list (if any)								
8	Address descriptor(s) (if any)							
n								

See the description of the READ DEFECT DATA (10) list header (see 5.16) for a description of the fields in this header.

5.18 READ LONG (10) command

The READ LONG (10) command (see table 46) requests that the device server transfer data from a single logical block to the application client. The data passed during the READ LONG (10) command is vendor-specific, but shall include the following items recorded on the medium:

a) user data or transformed user data;
b) protection information or transformed protection information, if any; and
c) any additional information (e.g., ECC bytes).

The most recent data written, or to be written, in the addressed logical block shall be returned. The values in the Read-Write Error Recovery mode page (see 6.3.4) do not apply to this command. The device server may perform retries while processing this command.

Table 46 — READ LONG (10) command

Byte\Bit	7	6	5	4	3	2	1	0
0	OPERATION CODE (3Eh)							
1	Reserved						CORRCT	Obsolete
2	(MSB)							
5	LOGICAL BLOCK ADDRESS							(LSB)
6	Reserved							
7	(MSB)							
8	BYTE TRANSFER LENGTH							(LSB)
9	CONTROL							

See the LOCK UNLOCK CACHE (10) command (see 5.5) for a definition of the LOGICAL BLOCK ADDRESS field.

If the additional bytes contain an ECC code, any other additional bytes that are correctable by ECC should be included (e.g., data synchronization mark within the area covered by ECC). It is not required for the ECC

bytes to be at the end of the data bytes; however, they should be in the same order as they are on the medium.

A correct (CORRCT) bit set to zero specifies that a logical block be read without any correction made by the device server. A CORRCT bit set to zero should result in GOOD status unless data is not transferred for some reason other than that the data is non-correctable. In this case the appropriate status and/or sense data shall be set. A CORRCT bit set to one specifies that the data be corrected by ECC before being transferred to the application client data-in buffer.

The BYTE TRANSFER LENGTH field specifies the number of bytes of data that shall be transferred from the specified logical block. If the BYTE TRANSFER LENGTH field is not set to zero and does not match the available data length, 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. In the sense data (see SPC-3), the VALID and ILI bits shall each be set to one and the INFORMATION field shall be set to the difference (i.e., residue) of the requested byte transfer length minus the actual available data length in bytes. Negative values shall be indicated by two's complement notation.

A BYTE TRANSFER LENGTH field set to zero specifies that no bytes shall be transferred. This condition shall not be considered an error.

5.19 READ LONG (16) command

The READ LONG (16) command (see table 47) requests that the device server transfer data from a single logical block to the application client. This command is implemented as a service action of the SERVICE ACTION IN operation code.

Table 47 — READ LONG (16) command

Byte\Bit	7	6	5	4	3	2	1	0
0	OPERATION CODE (9Eh)							
1	Reserved			SERVICE ACTION (11h)				
2	(MSB)							
9	LOGICAL BLOCK ADDRESS							(LSB)
10								
11	Reserved							
12	(MSB)							
13	BYTE TRANSFER LENGTH							(LSB)
14	Reserved							CORRCT
15	CONTROL							

See the READ LONG (10) command (see 5.18) for a description of the fields in this command.

5.20 REASSIGN BLOCKS command

The REASSIGN BLOCKS command (see table 48) requests the device server to reassign the defective logical blocks to another area on the medium set aside for this purpose. The device server should also record the location of the defective logical blocks in the GLIST if such a list is supported. More than one physical or logical block may be relocated by each address descriptor sent by the application client. This command shall not alter the contents of the PLIST (see 4.8).

The application client transfers a defect list that contains the LBAs to be reassigned. The device server shall reassign the parts of the medium used for each LBA in the list. If the device server is able to recover user data and protection information, if present, from the original logical block, it shall write the recovered data and any protection information to the reassigned logical block. If the device server is unable to recover user data and

protection information, if present, it shall write vendor specific data for user data and protection information, if enabled, to a default value of FFFFFFFF FFFFFFFFh. The data in all other logical blocks on the medium shall be preserved.

NOTE 17 - The effect of specifying a logical block to be reassigned that previously has been reassigned is to reassign the logical block again. Although not likely, over the life of the medium, a logical block may be assigned to multiple physical block addresses until no more spare locations remain on the medium.

Table 48 — REASSIGN BLOCKS command

Byte\Bit	7	6	5	4	3	2	1	0
0	OPERATION CODE (07h)							
1	Reserved						LONGLBA	LONGLIST
2	Reserved							
4								
5	CONTROL							

A long LBA (LONGLBA) bit set to zero specifies that the REASSIGN BLOCKS defect list contains four byte address descriptors. A LONGLBA bit set to one specifes that the REASSIGN BLOCKS defect list contains eight byte address descriptors.

The REASSIGN BLOCKS defect list (see table 49) contains a four-byte header followed by one or more address descriptors.

Table 49 — REASSIGN BLOCKS defect list

Byte\Bit	7	6	5	4	3	2	1	0
0	DEFECT LIST LENGTH HEADER							
3								
4	Address descriptor(s) (if any)							
n								

If LONGLIST is set to zero, the header is defined in table 50.

Table 50 — REASSIGN BLOCKS short defect list header

Byte\Bit	7	6	5	4	3	2	1	0
0	Reserved							
1								
2	(MSB) DEFECT LIST LENGTH							
3								(LSB)

If LONGLIST is set to one, the header is defined in table 51.

Table 51 — REASSIGN BLOCKS long defect list header

Byte\Bit	7	6	5	4	3	2	1	0
0	(MSB) DEFECT LIST LENGTH							
3								(LSB)

The DEFECT LIST LENGTH field indicates the total length in bytes of the address descriptors that follow. The DEFECT LIST LENGTH field does not include the defect list header length and is equal to either:

a) four times the number of address descriptors, if the LONGLBA bit is set to zero; or
b) eight times the number of address descriptors, if the LONGLBA bit is set to one.

The address descriptor contains the LBA of the defect. The LBA is a four-byte field if the LONGLBA bit is set to zero or an eight-byte field if the LONGLBA bit is set to one. The address descriptors shall be in ascending order.

If the block device has insufficient capacity to reassign all of the logical blocks specified in the address descriptors, the command shall terminate with CHECK CONDITION status and the sense key shall be set to HARDWARE ERROR with the additional sense code set to NO DEFECT SPARE LOCATION AVAILABLE.

If the block device is unable to successfully complete a REASSIGN BLOCKS command, the command shall terminate with CHECK CONDITION status with the appropriate sense information. The LBA of the first address descriptor not reassigned shall be returned in the COMMAND-SPECIFIC INFORMATION field of the sense data. If information about the first address descriptor not reassigned is not available, or if all the defects have been reassigned, the COMMAND-SPECIFIC INFORMATION field shall be set to FFFFFFFFh if the short sense data format is being used or FFFFFFFF FFFFFFFFh if the long sense data format is being used.

If the REASSIGN BLOCKS command failed due to an unexpected unrecoverable read error that would cause the loss of data in a logical block not specified in the defect list, the LBA of the unrecoverable block shall be returned in the INFORMATION field of the sense data and the VALID bit shall be set to one.

NOTE 18 - If the REASSIGN BLOCKS command returns CHECK CONDITION status and the sense data COMMAND-SPECIFIC INFORMATION field contains a valid LBA, the application client should remove all address descriptors from the defect list prior to the one returned in the COMMAND-SPECIFIC INFORMATION field. If the sense key is MEDIUM ERROR and the INFORMATION field contains the valid LBA, the application client should insert that new defective LBA into the defect list and reissue the REASSIGN BLOCKS command with the new defect list. Otherwise, the application client should perform any corrective action indicated by the sense data and then reissue the REASSIGN BLOCKS command with the new defect list.

5.21 START STOP UNIT command

The START STOP UNIT command provides an application client a method to control the power condition of a logical unit (see 4.14). This includes specifying that the device server enable or disable the block device for medium access operations by controlling certain power conditions and timers.

Logical units that contain cache memory shall write all cached data to the medium for the logical unit, the same as they would do in response to a SYNCHRONIZE CACHE command with the SYNC_NV bit set to zero (see 5.22), prior to entering into any power condition that prevents accessing the medium (e.g., before a hard drive stops its spindle motor during transition to the stopped power condition).

Table 52 defines the START STOP UNIT command CDB.

Table 52 — START STOP UNIT command

Byte\Bit	7	6	5	4	3	2	1	0
0	OPERATION CODE (1Bh)							
1	Reserved							IMMED
2	Reserved							
3								
4	POWER CONDITIONS				Reserved		LOEJ	START
5	CONTROL							

If the immediate (IMMED) bit is set to zero, then status shall be returned after the operation is completed. If the IMMED bit set to one, then status shall be returned as soon as the CDB has been validated.

The optional POWER CONDITION field is used to specify that the logical unit be placed into a power condition or to adjust a timer as defined in table 53. If this field is supported and has a value other than 0h then the START and LOEJ bits shall be ignored.

Table 53 — POWER CONDITIONS **field**

Code	Name	Description
0h	START_VALID	The START and LOEJ bits are valid.
1h	ACTIVE	Place the device into the active power condition.
2h	IDLE	Place the device into the idle power condition.
3h	STANDBY	Place the device into the standby power condition.
4h	Reserved	Reserved
5h	Obsolete	Obsolete
6h	Reserved	Reserved
7h	LU_CONTROL	Transfer control of power conditions to the logical unit.
8h - 9h	Reserved	Reserved
Ah	FORCE_IDLE_0	Force the idle condition timer to zero.
Bh	FORCE_STANDBY_0	Force the standby condition timer to zero.
Ch - Fh	Reserved	Reserved

If the START STOP UNIT command is issued with the POWER CONDITIONS field set to ACTIVE, IDLE, or STANDBY, then:

a) the logical unit shall transition to the specified power condition;
b) the logical unit shall change power conditions only after receipt of another START STOP UNIT command or a logical unit reset;
c) the device server shall disable the idle condition and standby condition timers if they are active (see SPC-3) until another START STOP UNIT command is received that returns control of the power condition to the logical unit, or a logical unit reset occurs; and
d) the device server shall terminate any command received that requires more power than allowed by the START STOP UNIT command's most recent power condition setting. The command shall be terminated with a CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to LOW POWER CONDITION ACTIVE.

If the START STOP UNIT command is issued with the POWER CONDITIONS field set to LU_CONTROL, then the device server shall enable the idle condition timer and the standby condition timer if they are active (see SPC-3).

If the START STOP UNIT command is issued with the POWER CONDITIONS field set to FORCE_IDLE_0 or FORCE_STANDBY_0, then the device server shall:

a) force the selected timer to zero, cause the logical unit to transition to the selected power condition, and return control of the power condition to the device server; or
b) terminate the START STOP UNIT command that selects a timer that is not supported by the device server or a timer that is not active. The command shall be terminated with a CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.

It is not an error to specify that the logical unit transition to a power condition in which it currently is.

If the load eject (LOEJ) bit is set to zero, then the logical unit shall take no action regarding loading or ejecting the medium. If the LOEJ bit is set to one, then the logical unit shall unload the medium if the START bit is set to zero. If the LOEJ bit is set to one, then the logical unit shall load the medium if the START bit is set to one.

If the START bit is set to zero, then the logical unit shall transition to the stopped power condition and, if activated, disable the idle condition and standby condition timers. If the START bit set to one, then the logical unit shall transition to the active power condition and, if activated, enable the idle condition and standby condition timers.

5.22 SYNCHRONIZE CACHE (10) command

The SYNCHRONIZE CACHE (10) command (see table 54) ensures that logical blocks within the specified range have their most recent data value recorded in non-volatile cache or on the medium, based on the SYNC_NV bit. Logical blocks include user data and protection information, if any. Logical blocks may or may not be removed from the volatile cache memory and non-volatile cache memory as a result of the synchronize cache operation. The synchronize cache function is also required implicitly by other SCSI functions as defined in other clauses of this standard.

Table 54 — SYNCHRONIZE CACHE (10) command

Byte\Bit	7	6	5	4	3	2	1	0
0	OPERATION CODE (35h)							
1	Reserved					SYNC_NV	IMMED	Obsolete
2	(MSB)							
5	LOGICAL BLOCK ADDRESS							(LSB)
6	Reserved			GROUP NUMBER				
7	(MSB)							
8	NUMBER OF BLOCKS							(LSB)
9	CONTROL							

See the LOCK UNLOCK CACHE (10) command (see 5.5) for a definition of the LOGICAL BLOCK ADDRESS field.

See the PRE-FETCH (10) command (see 5.7) and 4.16 for a description of the GROUP NUMBER field.

The SYNC_NV bit specifies whether the device server is required to synchronize volatile and non-volatile caches and is described in table 55.

Table 55 — SYNC_NV bit

SYNC_NV	Device server requirement to synchronize logical blocks currently in the	
	Volatile cache	**Non-volatile cache**
0	Device server shall synchronize to the medium.	Device server shall synchronize to the medium.
1	If a non-volatile cache is present, device server shall synchronize to non-volatile cache or the medium. If a non-volatile cache is not present, device server shall synchronize to the medium.	No requirement.

An immediate (IMMED) bit set to zero specifies that the status shall not be returned until the operation has been completed. An IMMED bit set to one specifies that the device server shall return status as soon as the CDB has been validated. If the IMMED bit is set to one and the device server does not support the IMMED bit, the command shall terminate with CHECK CONDITION status and the sense key shall be set to ILLEGAL REQUEST with the additional sense code set to INVALID FIELD IN CDB.

The NUMBER OF BLOCKS field specifies the number of contiguous logical blocks within the range. A NUMBER OF BLOCKS field set to zero specifies that the range contains all remaining logical blocks on the medium. If the logical block address plus the number of blocks exceeds the capacity of the medium, the device server shall return CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to LOGICAL BLOCK ADDRESS OUT OF RANGE.

A logical block within the specified range that is not in cache memory is not considered an error.

5.23 SYNCHRONIZE CACHE (16) command

The SYNCHRONIZE CACHE (16) command (see table 56) ensures that logical blocks within the specified range have their most recent data value recorded in non-volatile cache or on the medium, based on the SYNC_NV bit. Logical blocks include user data and protection information, if any. Logical blocks may or may not be removed from the volatile cache memory and non-volatile cache meory as a result of the synchronize cache operation. The synchronize cache function is also required implicitly by other SCSI functions as defined in other clauses of this standard.

Table 56 — SYNCHRONIZE CACHE (16) command

Byte\Bit	7	6	5	4	3	2	1	0
0	OPERATION CODE (91h)							
1	Reserved					SYNC_NV	IMMED	Reserved
2	(MSB)							
9	LOGICAL BLOCK ADDRESS							(LSB)
10	(MSB)							
13	NUMBER OF BLOCKS							(LSB)
14	Reserved			GROUP NUMBER				
15	CONTROL							

See the SYNCHRONIZE CACHE (10) command (see 5.22) for a description of the fields in this command.

5.24 VERIFY (10) command

The VERIFY (10) command (see table 57) requests that the device server verify the data on the medium. Data includes user data and protection information, if any. If the RTO_EN bit is set to one in the log read capacity data (see 5.15), the device server shall terminate this command with CHECK CONDITION status

with a sense key of ILLEGAL REQUEST and an additional sense code of INVALID COMMAND OPERATION CODE.

Table 57 — VERIFY (10) command

Byte\Bit	7	6	5	4	3	2	1	0
0	OPERATION CODE (2Fh)							
1	VRPROTECT			DPO	Reserved		BYTCHK	Obsolete
2	(MSB)							
5				LOGICAL BLOCK ADDRESS				(LSB)
6	Restricted for MMC-4	Reserved		GROUP NUMBER				
7	(MSB)							
8				VERIFICATION LENGTH				(LSB)
9	CONTROL							

See the READ (10) command (see 5.10) for a description of the DPO bit. See the LOCK UNLOCK CACHE (10) command (see 5.5) for a definition of the LOGICAL BLOCK ADDRESS field. See the PRE-FETCH (10) command (see 5.7) and 4.16 for a description of the GROUP NUMBER field.

If the MODE SELECT command is implemented, and the Verify Error Recovery mode page (see 6.3.5) is also implemented, then the current settings in that page specifies the verification criteria. If the Verify Error Recovery mode page is not implemented, then the verification criteria is vendor-specific.

If the byte check (BYTCHK) bit is set to zero, the device server shall:

a) perform a medium verification with no data comparison and not transfer any data from the application client data-out buffer; and
b) check protection information read from the medium based on the VRPROTECT field as described in table 58.

If the BYTCHK bit is set to one, the device server shall:

a) perform a byte-by-byte comparison of user data read from the medium and user data transferred from the application client data-out buffer;
b) check protection information read from the medium based on the VRPROTECT field as described in table 59;
c) check protection transferred from the application client data-out buffer based on the VRPROTECT field as described in table 60; and
d) perform a byte-by-byte comparison of protection information read from the medium and transferred from the application client data-out buffer based on the VRPROTECT field as described in table 61.

The order of the user data and protection information checks and comparisons is vendor-specific.

If a byte-by-byte comparison is unsuccessful for any reason, the device server shall return CHECK CONDITION status and the sense key shall be set to MISCOMPARE with the appropriate additional sense code for the condition.

The VERIFICATION LENGTH field specifies the number of contiguous logical blocks that shall be verified. A VERIFICATION LENGTH field set to zero specifies that no logical blocks shall be verified. This condition shall not be considered as an error. Any other value specifies the number of logical blocks that shall be verified. If the logical block address plus the verification length exceeds the capacity of the medium, the device server shall return CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to LOGICAL BLOCK ADDRESS OUT OF RANGE. The VERIFICATION LENGTH field is constrained by the MAXIMUM TRANSFER LENGTH field in the Block Limits VPD page (see 6.4.2).

If the BYTCHK bit is set to zero, the device server shall check the protection information read from the medium based on the VRPROTECT field as described in table 58.

Table 58 — VRPROTECT **field with** BYTCHK **set to zero - checking protection information read from the medium** (part 1 of 3)

Value	Logical unit formatted with protection information	Field in protection information [g]	Extended INQUIRY Data VPD page bit value [f]	If check fails [d] [e], additional sense code
000b	Yes	LOGICAL BLOCK GUARD	GRD_CHK = 1	LOGICAL BLOCK GUARD CHECK FAILED
			GRD_CHK = 0	No check performed
		LOGICAL BLOCK APPLICATION TAG	APP_CHK = 1 [c]	LOGICAL BLOCK APPLICATION TAG CHECK FAILED
			APP_CHK = 0	No check performed
		LOGICAL BLOCK REFERENCE TAG	REF_CHK = 1 [h]	LOGICAL BLOCK REFERENCE TAG CHECK FAILED
			REF_CHK = 0	No check performed
	No	No protection information on the medium to check. Only user data is checked.		
001b [b]	Yes	LOGICAL BLOCK GUARD	GRD_CHK = 1	LOGICAL BLOCK GUARD CHECK FAILED
			GRD_CHK = 0	No check performed
		LOGICAL BLOCK APPLICATION TAG	APP_CHK = 1 [c]	LOGICAL BLOCK APPLICATION TAG CHECK FAILED
			APP_CHK = 0	No check performed
		LOGICAL BLOCK REFERENCE TAG	REF_CHK = 1	LOGICAL BLOCK REFERENCE TAG CHECK FAILED
			REF_CHK = 0	No check performed
	No	Error condition [a]		
010b [b]	Yes	LOGICAL BLOCK GUARD	Shall not	No check performed
		LOGICAL BLOCK APPLICATION TAG	APP_CHK = 1 [c]	LOGICAL BLOCK APPLICATION TAG CHECK FAILED
			APP_CHK = 0	No check performed
		LOGICAL BLOCK REFERENCE TAG	REF_CHK = 1	LOGICAL BLOCK REFERENCE TAG CHECK FAILED
			REF_CHK = 0	No check performed
	No	Error condition [a]		

Table 58 — VRPROTECT **field with** BYTCHK **set to zero - checking protection information read from the medium** (part 2 of 3)

Value	Logical unit formatted with protection information	Field in protection information [g]	Extended INQUIRY Data VPD page bit value [f]	If check fails [d] [e], additional sense code
011b [b]	Yes	LOGICAL BLOCK GUARD	Shall not	No check performed
		LOGICAL BLOCK APPLICATION TAG	Shall not	No check performed
		LOGICAL BLOCK REFERENCE TAG	Shall not	No check performed
	No	Error condition [a]		

Table 58 — VRPROTECT **field with** BYTCHK **set to zero - checking protection information read from the medium** (part 3 of 3)

Value	Logical unit formatted with protection information	Field in protection information [g]	Extended INQUIRY Data VPD page bit value [f]	If check fails [d] [e], additional sense code
100b [b]	Yes	LOGICAL BLOCK GUARD	GRD_CHK = 1	LOGICAL BLOCK GUARD CHECK FAILED
			GRD_CHK = 0	No check performed
		LOGICAL BLOCK APPLICATION TAG	Shall not	No check performed
		LOGICAL BLOCK REFERENCE TAG	Shall not	No check performed
	No	Error condition [a]		
101b-111b	Reserved			

[a] A verify operation to a logical unit that supports protection information (see 4.15) and has not been formatted with protection information shall fail with a CHECK CONDITION status. The sense key shall be set to ILLEGAL REQUEST with the additional sense code set to INVALID FIELD IN CDB.

[b] If the logical unit does not support protection information the requested command should fail with CHECK CONDITION status with a sense key of ILLEGAL REQUEST and an additional sense code of INVALID FIELD IN CDB.

[c] The device server checks the logical block application tag only if it has knowledge of the contents of the LOGICAL BLOCK APPLICATION TAG field. This knowledge may be obtained by use of the VERIFY (32) command (see 5.27) or by a method not defined by this standard.

[d] If an error is reported, the sense key shall be set to ABORTED COMMAND.

[e] If multiple errors occur, the selection of which error to report is not defined by this standard.

[f] See the Extended INQUIRY Data VPD page (see SPC-3) for a description of the GRD_CHK, APP_CHK, and REF_CHK bits.

[g] If the application client or device server detects a LOGICAL BLOCK APPLICATION TAG field set to FFFFh, the checking of all protection information shall be disabled for the associated logical block.

[h] If the RTO_EN bit is set to zero in the long read capacity data (see 5.15), the device server checks the logical block reference tag by comparing it to the lower 4 bytes of the LBA associated with the logical block. If the RTO_EN bit is set to one, the device server checks the logical block reference tag only if it has knowledge of the contents of the LOGICAL BLOCK REFERENCE TAG field. The method for acquiring this knowledge is not defined by this standard.

If the BYTCHK bit is set to one, the device server shall check the protection information read from the medium based on the VRPROTECT field as described in table 59.

Table 59 — VRPROTECT field with BYTCHK set to one - checking protection information read from the medium

Value	Logical unit formatted with protection information	Field in protection information [g]	Extended INQUIRY Data VPD page bit value [f]	If check fails [d] [e], additional sense code
000b	Yes	LOGICAL BLOCK GUARD	GRD_CHK = 1	LOGICAL BLOCK GUARD CHECK FAILED
			GRD_CHK = 0	No check performed
		LOGICAL BLOCK APPLICATION TAG	APP_CHK = 1 [c] [g]	LOGICAL BLOCK APPLICATION TAG CHECK FAILED
			APP_CHK = 0	No check performed
		LOGICAL BLOCK REFERENCE TAG	REF_CHK = 1	LOGICAL BLOCK REFERENCE TAG CHECK FAILED
			REF_CHK = 0	No check performed
	No	No protection information on the medium available to check		
001b 010b 011b 100b [b]	Yes	LOGICAL BLOCK GUARD	Shall not	No check performed
		LOGICAL BLOCK APPLICATION TAG	Shall not	No check performed
		LOGICAL BLOCK REFERENCE TAG	Shall not	No check performed
	No	Error condition [a]		
101b - 111b	Reserved			

[a] A verify operation to a logical unit that supports protection information (see 4.15) and has not been formatted with protection information shall fail with a CHECK CONDITION status. The sense key shall be set to ILLEGAL REQUEST with the additional sense code set to INVALID FIELD IN CDB.

[b] If the logical unit does not support protection information the requested command should fail with CHECK CONDITION status with a sense key of ILLEGAL REQUEST and an additional sense code of INVALID FIELD IN CDB.

[c] The device server checks the logical block application tag only if it has knowledge of the contents of the LOGICAL BLOCK APPLICATION TAG field. This knowledge may be obtained by use of the VERIFY (32) command (see 5.27) or by a method not defined by this standard.

[d] If an error is reported, the sense key shall be set to ABORTED COMMAND.

[e] If multiple errors occur, the selection of which error to report is not defined by this standard.

[f] See the Extended INQUIRY Data VPD page (see SPC-3) for a description of the GRD_CHK, APP_CHK, and REF_CHK bits.

[g] If the application client or device server detects a LOGICAL BLOCK APPLICATION TAG field set to FFFFh, the checking of all protection information shall be disabled for the associated logical block.

If the BYTCHK bit is set to one, the device server shall check the protection information transferred from the application client data-out buffer based on the VRPROTECT field as described in table 60.

Table 60 — VRPROTECT **field with** BYTCHK **set to one - checking protection information from the application client** (part 1 of 2)

Value	Logical unit formatted with protection information	Field in protection information	Device server check	If check fails [d] [e], additional sense code
000b	Yes	No protection information received from application client to check		
	No	No protection information received from application client to check		
001b [b]	Yes	LOGICAL BLOCK GUARD	Shall	LOGICAL BLOCK GUARD CHECK FAILED
		LOGICAL BLOCK APPLICATION TAG	Shall not	No check performed
		LOGICAL BLOCK REFERENCE TAG	Shall	LOGICAL BLOCK REFERENCE TAG CHECK FAILED
	No	Error condition [a]		
010b [b]	Yes	LOGICAL BLOCK GUARD	Shall not	No check performed
		LOGICAL BLOCK APPLICATION TAG	May [c]	LOGICAL BLOCK APPLICATION TAG CHECK FAILED
		LOGICAL BLOCK REFERENCE TAG	May	LOGICAL BLOCK REFERENCE TAG CHECK FAILED
	No	Error condition [a]		
011b [b]	Yes	LOGICAL BLOCK GUARD	Shall not	No check performed
		LOGICAL BLOCK APPLICATION TAG	Shall not	No check performed
		LOGICAL BLOCK REFERENCE TAG	Shall not	No check performed
	No	Error condition [a]		

Table 60 — VRPROTECT **field with** BYTCHK **set to one - checking protection information from the application client** (part 2 of 2)

Value	Logical unit formatted with protection information	Field in protection information	Device server check	If check fails [d] [e], additional sense code
100b [b]	Yes	LOGICAL BLOCK GUARD	Shall	LOGICAL BLOCK GUARD CHECK FAILED
		LOGICAL BLOCK APPLICATION TAG	Shall not	No check performed
		LOGICAL BLOCK REFERENCE TAG	Shall not	No check performed
	No	Error condition [a]		
101b-111b	Reserved			

[a] A verify operation to a logical unit that supports protection information (see 4.15) and has not been formatted with protection information shall fail with a CHECK CONDITION status. The sense key shall be set to ILLEGAL REQUEST with the additional sense code set to INVALID FIELD IN CDB.

[b] If the logical unit does not support protection information the requested command should fail with CHECK CONDITION status with a sense key of ILLEGAL REQUEST and an additional sense code of INVALID FIELD IN CDB.

[c] The device server may check the logical block application tag if it has knowledge of the contents of the LOGICAL BLOCK APPLICATION TAG field. This knowledge may be obtained by use of the VERIFY (32) command (see 5.27) or by a method not defined by this standard.

[d] If an error is reported, the sense key shall be set to ABORTED COMMAND.

[e] If multiple errors occur, the selection of which error to report is not defined by this standard.

If the BYTCHK bit is set to one, the device server shall perform a byte-by-byte comparison of protection information transferred from the application client data-out buffer with protection information read from the medium based on the VRPROTECT field as described in table 61.

Table 61 — VRPROTECT **field with** BYTCHK **set to one - byte-by-byte comparison requirements** (part 1 of 2)

Value	Logical unit formatted with protection information	Field	Byte-by-byte Comparison	If compare fails [c] [d], additional sense code
000b	Yes	No protection information received from application client to compare. Only user data is compared within each logical block.		
	No	No protection information or the medium or received from application client to compare. Only user data is compared within each logical block.		
001b [b]	Yes	LOGICAL BLOCK GUARD	Shall	LOGICAL BLOCK GUARD CHECK FAILED
		LOGICAL BLOCK APPLICATION TAG (ATO = 1) [e]	Shall	LOGICAL BLOCK APPLICATION TAG CHECK FAILED
		LOGICAL BLOCK APPLICATION TAG (ATO = 0) [f]	Shall not	No compare performed
		LOGICAL BLOCK REFERENCE TAG	Shall	LOGICAL BLOCK REFERENCE TAG CHECK FAILED
	No	Error condition [a]		

Table 61 — VRPROTECT **field with** BYTCHK **set to one - byte-by-byte comparison requirements** (part 2 of 2)

Value	Logical unit formatted with protection information	Field	Byte-by-byte Comparison	If compare fails [c] [d], additional sense code
010b [b]	Yes	LOGICAL BLOCK GUARD	Shall not	No compare performed
		LOGICAL BLOCK APPLICATION TAG (ATO = 1) [e]	Shall	LOGICAL BLOCK APPLICATION TAG CHECK FAILED
		LOGICAL BLOCK APPLICATION TAG (ATO = 0) [f]	Shall not	No compare performed
		LOGICAL BLOCK REFERENCE TAG	Shall	LOGICAL BLOCK REFERENCE TAG CHECK FAILED
	No	Error condition [a]		
011b 100b [b]	Yes	LOGICAL BLOCK GUARD	Shall	LOGICAL BLOCK GUARD CHECK FAILED
		LOGICAL BLOCK APPLICATION TAG (ATO = 1) [e]	Shall	LOGICAL BLOCK APPLICATION TAG CHECK FAILED
		LOGICAL BLOCK APPLICATION TAG (ATO = 0) [f]	Shall not	No compare performed
		LOGICAL BLOCK REFERENCE TAG	Shall	LOGICAL BLOCK REFERENCE TAG CHECK FAILED
	No	Error condition [a]		
101b - 111b	Reserved			

[a] A verify operation to a logical unit that supports protection information (see 4.15) and has not been formatted with protection information shall fail with a CHECK CONDITION status. The sense key shall be set to ILLEGAL REQUEST with the additional sense code set to INVALID FIELD IN CDB.

[b] If the logical unit does not support protection information the requested command should fail with CHECK CONDITION status with a sense key of ILLEGAL REQUEST and an additional sense code of INVALID FIELD IN CDB.

[c] If an error is reported, the sense key shall be set to MISCOMPARE.

[d] If multiple errors occur, the selection of which error to report is not defined by this standard.

[e] If the ATO bit is set to one in the Control mode page (see SPC-3), the logical block application tag shall not be modified by a device server.

[f] If the ATO bit is set to zero in the Control mode page (see SPC-3), the logical block application tag may be modified by a device server.

5.25 VERIFY (12) command

The VERIFY (12) command (see table 62) requests that the device server verify the data on the medium. Data includes user data and protection information, if any. If the RTO_EN bit is set to one in the log read capacity data (see 5.15), the device server shall terminate this command with CHECK CONDITION status

with a sense key of ILLEGAL REQUEST and an additional sense code of INVALID COMMAND OPERATION CODE.

Table 62 — VERIFY (12) command

Byte\Bit	7	6	5	4	3	2	1	0
0	OPERATION CODE (AFh)							
1	VRPROTECT			DPO	Reserved		BYTCHK	Obsolete
2	(MSB)	LOGICAL BLOCK ADDRESS						
5								(LSB)
6	(MSB)	VERIFICATION LENGTH						
9								(LSB)
10	Restricted for MMC-4	Reserved		GROUP NUMBER				
11	CONTROL							

See the VERIFY (10) command (see 5.24) for a description of the fields in this command.

5.26 VERIFY (16) command

The VERIFY (16) command (see table 63) requests that the device server verify the data written on the medium. Data includes user data and protection information, if any. If the RTO_EN bit is set to one in the log read capacity data (see 5.15), the device server shall terminate this command with CHECK CONDITION status with a sense key of ILLEGAL REQUEST and an additional sense code of INVALID COMMAND OPERATION CODE.

Table 63 — VERIFY (16) command

Byte\Bit	7	6	5	4	3	2	1	0
0	OPERATION CODE (8Fh)							
1	VRPROTECT			DPO	Reserved		BYTCHK	Reserved
2	(MSB)	LOGICAL BLOCK ADDRESS						
9								(LSB)
10	(MSB)	VERIFICATION LENGTH						
13								(LSB)
14	Restricted for MMC-4	Reserved		GROUP NUMBER				
15	CONTROL							

See the VERIFY (10) command (see 5.24) for a description of the fields in this command.

5.27 VERIFY (32) command

The VERIFY (32) command (see table 64) requests that the device server transfer data to the application client from the medium. Data includes user data and protection information, if any.

Table 64 — VERIFY (32) command

Byte\Bit	7	6	5	4	3	2	1	0
0	OPERATION CODE (7Fh)							
1	CONTROL							
2	Reserved							
5								
6	Reserved			GROUP NUMBER				
7	ADDITIONAL CDB LENGTH (18h)							
8	(MSB)							
9	SERVICE ACTION (000Ah)							(LSB)
10	VRPROTECT			DPO	Reserved		BYTCHK	Reserved
12	(MSB)							
19	LOGICAL BLOCK ADDRESS							(LSB)
20	(MSB)							
23	INITIAL LOGICAL BLOCK REFERENCE TAG							(LSB)
24	(MSB)							
25	EXPECTED LOGICAL BLOCK APPLICATION TAG							(LSB)
26	(MSB)							
27	LOGICAL BLOCK APPLICATION TAG MASK							(LSB)
28	(MSB)							
31	VERIFICATION LENGTH							(LSB)

See the VERIFY (10) command (see 5.24) for a description of the GROUP NUMBER field, VRPROTECT field, DPO bit, BYTCHK bit, LOGICAL BLOCK ADDRESS field, and VERIFICATION LENGTH field.

If the RTO_EN bit is set to zero in the long read capacity data (see 5.15), the device server shall terminate the command with CHECK CONDITION status with a sense key of ILLEGAL REQUEST and an additional sense code set to INVALID COMMAND OPERATION CODE. If the RTO_EN bit is set to one, the device server may process the command.

The INITIAL LOGICAL BLOCK REFERENCE TAG field contains the value of the LOGICAL BLOCK REFERENCE TAG expected on the first logical block of the range of logical blocks for this command. The checking enables and requirements are controlled by the VRPROTECT field. See the VERIFY (10) command (see 5.24) for a definition description of the VRPROTECT field checking enables and requirements.

When checking of the LOGICAL BLOCK APPLICATION TAG is enabled (see table 58, table 59, table 60, and table 61 in 5.24), the EXPECTED LOGICAL BLOCK APPLICATION TAG field contains a value that is expected in the LOGICAL BLOCK APPLICATION TAG with the LOGICAL BLOCK APPLICATION TAG MASK applied in the protection information of logical blocks for this command.

When checking of the LOGICAL BLOCK APPLICATION TAG is enabled (see table 58, table 59, table 60, and table 61 in 5.24), the LOGICAL BLOCK APPLICATION MASK field contains a value that is a bit mask for enabling the checking of the LOGICAL BLOCK APPLICATION TAG in the protection information for each logical block of the

range of logical blocks for this command. A LOGICAL BLOCK APPLICATION TAG MASK bit set to one enables the checking of the corresponding bit in the EXPECTED LOGICAL BLOCK APPLICATION TAG field with the LOGICAL BLOCK APPLICATION TAG.

5.28 WRITE (6) command

The WRITE (6) command (see table 65) requests that the device server write the data transferred from the application client to the medium. Data transferred from the application client includes user data but does not include protection information.

Table 65 — WRITE (6) command

Byte\Bit	7	6	5	4	3	2	1	0
0	OPERATION CODE (0Ah)							
1	Reserved			(MSB)				
2	LOGICAL BLOCK ADDRESS							
3								(LSB)
4	TRANSFER LENGTH							
5	CONTROL							

The cache control bits are not provided for this command. Block devices with cache memory may have values for the cache control bits that may affect the WRITE (6) command, however no default value is defined by this standard. If explicit control is required, the WRITE (10) command should be used.

The LOGICAL BLOCK ADDRESS field specifies the logical block where the write operation shall begin.

The TRANSFER LENGTH field specifies the number of contiguous logical blocks of data that shall be transferred. A TRANSFER LENGTH field set to zero specifies that 256 logical blocks shall be transferred. Any other value specifies the number of logical blocks that shall be transferred. If the logical block address plus the transfer length exceeds the capacity of the medium, the device server shall return CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to LOGICAL BLOCK ADDRESS OUT OF RANGE. The TRANSFER LENGTH field is constrained by the MAXIMUM TRANSFER LENGTH field in the Block Limits VPD page (see 6.4.2).

NOTE 19 - For the WRITE (10) command, WRITE (12) command, WRITE (16) command, and WRITE (32) command, a TRANSFER LENGTH field set to zero specifies that no logical blocks are transferred.

If a WRITE (6) command is received after protection information is enabled the device server shall set the protection information (see 4.15) as follows as it writes the logical block to the medium:

a) the LOGICAL BLOCK GUARD field set to a properly generated CRC (see 4.15.3);
b) the LOGICAL BLOCK REFERENCE TAG field set to:
   A) the least significant four bytes of the LBA if the RTO_EN bit is set to zero in the long read capacity data (see 5.15); or
   B) FFFFFFFFh if the RTO_EN bit is set to one;

   and

c) the LOGICAL BLOCK APPLICATION TAG field set to:
   A) FFFFh if the ATO bit is set to one in the Control mode page (see SPC-3); or
   B) any value if the ATO bit is set to zero in the Control mode page (see SPC-3).

5.29 WRITE (10) command

The WRITE (10) command (see table 66) requests that the device server write the data transferred from the application client to the medium. Data transferred from the application client includes user data and includes protection information as required by the WRPROTECT field and the medium format.

Table 66 — WRITE (10) command

Byte\Bit	7	6	5	4	3	2	1	0
0	OPERATION CODE (2Ah)							
1	WRPROTECT			DPO	FUA	Reserved	FUA_NV	Obsolete
2	(MSB)							
5	LOGICAL BLOCK ADDRESS							(LSB)
6	Reserved			GROUP NUMBER				
7	(MSB)							
8	TRANSFER LENGTH							(LSB)
9	CONTROL							

See the READ (10) command (see 5.10) for a definition of the DPO bit. See the LOCK UNLOCK CACHE (10) command (see 5.5) for a definition of the LOGICAL BLOCK ADDRESS field. See the PRE-FETCH (10) command (see 5.7) and 4.16 for a description of the GROUP NUMBER field.

The force unit access (FUA and FUA_NV) bits are defined in table 67.

Table 67 — Force unit access for writes

FUA	FUA_NV	**Description**
0	0	The device server shall write the logical blocks to volatile cache, non-volatile cache, and/or the medium.
0	1	If the NV_SUP bit is set to one in the Extended INQUIRY Data VPD page (see SPC-3), the device server shall write the logical blocks to non-volatile cache and/or the medium. If the NV_SUP bit is set to zero in the Extended INQUIRY Data VPD page (see SPC-3), the device server shall write the logical blocks to volatile cache, non-volatile cache, and/or the medium.
1	0 or 1	The device server shall write the logical blocks to the medium, and shall not return GOOD status until the logical blocks have actually been written on the medium.

If logical blocks are transferred directly to a cache memory, GOOD status may be returned to the application client prior to writing the logical blocks to the medium. Any error that occurs after the GOOD status is returned is a deferred error, and information regarding the error is not reported until a subsequent command.

The TRANSFER LENGTH field specifies the number of contiguous logical blocks of data that shall be transferred. A TRANSFER LENGTH field set to zero specifies that no logical blocks shall be transferred. This condition shall not be considered an error and no data shall be written. Any other value specifies the number of logical blocks that shall be transferred. If the logical block address plus the transfer length exceeds the capacity of the medium, the device server shall return CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to LOGICAL BLOCK ADDRESS OUT OF RANGE. The TRANSFER LENGTH field is constrained by the MAXIMUM TRANSFER LENGTH field in the Block Limits VPD page (see 6.4.2).

NOTE 20 - For the WRITE (6) command, a TRANSFER LENGTH field set to zero specifies that 256 logical blocks are transferred.

The device server shall check the protection information transferred from the application client data-out buffer based on the WRPROTECT field as described in table 68.

Table 68 — WRPROTECT **field** (part 1 of 2)

Value	Logical unit formatted with protection information	Field in protection information	Device server check	If check fails [d] [i], additional sense code
000b	Yes [f] [g] [h]	No protection information received from application client to check		
	No	No protection information received from application client to check		
001b [b] [j]	Yes [e]	LOGICAL BLOCK GUARD	Shall	LOGICAL BLOCK GUARD CHECK FAILED
		LOGICAL BLOCK APPLICATION TAG	Shall not	No check performed
		LOGICAL BLOCK REFERENCE TAG	Shall	LOGICAL BLOCK REFERENCE TAG CHECK FAILED
	No [a]	No protection information available to check		
010b [b] [j]	Yes [e]	LOGICAL BLOCK GUARD	Shall not	No check performed
		LOGICAL BLOCK APPLICATION TAG	May [c]	LOGICAL BLOCK APPLICATION TAG CHECK FAILED
		LOGICAL BLOCK REFERENCE TAG	May	LOGICAL BLOCK REFERENCE TAG CHECK FAILED
	No [a]	No protection information available to check		

Table 68 — WRPROTECT **field** (part 2 of 2)

Value	Logical unit formatted with protection information	Field in protection information	Device server check	If check fails [d] [i], additional sense code
011b [b] [j]	Yes [e]	LOGICAL BLOCK GUARD	Shall not	No check performed
		LOGICAL BLOCK APPLICATION TAG	Shall not	No check performed
		LOGICAL BLOCK REFERENCE TAG	Shall not	No check performed
	No [a]	No protection information available to check		
100b [b] [j]	Yes [e]	LOGICAL BLOCK GUARD	Shall	LOGICAL BLOCK GUARD CHECK FAILED
		LOGICAL BLOCK APPLICATION TAG	Shall not	No check performed
		LOGICAL BLOCK REFERENCE TAG	Shall not	No check performed
	No [a]	No protection information available to check		
101b - 111b	Reserved			

[a] A write operation to a logical unit that supports protection information (see 4.15) and has not been formatted with protection information shall fail with a CHECK CONDITION status. The sense key shall be set to ILLEGAL REQUEST with the additional sense code set to INVALID FIELD IN CDB.

[b] If the logical unit does not support protection information the requested command should fail with CHECK CONDITION status with a sense key of ILLEGAL REQUEST and an additional sense code of INVALID FIELD IN CDB.

[c] The device server may check the logical block application tag if it has knowledge of the contents of the LOGICAL BLOCK APPLICATION TAG field. This knowledge may be obtained by use of the WRITE (32) command (see 5.32) or by a method not defined by this standard.

[d] If an error is reported the sense key shall be set to ABORTED COMMAND.

[e] Device server shall preserve the contents of protection information (e.g., write to medium, store in non-volatile memory).

[f] The device server shall write a properly generated CRC (see 4.15.3.2) into the LOGICAL BLOCK GUARD field.

[g] If the RTO_EN bit is set to zero in the long read capacity data (see 5.15), the device server shall write the least significant four bytes of each LBA into the LOGICAL BLOCK REFERENCE TAG field of each of the written logical blocks. If the RTO_EN bit is set to one, the device server shall write a value of FFFFFFFFh into the LOGICAL BLOCK REFERENCE TAG field of each of the written logical blocks.

[h] If the ATO bit is set to one in the Control mode page (see SPC-3), the device server shall write FFFFh into each LOGICAL BLOCK APPLICATION TAG field. If the ATO bit is set to zero, the device server may write any value into each LOGICAL BLOCK APPLICATION TAG field.

[i] If multiple errors occur, the selection of which error to report is not defined by this standard.

[j] If the RTO_EN bit is set to zero in the long read capacity data (see 5.15), the device server may process the command. If the RTO_EN bit is set to one, WRITE (10), WRITE (12),and WRITE (16) commands with the WRPROTECT field set to 000b may be processed by the device server. If the RTO_EN bit is set to one, the device server shall terminate WRITE (10), WRITE (12), and WRITE (16) commands with the WRPROTECT field not set to 000b with CHECK CONDITION status with a sense key of ILLEGAL REQUEST and an additional sense code of INVALID COMMAND OPERATION CODE.

5.30 WRITE (12) command

The WRITE (12) command (see table 69) requests that the device server write the data transferred from the application client to the medium. Data transferred from the application client includes user data and includes protection information as required by the WRPROTECT field and the medium format.

Table 69 — WRITE (12) command

Byte\Bit	7	6	5	4	3	2	1	0
0	OPERATION CODE (AAh)							
1	WRPROTECT			DPO	FUA	Reserved	FUA_NV	Obsolete
2	(MSB)							
5	LOGICAL BLOCK ADDRESS							(LSB)
6	(MSB)							
9	TRANSFER LENGTH							(LSB)
10	Restricted for MMC-4	Reserved		GROUP NUMBER				
11	CONTROL							

See the WRITE (10) command (see 5.29) for a description of the fields in this command.

5.31 WRITE (16) command

The WRITE (16) command (see table 70) requests that the device server write the data transferred from the application client to the medium. Data transferred from the application client includes user data and includes protection information as required by the WRPROTECT field and the medium format.

Table 70 — WRITE (16) command

Byte\Bit	7	6	5	4	3	2	1	0
0	OPERATION CODE (8Ah)							
1	WRPROTECT			DPO	FUA	Reserved	FUA_NV	Reserved
2	(MSB)							
9	LOGICAL BLOCK ADDRESS							(LSB)
10	(MSB)							
13	TRANSFER LENGTH							(LSB)
14	Restricted for MMC-4	Reserved		GROUP NUMBER				
15	CONTROL							

See the WRITE (10) command (see 5.29) for a description of the fields in this command.

5.32 WRITE (32) command

The WRITE (32) command (see table 71) requests that the device server write the data transferred from the application client to the medium. Data includes user data and protection information, if any.

Table 71 — WRITE (32) command

Byte\Bit	7	6	5	4	3	2	1	0
0	OPERATION CODE (7Fh)							
1	CONTROL							
2	Reserved							
5								
6	Reserved			GROUP NUMBER				
7	ADDITIONAL CDB LENGTH (18h)							
8	(MSB)							
9	SERVICE ACTION (000Bh)							(LSB)
10	WRPROTECT			DPO	FUA	Reserved	FUA_NV	Reserved
12	(MSB)							
19	LOGICAL BLOCK ADDRESS							(LSB)
20	(MSB)							
23	INITIAL LOGICAL BLOCK REFERENCE TAG							(LSB)
24	(MSB)							
25	EXPECTED LOGICAL BLOCK APPLICATION TAG							(LSB)
26	(MSB)							
27	LOGICAL BLOCK APPLICATION TAG MASK							(LSB)
28	(MSB)							
31	TRANSFER LENGTH							(LSB)

See the WRITE (10) command (see 5.29) for a description of the GROUP NUMBER field, WRPROTECT field, DPO bit, FUA bit, FUA_NV bit, LOGICAL BLOCK ADDRESS field, and TRANSFER LENGTH field.

If the RTO_EN bit is set to zero in the long read capacity data (see 5.15), the device server shall terminate the WRITE (32) command with CHECK CONDITION status with a sense key of ILLEGAL REQUEST and an additional sense code set to INVALID COMMAND OPERATION CODE. If the RTO_EN bit is set to one, the device server may process the command.

The INITIAL LOGICAL BLOCK REFERENCE TAG field contains the value of the LOGICAL BLOCK REFERENCE TAG expected on the first logical block of the range of logical blocks for this command. The checking enables and requirements are controlled by the WRPROTECT field. See the WRITE (10) command (see 5.29) for a definition description of the WRPROTECT field checking enables and requirements.

When checking of the LOGICAL BLOCK APPLICATION TAG is enabled (see table 68 in 5.29), the EXPECTED LOGICAL BLOCK APPLICATION TAG field contains a value that is expected in the LOGICAL BLOCK APPLICATION TAG with the LOGICAL BLOCK APPLICATION TAG MASK applied in the protection information of logical blocks for this command.

When checking of the LOGICAL BLOCK APPLICATION TAG is enabled (see table 68 in 5.29), the LOGICAL BLOCK APPLICATION MASK field contains a value that is a bit mask for enabling the checking of the LOGICAL BLOCK APPLICATION TAG in the protection information for each logical block of the range of logical blocks for this

command. A LOGICAL BLOCK APPLICATION TAG MASK bit set to one enables the checking of the corresponding bit in the EXPECTED LOGICAL BLOCK APPLICATION TAG field with the LOGICAL BLOCK APPLICATION TAG.

5.33 WRITE AND VERIFY (10) command

The WRITE AND VERIFY (10) command (see table 72) requests that the device server write the data transferred from the application client to the medium and then verify that the data is correctly written. Data transferred from the application client includes user data and includes protection information as required by the WRPROTECT field and the medium format. The data is only transferred once from the application client to the device server. If the RTO_EN bit is set to one in the long read capacity data (see 5.15), the device server shall terminate this command with CHECK CONDITION status with a sense key of ILLEGAL REQUEST and an additional sense code set to INVALID COMMAND OPERATION CODE.

Table 72 — WRITE AND VERIFY (10) command

Byte\Bit	7	6	5	4	3	2	1	0
0	OPERATION CODE (2Eh)							
1	WRPROTECT			DPO	Reserved		BYTCHK	Obsolete
2	(MSB)							
5	LOGICAL BLOCK ADDRESS							(LSB)
6	Reserved			GROUP NUMBER				
7	(MSB)							
8	TRANSFER LENGTH							(LSB)
9	CONTROL							

See the LOCK UNLOCK CACHE (10) command (see 5.5) for a definition of the LOGICAL BLOCK ADDRESS field. See the PRE-FETCH (10) command (see 5.7) and 4.16 for a description of the GROUP NUMBER field. See the WRITE (10) command (see 5.29) for a definition of the TRANSFER LENGTH field and the WRPROTECT field. See the READ (10) command (see 5.10) for a description of the DPO bit.

If the MODE SELECT command is implemented, and the Verify Error Recovery mode page (see 6.3.5) is also implemented, then the current settings in that mode page along with the AWRE bit in the Read-Write Error Recovery mode page (see 6.3.4) specify the verification error criteria. If these mode pages are not implemented, then the verification criteria is vendor-specific.

A byte check (BYTCHK) bit set to zero specifies that, after writing, the device server perform a medium verification with no data comparison. A BYTCHK bit set to one specifies that, after writing, the device server perform a byte-by-byte comparison of data written on the medium with the data just written. If the comparison is unsuccessful for any reason, the device server shall return CHECK CONDITION status with the sense key set to MISCOMPARE with the appropriate additional sense code for the condition.

5.34 WRITE AND VERIFY (12) command

The WRITE AND VERIFY (12) command (see table 73) requests that the device server write the data transferred from the application client to the medium and then verify that the data is correctly written. Data transferred from the application client includes user data and includes protection information as required by the WRPROTECT field and the medium format. The data is only transferred once from the application client to the device server. If the RTO_EN bit is set to one in the long read capacity data (see 5.15), the device server

shall terminate this command with CHECK CONDITION status with a sense key of ILLEGAL REQUEST and an additional sense code set to INVALID COMMAND OPERATION CODE.

Table 73 — WRITE AND VERIFY (12) command

Byte\Bit	7	6	5	4	3	2	1	0
0	OPERATION CODE (AEh)							
1	WRPROTECT			DPO	Reserved		BYTCHK	Obsolete
2	(MSB)							
5	LOGICAL BLOCK ADDRESS							(LSB)
6	(MSB)							
9	TRANSFER LENGTH							(LSB)
10	Restricted for MMC-4	Reserved		GROUP NUMBER				
11	CONTROL							

See the WRITE AND VERIFY (10) command (see 5.33) for a description of the bits in this command.

5.35 WRITE AND VERIFY (16) command

The WRITE AND VERIFY (16) command (see table 74) requests that the device server write the data transferred from the application client to the medium and then verify that the data is correctly written. Data transferred from the application client includes user data and includes protection information as required by the WRPROTECT field and the medium format. The data is only transferred once from the application client to the device server. If the RTO_EN bit is set to one in the long read capacity data (see 5.15), the device server shall terminate this command with CHECK CONDITION status with a sense key of ILLEGAL REQUEST and an additional sense code set to INVALID COMMAND OPERATION CODE

Table 74 — WRITE AND VERIFY (16) command

Byte\Bit	7	6	5	4	3	2	1	0
0	OPERATION CODE (8Eh)							
1	WRPROTECT			DPO	Reserved		BYTCHK	Reserved
2	(MSB)							
9	LOGICAL BLOCK ADDRESS							(LSB)
10	(MSB)							
13	TRANSFER LENGTH							(LSB)
14	Restricted for MMC-4	Reserved		GROUP NUMBER				
15	CONTROL							

See the WRITE AND VERIFY (10) command (see 5.33) for a description of the fields in this command.

5.36 WRITE AND VERIFY (32) command

The WRITE AND VERIFY (32) command (see table 75) requests that the device server transfer data to the application client from the medium. Data includes user data and protection information, if any.

Table 75 — WRITE AND VERIFY (32) command

Byte\Bit	7	6	5	4	3	2	1	0
0	OPERATION CODE (7Fh)							
1	CONTROL							
2	Reserved							
5								
6	Reserved			GROUP NUMBER				
7	ADDITIONAL CDB LENGTH (18h)							
8	(MSB)							
9	SERVICE ACTION (000Ch)							(LSB)
10	WRPROTECT			DPO	Reserved		BYTCHK	Reserved
12	(MSB)							
19	LOGICAL BLOCK ADDRESS							(LSB)
20	(MSB)							
23	INITIAL LOGICAL BLOCK REFERENCE TAG							(LSB)
24	(MSB)							
25	EXPECTED LOGICAL BLOCK APPLICATION TAG							(LSB)
26	(MSB)							
27	LOGICAL BLOCK APPLICATION TAG MASK							(LSB)
28	(MSB)							
31	TRANSFER LENGTH							(LSB)

See the WRITE AND VERIFY (10) command (see 5.33) for a description of the GROUP NUMBER field, WRPROTECT field, DPO bit, BYTCHK bit, LOGICAL BLOCK ADDRESS field, and TRANSFER LENGTH field.

If the RTO_EN bit is set to zero in the long read capacity data (see 5.15), the device server shall terminate the command with CHECK CONDITION status with a sense key of ILLEGAL REQUEST and an additional sense code set to INVALID COMMAND OPERATION CODE. If the RTO_EN bit is set to one, the device server may process the command.

The INITIAL LOGICAL BLOCK REFERENCE TAG field contains the value of the LOGICAL BLOCK REFERENCE TAG expected on the first logical block of the range of logical blocks for this command. The checking enables and requirements are controlled by the WRPROTECT field. See the WRITE AND VERIFY (10) command (see 5.33) for a definition description of the WRPROTECT field checking enables and requirements.

When checking of the LOGICAL BLOCK APPLICATION TAG is enabled (see table 68 in 5.29), the EXPECTED LOGICAL BLOCK APPLICATION TAG field contains a value that is expected in the LOGICAL BLOCK APPLICATION TAG with the LOGICAL BLOCK APPLICATION TAG MASK applied in the protection information of logical blocks for this command.

When checking of the LOGICAL BLOCK APPLICATION TAG is enabled (see table 68 in 5.29), the LOGICAL BLOCK APPLICATION MASK field contains a value that is a bit mask for enabling the checking of the LOGICAL BLOCK APPLICATION TAG in the protection information for each logical block of the range of logical blocks for this

command. A LOGICAL BLOCK APPLICATION TAG MASK bit set to one enables the checking of the corresponding bit in the EXPECTED LOGICAL BLOCK APPLICATION TAG field with the LOGICAL BLOCK APPLICATION TAG.

5.37 WRITE LONG (10) command

The WRITE LONG (10) command (see table 76) requests that the device server write the data transferred from the application client to one logical block on the medium. The data written shall be the same length and shall be in the same order as the data returned by the READ LONG command (see 5.18).

Table 76 — WRITE LONG (10) command

Byte\Bit	7	6	5	4	3	2	1	0
0	OPERATION CODE (3Fh)							
1	Reserved							Obsolete
2	(MSB)	LOGICAL BLOCK ADDRESS						
5								(LSB)
6	Reserved							
7	(MSB)	BYTE TRANSFER LENGTH						
8								(LSB)
9	CONTROL							

See the LOCK UNLOCK CACHE (10) command (see 5.5) for a definition of the LOGICAL BLOCK ADDRESS field.

NOTE 21 - Any other bytes that can be corrected by ECC should be included (e.g., a data synchronization mark within the area covered by ECC). A READ LONG command may be issued before issuing a WRITE LONG command.

The BYTE TRANSFER LENGTH field specifies the number of bytes of data that the device server shall transfer to the specified logical block. If the BYTE TRANSFER LENGTH field is not set to zero and does not exactly match the data length the device server returns for a READ LONG command, 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. In the sense data (see SPC-3), the ILI and VALID bits shall be set to one and the INFORMATION field shall be set to the difference (i.e., residue) of the requested length minus the actual length in bytes. Negative values shall be indicated by two's complement notation. A BYTE TRANSFER LENGTH field set to zero specifies that no bytes shall be transferred. This condition shall not be considered an error.

5.38 WRITE LONG (16) command

The WRITE LONG (16) command (see table 77) requests that the device server write the data transferred from the application client to one logical block on the medium. This command is implemented as a service action of the SERVICE ACTION OUT operation code.

Table 77 — WRITE LONG (16) command

Byte\Bit	7	6	5	4	3	2	1	0
0	OPERATION CODE (9Fh)							
1	Reserved			SERVICE ACTION (11h)				
2	(MSB)							
9				LOGICAL BLOCK ADDRESS				(LSB)
10								
11				Reserved				
12	(MSB)							
13				BYTE TRANSFER LENGTH				(LSB)
14	Reserved							CORRCT
15	CONTROL							

See the WRITE LONG (10) command (see 5.37) for a description of the fields in this command.

5.39 WRITE SAME (10) command

The WRITE SAME (10) command (see table 78) requests that the device server write the single block of data transferred from the application client to the medium multiple times to consecutive multiple logical blocks. Data transferred from the application client includes user data and includes protection information as required by the WRPROTECT field and the medium format. If the RTO_EN bit is set to one in the long read capacity data (see 5.15), the device server shall terminate this command with CHECK CONDITION status with a sense key of ILLEGAL REQUEST and an additional sense code set to INVALID COMMAND OPERATION CODE.

NOTE 22 - This command may be useful if large areas of the medium need to be written, prepared for certification, or otherwise initialized without the application client having to transfer all the data.

Table 78 — WRITE SAME (10) command

Byte\Bit	7	6	5	4	3	2	1	0
0	OPERATION CODE (41h)							
1	WRPROTECT			Reserved		PBDATA	LBDATA	Obsolete
2	(MSB)							
5				LOGICAL BLOCK ADDRESS				(LSB)
6	Reserved			GROUP NUMBER				
7	(MSB)							
8				NUMBER OF BLOCKS				(LSB)
9	CONTROL							

See the LOCK UNLOCK CACHE (10) command (see 5.5) for a definition of the LOGICAL BLOCK ADDRESS field. See the PRE-FETCH (10) command (see 5.7) and 4.16 for a description of the GROUP NUMBER field.

Table 79 describes the LBDATA bit and the PBDATA bit.

Table 79 — LBDATA **bit and** PBDATA **bit**

LBDATA	PBDATA	**Description**
0	0	The device server shall write the single block of user data received from the application client data-out buffer to each logical block without modification. If the medium is formatted with protection information, the value in the LOGICAL BLOCK REFERENCE TAG field received from the application client shall be placed into the LOGICAL BLOCK REFERENCE TAG field of the first logical block written to the medium. Into each of the following logical blocks the logical block reference tag received in the data transferred from the application client, incremented by one, shall be placed into the LOGICAL BLOCK REFERENCE TAG field of that logical block (i.e., each logical block written to the medium has a logical block reference tag value of one greater than the previous logical block). If the ATO bit is set to one in the Control mode page (see SPC-3), the logical block application tag received in the single block of data shall be placed in the LOGICAL BLOCK APPLICATION TAG field of each logical block. If the ATO bit is set to zero, the logical block application tag received in the single block of data may be placed in the LOGICAL BLOCK APPLICATION TAG field of each logical block.
0	1 [a]	The device server shall replace the first eight bytes of the block received from the application client data-out buffer to each physical sector with the physical address of the sector being written using the physical sector format (see 5.4.2.4.5).
1 [a]	0	The device server shall replace the first four bytes of the block received from the application client data-out buffer with the least significant four bytes of the LBA of the block being written. The most significant byte of the four bytes shall be written first.
1	1	The device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.

[a] If the medium is formatted with protection information then the protection information shall be written to a default value of FFFFFFFF FFFFFFFFh in each of the written logical blocks.

The NUMBER OF BLOCKS field specifies the number of contiguous logical blocks to be written. A NUMBER OF BLOCKS field set to zero specifies that the device server write all the remaining logical blocks on the medium. If the logical block address plus the number of blocks exceeds the capacity of the medium, the device server shall return CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to LOGICAL BLOCK ADDRESS OUT OF RANGE.

5.40 WRITE SAME (16) command

The WRITE SAME (16) command (see table 80) requests that the device server write the single block of data transferred from the application client to the medium multiple times to consecutive multiple logical blocks. Data transferred from the application client includes user data and includes protection information as required by the WRPROTECT field and the medium format. If the RTO_EN bit is set to one in the long read capacity data

(see 5.15), the device server shall terminate this command with CHECK CONDITION status with a sense key of ILLEGAL REQUEST and an additional sense code set to INVALID COMMAND OPERATION CODE

Table 80 — WRITE SAME (16) command

Byte\Bit	7	6	5	4	3	2	1	0
0	OPERATION CODE (93h)							
1	WRPROTECT			Reserved		PBDATA	LBDATA	Reserved
2	(MSB)							
9	LOGICAL BLOCK ADDRESS							(LSB)
10	(MSB)							
13	NUMBER OF BLOCKS							(LSB)
14	Reserved			GROUP NUMBER				
15	CONTROL							

See the WRITE SAME (10) command (see 5.39) for a description of the fields in this command.

5.41 WRITE SAME (32) command

The WRITE SAME (32) command (see table 81) requests that the device server transfer data to the application client from the medium. Data includes user data and protection information, if any.

Table 81 — WRITE SAME (32) command

Byte\Bit	7	6	5	4	3	2	1	0
0	OPERATION CODE (7Fh)							
1	CONTROL							
2	Reserved							
5								
6	Reserved			GROUP NUMBER				
7	ADDITIONAL CDB LENGTH (18h)							
8	(MSB)							
9	SERVICE ACTION (000Dh)							(LSB)
10	WRPROTECT			Reserved		PBDATA	LBDATA	Reserved
12	(MSB)							
19	LOGICAL BLOCK ADDRESS							(LSB)
20	(MSB)							
23	INITIAL LOGICAL BLOCK REFERENCE TAG							(LSB)
24	(MSB)							
25	EXPECTED LOGICAL BLOCK APPLICATION TAG							(LSB)
26	(MSB)							
27	LOGICAL BLOCK APPLICATION TAG MASK							(LSB)
28	(MSB)							
31	NUMBER OF BLOCKS							(LSB)

See the WRITE SAME (10) command (see 5.39) for a description of the GROUP NUMBER field, WRPROTECT field, PBDATA bit, LBDATA bit, LOGICAL BLOCK ADDRESS field, and NUMBER OF BLOCKS field.

If the RTO_EN bit is set to zero in the long read capacity data (see 5.15), the device server shall terminate the command with CHECK CONDITION status with a sense key of ILLEGAL REQUEST and an additional sense code set to INVALID COMMAND OPERATION CODE. If the RTO_EN bit is set to one, the device server may process the command.

The INITIAL LOGICAL BLOCK REFERENCE TAG field contains the value of the LOGICAL BLOCK REFERENCE TAG expected on the first logical block of the range of logical blocks for this command. The checking enables and requirements are controlled by the WRPROTECT field. See the WRITE SAME (10) command (see 5.39) for a definition description of the WRPROTECT field checking enables and requirements.

When checking of the LOGICAL BLOCK APPLICATION TAG is enabled (see table 68 in 5.29), the EXPECTED LOGICAL BLOCK APPLICATION TAG field contains a value that is expected in the LOGICAL BLOCK APPLICATION TAG with the LOGICAL BLOCK APPLICATION TAG MASK applied in the protection information of logical blocks for this command.

When checking of the LOGICAL BLOCK APPLICATION TAG is enabled (see table 68 in 5.29), the LOGICAL BLOCK APPLICATION MASK field contains a value that is a bit mask for enabling the checking of the LOGICAL BLOCK APPLICATION TAG in the protection information for each logical block of the range of logical blocks for this

command. A LOGICAL BLOCK APPLICATION TAG MASK bit set to one enables the checking of the corresponding bit in the EXPECTED LOGICAL BLOCK APPLICATION TAG field with the LOGICAL BLOCK APPLICATION TAG.

5.42 XDREAD (10) command

The XDREAD (10) command (see table 82) requests that the target transfer to the initiator the XOR data generated by an XDWRITE command. Data transferred from the device server includes user data and includes protection information as required by the XORPINFO bit and the medium format.

Table 82 — XDREAD (10) command

Byte\Bit	7	6	5	4	3	2	1	0
0	OPERATION CODE (52h)							
1	Reserved							XORPINFO
2	(MSB)							
5	LOGICAL BLOCK ADDRESS							(LSB)
6	Reserved			GROUP NUMBER				
7	(MSB)							
8	TRANSFER LENGTH							(LSB)
9	CONTROL							

See the PRE-FETCH (10) command (see 5.7) and 4.16 for a description of the GROUP NUMBER field.

If the XOR protection information (XORPINFO) bit is set to zero, the device server shall not check or transmit protection information. If the XORPINFO bit is set to one, the device server supports protection information, and the medium has been formatted with protection information, the device server shall transmit protection information but shall not check any of the fields. If the XORPINFO bit is set to one, the device server supports protection information, and the medium has not been formatted with protection information, the device server shall terminate the command with CHECK CONDITION status with a sense key of ILLEGAL REQUEST and an additional sense code of INVALID FIELD IN CDB. If the XORPINFO bit is set to one and the device server does not support protection information, the device server should terminate the command with CHECK CONDITION status with a sense key of ILLEGAL REQUEST and an additional sense code of INVALID FIELD IN CDB.

The XOR data transferred is identified by the LOGICAL BLOCK ADDRESS field and the TRANSFER LENGTH field. The LOGICAL BLOCK ADDRESS field and TRANSFER LENGTH field shall be the same as, or a subset of, those specified in a prior XDWRITE command. If a match is not found the command is terminated with a CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB. The TRANSFER LENGTH field is constrained by the MAXIMUM TRANSFER LENGTH field in the Block Limits VPD page (see 6.4.2).

5.43 XDREAD (32) command

The XDREAD (32) command (see table 83) requests that the target transfer to the initiator the XOR data generated by an XDWRITE command. Data transferred from the device server includes user data and includes protection information as required by the XORPINFO bit and the medium format.

Table 83 — XDREAD (32) command

Byte\Bit	7	6	5	4	3	2	1	0
0	OPERATION CODE (7Fh)							
1	CONTROL							
2	Reserved							
5								
6	Reserved			GROUP NUMBER				
7	ADDITIONAL CDB LENGTH (18h)							
8	(MSB)							
9	SERVICE ACTION (0003h)							(LSB)
10	Reserved							XORPINFO
11	Reserved							
12	(MSB)							
19	LOGICAL BLOCK ADDRESS							(LSB)
20	Reserved							
27								
28	(MSB)							
31	TRANSFER LENGTH							(LSB)

See the XDREAD (10) command (see 5.42) and SPC-3 for a description of the fields in this command.

5.44 XDWRITE (10) command

The XDWRITE (10) command (see table 84) requests that the target XOR the data transferred from the application client with the data on the medium. Data transferred from the application client includes user data and includes protection information as required by the WRPROTECT field and the medium format. The resulting XOR data is stored by the target until it is retrieved by an XDREAD (10) command. If the RTO_EN bit is set to one in the long read capacity data (see 5.15), the device server shall terminate this command with CHECK

CONDITION status with a sense key of ILLEGAL REQUEST and an additional sense code set to INVALID COMMAND OPERATION CODE.

Table 84 — XDWRITE (10) command

Byte\Bit	7	6	5	4	3	2	1	0
0	OPERATION CODE (50h)							
1	WRPROTECT			DPO	FUA	DISABLE WRITE	FUA_NV	Reserved
2	(MSB)							
5				LOGICAL BLOCK ADDRESS				(LSB)
6		Reserved		GROUP NUMBER				
7	(MSB)							
8				TRANSFER LENGTH				(LSB)
9	CONTROL							

See the WRITE (10) command (see 5.29) for a definition of the WRPROTECT field, the FUA bit, and the FUA_NV bit.

See the READ (10) command (see 5.10) for a definition of the DPO bit.

See the PRE-FETCH (10) command (see 5.7) and 4.16 for a description of the GROUP NUMBER field.

A DISABLE WRITE bit set to zero specifies that the data transferred from the application client shall be written to the medium after the XOR operation is complete. A DISABLE WRITE bit set to one specifies that the data shall not be written to the medium.

The LOGICAL BLOCK ADDRESS field specifies the starting LBA of the data on which an XOR operation shall be performed with the data from the medium.

The TRANSFER LENGTH field specifies the number of logical blocks that shall be transferred from the application client and the number of logical blocks on which an XOR operation shall be performed with the data from the medium. If the logical block address plus the transfer length exceeds the capacity of the medium, the device server shall return CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to LOGICAL BLOCK ADDRESS OUT OF RANGE. The TRANSFER LENGTH field is constrained by the MAXIMUM TRANSFER LENGTH field in the Block Limits VPD page (see 6.4.2).

The resulting XOR data is retrieved by an XDREAD command with starting LOGICAL BLOCK ADDRESS field and TRANSFER LENGTH field that match, or are a subset of, the starting LOGICAL BLOCK ADDRESS field and TRANSFER LENGTH field of this command.

5.45 XDWRITE (32) command

The XDWRITE (32) command (see table 85) requests that the target XOR the data transferred from the application client with the data on the medium. Data transferred from the application client includes user data and includes protection information as required by the WRPROTECT field and the medium format. The resulting XOR data is stored by the target until it is retrieved by an XDREAD (32) command. If the RTO_EN bit is set to one in the long read capacity data (see 5.15), the device server shall terminate this command with CHECK

CONDITION status with a sense key of ILLEGAL REQUEST and an additional sense code set to INVALID COMMAND OPERATION CODE.

Table 85 — XDWRITE (32) command

Byte\Bit	7	6	5	4	3	2	1	0
0	OPERATION CODE (7Fh)							
1	CONTROL							
2	Reserved							
5								
6	Reserved			GROUP NUMBER				
7	ADDITIONAL CDB LENGTH (18h)							
8	(MSB)							
9	SERVICE ACTION (0004h)							(LSB)
10	WRPROTECT			DPO	FUA	DISABLE WRITE	FUA_NV	Reserved
11	Reserved							
12	(MSB)							
19	LOGICAL BLOCK ADDRESS							(LSB)
20	Reserved							
27								
28	(MSB)							
31	TRANSFER LENGTH							(LSB)

See the XDWRITE (10) command (see 5.44) and SPC-3 for a description of the fields in this command.

5.46 XDWRITEREAD (10) command

The XDWRITEREAD (10) command (see table 86) requests that the target XOR the data transferred from the application client with the data on the medium and return the resulting XOR data to the application client. Data transferred to and from the application client includes user data and includes protection information as required by the WRPROTECT field, the XORPINFO bit, and the medium format. This is the equivalent to an XDWRITE (10) followed by an XDREAD (10) with the same LBA and transfer length. This command is only available on transport protocols supporting bidirectional commands. If the RTO_EN bit is set to one in the long read capacity data (see 5.15), the device server shall terminate this command with CHECK CONDITION

status with a sense key of ILLEGAL REQUEST and an additional sense code set to INVALID COMMAND OPERATION CODE.

Table 86 — XDWRITEREAD (10) command

Byte\Bit	7	6	5	4	3	2	1	0
0	OPERATION CODE (53h)							
1	WRPROTECT			DPO	FUA	DISABLE WRITE	FUA_NV	XORPINFO
2	(MSB)	LOGICAL BLOCK ADDRESS						
5								(LSB)
6	Reserved			GROUP NUMBER				
7	(MSB)	TRANSFER LENGTH						
8								(LSB)
9	CONTROL							

See the XDWRITE (10) command (see 5.44) and XDREAD (10) command (see 5.42) for a description of the fields in this command.

5.47 XDWRITEREAD (32) command

The XDWRITEREAD (32) command (see table 87) requests that the target XOR the data transferred from the application client with the data on the medium and return the resulting XOR data to the application client. Data transferred to and from the application client includes user data and includes protection information as required by the WRPROTECT field, the XORPINFO bit, and the medium format. This is the equivalent to an XDWRITE (32) followed by an XDREAD (32) with the same LBA and transfer length. This command is only available on transport protocols supporting bidirectional commands. If the RTO_EN bit is set to one in the long read capacity data (see 5.15), the device server shall terminate this command with CHECK CONDITION

status with a sense key of ILLEGAL REQUEST and an additional sense code set to INVALID COMMAND OPERATION CODE.

Table 87 — XDWRITEREAD (32) command

Byte\Bit	7	6	5	4	3	2	1	0
0	OPERATION CODE (7Fh)							
1	CONTROL							
2	Reserved							
5								
6	Reserved			GROUP NUMBER				
7	ADDITIONAL CDB LENGTH (18h)							
8	(MSB)							
9	SERVICE ACTION (0007h)							(LSB)
10	WRPROTECT			DPO	FUA	DISABLE WRITE	FUA_NV	XORPINFO
11	Reserved							
12	(MSB)							
19	LOGICAL BLOCK ADDRESS							(LSB)
20	Reserved							
27								
28	(MSB)							
31	TRANSFER LENGTH							(LSB)

See the XDWRITEREAD (10) command (see 5.46) and SPC-3 for a description of the fields in this command.

5.48 XPWRITE (10) command

The XPWRITE (10) command (see table 88) requests that the target XOR the data transferred from the application client with the data on the medium and then write the XOR data to the medium. Data transferred from the device server includes user data and includes protection information as required by the XORPINFO bit and the medium format.

Table 88 — XPWRITE (10) command

Byte\Bit	7	6	5	4	3	2	1	0
0	OPERATION CODE (51h)							
1	Reserved			DPO	FUA	Reserved	FUA_NV	XORPINFO
2	(MSB)							
5	LOGICAL BLOCK ADDRESS							(LSB)
6	Reserved			GROUP NUMBER				
7	(MSB)							
8	TRANSFER LENGTH							(LSB)
9	CONTROL							

See the READ (10) command (see 5.10) for a definition of the DPO bit. See the WRITE (10) command (see 5.29) for a definition of the FUA bit and the FUA_NV bit. See the PRE-FETCH (10) command (see 5.7) and 4.16 for a description of the GROUP NUMBER field.

If the XOR protection information (XORPINFO) bit is set to zero, the device server supports protection information, and the medium has been formatted with protection information, the device server shall terminate the command with CHECK CONDITION status with a sense key of ILLEGAL REQUEST and an additional sense code of INVALID FIELD IN CDB.

If the XORPINFO bit is set to one, the device server supports protection information, and the medium has been formatted with protection information, the device server shall XOR the data and protection information transferred from the application client with the data and protection information on the medium and then write the XOR result to the medium. The device server shall not check any of the fields.

If the XORPINFO bit is set to one, the device server supports protection information, and the medium has not been formatted with protection information, the device server shall terminate the command with CHECK CONDITION status with a sense key of ILLEGAL REQUEST and an additional sense code of INVALID FIELD IN CDB.

If the XORPINFO bit is set to one and the device server does not support protection information, the device server should terminate the command with CHECK CONDITION status with a sense key of ILLEGAL REQUEST and an additional sense code of INVALID FIELD IN CDB.

The LOGICAL BLOCK ADDRESS field specifies the starting LBA where the target shall read data from its medium. It also specifies the starting LBA where the XOR result data shall be written to the medium.

The TRANSFER LENGTH field specifies the number of logical blocks that shall be read from the medium. It also specifies the number of logical blocks that shall be written to the medium. If the logical block address plus the transfer length exceeds the capacity of the medium, the device server shall return CHECK CONDITION statuswith the sense key set to ILLEGAL REQUEST and the additional sense code set to LOGICAL BLOCK ADDRESS OUT OF RANGE. The TRANSFER LENGTH field is constrained by the MAXIMUM TRANSFER LENGTH field in the Block Limits VPD page (see 6.4.2).

5.49 XPWRITE (32) command

The XPWRITE (32) command (see table 89) requests that the target XOR the data transferred from the application client with the data on the medium and then write the XOR data to the medium. Data transferred

from the device server includes user data and includes protection information as required by the XORPINFO bit and the medium format.

Table 89 — XPWRITE (32) command

Byte\Bit	7	6	5	4	3	2	1	0
0	OPERATION CODE (7Fh)							
1	CONTROL							
2	Reserved							
5								
6	Reserved			GROUP NUMBER				
7	ADDITIONAL CDB LENGTH (18h)							
8	(MSB)							
9	SERVICE ACTION (0006h)							(LSB)
10	Reserved			DPO	FUA	Reserved	FUA_NV	XORPINFO
11	Reserved							
12	(MSB)							
19	LOGICAL BLOCK ADDRESS							(LSB)
20	Reserved							
27								
28	(MSB)							
31	TRANSFER LENGTH							(LSB)

See the XPWRITE (10) command (see 5.48) and SPC-3 for a description of the fields in this command.

6 Parameters for direct-access block devices

6.1 Diagnostic parameters

6.1.1 Diagnostic parameters overview

This subclause defines the descriptors and pages for diagnostic parameters used with direct-access devices. The diagnostic page codes for direct-access devices are defined in table 90.

Table 90 — Diagnostic page codes

Diagnostic page code	Description	Reference
00h	Supported diagnostic pages	SPC-3
01h - 2Fh	SCSI enclosure services diagnostic pages	SES-2
30h - 3Fh	Diagnostic pages assigned by SPC-3	SPC-3
40h	Translate Address Output diagnostic page	6.1.2
	Translate Address Input diagnostic page	6.1.3
41h	Obsolete (Device Status diagnostic pages)	
42h - 7Fh	Reserved for this standard	
80h - FFh	Vendor-specific diagnostic pages	

6.1.2 Translate Address Output diagnostic page

The Translate Address diagnostic pages allow the application client to translate an address in one of the formats supported by the FORMAT UNIT command (see 5.4.2.4) - a short block format address, a long block format address, a physical sector format address, or a bytes from index format address - into any one of the other formats. The address to be translated is passed to the device server with the SEND DIAGNOSTIC command and the results are returned to the application client by the RECEIVE DIAGNOSTIC RESULTS command. The format of the Translate Address Output diagnostic page sent with SEND DIAGNOSTIC is shown in table 91. The translated address is returned in the Translate Address Input diagnostic page (see table 92).

Table 91 — Translate Address Output diagnostic page

Byte\Bit	7	6	5	4	3	2	1	0
0	PAGE CODE (40h)							
1	Reserved							
2	(MSB)							
3	PAGE LENGTH (000Ah)							(LSB)
4	Reserved					SUPPLIED FORMAT		
5	Reserved					TRANSLATE FORMAT		
6	(MSB)							
13	ADDRESS TO TRANSLATE							(LSB)

The SUPPLIED FORMAT field specifies the format of ADDRESS TO TRANSLATE field. Valid values for this field are defined in the DEFECT LIST FORMAT field of the FORMAT UNIT command (see 5.4). If the device server does not support the requested format it shall terminate the SEND DIAGNOSTIC command with CHECK

CONDITION status. The sense key shall be set to ILLEGAL REQUEST and the additional sense code shall be set to INVALID FIELD IN PARAMETER LIST.

The TRANSLATE FORMAT field specifies the format the device server shall use for the result of the address translation. Valid values for this field are defined in the DEFECT LIST FORMAT field of the FORMAT UNIT command. If the device server does not support the specified format it shall terminate the command with CHECK CONDITION status, with a sense key set to ILLEGAL REQUEST and an additional sense code set to INVALID FIELD IN PARAMETER LIST.

The ADDRESS TO TRANSLATE field contains a single address the application client is requesting the device server to translate. The format of this field depends on the value in the SUPPLIED FORMAT field. The formats are described in 5.4.2.4. If the short block format address descriptor is specified, the first four bytes of ADDRESS TO TRANSLATE field field shall contain the short block format address descriptor and the last four bytes shall contain 00000000h.

6.1.3 Translate Address Input diagnostic page

Table 92 defines the Translate Address Input diagnostic page retrieved with RECEIVE DIAGNOSTIC RESULTS after the Translate Address Output diagnostic page has been sent with SEND DIAGNOSTIC. If a Translate Address Output diagnostic page has not yet been processed, the results of a RECEIVE DIAGNOSTIC RESULTS command requesting this diagnostic page are vendor-specific.

Table 92 — Translate Address Input diagnostic page

Byte\Bit	7	6	5	4	3	2	1	0
0	PAGE CODE (40h)							
1	Reserved							
2	(MSB)	PAGE LENGTH (n - 3)						
3								(LSB)
4	Reserved					SUPPLIED FORMAT		
5	RAREA	ALTSEC	ALTTRK	Reserved		TRANSLATED FORMAT		
Translated address(es)								
6	(MSB)	TRANSLATED ADDRESS 1						
13								(LSB)
	…							
n - 7	(MSB)	TRANSLATED ADDRESS x (if required)						
n								(LSB)

The Translate Address Input diagnostic page contains a four-byte page header that specifies the page code and length followed by two bytes that describe the translated address followed by zero or more translated address(s).

The PAGE LENGTH field contains the number of parameter bytes that follow.

The SUPPLIED FORMAT field contains the value from the SUPPLIED FORMAT field in the previous Translate Address Output diagnostic page (see 6.1.2).

A reserved area (RAREA) bit set to zero indicates that no part of the translated address falls within a reserved area of the medium. A RAREA bit set to one indicates that all or part of the translated address falls within a reserved area of the medium (e.g., speed tolerance gap, alternate sector, vendor reserved area, etc.). If the entire translated address falls within a reserved area, the device server may not return a translated address.

An alternate sector (ALTSEC) bit set to zero indicates that no part of the translated address is located in an alternate sector of the medium or that the device server is unable to determine this information. An ALTSEC bit

set to one indicates that the translated address is physically located in an alternate sector of the medium. If the device server is unable to determine if all or part of the translated address is located in an alternate sector it shall set this bit to zero.

An alternate track (ALTTRK) bit set to zero indicates that no part of the translated address is located on an alternate track of the medium. An ALTTRK bit set to one indicates that part or all of the translated address is located on an alternate track of the medium or the device server is unable to determine if all or part of the translated address is located on an alternate track.

The TRANSLATED FORMAT field contains the value from the TRANSLATE FORMAT field in the previous Translate Address Output diagnostic page (see 6.1.2).

The TRANSLATED ADDRESS field(s) contains the address(es) the device server translated from the address supplied by the application client in the previous Translate Address Output diagnostic page. Each field shall be in the format specified in the TRANSLATE FORMAT field. The different formats are described in 5.4.2.4. If the short block format address descriptor is specified, the first four bytes of the TRANSLATED ADDRESS field shall contain the short block format address descriptor and the last four bytes shall contain 00000000h.

If the returned data is in short block format, long block format, or physical sector format and the ADDRESS TO TRANSLATE field in the previous Translate Address Output diagnostic page covers more than one address after it has been translated (e.g., because of multiple physical sectors within a single logical block or multiple logical blocks within a single physical sector) the device server shall return all possible addresses that are contained in the area specified by the address to be translated. If the returned data is in bytes from index format, the device server shall return a pair of translated values for each of the possible addresses that are contained in the area specified by the ADDRESS TO TRANSLATE field in the previous Translate Address Output diagnostic page. Of the pair of translated values returned, the first indicates the starting location and the second the ending location of the area.

6.2 Log parameters

6.2.1 Log parameters overview

This subclause defines the descriptors and pages for log parameters used with direct-access devices. See SPC-3 for a detailed description of logging operations. The log page codes for direct-access devices are defined in table 93.

Table 93 — Log page codes

Log page code	Description	Reference
00h	Supported Log Pages log page	SPC-3
01h	Buffer Over-Run/Under-Run log page	SPC-3
02h	Write Error Counter log page	SPC-3
03h	Read Error Counter log page	SPC-3
04h	Reserved	
05h	Verify Error Counter log page	SPC-3
06h	Non-Medium Error log page	SPC-3
07h	Last n Error Events log page	SPC-3
08h	Format Status log page	6.2.2
09h	Restricted (see SPC-3)	
0Ah	Restricted (see SPC-3)	
0Bh	Last n Deferred Errors Or Asynchronous Events log page	SPC-3
0Ch	Reserved	
0Dh	Temperature log page	SPC-3
0Eh	Start-Stop Cycle Counter log page	SPC-3
0Fh	Application Client log page	SPC-3
10h	Self-Test Results log page	SPC-3
11h - 16h	Reserved	
17h	Non-volatile Cache log page	6.2.3
18h	Protocol-Specific Port log page	SPC-3
19h - 2Eh	Reserved	
2Fh	Informational Exceptions log page	SPC-3
30h - 3Eh	Vendor-specific	
3Fh	Reserved	

6.2.2 Format Status log page

The Format Status log page (log page code 08h) captures the state of the block device since the most recent successful FORMAT UNIT command (see 5.4) was completed. Additionally, this log page provides Defect

Management information for the device server. Table 94 defines the parameter codes for the Format Status log page.

Table 94 — Format Status log page parameter codes

Parameter code	Description
0000h	Format DATA OUT
0001h	Grown defects during certification
0002h	Total blocks reallocated during format
0003h	Total new blocks reallocated
0004h	Power on minutes since format
0005h - 7FFFh	Reserved
8000h - FFFFh	Vendor-specific parameters

Event counts are returned as a result of the LOG SENSE command. The default value for each event count listed table 94 shall be zero. Attempts to change these event counts by issuing a LOG SELECT with these fields set to non-zero values is not considered an error and shall have no effect on the saved values.

If information about a log parameter is not available, the device server shall return a value of FFh. If the most recent FORMAT UNIT command failed, the device server shall return a value of FFh for each log parameter.

The FORMAT DATA OUT field contains the entire FORMAT UNIT parameter list (see 5.4.2) of the most recently successful FORMAT UNIT command completed. This includes:

a) the parameter list header;
b) the initialization pattern descriptor, if any; and
c) the defect list, if any.

The GROWN DEFECTS DURING CERTIFICATION field is a count of the number of defects detected as a result of performing certification during processing of a FORMAT UNIT command. This count reflects only those defects detected and replaced that were not already part of the PLIST or GLIST. If a certification pass was not performed the GROWN DEFECTS DURING CERTIFICATION field shall be set to zero.

The TOTAL BLOCKS REALLOCATED DURING FORMAT field is a count of the total number of blocks that have been reallocated since the completion of the last successful FORMAT UNIT command.

The POWER ON MINUTES SINCE FORMAT field represents the unsigned number of usage minutes (i.e., power applied regardless of power state) that have elapsed since the most recently successful FORMAT UNIT command.

Upon receiving the FORMAT UNIT command, the device server should set all fields within the Format Status log page to reflect no such information being available. Only upon successful completion of the FORMAT UNIT command should the device server update the affected fields.

The target save disable (TSD) bit shall always be set to zero to indicate that the device server provides an implicit saving frequency.

NOTE 23 - Removable media device servers may save log page information with the medium in a vendor-specific manner and location.

6.2.3 Non-volatile Cache log page

The Non-volatile Cache log page defined in table 95 indicates the status of battery backup for a non-volatile cache.

Table 95 — Non-volatile Cache log page

Byte\Bit	7	6	5	4	3	2	1	0
0	PAGE CODE (17h)							
1	Reserved							
2	(MSB)							
3	PAGE LENGTH (n - 3)							(LSB)
4								
n	Non-volatile cache log parameters							

Table 96 defines the parameter codes.

Table 96 — Non-volatile Cache log parameters

Code	Description
0000h	Remaining Non-volatile Time
0001h	Maximum Non-volatile Time
All others	Reserved

The Remaining Non-volatile Time parameter has the format shown in table 97.

Table 97 — Remaining Non-volatile Time parameter data

Byte\Bit	7	6	5	4	3	2	1	0
0	PARAMETER LENGTH (3h)							
1	(MSB)							
3	REMAINING NON-VOLATILE TIME							(LSB)

The REMAINING NON-VOLATILE TIME field is defined in table 98 .

Table 98 — Remaining non-volatile time

Code	Description
000000h	Non-volatile cache is volatile (either permanently or temporarily, e.g., if batteries need to be recharged).
000001h	Non-volatile cache is expected to remain non-volatile for an unknown amount of time (e.g., if battery status is unknown)
000002h to FFFFFEh	Non-volatile cache is expected to remain non-volatile for the number of minutes indicated (e.g., battery-backed random access memory).
FFFFFFh	Non-volatile cache is indefinitely non-volatile.

The Maximum Non-volatile Time parameter has the format shown in table 99.

Table 99 — Maximum Non-volatile Time parameter data

Byte\Bit	7	6	5	4	3	2	1	0
0	PARAMETER LENGTH (3h)							
1	(MSB)	MAXIMUM NON-VOLATILE TIME						
3								(LSB)

The MAXIMUM NON-VOLATILE TIME field is defined in table 100.

Table 100 — Maximum non-volatile time

Code	Description
000000h	Non-volatile cache is volatile
000001h	Reserved
000002h to FFFFFEh	Non-volatile cache is capable of being non-volatile for the estimated number of minutes indicated. If the time is based on batteries, it shall be based on the last full charge capacity rather than the design capacity of the batteries.
FFFFFFh	Non-volatile cache is indefinitely non-volatile.

6.3 Mode parameters

6.3.1 Mode parameters overview

This subclause defines the block descriptors and mode pages used with direct-access devices.

The mode parameter list, including the mode parameter header, is described in SPC-3. Direct-access devices support zero or one mode parameter block descriptors (i.e., the block descriptor is shared by all the logical blocks on the medium).

The MEDIUM TYPE field in the mode parameter header (see SPC-3) shall be set to 00h.

The DEVICE-SPECIFIC PARAMETER field in the mode parameter header (see SPC-3) is defined for direct-access devices in table 101.

Table 101 — DEVICE-SPECIFIC PARAMETER **field for direct-access devices**

Bit	7	6	5	4	3	2	1	0
	WP	Reserved		DPOFUA	Reserved			

When used with the MODE SELECT command, the write protect (WP) bit is not defined.

When used with the MODE SENSE command, a WP bit set to zero indicates that the medium is write enabled. A WP bit set to one indicates that the medium is write protected.

When used with the MODE SELECT command, the DPOFUA bit is reserved.

When used with the MODE SENSE command, a DPOFUA bit set to zero indicates that the device server does not support the DPO and FUA bits. When used with the MODE SENSE command, a DPOFUA bit set to one indicates that the device server supports the DPO and FUA bits (see 4.9).

The mode page codes and subpage codes for direct-access devices are shown in table 102.

Table 102 — Mode page codes for direct-access devices

Mode page code	Description	Reference
00h	Vendor-specific (does not require page format)	
00h-3Eh/FFh	Return all subpages [a]	SPC-3
01h	Read-Write Error Recovery mode page	6.3.4
02h	Disconnect-Reconnect mode page	SPC-3
03h	Obsolete (Format Device mode page)	
04h	Obsolete (Rigid Disk Geometry mode page)	
05h	Obsolete (Flexible Disk mode page)	
06h	Reserved	
07h	Verify Error Recovery mode page	6.3.5
08h	Caching mode page	6.3.3
09h	Obsolete	
0Ah/00h	Control mode page	SPC-3
0Ah/01h	Control Extension mode page	SPC-3
0Ah/02h - 3Eh	Reserved	
0Bh	Obsolete (Medium Types Supported mode page)	
0Ch	Obsolete (Notch And Partition mode page)	
0Dh	Obsolete	
0Eh - 0Fh	Reserved	
10h	XOR Control mode page	6.3.6
11h - 13h	Reserved	
14h	Enclosure Services Management [b]	SES-2
15h - 17h	Reserved	
18h	Protocol-Specific LUN mode page	SPC-3
19h	Protocol-Specific Port mode page	SPC-3
1Ah	Power Condition mode page	SPC-3
1Bh	Reserved	
1Ch	Informational Exceptions Control mode page	SPC-3
1Dh - 1Fh	Reserved	
20h - 3Eh	Vendor-specific (does not require page format)	
3Fh/00h	Return all mode pages [a]	SPC-3
3Fh/01h - 3Eh	Reserved	
3Fh/FFh	Return all mode pages and subpages [a]	SPC-3

[a] Valid only for the MODE SENSE command
[b] Valid only if the ENCSERV bit is set to one in the standard INQUIRY data (see SPC-3)

6.3.2 Mode parameter block descriptors

6.3.2.1 Mode parameter block descriptors overview

If it returns a mode parameter block descriptor, the device server shall return a short LBA mode parameter block descriptor (see 6.3.2.2) in the mode parameter data in response to:

a) a MODE SENSE (6) command; or
b) a MODE SENSE (10) command with the LLBAA bit set to zero.

If it returns a mode parameter block descriptor and the number of blocks is greater than FFFFFFFFh, the device server may return a long LBA mode parameter block descriptor (see 6.3.2.3) in the mode parameter data in response to a MODE SENSE (10) command with the LLBAA bit set to one.

If it sends a mode parameter block descriptor in the mode parameter list, the application client shall send a short LBA mode parameter block descriptor (see 6.3.2.2) for a MODE SELECT (6) command.

If it sends a mode parameter block descriptor in the mode parameter list, the application client may send a long LBA mode parameter block descriptor (see 6.3.2.3) for a MODE SELECT (10) command.

Support for the mode parameter block descriptors is optional. A unit attention condition (see SPC-3 and SAM-3) shall be generated when the block descriptor values are changed.

6.3.2.2 Short LBA mode parameter block descriptor

Table 103 defines the block descriptor for direct-access devices used:

a) with the MODE SELECT (6) and MODE SENSE (6) commands; and
b) with the MODE SELECT (10) and MODE SENSE (10) commands when the LONGLBA bit is set to zero in the mode parameter header (see SPC-3).

Table 103 — Short LBA mode parameter block descriptor

Byte\Bit	7	6	5	4	3	2	1	0
0	(MSB)							
				NUMBER OF BLOCKS				
3								(LSB)
4					Reserved			
5	(MSB)							
				BLOCK LENGTH				
7								(LSB)

A device server shall respond to a MODE SENSE command (see SPC-3) by reporting the number of blocks specified in the NUMBER OF BLOCKS field sent in the last MODE SELECT command that contained a mode parameter block descriptor. If no MODE SELECT command with a mode parameter block descriptor has been received then the current number of blocks shall be returned. To determine the number of blocks at which the logical unit is currently formatted the application client shall use the READ CAPACITY command (see 5.15).

On a MODE SENSE command, the NUMBER OF BLOCKS field indicates the number of logical blocks on the medium to which the BLOCK LENGTH field applies. A value of zero indicates that all of the remaining logical blocks of the logical unit have the medium characteristics specified.

On a MODE SENSE command, 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 FFFFFFFFh indicates that all of the remaining logical blocks of the logical unit have the medium characteristics specified.

If the SCSI device does not support changing its capacity by changing the NUMBER OF BLOCKS field using the MODE SELECT command (see SPC-3), the value in the NUMBER OF BLOCKS field is ignored. If the device

supports changing its capacity by changing the NUMBER OF BLOCKS field, then the NUMBER OF BLOCKS field is interpreted as follows:

a) If the NUMBER OF BLOCKS field is set to zero, the device shall retain its current capacity if the block length has not changed. If the NUMBER OF BLOCKS field is set to zero and the content of the BLOCK LENGTH field (i.e., new block length) is different than the current block length, the device shall be set to its maximum capacity when the new block length takes effect (i.e., after a successful FORMAT UNIT command);
b) If the NUMBER OF BLOCKS field is greater than zero and less than or equal to its maximum capacity, the device shall be set to that number of blocks. If the content of the BLOCK LENGTH field is the same as the current block length, the device shall not become format corrupted. This capacity setting shall be retained through power cycles, hard resets, logical unit resets, and I_T nexus losses. If the content of the BLOCK LENGTH field is the same as the current block length this capacity setting shall take effect on successful completion of the MODE SELECT command. If the content of the BLOCK LENGTH field (i.e., new block length) is different than the current block length this capacity setting shall take effect when the new block length takes effect (i.e., after a successful FORMAT UNIT command);
c) If the NUMBER OF BLOCKS field is set to a value greater than the maximum capacity of the device and less than FFFFFFFFh, then the command is terminated with a CHECK CONDITION status. The sense key is set to ILLEGAL REQUEST. The device shall retain its previous block descriptor settings; or
d) If the NUMBER OF BLOCKS field is set to FFFFFFFFh, the device shall be set to its maximum capacity. If the content of the BLOCK LENGTH field is the same as the current block length, the device shall not become format corrupted. This capacity setting shall be retained through power cycles, hard resets, logical unit resets, and I_T nexus losses. If the content of the BLOCK LENGTH field is the same as the current block length this capacity setting shall take effect on successful completion of the MODE SELECT command. If the content of the BLOCK LENGTH field (i.e., new block length) is different than the current block length this capacity setting shall take effect when the new block length takes effect (i.e., after a successful FORMAT UNIT command).

The BLOCK LENGTH field specifies the length in bytes of each logical block. No change shall be made to any logical blocks on the medium until a format operation (see 5.4) is initiated by an application client.

A device server shall respond to a MODE SENSE command (see SPC-3) by reporting the length of the logical blocks as specified in the BLOCK LENGTH field sent in the last MODE SELECT command that contained a mode parameter block descriptor. If no MODE SELECT command with a block descriptor has been received then the current block length shall be returned (e.g., if the block length is 512 bytes and a MODE SELECT command occurs with the BLOCK LENGTH field set to 520 bytes, any MODE SENSE commands would return 520 in the BLOCK LENGTH field). To determine the block length at which the logical unit is currently formatted the application client shall use the READ CAPACITY command (see 5.15).

6.3.2.3 Long LBA mode parameter block descriptor

Table 104 defines the block descriptor for direct-access devices used with the MODE SELECT (10) and MODE SENSE (10) commands when the LONGLBA bit is set to one in the mode parameter header (see SPC-3).

Table 104 — Long LBA mode parameter block descriptor

Byte\Bit	7	6	5	4	3	2	1	0
0	(MSB)							
7				NUMBER OF BLOCKS				(LSB)
8								
11				Reserved				
12	(MSB)							
15				BLOCK LENGTH				(LSB)

A device server shall respond to a MODE SENSE command (see SPC-3) by reporting the number of blocks specified in the NUMBER OF BLOCKS field sent in the last MODE SELECT command that contained a mode parameter block descriptor. If no MODE SELECT command with a mode parameter block descriptor has been received then the current number of blocks shall be returned. To determine the number of blocks at which the logical unit is currently formatted the application client shall use the READ CAPACITY command (see 5.15).

On a MODE SENSE command the NUMBER OF BLOCKS field indicates the number of logical blocks on the medium to which the BLOCK LENGTH field applies. A value of zero indicates that all of the remaining logical blocks of the logical unit have the medium characteristics specified.

If the SCSI device does not support changing its capacity by changing the NUMBER OF BLOCKS field using the MODE SELECT command (see SPC-3), the value in the NUMBER OF BLOCKS field is ignored. If the device supports changing its capacity by changing the NUMBER OF BLOCKS field, then the NUMBER OF BLOCKS field is interpreted as follows:

a) If the NUMBER OF BLOCKS field is set to zero, the device shall retain its current capacity if the block length has not changed. If the NUMBER OF BLOCKS field is set to zero and the content of the BLOCK LENGTH field (i.e., new block length) is different than the current block length, the device shall be set to its maximum capacity when the new block length takes effect (i.e., after a successful FORMAT UNIT command);
b) If the NUMBER OF BLOCKS field is greater than zero and less than or equal to its maximum capacity, the device shall be set to that number of blocks. If the content of the BLOCK LENGTH field is the same as the current block length, the device shall not become format corrupted. This capacity setting shall be retained through power cycles, hard resets, logical unit resets, and I_T nexus losses. If the content of the BLOCK LENGTH field is the same as the current block length this capacity setting shall take effect on successful completion of the MODE SELECT command. If the content of the BLOCK LENGTH field (i.e., new block length) is different than the current block length this capacity setting shall take effect when the new block length takes effect (i.e., after a successful FORMAT UNIT command);
c) If the NUMBER OF BLOCKS field is set to a value greater than the maximum capacity of the device and less than FFFFFFFF FFFFFFFFh, then the command is terminated with a CHECK CONDITION status. The sense key is set to ILLEGAL REQUEST. The device shall retain its previous block descriptor settings; or
d) If the NUMBER OF BLOCKS field is set to FFFFFFFF FFFFFFFFh, the device shall be set to its maximum capacity. If the content of the BLOCK LENGTH field is the same as the current block length, the device shall not become format corrupted. This capacity setting shall be retained through power cycles, hard resets, logical unit resets, and I_T nexus losses. If the content of the BLOCK LENGTH field is the same as the current block length this capacity setting shall take effect on successful completion of the MODE SELECT command. If the content of the BLOCK LENGTH field (i.e., new block length) is different than the current block length this capacity setting shall take effect when the new block length takes effect (i.e., after a successful FORMAT UNIT command).

The BLOCK LENGTH field specifies the length in bytes of each logical block. No change shall be made to any logical blocks on the medium until a format operation (see 5.4) is initiated by an application client.

A device server shall respond to a MODE SENSE command (see SPC-3) by reporting the length of the logical blocks as specified in the BLOCK LENGTH field sent in the last MODE SELECT command that contained a mode parameter block descriptor. If no MODE SELECT command with a block descriptor has been received then the current block length shall be returned (e.g., if the block length is 512 bytes and a MODE SELECT command occurs with the BLOCK LENGTH field set to 520 bytes, any MODE SENSE commands would return 520 in the BLOCK LENGTH field). To determine the block length at which the logical unit is currently formatted the application client shall use the READ CAPACITY command (see 5.15).

6.3.3 Caching mode page

The Caching mode page (see table 105) defines the parameters that affect the use of the cache.

Table 105 — Caching mode page

Byte\Bit	7	6	5	4	3	2	1	0
0	PS	Reserved	PAGE CODE (08h)					
1	PAGE LENGTH (12h)							
2	IC	ABPF	CAP	DISC	SIZE	WCE	MF	RCD
3	DEMAND READ RETENTION PRIORITY				WRITE RETENTION PRIORITY			
4	(MSB)	DISABLE PRE-FETCH TRANSFER LENGTH						
5								(LSB)
6	(MSB)	MINIMUM PRE-FETCH						
7								(LSB)
8	(MSB)	MAXIMUM PRE-FETCH						
9								(LSB)
10	(MSB)	MAXIMUM PRE-FETCH CEILING						
11								(LSB)
12	FSW	LBCSS	DRA	Vendor specific		Reserved		NV_DIS
13	NUMBER OF CACHE SEGMENTS							
14	(MSB)	CACHE SEGMENT SIZE						
15								(LSB)
16	Reserved							
17	(MSB)	NON CACHE SEGMENT SIZE						
19								(LSB)

The parameters savable (PS) bit is only used with the MODE SENSE command. This bit is reserved with the MODE SELECT command. A PS bit set to one indicates that the device server is capable of saving the mode page in a non-volatile vendor-specific location. If the PS is set to one in MODE SENSE data then the mode page shall be savable by issuing a MODE SELECT command with the SP bit set to one.

An initiator control (IC) enable bit set to one specifies that the device server use the NUMBER OF CACHE SEGMENTS field or the CACHE SEGMENT SIZE field, dependent upon the SIZE bit, to control the caching algorithm rather than the device server's own adaptive algorithm.

An abort pre-fetch (ABPF) bit set to one, with the DRA bit set to zero, specifies that the device server abort a pre-fetch upon receipt of a new command. An ABPF bit set to one takes precedence over the value specified in the MINIMUM PRE-FETCH field. An ABPF bit set to zero, with the DRA bit set to zero, specifies that the termination of any active pre-fetch is dependent upon Caching mode page bytes 4 through 11 and is operation and/or vendor-specific.

A caching analysis permitted (CAP) bit set to one specifies that the device server perform caching analysis during subsequent operations. A CAP bit set to zero specifies that caching analysis be disabled to reduce overhead time or to prevent nonpertinent operations from impacting tuning values.

A discontinuity (DISC) bit set to one specifies that the device server continue the pre-fetch across time discontinuities (e.g., across cylinders) up to the limits of the buffer, or segment, space available for the pre-fetch. A DISC bit set to zero specifies that pre-fetches be truncated or wrapped at time discontinuities.

A size enable (SIZE) bit set to one specifies that the CACHE SEGMENT SIZE field be used to control caching segmentation. A SIZE bit set to zero specifies that the NUMBER OF CACHE SEGMENTS field be used to control caching segmentation. Simultaneous use of both the number of segments and the segment size is vendor-specific.

A writeback cache enable (WCE) bit set to zero specifies that the device server shall return GOOD status for a WRITE command only after successfully writing all of the data to the medium. A WCE bit set to one specifies that the device server may return GOOD status for a WRITE command after successfully receiving the data and prior to having successfully written it to the medium.

A multiplication factor (MF) bit set to zero specifies that the device server shall interpret the MINIMUM and MAXIMUM PRE-FETCH fields in terms of the number of logical blocks for each of the respective types of pre-fetch. An MF bit set to one specifies that the device server shall interpret the MINIMUM and MAXIMUM PRE-FETCH fields to be specified in terms of a scalar number that, when multiplied by the number of logical blocks to be transferred for the current command, yields the number of logical blocks for each of the respective types of pre-fetch.

A read cache disable (RCD) bit set to zero specifies that the device server may return data requested by a READ command by accessing either the cache or medium. A RCD bit set to one specifies that the device server shall transfer all of the data requested by a READ command from the medium (i.e., data shall not be transferred from the cache).

The DEMAND READ RETENTION PRIORITY field (see table 106) advises the device server the retention priority to assign for data read into the cache that has also been transferred from the logical unit to the application client.

Table 106 — Demand read retention priority and write retention priority

Value	Description
0h	The device server should not distinguish between retaining the indicated data and data placed into the cache memory by other means (e.g., pre-fetch).
1h	Demand read retention priority: Data put into the cache via a READ command should be replaced sooner (i.e., has lower priority) than data placed into the cache by other means (e.g., pre-fetch). Write retention priority: Data put into the cache during a WRITE or WRITE AND VERIFY command should be replaced sooner (i.e., has lower priority) than data placed into the cache by other means (e.g., pre-fetch).
2h - Eh	Reserved
Fh	Demand read retention priority: Data put into the cache via a READ command should not be replaced if there is other data in the cache that was placed into the cache by other means (e.g., pre-fetch) and it may be replaced (i.e., it is not locked). Write retention priority: Data put into the cache during a WRITE or WRITE AND VERIFY command should not be replaced if there is other data in the cache that was placed into the cache by other means (e.g., pre-fetch) and it may be replaced (i.e., it is not locked).

The WRITE RETENTION PRIORITY field advises the device server the retention priority to assign for data written into the cache that has also been transferred from the cache memory to the medium.

An anticipatory pre-fetch occurs when data is placed in the cache that has not been requested. This may happen in conjunction with the reading of data that has been requested. All the following parameters give an indication to the device server how it should manage the cache based on the last READ command. An anticipatory pre-fetch may occur based on other information. All the remaining caching parameters are only recommendations to the device server and should not cause a CHECK CONDITION to occur if the device server is not able to satisfy the request.

The DISABLE PRE-FETCH TRANSFER LENGTH field specifies the selective disabling of anticipatory pre-fetch on long transfer lengths. The value in this field is compared to the number of blocks requested by the current

READ command. If the number of blocks is greater than the disable pre-fetch transfer length, then an anticipatory pre-fetch is not done for the command. Otherwise the device server should attempt an anticipatory pre-fetch. If the pre-fetch disable transfer length is zero, then all anticipatory pre-fetching is disabled for any request for data, including those for zero logical blocks.

The MINIMUM PRE-FETCH field specifies either a number of blocks or a scalar multiplier of the TRANSFER LENGTH, depending upon the setting of the MF bit. In either case, the resulting number of blocks is the number to pre-fetch regardless of the delays it might cause in processing subsequent commands.

The pre-fetching operation begins at the logical block after the last logical block of the previous READ command. Pre-fetching shall always halt before exceeding the end of the medium. Errors that occur during the pre-fetching operation shall not be reported to the application client unless the device server is unable to, as a result of the error, process subsequent commands correctly. In this case the error may be reported either as an error for the current READ command, or as a deferred error, at the discretion of the device server and according to the rules for reporting deferred errors.

If the pre-fetch has read more than the amount of data specified by the MINIMUM PRE-FETCH field, then pre-fetching should be terminated whenever another command is ready to be processed. This consideration is ignored when the MINIMUM PRE-FETCH field value is equal to the MAXIMUM PRE-FETCH field value.

The MAXIMUM PRE-FETCH field specifies either a number of blocks or a scalar multiplier of the TRANSFER LENGTH, depending upon the setting of the MF bit. In either case, the resulting number of blocks is the number to pre-fetch if the pre-fetch does not delay processing of subsequent commands.

The MAXIMUM PRE-FETCH field contains the maximum amount of data to pre-fetch into the cache as a result of one READ command. It is used in conjunction with the DISABLE PRE-FETCH TRANSFER LENGTH and MAXIMUM PRE- FETCH CEILING parameters to trade off pre-fetching new data with displacing old data already stored in the cache.

The MAXIMUM PRE-FETCH CEILING field specifies an upper limit on the number of logical blocks computed as the maximum pre-fetch. If this number of blocks is greater than the MAXIMUM PRE-FETCH, then the number of logical blocks to pre-fetch shall be truncated to the value stored in the MAXIMUM PRE-FETCH CEILING field.

NOTE 24 - If the MF bit is set to one the MAXIMUM PRE-FETCH CEILING field is useful in limiting the amount of data to be pre-fetched.

A force sequential write (FSW) bit set to one specifies that multiple block writes are to be transferred from the application client data-out buffer and written to the medium in an ascending, sequential, logical block order. An FSW bit set to zero specifies that the device server is allowed to reorder the sequence of writing addressed logical blocks in order to achieve a faster command completion.

A logical block cache segment size (LBCSS) bit set to one specifies that the CACHE SEGMENT SIZE field units shall be interpreted as logical blocks. An LBCSS bit set to zero specifies that the CACHE SEGMENT SIZE field units shall be interpreted as bytes. The LBCSS shall not impact the units of other fields.

A disable read-ahead (DRA) bit set to one specifies that the device server not read into the buffer any logical blocks beyond the addressed logical block(s). A DRA bit set to zero specifies that the device server may continue to read logical blocks into the buffer beyond the addressed logical block(s).

The NUMBER OF CACHE SEGMENTS field specifies the number of segments into which the device server shall divide the cache.

The CACHE SEGMENT SIZE field specifies the segment size in bytes. The CACHE SEGMENT SIZE field is valid only when the SIZE bit is set to one.

An NV_DIS bit set to one specifies that the device server shall disable a non-volatile cache and indicates that a non-volatile cache is supported but disabled. An NV_DIS bit set to zero specifies that the device server may use a non-volatile cache and indicates that a non-volatile cache may be present and enabled.

A NON CACHE SEGMENT SIZE field that is greater than zero specifies that the device server allocate for a buffer function when all other cache segments are occupied by data to be retained. If the number is at least one, caching functions in the other segments need not be impacted by cache misses to perform the SCSI buffer function. A NON CACHE SEGMENT SIZE field set to zero is vendor-specific. If the sum of the NON CACHE SEGMENT

SIZE field value plus the CACHE SEGMENT SIZE field value is greater than the buffer size, the result is vendor-specific.

6.3.4 Read-Write Error Recovery mode page

The Read-Write Error Recovery mode page (see table 107) specifies the error recovery parameters the device server shall use during any command that performs a read or write operation to the medium (e.g., READ, WRITE, WRITE AND VERIFY, etc.).

Table 107 — Read-Write Error Recovery mode page

Byte\Bit	7	6	5	4	3	2	1	0
0	PS	Reserved	PAGE CODE (01h)					
1	PAGE LENGTH (0Ah)							
2	AWRE	ARRE	TB	RC	EER	PER	DTE	DCR
3	READ RETRY COUNT							
4	Obsolete							
5	Obsolete							
6	Obsolete							
7	Reserved							Restricted for MMC-4
8	WRITE RETRY COUNT							
9	Reserved							
10	(MSB)	RECOVERY TIME LIMIT						
11								(LSB)

The parameters savable (PS) bit is only used with the MODE SENSE command. This bit is reserved with the MODE SELECT command. A PS bit set to one indicates that the device server is capable of saving the mode page in a non-volatile vendor-specific location.

An automatic write reallocation enabled (AWRE) bit set to zero specifies that the device server shall not perform automatic reallocation of defective logical blocks during write operations.

An AWRE bit set to one specifies that the device server shall enable automatic reallocation to be performed during write operations. The automatic reallocation shall be performed only if the device server has the valid data (e.g., original data in the buffer or recovered from the medium). The valid data shall be placed in the reallocated block. Error reporting as required by the error recovery bits (EER, PER, DTE, and DCR) shall be performed only after completion of the reallocation. The reallocation operation shall report any failures that occur. See the REASSIGN BLOCKS command (see 5.20) for error procedures.

An automatic read reallocation enabled (ARRE) bit set to zero specifies that the device server shall not perform automatic reallocation of defective logical blocks during read operations.

An ARRE bit set to one specifies that the device server shall enable automatic reallocation of defective logical blocks during read operations. All error recovery actions required by the error recovery bits (TB, EER, PER, DTE, and DCR) shall be processed. The automatic reallocation shall then be performed only if the device server successfully recovers the data. The recovered data shall be placed in the reallocated block. Error reporting as required by the error recovery bits shall be performed only after completion of the reallocation. The reallocation process shall present any failures that occur. See the REASSIGN BLOCKS command (see 5.20) for error procedures.

A transfer block (TB) bit set to zero specifies that a logical block that is not recovered within the recovery limits specified shall not be transferred to the application client. A TB bit set to one specifies that a logical block that is not recovered within the recovery limits specified shall be transferred to the application client before CHECK CONDITION status is returned. The TB bit does not affect the action taken for recovered data.

NOTE 25 - Fabricated data may be data already in the buffer or any other vendor-specific data. This bit may be used in image processing, audio, or video applications.

A read continuous (RC) bit set to zero specifies that error recovery operations that cause delays are acceptable during the data transfer. Data shall not be fabricated.

An RC bit set to one specifies the device server shall transfer the entire requested length of data without adding delays to perform error recovery procedures. This implies that the device server may send data that is erroneous or fabricated in order to maintain a continuous flow of data. The device server shall assign priority to this bit over conflicting error control bits (EER, DCR, DTE, and PER) within this byte.

The individual bit definitions for EER, PER, DTE and DCR are contained in table 108.

Table 108 — Error recovery bit definitions

EER	PER	DTE	DCR	**Description**
1	-	-	-	An enable early recovery (EER) bit set to one specifies that the device server shall use the most expedient form of error recovery first. This bit only applies to data error recovery and it does not affect positioning retries and the message system error recovery procedures.
0	-	-	-	An EER bit set to zero specifies that the device server shall use as error recovery procedure that minimizes the risk of mis-detection or mis-correction.
-	1	-	-	A post error (PER) bit set to one specifies that the device server shall report recovered errors.
-	0	-	-	A PER bit set to zero specifies that the device server shall not report recovered errors. Error recovery procedures shall be performed within the limits established by the error recovery parameters.
-	-	1	-	A DTE bit set to one specifies that the device server shall terminate the data-in or data-out buffer transfer upon detection of a recovered error.
-	-	0	-	A DTE bit set to zero specifies that the device server shall not terminate the data-in or data-out buffer transfer upon detection of a recovered error.
-	-	-	1	A disable correction (DCR) bit set to one specifies that error condition codes shall not be used for data error recovery.
-	-	-	0	A DCR bit set to zero allows the use of error condition codes for data error recovery.

NOTE 26 - An EER bit set to one may imply an increase in the probability of mis-detection or mis-correction. An EER bit set to zero allows the specified retry limit to be exhausted prior to using error correction codes.

The combinations of these bits are explained in table 109.

Table 109 — Combined error recovery parameter descriptions (part 1 of 2)

EER	PER	DTE	DCR	Description
0	0	0	0	The full number of retries (specified in the READ, WRITE or VERIFY RETRY COUNT field) and error correction are attempted to recover the data (EER and DCR bits set to zero). A CHECK CONDITION is not reported at the completion of the command for recovered errors (PER bit set to zero). The command terminates with CHECK CONDITION status before the transfer count is exhausted only if an unrecoverable error is detected. If an unrecoverable data error occurred, the data in the block with the unrecoverable error may or may not be transferred to the application client depending on the setting of the transfer block (TB) bit (read operation only).
0	0	0	1	Error correction is disabled (DCR bit set to one) so only the full number of retries (specified in the READ, WRITE or VERIFY RETRY COUNT field) are attempted to recover the data (EER bit set to zero). A CHECK CONDITION is not reported at the completion of the command for recoverable errors (PER bit set to zero). The command terminates with CHECK CONDITION status before the transfer count is exhausted only if an unrecoverable error is detected. If an unrecoverable data error occurred, the data in the block with the unrecoverable error may or may not be transferred to the application client depending on the setting of the transfer block (TB) bit (read operation only).
0	0	1	0	Invalid mode (the PER bit shall be set to one if DTE is set to one).[a]
0	0	1	1	Invalid mode (the PER bit shall be set to one if DTE is set to one).[a]
0	1	0	0	The full number of retries (specified in the READ, WRITE or VERIFY RETRY COUNT field) and error correction are attempted to recover the data (EER and DCR bits set to zero). The command terminates with CHECK CONDITION status before the transfer count is exhausted only if an unrecoverable error is detected. If an unrecoverable data error occurred, the data in the block with the unrecoverable error may or may not be transferred to the application client depending on the setting of the transfer block (TB) bit (read operation only). A CHECK CONDITION with a sense key of RECOVERED ERROR is reported at the completion of the command for any recoverable error that occurs (PER bit set to one). The INFORMATION field in the sense data shall contain the LBA of the last recovered error that occurred during the transfer.
0	1	0	1	Error correction is disabled (DCR bit set to one) so only the full number of retries (specified in the READ, WRITE or VERIFY RETRY COUNT field) are attempted to recover the data (EER bit set to zero). The command terminates with CHECK CONDITION status before the transfer count is exhausted only if an unrecoverable error is detected. If an unrecoverable data error occurred, the data in the block with the unrecoverable error may or may not be transferred to the application client depending on the setting of the transfer block (TB) bit (read operation only). A CHECK CONDITION with a sense key of RECOVERED ERROR is reported at the completion of the command for any recoverable error that occurs (PER bit set to one). The INFORMATION field in the sense data shall contain the LBA of the last recovered error that occurred during the transfer.

Table 109 — Combined error recovery parameter descriptions (part 2 of 2)

EER	PER	DTE	DCR	Description
0	1	1	0	The full number of retries (specified in the READ, WRITE or VERIFY RETRY COUNT field) and error correction are attempted to recover the data (EER and DCR bits set to zero). The command terminates with CHECK CONDITION status before the transfer count is exhausted if any error (recoverable or unrecoverable) is detected (DTE bit set to one). The INFORMATION field in the sense data shall contain the LBA of the block in error. If an unrecoverable data error occurs the data in the block with the error may or may not be transferred to the application client depending on the setting of the transfer block (TB) bit (read operation only.)
0	1	1	1	Error correction is disabled (DCR bit set to one) so only the full number of retries (specified in the READ, WRITE or VERIFY RETRY COUNT field) are attempted to recover the data (EER bit set to zero). The command terminates with CHECK CONDITION status before the transfer count is exhausted if any error (recoverable or unrecoverable) is detected (DTE bit set to one). The INFORMATION field in the sense data shall contain the LBA of the block in error. If an unrecoverable data error occurs the data in the block with the error may or may not be transferred to the application client depending on the setting of the transfer block (TB) bit (read operation only).
1	0	0	0	The fewest possible retries and error correction are attempted to recover the data (EER bit set to one and DCR bit set to zero). A CHECK CONDITION is not reported at the completion of the command for recoverable errors (PER bit set to zero). The command terminates with CHECK CONDITION status before the transfer count is exhausted only if an unrecoverable error is detected. If an unrecoverable data error occurred, the data in the block with the unrecoverable error may or may not be transferred to the application client depending on the setting of the transfer block (TB) bit (read operation only).
1	0	0	1	Invalid mode (the DCR bit shall be set to zero if EER is set to one).[a]
1	1	0	0	The fewest possible retries and error correction are attempted to recover the data (EER bit set to one and DCR bit set to zero). The command terminates with CHECK CONDITION status before the transfer count is exhausted only if an unrecoverable error is detected. If an unrecoverable data error occurred, the data in the block with the unrecoverable error may or may not be transferred to the application client depending on the setting of the transfer block (TB) bit (read operation only). A CHECK CONDITION with a sense key of RECOVERED ERROR is reported at the completion of the command for any recoverable error that occurs (PER bit set to one). The INFORMATION field in the sense data shall contain the LBA of the last recovered error that occurred during the transfer.
1	1	0	1	Invalid mode (the DCR bit shall be set to zero if EER is set to one).[a]
1	1	1	0	The fewest possible retries and error correction are attempted to recover the data (EER bit set to one and DCR bit set to zero). The command terminates with CHECK CONDITION status before the transfer count is exhausted if any error (recoverable or unrecoverable) is detected (DTE bit set to one). The INFORMATION field in the sense data shall contain the LBA of the block in error. If an unrecoverable data error occurs the data in the block with the error may or may not be transferred to the application client depending on the setting of the transfer block (TB) bit (read operation only).
1	1	1	1	Invalid mode (the DCR bit shall be set to zero if EER is set to one).[a]

[a] If an invalid mode for the error recovery combination is sent by the application client the device server shall return CHECK CONDITION status and the sense key shall be set to ILLEGAL REQUEST with the additional sense code set to INVALID FIELD IN PARAMETER LIST.

The READ and WRITE RETRY COUNT fields specify the number of times that the device server shall attempt its recovery algorithm during read and write operations, respectively. If the RETRY COUNT field and the RECOVERY TIME LIMIT field are both specified in a MODE SELECT command, the field that requires the least time for data error recovery actions shall have priority.

The RECOVERY TIME LIMIT field specifies in milliseconds the maximum time duration that the device server shall use for data error recovery procedures. The device server may round this value as described in SPC-3. The limit in this field specifies the maximum error recovery time allowed for any individual logical block. A RECOVERY TIME LIMIT field set to zero specifies that the device server shall use its default value.

If both RETRY COUNT and RECOVERY TIME LIMIT are specified, the field that specifies the recovery action of least duration shall have priority.

6.3.5 Verify Error Recovery mode page

The Verify Error Recovery mode page (see table 110) specifies the error recovery parameters the device server shall use during the VERIFY command and the verify operation of the WRITE AND VERIFY command.

Table 110 — Verify Error Recovery mode page

Byte\Bit	7	6	5	4	3	2	1	0
0	PS	Reserved	PAGE CODE (07h)					
1	PAGE LENGTH (0Ah)							
2	Reserved				EER	PER	DTE	DCR
3	VERIFY RETRY COUNT							
4	Obsolete							
5	Reserved							
9								
10	(MSB) VERIFY RECOVERY TIME LIMIT							
11								(LSB)

The parameters savable (PS) bit is only used with the MODE SENSE command. This bit is reserved with the MODE SELECT command. A PS bit set to one indicates that the device server is capable of saving the mode page in a non-volatile vendor-specific location.

The AWRE bit as defined in the Read-Write Error Recovery mode page (see 6.3.4) applies to the WRITE AND VERIFY command. The VERIFY command shall not perform automatic reallocation.

The EER, PER, DTE, and DCR bits are defined in 6.3.4. The combinations of these bits are defined in table 109 (see 6.3.4).

The VERIFY RETRY COUNT field specifies the number of times that the device server shall attempt its recovery algorithm during a verify operation. If the verify retry count and the VERIFY RECOVERY TIME LIMIT are both specified, the one that requires the least time for data error recovery actions shall have priority.

The VERIFY RECOVERY TIME LIMIT field specifies in milliseconds the maximum time duration that the device server shall use error recovery procedures to recover data for an individual logical block. The device server may round this value as described in SPC-3. If the VERIFY RETRY COUNT and the VERIFY RECOVERY TIME LIMIT are both specified, the one that requires the least time for data error recovery actions shall have priority.

NOTE 27 - To disable all types of correction and retries the application client should set the EER bit to zero, the PER, DTE, and DCR bits to one and the number of retries and recovery time limit to zero.

6.3.6 XOR Control mode page

The XOR Control mode page (see table 111) provides the initiator with the means to obtain or modify certain XOR operating parameters of the target.

Table 111 — XOR Control mode page

Byte\Bit	7	6	5	4	3	2	1	0
0	PS	Reserved	PAGE CODE (10h)					
1	PAGE LENGTH (16h)							
2	Reserved						XORDIS	Reserved
3	Reserved							
4	(MSB)	MAXIMUM XOR WRITE SIZE						
7								(LSB)
8	Reserved							
11								
12	Obsolete							
19								
20	Reserved							
21								
22	Obsolete							
23								

The parameters savable (PS) bit is only used with the MODE SENSE command. This bit is reserved with the MODE SELECT command. A PS bit set to one indicates that the device server is capable of saving the mode page in a non-volatile vendor-specific location.

An XOR disable (XORDIS) bit set to zero enables the XOR operations within a device. An XORDIS bit set to one shall disable the XOR operations within a device. If the XORDIS bit is set to one and an XOR command is sent to the logical unit, the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST with the additional sense code set to INVALID COMMAND OPERATION CODE.

The MAXIMUM XOR WRITE SIZE field specifies the maximum transfer length in blocks that the target accepts for a single XDWRITE or XPWRITE command.

6.4 Vital product data (VPD) parameters

6.4.1 VPD parameters overview

This subclause defines the VPD pages used only with direct-access block type devices. See SPC-3 for VPD pages used with all device types.

The VPD page codes specific to direct-access devices are defined in table 112.

Table 112 — Direct-access device VPD page codes

VPD page code	VPD page name	Reference	Support requirements
B0h	Block Limits VPD page	6.4.2	Optional
B1h - BFh	Reserved for this device type		

6.4.2 Block Limits VPD page

The Block Limits VPD page (see table 113) provides the application client with the means to obtain certain operating parameters of the logical unit.

Table 113 — Block Limits VPD page

Byte\Bit	7	6	5	4	3	2	1	0
0	PERIPHERAL QUALIFIER			PERIPHERAL DEVICE TYPE				
1	PAGE CODE (B0h)							
2	Reserved							
3	PAGE LENGTH (0Ch)							
4	Reserved							
5								
6	(MSB)	OPTIMAL TRANSFER LENGTH GRANULARITY						
7								(LSB)
8	(MSB)	MAXIMUM TRANSFER LENGTH						
11								(LSB)
12	(MSB)	OPTIMAL TRANSFER LENGTH						
15								(LSB)

The PERIPHERAL QUALIFIER field and the PERIPHERAL DEVICE TYPE field are defined in SPC-3.

The PAGE CODE field shall be set to B0h.

The PAGE LENGTH field is defined in SPC-3.

The OPTIMAL TRANSFER LENGTH GRANULARITY field indicates the optimal transfer length granularity in blocks for a single PRE-FETCH, READ, VERIFY, WRITE, WRITE AND VERIFY, XDREAD, XDWRITE, XDWRITEREAD, or XPWRITE command. Transfers with transfer lengths not equal to a multiple of this value may incur significant delays in processing.

The MAXIMUM TRANSFER LENGTH field indicates the maximum transfer length in blocks that the target accepts for a single PRE-FETCH, READ, VERIFY, WRITE, WRITE AND VERIFY, XDREAD, XDWRITE, XDWRITEREAD, or XPWRITE command. Requests for transfer lengths exceeding this limit result in CHECK CONDITION status with a sense key of ILLEGAL REQUEST and an additional sense code of INVALID FIELD IN CDB. A MAXIMUM TRANSFER LENGTH field set to zero indicates that there is no reported limit on the transfer length.

The OPTIMAL TRANSFER LENGTH field indicates the optimal transfer length in blocks for a single PRE-FETCH, READ, VERIFY, WRITE, WRITE AND VERIFY, XDREAD, XDWRITE, XDWRITEREAD, or XPWRITE command. Transfers with transfer lengths exceeding this value may incur significant delays in processing.

Annex A
(informative)

XOR command examples

A.1 XOR command examples overview

This annex provides XOR command examples in storage array controller supervised configurations.

A.2 Update write operation

Figure A.1 illustrates a read-modify-write operation supervised by a storage array controller. The example uses a supervising storage array controller, a data disk device (holding protected user data), and a parity disk device (holding check data). In this example, the data and parity devices are on separate SCSI physical interconnects, and thus are not capable of peer-to-peer interaction. Three SCSI commands are used: XDWRITE, XDREAD, and XPWRITE. XDWRITEREAD may be used in place of any sequence of XDWRITE followed by XDREAD.

The supervising storage array controller begins by sending user data to the data disk device using an XDWRITE command. It also initiates an XPWRITE command to the parity disk device (the supervising storage array controller does not yet have the intermediate XOR data for this command; the purpose of issuing the XPWRITE command at this time is to cause the parity disk device to begin reading XOR data from its medium to its buffer memory).

The data disk device reads old user data from its medium, performs an XOR operation using the old user data and the user data from the supervising storage array controller, stores the resulting intermediate XOR data in its buffer memory, and writes the user data from the supervising storage array controller to its medium. The supervising storage array controller reads the resulting intermediate XOR data from the buffer memory by sending the data disk device an XDREAD command.

The supervising storage array controller makes the resulting intermediate XOR data (read with the XDREAD command) available to the parity disk device for the already issued XPWRITE command. The parity disk

device performs an XOR operation using the intermediate XOR data and the XOR data in its buffer memory. The resulting new XOR data is written to the medium.

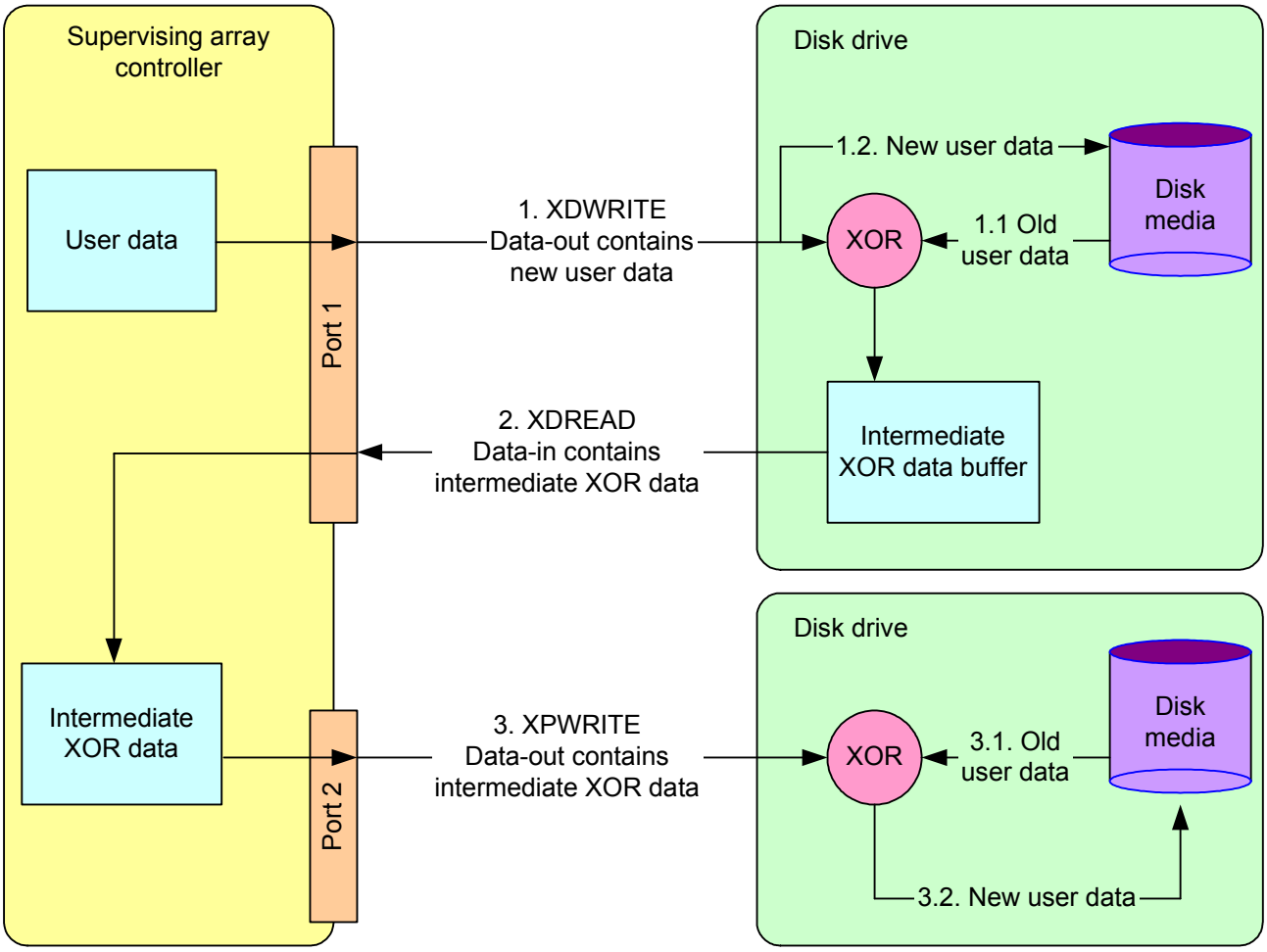


Figure A.1 — Update write operation (storage array controller supervised)

A.3 Regenerate operation

Figure A.2 illustrates a regenerate operation supervised by a storage array controller. The example uses a supervising storage array controller and three disk devices. In this example, all three disk devices are on separate SCSI physical interconnects, and thus are not capable of peer-to-peer interaction. Three SCSI commands are used: READ, XDWRITE, and XDREAD. XDWRITEREAD may be used in place of any sequence of XDWRITE followed by XDREAD.

The supervising storage array controller begins by issuing a READ command to disk device 1. The data received from this command is sent by the supervising storage array controller to disk device 2 using an XDWRITE command with a DISABLE WRITE bit set to one. Disk device 2 reads data from its medium, performs an XOR operation using that data and the data received from the supervising storage array controller, and stores the resulting intermediate XOR data in its buffer memory. The supervising storage array controller retrieves the intermediate XOR data from the buffer memory by issuing an XDREAD command to disk device 2. The supervising storage array controller issues XDWRITE and XDREAD commands in the same manner to disk device 3.

The resulting data from disk device 3 is the regenerated user data.

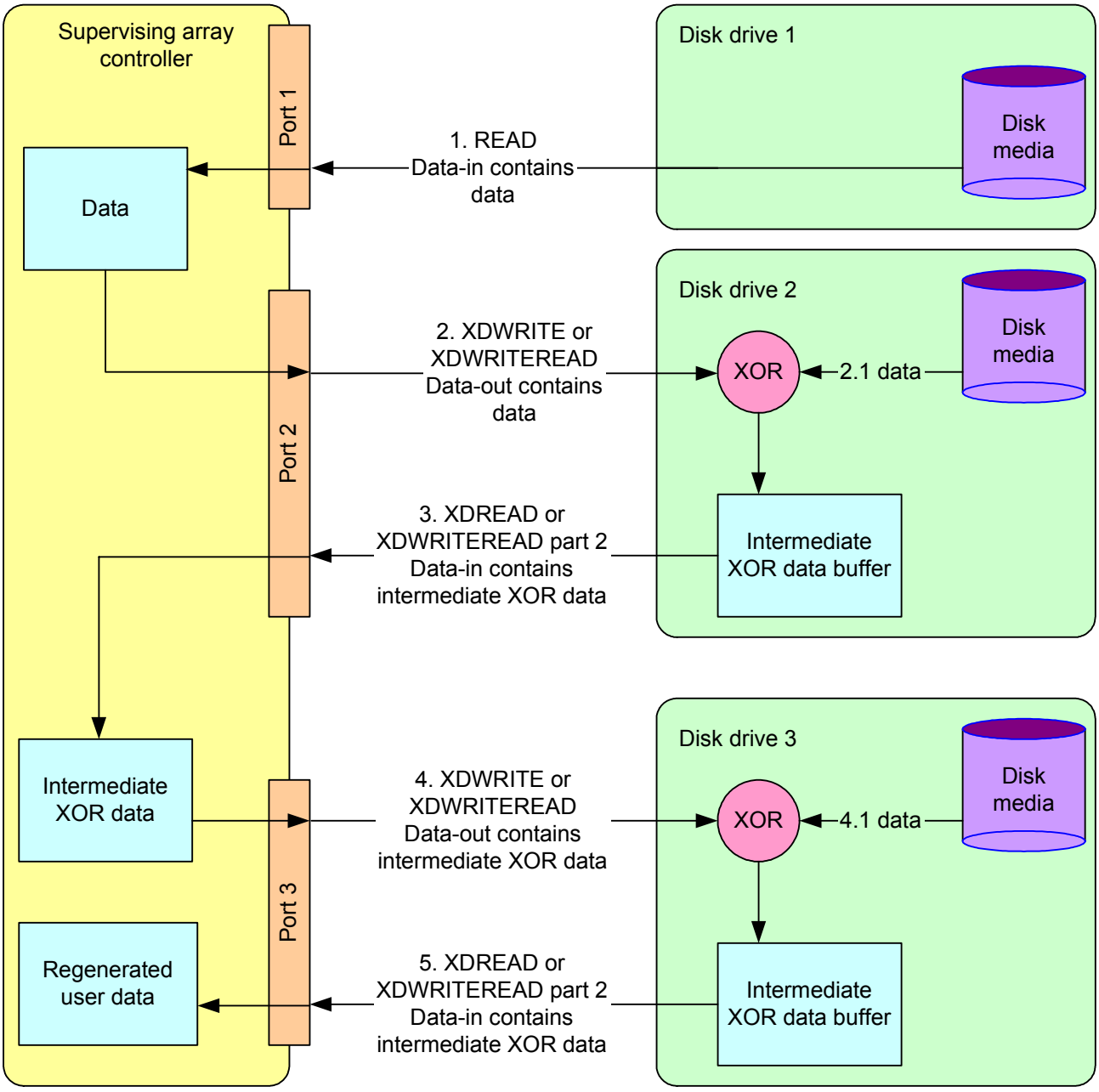


Figure A.2 — Regenerate operation (storage array controller supervised)

A.4 Rebuild operation

Figure A.3 illustrates a rebuild operation supervised by a storage array controller. The example uses a supervising storage array controller, two disk devices as the source devices, and one disk device as the rebuild device. In this example, all three disk devices are on separate SCSI physical interconnects, and thus are not capable of peer-to-peer interaction. Four SCSI commands are used: READ, XDWRITE, XDREAD, and WRITE. XDWRITEREAD may be used in place of any sequence of XDWRITE followed by XDREAD.

The supervising storage array controller begins by issuing a READ command to disk device 1. The data received from the READ command is sent by the supervising storage array controller to disk device 2 using an XDWRITE command with a DISABLE WRITE bit set to one. Disk device 2 reads data from its medium, performs an XOR operation using that data and the data received from the supervising storage array controller, and

stores the resulting intermediate XOR data in its buffer memory. The supervising storage array controller retrieves the intermediate XOR data by sending an XDREAD command to disk device 2.

The resulting data from disk device 2 is the "rebuilt" data and is sent to the device being rebuilt (disk device 3) using a WRITE command.

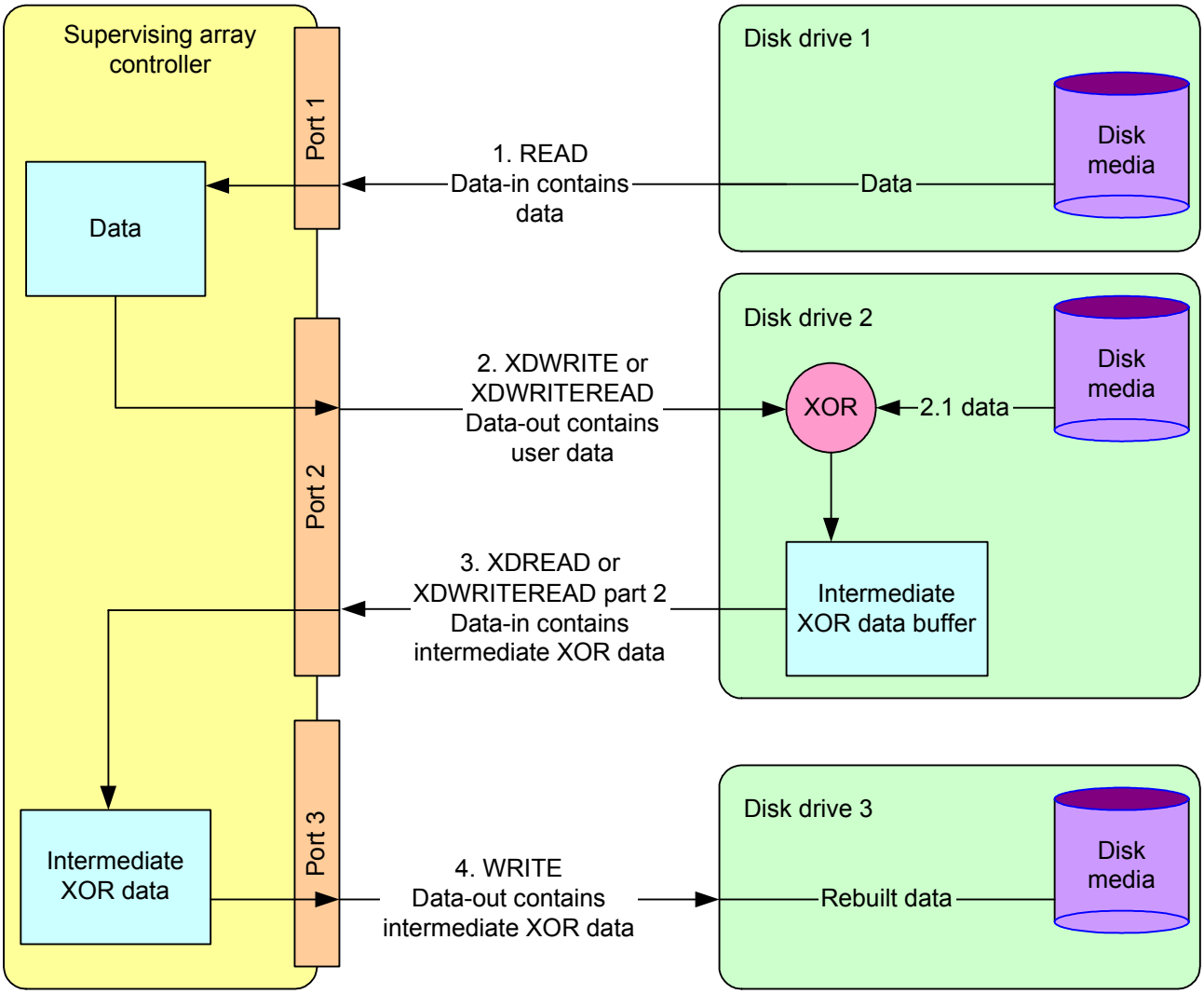


Figure A.3 — Rebuild operation (storage array controller supervised)

Annex B
(informative)

CRC example in C

The following is an example C program that generates the value for the LOGICAL BLOCK GUARD field in protection information (see 4.15).

```
// dpcrc.cpp : SCSI SBC-2 Data Protection CRC generator
#include "stdafx.h"
#include <stdio.h>
#include <malloc.h>

/* return crc value */
unsigned short calculate_crc(unsigned char *frame, unsigned long length) {
unsigned short const poly = 0x8BB7L;  /* Polynomial */
unsigned const int poly_length = 16;
unsigned short crc_gen;
unsigned short x;
unsigned int i, j, fb;
unsigned const int invert = 0;/* 1=seed with 1s and invert the CRC */

crc_gen = 0x0000;
crc_gen ^= invert? 0xFFFF: 0x0000;   /* seed generator */

for (i = 0; i < length; i += 2) {
      /* assume little endian */
      x = (frame[i] << 8) | frame[i+1];

      /* serial shift register implementation */
        for (j = 0; j < poly_length; j++) {
            fb = ((x & 0x8000L) == 0x8000L) ^ ((crc_gen & 0x8000L) ==
0x8000L);
            x <<= 1;
            crc_gen <<= 1;
            if (fb)
                  crc_gen ^= poly;
      }
}

return crc_gen ^ (invert? 0xFFFF: 0x0000); /* invert output */
} /* calculate_crc */

/* function prototype */
unsigned short calculate_crc(unsigned char *, unsigned long);

void main (void) {
unsigned char *buffer;
unsigned long buffer_size = 32;
unsigned short crc;
unsigned int i;

/* 32 0x00 */
buffer = (unsigned char *) malloc (buffer_size);
for (i = 0; i < buffer_size; i++) {
      buffer[i] = 0x00;
```

```
}
crc = calculate_crc(buffer, buffer_size);
printf ("Example CRC all-zeros is %04x\n", crc);
free (buffer);

/* 32 0xFF */
buffer = (unsigned char *) malloc (buffer_size);
for (i = 0; i < buffer_size; i++) {
      buffer[i] = 0xFF;
}
crc = calculate_crc(buffer, buffer_size);
printf ("Example CRC all-ones is %04x\n", crc);
free (buffer);

/* 0x00 incrementing to 0x1F */
buffer = (unsigned char *) malloc (buffer_size);
for (i = 0; i < buffer_size; i++) {
      buffer[i] = i;
}
crc = calculate_crc(buffer, buffer_size);
printf ("Example CRC incrementing is %04x\n", crc);
free (buffer);

/* 0xFF 0xFF then 30 zeros */
buffer = (unsigned char *) malloc (buffer_size);
buffer[0] = 0xff;
buffer[1] = 0xff;
for (i = 2; i < buffer_size; i++) {
      buffer[i] = 0x00;
}
crc = calculate_crc(buffer, buffer_size);
printf ("Example CRC FF FF then 30 zeros is %04x\n", crc);
free (buffer);

/* 0xFF decrementing to 0xE0 */
buffer = (unsigned char *) malloc (buffer_size);
for (i = 0; i < buffer_size; i++) {
      buffer[i] = 0xff - i;
}
crc = calculate_crc(buffer, buffer_size);
printf ("Example CRC FF decrementing to E0 is %04x\n", crc);
free (buffer);

} /* main */
```