Working Draft American National Standard

Project T10/BSR INCITS 571

Revision 6

15 November 2023

Information technology - SCSI Block Commands – 5 (SBC-5)

T10 Technical Editor:

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

Telephone: (408) 449-1839
Email: bill.martin@samsung.com

Reference number
ISO/IEC 14776-324:201x
BSR INCITS 571:202x

Points of contact

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

T10 Chair
William Martin
Samsung Semiconductor, Inc
7213 Marblethorpe Drive
Roseville, CA 95747
USA

Telephone: (408) 499-1839
Email: bill.martin@samsung.com

T10 Vice-Chair
Curtis Ballard
Hewlett Packard Enterprise
3404 East Harmony Road
Fort Collins CO 80528
USA

Telephone: (970) 898-3013
Email: curtis.ballard@hpe.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
1101 K Street, NW
Suite 610
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

Purchase INCITS Standards
Web site: http://www.incits.org/standards-information/purchase-standards-or-download-dpans

American National Standard
for Information Technology

SCSI Block Commands – 5 (SBC-5)

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 - 5 (SBC-5) command set. SBC-5 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 -4 (SBC-4) command set, INCITS 506-2021, 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 review by ANSI that the requirements for due process, consensus, and other criteria for approval have been met by the standards developer. Consensus is established when, in the judgement of the ANSI Board of Standards Review, substantial agreement has been reached by directly and materially affected interests. Substantial agreement means much more than a simple majority, but not necessarily unanimity. Consensus requires that all views and objections be considered, and that a concerted effort be made towards their resolution.

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

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

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

CAUTION: The developers of this standard have requested that holders of patents that may be required for the implementation of the standard, disclose such patents to the publisher. However, neither the developers nor the publisher have undertaken a patent search in order to identify which, if any, patents may apply to this standard. As of the date of publication of this standard, following calls for the identification of patents that may be required for the implementation of the standard, 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, Inc.
25 West 43rd Street, 4th floor, New York, New York 10036-7422

Revision History

This revision history is not part of American National Standard INCITS 571:202x.

R.1 Revision 0 (14 April 2021)

Revision 0 of SBC-5 is substantially equal to revision 22 of SBC-4. The only differences arise from changes made in SBC-4 discovered during the ISO process.

R.2 Revision 1

Incorporates the following proposals:

a) SBC-5, ZBC-2: Obsolete the ZONED field (20-054r1);
b) SBC-5: Define zone domains and realms schema type (20-030r0);
c) SBC-4 Add multi-actuator bit to Block Device VPD page (20-049r1);
d) SBC-4 Add depopulation revocation codes for GET PHYSICAL HEALTH STATUS command (20-052r1);
e) SBC-5: Informative Annex for W-LUNs Dual-Actuator Support (19-129r3);
f) ZBC-2 SBC-5 SPC-6: REMOVE ELEMENT AND MODIFY ZONES (20-099r2);
g) SBC-5 Create Concurrent Positioning Ranges VPD page (20-089r2); and
h) SPC-6 SBC-5 SAT-5 Translate the Command Duration Limits T2 mode pages (20-085r5).

R.3 Revision 2 (8 March 2022)

a) SBC-5 Verify Recovery Mode Page Clarification (21-0678r0)
b) ZBC-2 SBC-5: Adding Zone Capacities that are different from Zone Lengths (21-081r4)
   A) This proposal was overtaken by the subsequently approved proposal 21-118r0
c) SBC-5 SPC-6 SPL-5: START STOP UNIT Halting Power Condition Timers (21-106r3)
d) ZBC-2 SBC-5 Eject Mode Field From VPD Page (21-118r0)
e) Moved Annex H SBC feature sets, to the beginning of the Annexes as it is a normative annex

R.4 Revision 3 (28 July 2022)

a) SBC-5 ZBC-3 Restore Elements and Rebuild clarifications (22-036r2)
b) Minor editorial changes

R.5 Revision 4 (26 January 2023)

a) SBC-5 Enhanced INQUIRY data changes when capacity changes (22-057r6)
b) SBC-5: Write and Verify command operation ambiguity (22-096r2)
c) SBC-5, ZBC-3: Limit Excessive Depopulating of Physical Elements (22-107r2)
d) SPC-6 SBC-5 SAT-5 Obsolete the 6 byte MODE SELECT and MODE SENSE commands (22-063r5)
e) SBC-5 Add new status to GET LBA STATUS commands (22-108r1)

R.6 Revision 5 (4 October 2023)

a) SBC-5: Does Sanitize EXIT FAILURE MODE need IMMED=1? (22-097R5)
b) ZBC-3, SBC-5 Obsolete host aware (23-008R1)
c) SBC-5 Constrained Streams with Data Lifetimes (23-024R3)
d) SBC-5: Focus on Standardized Grouping Function (23-027R2)
e) Minor editorial fixes

R.7 Revision 6 (16 November 2023)

a) SBC-5, SPC-6: Interrupted Depopulation and Interrupted Restoration (23-071r0)
b) SBC5: CAPPID Modifications (23-073r1)
c) SBC-5 Nitpicking the Grouping Functions model (23-076r1)
d) SBC-5 Nitpicking in the Stream Control model (23-078r1)

Contents

Page

Tables

Figures

Foreword

This foreword is not part of American National Standard BSR INCITS 506.

The purpose of this standard is to define the model and command set extensions to be used in conjunction with the SCSI Primary Command Set standard – 6 (SPC-6) to facilitate operation of SCSI direct access block devices (e.g., hard disk drives).

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 Council, Suite 610 K Street, NW, Washington, DC 20005.

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

Karen Higginbottom, Chair

David Michael, Vice-Chair

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

William Martin, Chair

Curtis Ballard, Vice-Chair

Curtis Stevens, Secretary

Organization Represented *Name of Representative*

....

SCSI standards family

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

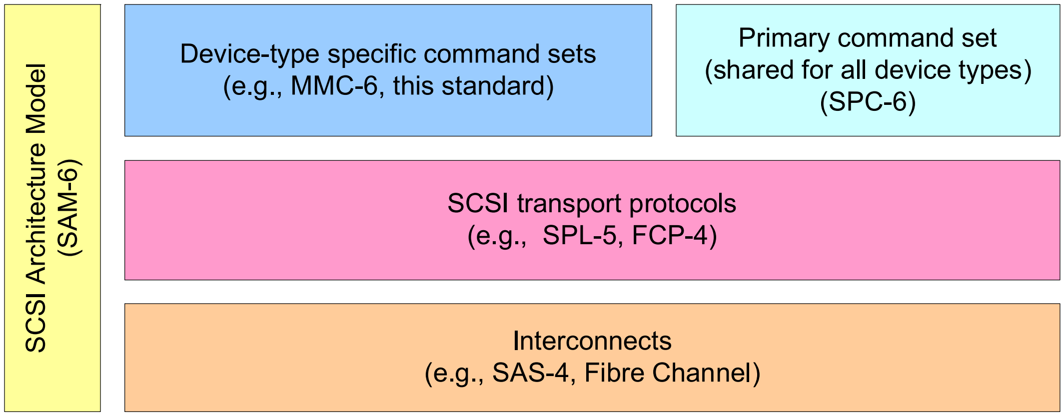


Figure 0 — SCSI document relationships

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

The functional areas identified in figure 0 characterize the scope of standards within a group as follows:

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

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

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

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

Interconnects: Implementation standards that define the communications mechanism employed by the SCSI transport protocols. These standards may describe the electrical and signaling requirements essential for SCSI devices to interoperate over a given interconnect. Interconnect standards may allow the interconnection of devices other than SCSI devices in ways that are not defined by this standard.

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

1 Scope

This standard defines the command set extensions to facilitate operation of SCSI direct access block devices. The clauses in this standard, implemented in conjunction with the applicable clauses of SPC-6, specify the standard command set for SCSI direct access block devices.

The objectives of this standard are to:

a) permit an application client to communicate over a SCSI service delivery subsystem (see SAM-6) with a logical unit that declares itself to be a direct access block device in the PERIPHERAL DEVICE TYPE field of the standard INQUIRY data (see SPC-6); and
b) define commands and parameters unique to the direct access block device type.

This standard makes obsolete the following concepts from SBC-3:

a) the EER bit and DCR bit (see 6.5.10);
b) the TMC field and ETC bit in various log parameters;
c) the READ LONG command and WRITE LONG command except for write uncorrectable; and
d) the XOR command.

2 Normative references

The following documents, in whole or in part, are normatively referenced in this document and are indispensable for its application. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO/IEC 14776-342, Information technology – Small Computer System Interface (SCSI) – Part 342: *SCSI-3 Controller Commands - 2 (SCC-2)*

INCITS 468-2010, *Multi-Media Commands - 6* (MMC-6)

INCITS 468-2010/AM 1 *MultiMedia Command Set - 6 - Amendment 1* (MMC-6/AM 1)

INCITS 491-2018, *SCSI / ATA Translation - 4* (SAT-4)

INCITS 497-2012, *Automation/Drive Interface Commands* - 3 (ADC-3)

INCITS 518, *SCSI Enclosure Services - 3 (SES-3)* (planned as ISO/IEC 14776-373)

INCITS 536, *Zoned Block Commands* (ZBC) (planned as ISO/IEC 14776-345)

INCITS 546, *SCSI Architecture Model - 6* (SAM-6) (under consideration)

INCITS 550, *Zoned Block Commands* - 2 (ZBC-2) (under consideration)

INCITS 554, SAS Protocol Layer - 5 (SPL-5) (under consideration)

INCITS 566, *SCSI Primary Commands - 6* (SPC-6) (under consideration)

3 Definitions, symbols, abbreviations, keywords, and conventions Introduction

Table 1 shows the topics in clause 3 and a reference to the subclause where each topic is described.

Table 1 — Direct access block device type mode topics and references

Topic	Reference
Terms and Definitions	3.1
Symbols	3.2
Abbreviations	3.3
Keywords	3.4
Editorial conventions	3.5
Numeric conventions	3.6
State machine conventions	3.7

3.1 Terms and Definitions

3.1.1 additional sense code

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

Note 1 to entry: See SPC-6.

3.1.2 advanced background operation

background operation that may impact device server response time to affected LBAs and may include garbage collection operations

3.1.3 AND

Boolean arithmetic function (see 3.1.13) on two binary input values that results in an output value of one if both of the input values are one or zero if either of the input values is zero

3.1.4 AND operation

performance of an AND (see 3.1.3) bitwise on two multiple-bit input values both having the same number of bits

Note 1 to entry: An example of multiple-bit inputs is the current content of a logical block and the content contained in the Data-Out Buffer having the same number of bytes.

3.1.5 anchored

logical block provisioning state of an LBA (see 4.7.1) in which physical capacity has been reserved for the referenced logical block (see 4.7.4.7)

3.1.6 application client

object that is the source of SCSI commands

Note 1 to entry: See SAM-6.

3.1.7 automatic read reassignment

sequence after the device server detects a recovered read error during which the device server, without intervention from an application client, performs a reassign operation on the LBA for which the error was detected

3.1.8 atomic write command
command that performs one or more atomic write operations

Note 1 to entry: See 4.29.

3.1.9 atomic write operation
process by which a device server performs a write operation that is either completed in its entirety or has no effects on stored logical block data

Note 1 to entry: See 4.29.

3.1.10 automatic write reassignment
sequence after the device server detects a recovered error or an unrecovered error during which the device server, without intervention from an application client, performs a reassign operation on the LBA for which the error was detected

3.1.11 background function
either a background scan operation (see 4.23) or a device specific background function (see 3.1.28)

3.1.12 bitmap buffer
temporary buffer within a device server (e.g., for one or more bytes of the result of an AND operation (see 3.1.4) or an OR operation (see 3.1.60))

3.1.13 Boolean arithmetic function
function that produces an output from one or more inputs according to the rules of AND (see 3.1.3), XOR (see 3.1.32), and OR (see 3.1.59)

3.1.14 byte
sequence of eight contiguous bits considered as a unit

3.1.15 cache
temporary data storage area that is capable of containing a subset of the logical block data stored by the logical unit and is either volatile or non-volatile

3.1.16 check data
information contained within a redundancy group (see 3.1.82) that may allow lost or destroyed XOR-protected data (see 3.1.120) to be recreated

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

Note 1 to entry: See SAM-6.

3.1.18 command descriptor block (CDB)
structure used to communicate commands from an application client to a device server

Note 1 to entry: See SPC-6.

3.1.19 compare operation
process by which a device server compares two sets of data for equality

3.1.20 cyclic redundancy check (CRC)
error checking mechanism that checks data integrity by computing a polynomial algorithm based checksum (see 4.21.4)

3.1.21 Data-In Buffer
buffer specified by the application client to receive data from the device server during the processing of a command

Note 1 to entry: See SAM-6 and SPC-6.

3.1.22 Data-Out Buffer
buffer specified by the application client to supply data that is sent from the application client to the device server during the processing of a command

Note 1 to entry: See SAM-6 and SPC-6.

3.1.23 deallocated
logical block provisioning state of an LBA (see 4.7.1) in which physical capacity has not been reserved for the referenced logical block (see 4.7.4.6)

3.1.24 defect list
GLIST (see 4.13) or PLIST (see 4.13)

3.1.25 depopulate operation
process by which a SCSI device removes the capability of a storage element to store logical block data

Note 1 to entry: See 4.36.4.2.

3.1.26 depopulated
the condition of a physical element that has been affected by a depopulate operation

3.1.27 device server
object within a logical unit (see 3.1.50) that processes SCSI commands according to the rules of command management

Note 1 to entry: See SAM-6.

3.1.28 device specific background functions
SCSI target device specific functions that a SCSI target device may perform that have no specific association with application client-initiated operations

Note 1 to entry: See SPC-6.

3.1.29 device type
device model implemented by the logical unit and indicated to the application client by the PERIPHERAL DEVICE TYPE field of the standard INQUIRY data (see SPC-6)

3.1.30 direct access block device
device that is capable of containing data stored in logical blocks that each have a unique LBA (see 4.2)

3.1.31 error correcting code (ECC)
error checking mechanism that checks data integrity and enables some errors in the logical block data to be corrected

3.1.32 exclusive-or (XOR)
boolean arithmetic function on two binary input values that results in an output value of one if one and only one of the input values is one, or zero if both of the input values are either zero or one

3.1.33 extent
set of logical blocks occupying contiguous LBAs on a logical unit

3.1.34 field
group of one or more contiguous bits, a part of a larger structure (e.g., a CDB (see 3.1.18) or sense data (see SPC-6))

3.1.35 format corrupt
vendor specific condition in which the device server may not be able to perform logical block access commands

Note 1 to entry: See 4.10, 4.33, and 5.26.

3.1.36 format operation
process by which a device server initializes the medium in a logical unit

Note 1 to entry: See 4.10 and 4.33.

3.1.37 fully provisioned logical unit
logical unit that stores logical block data for every LBA and has assigned physical capacity for every LBA

Note 1 to entry: See 4.7.2.

3.1.38 garbage collection operation
process that prepares resources for future allocation to LBAs

3.1.39 grown defect list (GLIST)
list of physical blocks that the device server has detected as containing medium defects or that the application client has specified as containing medium defects

Note 1 to entry: See 4.13.

3.1.40 hard reset
condition resulting from the events defined by SAM-6 during which the SCSI device performs the hard reset operations described in SAM-6, this standard, and other applicable command standards (see table 34)

3.1.41 I_T nexus
relationship between a SCSI initiator port and a SCSI target port

Note 1 to entry: See SAM-6.

3.1.42 I_T nexus loss
condition resulting from the events defined by SAM-6 during which the SCSI device performs the I_T nexus loss operations described in SAM-6, this standard, and other applicable command standards (see table 34)

3.1.43 LBA resource
resource used for storing logical block data

3.1.44 LBA mapping resource
resource used by a logical unit that supports logical block provisioning management

Note 1 to entry: An example of a mapping resource is a a physical block or a data structure associated with tracking resource usage.

3.1.45 logical block
set of data bytes accessed and referenced as a unit

Note 1 to entry: See 4.5.

3.1.46 logical block access command
command that requests access to one or more logical blocks that may require access to the medium

Note 1 to entry: See 4.2.2.

3.1.47 logical block address (LBA)
value used to reference a logical block (see 4.5)

3.1.48 logical block data
user data and protection information, if any

3.1.49 logical block length
number of bytes of user data in a logical block (see 4.5)

3.1.50 logical unit
externally addressable entity within a SCSI target device (see 3.1.86) that implements a SCSI device model

Note 1 to entry: See SAM-6.

3.1.51 logical unit reset
condition resulting from the events defined by SAM-6 in which the logical unit performs the logical unit reset operations described in SAM-6, this standard, and other applicable command standards (see table 34)

3.1.52 mapped
logical block provisioning state of an LBA (see 4.7.1) in which physical capacity has been assigned to the referenced logical block (see 4.7.4.5)

3.1.53 medium
material that is not cache on which data is stored

Note 1 to entry: The plural of medium is media.

Note 2 to entry: An example of a medium in which data is stored is a magnetic disk.

3.1.54 medium defect
area of the medium that results in a recovered error or an unrecovered error when a read medium operation or a write medium operation is performed

Note 1 to entry: See 4.13.

3.1.55 misaligned write command
write command with fields set as described in 4.6.2

3.1.56 non-advanced background operation
background operation that does not impact device server response time to affected LBAs and may include garbage collection operations

3.1.57 non-volatile cache
cache that retains logical block data through any power cycle

3.1.58 non-volatile medium
medium that retains logical block data through any power cycle

3.1.59 OR
Boolean arithmetic function (see 3.1.13) on two binary input values that results in an output value of one if either of the input values are one or zero if both of the input values are zero

3.1.60 OR operation
performance of an OR (see 3.1.59) bitwise on two multiple-bit input values both having the same number of bits

Note 1 to entry: An example of multiple-bit input is the current content of a logical block and the content contained in the Data-Out Buffer having the same number of bytes.

3.1.61 point in time ROD token
ROD token with a ROD type that is a point in time copy ROD

Note 1 to entry: See SPC-6.

3.1.62 physical block
set of data bytes accessed as a unit by the device server (see 4.6)

3.1.63 physical block length
number of bytes of logical block data in a physical block (see 4.6)

3.1.64 physical element
subcomponent of a SCSI device

3.1.65 power cycle
sequence of power being removed followed by power being applied to a SCSI device

3.1.66 power on
condition resulting from the events defined by SAM-6 during which a SCSI device performs the power on operations described in SAM-6, this standard, and other applicable command standards (see table 34)

3.1.67 primary defect list (PLIST)
list of physical blocks containing medium defects that are considered permanent

Note 1 to entry: See 4.13.

3.1.68 protection information
group of fields at the end of each logical block or at specified intervals within each logical block that contain a logical block guard, an application tag, and a reference tag

Note 1 to entry: See 4.21.

3.1.69 protection information interval
length of user data that occurs within a logical block before each protection information

3.1.70 provisioning initialization pattern
non-zero pattern that is the length of one logical block

3.1.71 pseudo read data
indeterminate logical block data

3.1.72 pseudo unrecovered error
simulated error (e.g., created by a WRITE LONG command (see 5.50 and 5.51)) for which a device server reports that it is unable to read or write a logical block, regardless of whether the data on the medium is valid, recoverable, or unrecoverable

3.1.73 pseudo unrecovered error with correction disabled
pseudo unrecovered error for which a device server performs no error recovery

Note 1 to entry: See 4.18.2.

3.1.74 read cache operation
process by which a device server reads logical blocks for one or more LBAs from cache as described in 4.15

3.1.75 read command
command that requests read operations

Note 1 to entry: See 4.2.2.

3.1.76 read medium operation
process by which a device server reads logical blocks for one or more LBAs from the medium using the parameters specified in the Read-Write Error Recovery mode page (see 6.5.10)

3.1.77 read operation
process by which a device server performs operations as described in this standard

Note 1 to entry: Examples of read operations are read cache operations and read medium operations.

Note 2 to entry: See 4.2.3.

3.1.78 reassign
perform a reassign operation

3.1.79 reassign operation
operation during which the device server changes the assignment of an LBA from a specified physical block to another physical block and adds the specified physical block to the GLIST

3.1.80 recovered error
error for which a device server is able to read or write a logical block within the recovery limits specified in the Read-Write Error Recovery mode page (see 6.5.10) or the Verify Error Recovery mode page (see 6.5.11)

3.1.81 recovered read error
recovered error that occurs during a read medium operation

3.1.82 redundancy group
grouping of XOR-protected data (see 3.1.120) and associated check data (see 3.1.16) into a single type of data redundancy (see SCC-2)

3.1.83 resource provisioned logical unit
logical unit that may or may not store logical block data for every LBA and that provides enough LBA mapping resources (see 3.1.44) to map every LBA

Note 1 to entry: See 4.7.3.2.

3.1.84 sanitize operation
process by which a device server alters information on a logical unit such that recovery of previous logical block data from the cache and the medium is not possible

Note 1 to entry: See 4.11.

3.1.85 scattered write command
command that requests write operations to LBA ranges that may not be contiguous

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

Note 1 to entry: See SAM-6.

3.1.87 sense data
data describing an error, exceptional condition, or completion information

Note 1 to entry: See SPC-6.

3.1.88 sense key
contents of the SENSE KEY field in the sense data

Note 1 to entry: See SAM-6.

3.1.89 status
one byte of response information that contains a coded value defined in SAM-6, transferred from a device server to an application client upon completion of each command

Note 1 to entry: See SAM-6.

3.1.90 stopped power condition
power condition in which a device server terminates TEST UNIT READY commands and logical block access commands

Note 1 to entry: See 4.20.4.

3.1.91 storage element
physical element that provides non-volatile storage for an associated group of logical blocks

Note 1 to entry: See 4.36.

3.1.92 stream block
stream granularity size (see 6.6.5) area of physical media that is able to contain logical block data

3.1.93 stream identifier
identifier supplied by the device server and used by the application client to identify a stream

Note 1 to entry: See 4.32.

3.1.94 synchronize cache operation
process by which a device server synchronizes logical blocks within the volatile cache with the non-volatile cache or the medium

3.1.95 thin provisioned logical unit
logical unit that may or may not store logical block data for every LBA and that may or may not provide enough LBA mapping resources (see 3.1.44) to map every LBA

Note 1 to entry: See 4.7.3.3.

3.1.96 threshold set
set of two or more logical blocks used for tracking logical block provisioning thresholds

Note 1 to entry: See 4.7.3.7.

3.1.97 threshold set size
number of LBAs in a threshold set

Note 1 to entry: See 4.7.3.7.

3.1.98 token
representation of a collection of data

Note 1 to entry: See SPC-6.

3.1.99 truncate operation
process by which a device server reduces the logical unit’s capacity

Note 1 to entry: See 4.36.4.3.

3.1.100 unit attention condition
asynchronous status information that a logical unit establishes to report to the initiator ports associated with one or more I_T nexuses

Note 1 to entry: See SAM-6.

3.1.101 unmap command
command that requests an unmap operation

Note 1 to entry: See 4.2.2.

3.1.102 unmap operation
process by which a device server either deallocates or anchors a single LBA

Note 1 to entry: See 4.2.3 and 4.7.3.4.

3.1.103 unmapped
logical block provisioning state of an LBA (see 4.7.1) in which the LBA is either anchored or deallocated

3.1.104 unrecovered error
error for which a device server is unable to read a logical block or write a logical block within the recovery limits specified in the Read-Write Error Recovery mode page (see 6.5.10) and/or the Verify Error Recovery mode page (see 6.5.11)

3.1.105 unrecovered read error
unrecovered error that occurs during a read medium operation

3.1.106 unrecovered write error
unrecovered error that occurs during a write medium operation

3.1.107 user data
data contained in logical blocks that is accessible by an application client and is neither protection information nor other information that may not be accessible to the application client

3.1.108 user data segment
contiguous sequence of logical blocks

Note 1 to entry: See 4.26.

3.1.109 verify command
command that requests verify operations

Note 1 to entry: See 4.2.2.

3.1.110 verify medium operation
process by which a device server reads logical blocks for one or more LBAs from the medium using the parameters specified in the Verify Error Recovery mode page (see 6.5.11)

3.1.111 verify operation
process by which the device server performs operations as described in this standard

Note 1 to entry: An example of a verify operation is a verify medium operation.

Note 2 to entry: See 4.2.3.

3.1.112 volatile cache
cache that does not retain logical block data between power cycles

3.1.113 volatile medium
medium that does not retain logical block data between power cycles

Note 1 to entry: An example of volatile medium is a silicon memory device that loses data written to it if device power is lost.

3.1.114 write cache operation
process by which a device server writes logical blocks for one or more LBAs to the cache (see 4.7.1)

3.1.115 write command
command that requests write operations

Note 1 to entry: See 4.2.2.

3.1.116 write medium operation
process by which a device server writes logical blocks for one or more LBAs to the medium using the parameters specified in the Read-Write Error Recovery mode page (see 6.5.10)

3.1.117 write operation
process by which a device server performs operations as described in this standard

Note 1 to entry: See 4.2.3.

3.1.118 write stream command
write command that allows the application client to specify a stream identifier (see 4.32) in the CDB
Note 1 to entry: See 4.32.2.

3.1.119XOR operation processing of an XOR bitwise on two identical-sized multiple-bit input values

Note 1 to entry: An example of multiple-bit inputs is the current value of a logical block and the new value for that logical block.

3.1.120 XOR-protected data
logical blocks (i.e., including logical block data) that are part of a redundancy group

3.1.121 zoned block device
block device based on this standard that implements one or more of the models defined in ZBC-2

3.2 Symbols

Symbols used in this standard include:

Symbol	Meaning
+	plus
–	minus
×	multiplied by
÷	divided by
=	equals
≠	not equal
<	less than
>	greater than

3.3 Abbreviations

Abbreviations used in this standard include:

Abbreviation	Meaning
CDB	command descriptor block (see 3.1.18)
CRC	cyclic redundancy check (see 3.1.20)
ECC	error correcting code (see 3.1.31)
GLIST	grown defect list (see 3.1.39)
LBA	logical block address (see 3.1.47)
LBM	Logical Block Markup (see 6.8)
LBP	logical block provisioning (see 4.7.4)

LSB	least significant bit
LUN	logical unit number
M	implementation is mandatory
MMC-6	SCSI Multimedia Commands - 6 (see clause 2)
MSB	most significant bit
O	implementation is optional
PLIST	primary defect list (see 3.1.67)
n/a	not applicable
ROD	representation of data (see SPC-6)
SAM-6	SCSI Architecture Model - 6 (see clause 2)
SCSI	Small Computer System Interface family of standards
SCC-2	SCSI-3 Controller Commands - 2 (see clause 2)
SES-3	SCSI Enclosure Services - 3 (see clause 2)
SPC-6	SCSI Primary Commands - 6 (see clause 2)
SPL-5	SAS Protocol Layer-5 (see clause 2)
VPD	Vital product data (see 6.6)
XOR	exclusive-or (see 3.1.32)
ZBC-2	Zoned Block Commands - 2 (see clause 2)

3.4 Keywords

3.4.1 invalid
keyword used to describe an illegal or unsupported bit, byte, word, field or code value

Note 1 to entry: Receipt of an invalid bit, byte, word, field or code value shall be reported as an error.

3.4.2 mandatory
keyword indicating an item that is required to be implemented as defined in this standard

3.4.3 may
keyword that indicates flexibility of choice with no implied preference

Note 1 to entry: “May” is equivalent to “may or may not”.

3.4.4 may not
keywords that indicate flexibility of choice with no implied preference

Note 1 to entry: “May not” is equivalent to “may or may not”.

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

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

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

3.4.8 reserved
keyword referring to bits, bytes, words, fields and code values that are set aside for future standardization

Note 1 to entry: A reserved bit, byte, word or field shall be set to zero, or in accordance with a future extension to this standard.

Note 2 to entry: Recipients are not required to check reserved bits, bytes, words or fields for zero values.

Note 3 to entry: Receipt of reserved code values in defined fields shall be reported as an error.

3.4.9 restricted
keyword referring to bits, bytes, words, and fields that are set aside for other identified standardization purposes

Note 1 to entry: A restricted bit, byte, word, or field shall be treated as a reserved bit, byte, word or field in the context where the restricted designation appears.

3.4.10 shall
keyword indicating a mandatory requirement

Note 1 to entry: Designers are required to implement all such mandatory requirements to ensure interoperability with other products that conform to this standard.

3.4.11 should
keyword indicating flexibility of choice with a strongly preferred alternative

Note 1 to entry: “Should” is equivalent to the phrase “it is strongly recommended”.

3.4.12 vendor specific
something (e.g., a bit, field, code value) that is not defined by this standard

Note 1 to entry: Specification of the referenced item is determined by the SCSI device vendor and may be used differently in various implementations.

3.5 Editorial conventions

Certain words and terms used in this standard have a specific meaning beyond the normal English meaning. These words and terms are defined either in 3.5 or in the text where they first appear.

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

Upper case is used when referring to names of commands, status codes, sense keys, and additional sense codes (e.g., REQUEST SENSE command).

If there is more a CDB length specified for a particular command (e.g., ORWRITE (16) and ORWRITE (32)), and the name of the command is used in a sentence without any CDB length descriptor (e.g., ORWRITE), then the condition described in the sentence applies to all CDB lengths for that command.

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 also be used for readability (e.g., the LOGERR bit). 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.

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

EXAMPLE 1 - The following list shows no relationship between the listed items:

a) red (i.e., one of the following colors):
   A) crimson: or
   B) amber;
b) blue; or
c) green.

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

EXAMPLE 2 - The following list shows an ordered relationship between the named items:

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

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

In the event of conflicting information the precedence for requirements defined in this standard is:

1) text;
2) tables; then
3) figures.

Tables show data format and values. Not all tables or figures are fully described in the text.

Notes and examples do not constitute any requirements for implementers, and notes are numbered consecutively throughout this standard.

3.6 Numeric and character conventions

3.6.1 Numeric conventions

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

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

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

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

This standard uses the following conventions for representing decimal numbers:

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

Table 2 shows some examples of decimal numbers represented using various conventions.

Table 2 — Numbering convention examples

ISO/IEC	United States	This standard
0,6	0.6	0.6
3,141 592 65	3.14159265	3.141 592 65
1 000	1,000	1 000
1 323 462,95	1,323,462.95	1 323 462.95

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

3.6.2 Units of measure

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

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

   and

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

Table 3 compares the prefix, symbols, and power of the binary and decimal units.

Table 3 — Comparison of decimal prefixes and binary prefixes

Decimal			Binary		
Prefix name	**Prefix symbol**	**Power (base-10)**	**Prefix name**	**Prefix symbol**	**Power (base-2)**
kilo	k	10^{3}	kibi	Ki	2^{10}
mega	M	10^{6}	mebi	Mi	2^{20}
giga	G	10^{9}	gibi	Gi	2^{30}
tera	T	10^{12}	tebi	Ti	2^{40}
peta	P	10^{15}	pebi	Pi	2^{50}
exa	E	10^{18}	exbi	Ei	2^{60}
zetta	Z	10^{21}	zebi	Zi	2^{70}
yotta	Y	10^{24}	yobi	Yi	2^{80}

3.7 State machine conventions

Figure 1 shows how state machines are described in this standard.

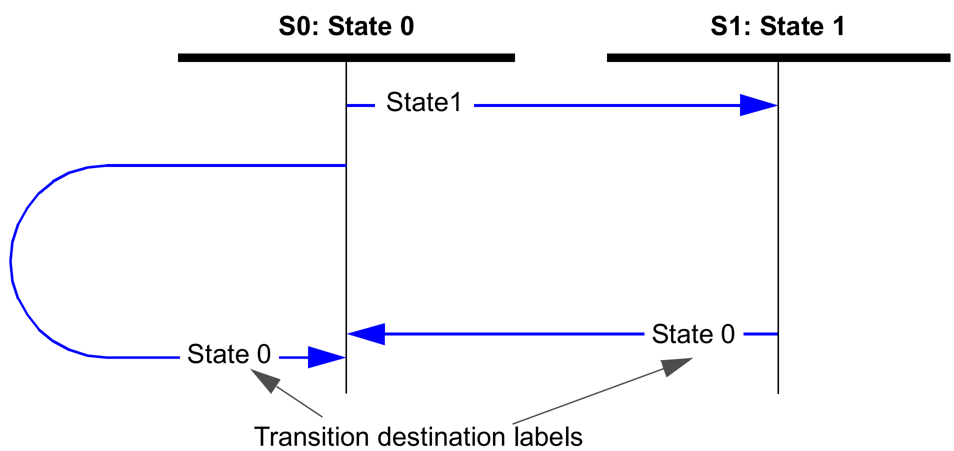


Figure 1 — Example state machine figure

The state machine figure is followed by subclauses describing the states and state transitions.

Each state and state transition is described in the list with particular attention to the conditions that cause the transition to occur and special conditions related to the transition.

A system specified in this manner has the following properties:

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

4 Direct access block device type model

4.1 Direct access block device type model introduction

Table 4 shows the topics in clause 4 and a reference to the subclause where each topic is described.

Table 4 — Direct access block device type model topics (part 1 of 2)

Topic	Reference
Direct access block device type model overview	4.2
Media examples	4.3
Removable media	4.4
Logical blocks	4.5
Physical blocks	4.6
Logical block provisioning	4.7
Data de-duplication	4.8
Ready state	4.9
Initialization	4.10
Sanitize operations	4.11
Write protection	4.12
Medium defects	4.13
Write and unmap failures	4.14
Caches	4.15
Implicit HEAD OF QUEUE command processing	4.16
Reservations	4.17
Error reporting	4.18
Rebuild assist mode	4.19
START STOP UNIT and power conditions	4.20
Protection information model	4.21
Grouping function	4.22
Background scan operations	4.23
Deferred microcode activation	4.24
Model for uninterrupted sequences on LBA ranges	4.25
Referrals	4.26

Table 4 — Direct access block device type model topics (part 2 of 2)

Topic	Reference
ORWRITE commands	4.27
Block device ROD token operations	4.28
Atomic Writes	4.29
IO Advice Hints	4.30
Background operation control	4.31
Stream control	4.32
Format operations	4.33
Transfer limits	4.34
Scattered writes	4.35
Storage element depopulation and restoration	4.36

4.2 Direct access block device type model

4.2.1 Direct access block device type model overview

SCSI devices that conform to this standard are referred to as direct access block devices (e.g., hard disk drives, removable rigid disks, and solid state drives).

This standard is intended to be used in conjunction with SAM-6, SPC-6, SCC-2, and SES-3.

Direct access block devices store data in logical blocks for later retrieval.

Logical blocks are stored by a process that causes localized changes or transitions within a 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).

4.2.2 Logical block access command types

The following are logical block access command types:

a) read commands;
b) unmap commands;
c) verify commands;
d) write commands; and
e) other commands (e.g., a FORMAT UNIT command or a SANITIZE command).

See table 34 for a list of commands for direct access block devices, including the logical block access command type. Some commands may be more than one type of logical block access command (e.g., a COMPARE AND WRITE command is both a read command and a write command).

4.2.3 Logical block access operation types

Each named command type (see 4.2.2) is processed by performing one or more of the following:

a) read operations;
b) unmap operations;
c) verify operations; and
d) write operations.

A device server that supports optional features (e.g., caches (see 4.15)) may be required to support additional requirements (e.g., cache coherency, LBA mapping resource allocations) that are related to those features for specific operations (e.g., read operations, write operations).

In a device server that does not support any optional features:

a) any read operation causes only read medium operations to be performed;
b) any verify operation causes only verify medium operations to be performed; and
c) any write operation causes only write medium operations to be performed.

The requirements for any optional feature (e.g., caches) may include additional requirements for the operations described in this subclause.

If an optional feature (e.g., caches) defines requirements for read operations, then the device server shall support those requirements for both verify operations and read operations.

4.3 Media examples

4.3.1 Media examples overview

Examples of types of media used by the direct access block device are:

a) a rotating medium (see 4.3.2); and
b) a memory medium (see 4.3.3).

Other types of media are possible.

4.3.2 Rotating media

A rotating medium is one or more spinning disks, each coated with a magnetic material that allows flux changes to be induced and recorded. An actuator positions a read-write head radially across the spinning disk, allowing the device to randomly read or write the information at any radial position. Data is stored by using the write portion of the head to record flux changes and the recorded data is read by using the read portion of the head.

The circular path followed by the read-write head at a particular radius is called a track. A track is divided into sectors each containing blocks of stored data. If there is more than one disk spinning on a single axis and the actuator has a read-write head to access each of the disk surfaces, then the collection of tracks at a particular radius is called a cylinder.

A logical block is stored in one or more sectors, or a sector may store more than one logical block. Sectors may also contain information for accessing, synchronizing, and protecting the integrity of the logical blocks.

A rotating medium direct access block device is ready if:

a) the disks are rotating at the correct speed; and
b) the read-write circuitry is powered and ready to access the data.

A START STOP UNIT command (see 5.31) may be required to bring the logical unit to the ready state.

The rotating medium in a direct access block device is non-volatile.

4.3.3 Memory media

A memory medium is solid state, random access memory (RAM) (e.g., static RAM (SRAM), dynamic RAM (DRAM), magnetoresistive RAM (MRAM), ferroelectric RAM (FeRAM), or flash memory).

A memory medium direct access block device may be ready after power on and may not require a START STOP UNIT command (see 5.31) to bring the logical unit to a ready state.

These logical units may be nonmechanical, and logical blocks may be accessed with similar access times regardless of their location on the medium. Memory medium direct access block devices may store less data than disks or tapes and may be volatile.

A memory medium may be volatile (e.g., SRAM or DRAM) or non-volatile (e.g., SRAM or DRAM with battery backup, MRAM, FeRAM, or flash memory).

4.4 Removable media

The medium may be removable or non-removable. A 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 demounted on a suitable transport mechanism in a direct access block device. A removable medium is mounted when the direct access block device is capable of accessing its medium (e.g., performing read medium operations and write medium operations). A removable medium is demounted at any other time (e.g., during loading, unloading, or storage).

An application client may check whether a removable medium is mounted by sending a TEST UNIT READY command (see SPC-6). A direct access block device containing a removable medium may not be accessible for read operations, unmap operations, and write operations until it receives a START STOP UNIT command with the START bit set to one (see 5.31).

If a direct access block device implements cache, either volatile or non-volatile, then the device server ensures that all logical blocks on the medium contain the most recent logical block data prior to permitting demounting of the removable medium.

If the medium in a direct access block device is removable, and the medium is removed, then the device server shall establish a unit attention condition with the additional sense code set to the appropriate value (e.g., MEDIUM NOT PRESENT). The PREVENT ALLOW MEDIUM REMOVAL command (see 5.15) allows an application client to restrict the demounting of the removable medium.

If an application client sends a START STOP UNIT command to request that the removable medium to be ejected and the direct access block device is prevented from demounting the medium by a previous PREVENT ALLOW MEDIUM REMOVAL command, then the START STOP UNIT command is terminated by the device server.

4.5 Logical blocks

Logical blocks are stored on the medium. Logical blocks:

a) contain logical block data that contains:
   A) user data; and
   B) protection information, if any;
   and
b) may contain additional information (e.g., an ECC which may be used for medium defect management (see 4.13)), which may not be accessible to the application client.

The number of bytes of user data contained in each logical block is the logical block length. The logical block length is greater than or equal to one byte and should be an even number of bytes (e.g., 512 bytes, 520 bytes, 4 096 bytes, or 4 104 bytes). The logical block length does not include the length of protection information, if any, and additional information, if any, that are contained in the logical block. The logical block length is the same for all logical blocks in the logical unit. The LOGICAL BLOCK LENGTH IN BYTES field (see 5.20.2) and the READ CAPACITY (16) parameter data (see 5.21.2) indicates the logical block length. The FORMAT UNIT command (see 5.4) and the mode parameter block descriptor (see 6.5.2) are used together by an application client to change the logical block length in direct access block devices that support changeable logical block lengths.

Each logical block is referenced by a unique LBA, which is either four bytes in length or eight bytes in length. The LBAs on a logical unit shall begin with zero and shall be contiguous up to the last LBA on the logical unit. The last LBA is [n-1], where [n] is the number of logical blocks accessible by an application client.

For this standard, the RETURNED LOGICAL BLOCK ADDRESS field in the READ CAPACITY (10) parameter data (see 5.20.2) and the READ CAPACITY (16) parameter data (see 5.21.2) indicates the value of [n-1].

Other command standards (e.g., ZBC-2) may define the contents of the RETURNED LOGICAL BLOCK ADDRESS field to be less than [n-1].

Each LBA has a logical block provisioning state (see 4.7) of mapped, deallocated, or anchored.

Some commands support only four-byte LOGICAL BLOCK ADDRESS fields (e.g., READ (10), and WRITE (10)).

If the capacity of the logical unit exceeds that accessible with four-byte LBAs, then the device server returns the RETURNED LOGICAL BLOCK ADDRESS field set to FFFF_FFFFh in the READ CAPACITY (10) parameter data, indicating that an application client should:

a) enable descriptor format sense data (see SPC-6) in the Control mode page (see SPC-6) and in any REQUEST SENSE commands (see SPC-6) it sends; and
b) use commands with eight-byte LOGICAL BLOCK ADDRESS fields (e.g., READ (16), and WRITE (16)).

NOTE 1 - If a command requests access to an LBA greater than FFFF_FFFFh, fixed format sense data is used, and an error occurs for that LBA, then there is no field in the sense data large enough to report that LBA as having an error (see 4.18).

If a command is received that references or attempts to access a logical block that exceeds the capacity of the medium, then the device server shall terminate the command (e.g., with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to LOGICAL BLOCK ADDRESS OUT OF RANGE). The device server:

a) should terminate the command before processing; and
b) may terminate the command after the device server has transferred some, all, or none of the data.

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 direct access block device with rotating media (see 4.3.2), the time to access a logical block at LBA [x+1] after accessing LBA [x] is often less than the time to access some other logical block.

4.6 Physical blocks

4.6.1 Overview

A physical block is a set of data bytes on the medium accessed by the device server as a unit. A physical block may contain:

a) a portion of a logical block (i.e., there are multiple physical blocks in the logical block)(e.g., a physical block length of 512 bytes with a logical block length of 2 048 bytes);
b) a single complete logical block; or
c) more than one logical block (i.e., there are multiple logical blocks in the physical block)(e.g., a physical block length of 4 096 bytes with a logical block length of 512 bytes).

Each physical block may include additional information (e.g., an ECC which may be used for medium defect management (see 4.13)), which may not be accessible to the application client.

If the device server supports the creation of pseudo unrecovered errors (see 4.18.2), then the device server shall have the capability of marking individual logical blocks as containing pseudo unrecovered errors.

Logical blocks may or may not be aligned to physical block boundaries. A mechanism for establishing the alignment is not defined by this standard.

Figure 2 shows examples of where there are one or more physical blocks per logical block, and LBA 0 is aligned to a physical block boundary. The LOGICAL BLOCKS PER PHYSICAL BLOCK EXPONENT field and the LOWEST ALIGNED LOGICAL BLOCK ADDRESS field (see 5.21.2) indicate the alignment.

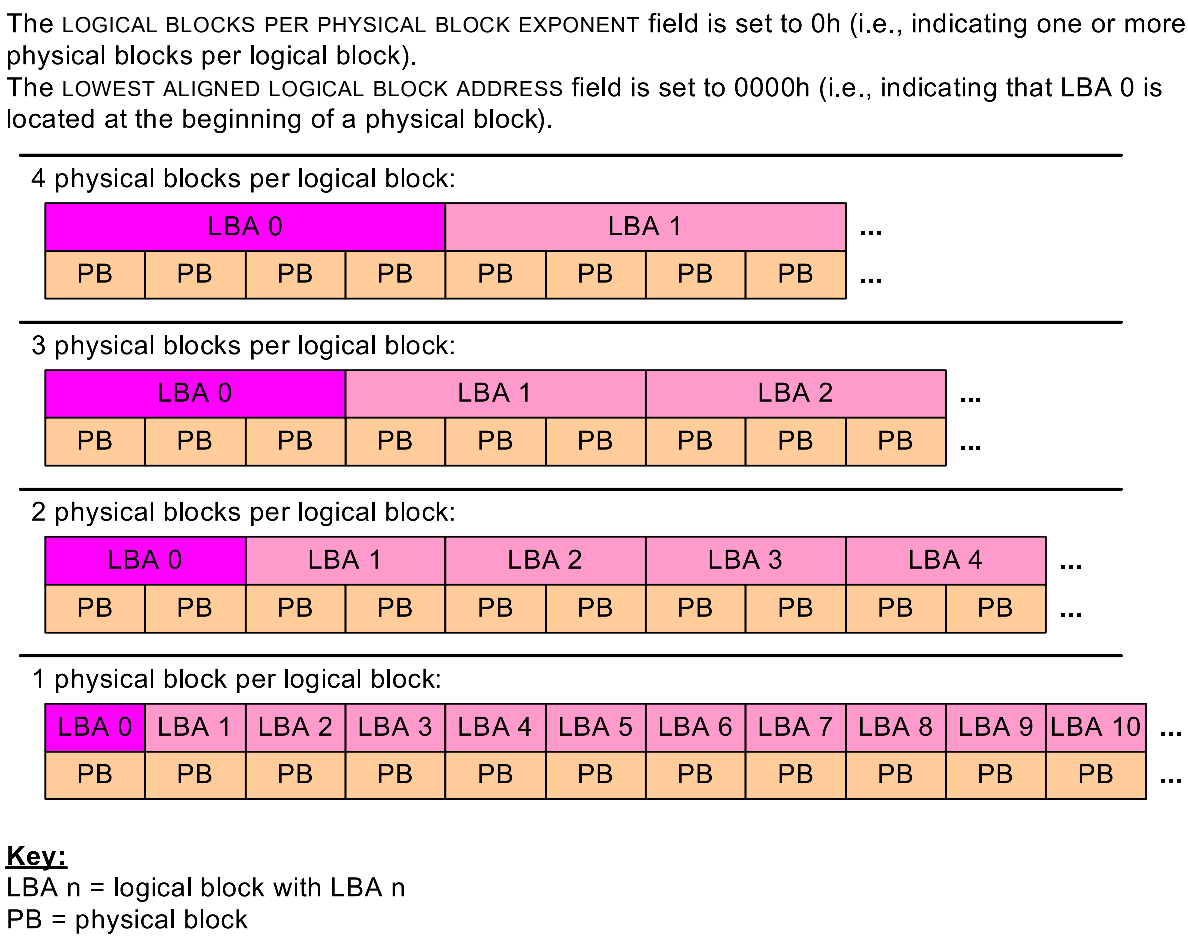
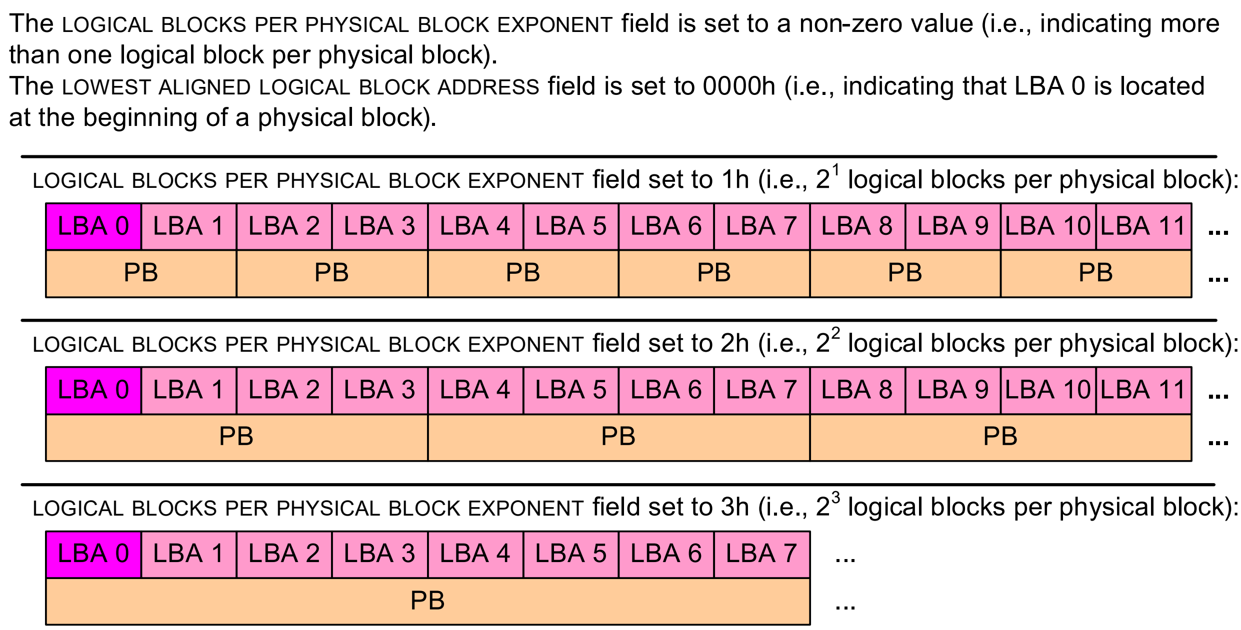


Figure 2 — One or more physical blocks per logical block examples

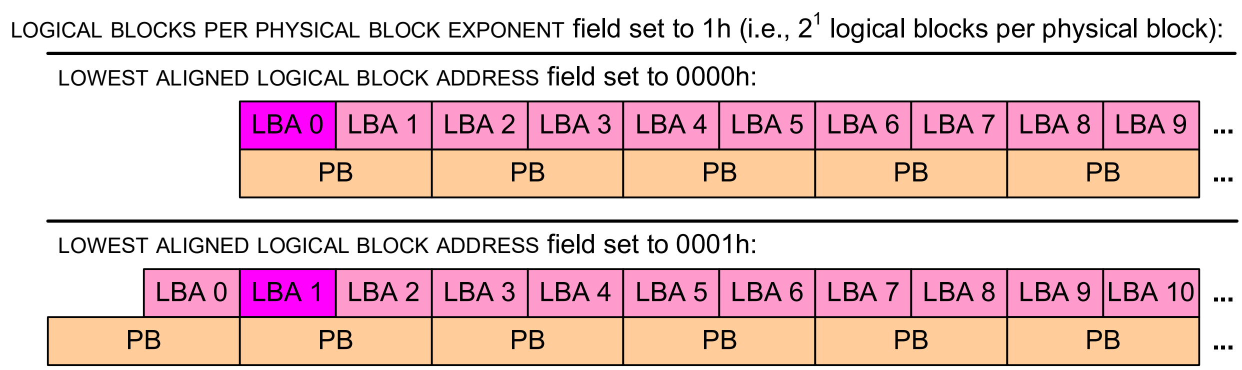
Figure 3 shows examples of where there are one or more logical blocks per physical block, and LBA 0 is aligned to a physical block boundary. The LOGICAL BLOCKS PER PHYSICAL BLOCK EXPONENT field and the LOWEST ALIGNED LOGICAL BLOCK ADDRESS field (see 5.21.2) indicate the alignment.



Key:
LBA n = logical block with LBA n
PB = physical block

Figure 3 — One or more logical blocks per physical block examples

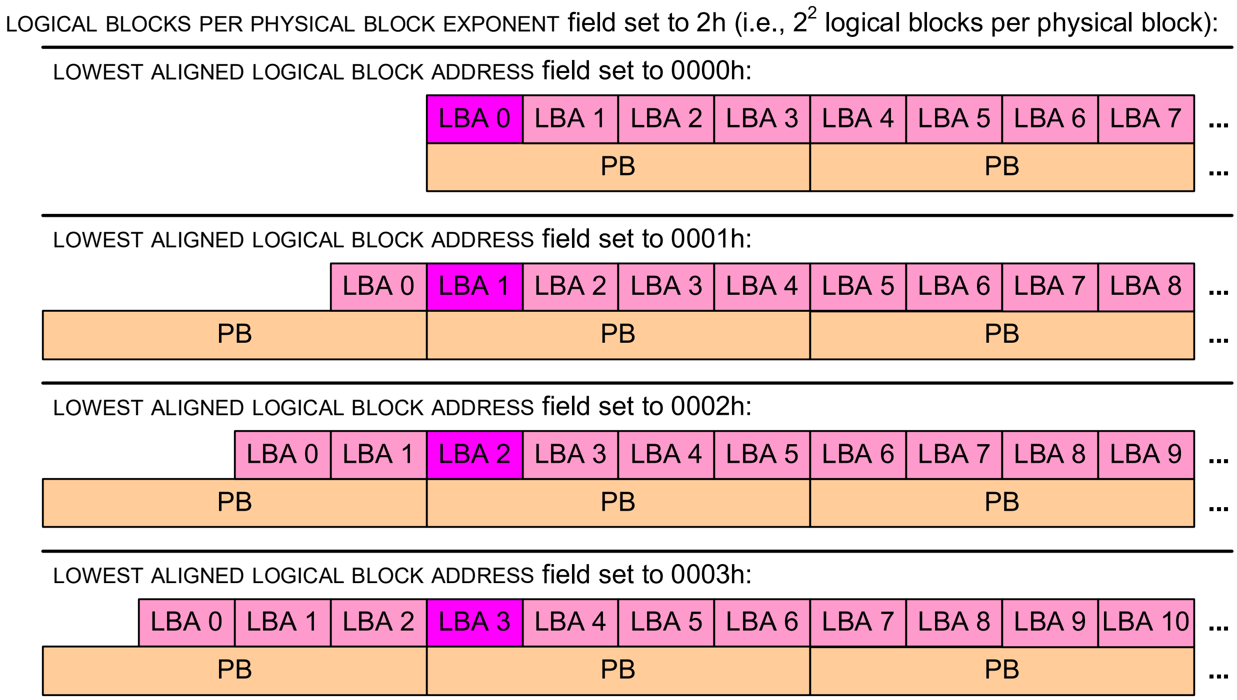
Figure 4 shows examples of where there are two logical blocks per physical block, and different LBAs are aligned to physical block boundaries. The LOGICAL BLOCKS PER PHYSICAL BLOCK EXPONENT field and the LOWEST ALIGNED LOGICAL BLOCK ADDRESS field (see 5.21.2) indicate the alignment.



Key:
LBA n = logical block with LBA n
PB = physical block

Figure 4 — Two logical blocks per physical block alignment examples

Figure 5 shows examples of where there are four logical blocks per physical block, and different LBAs are aligned to physical block boundaries. The LOGICAL BLOCKS PER PHYSICAL BLOCK EXPONENT field and the LOWEST ALIGNED LOGICAL BLOCK ADDRESS field (see 5.21.2) indicate the alignment.



Key:
LBA n = logical block with LBA n
PB = physical block

Figure 5 — Four logical blocks per physical block alignment examples

If there is more than one logical block per physical block, then not all of the logical blocks are aligned to the physical block boundaries. When using logical block access commands (see 4.2.2), application clients should:

a) specify an LBA that is aligned to a physical block boundary; and
b) access an integral number of physical blocks, provided that the access does not go beyond the last LBA on the medium.

See annex E for an example method in which application clients may use alignment information to determine optimal performance for logical block access.

4.6.2 Physical block misaligned write reporting

A misaligned write command is a write command that specifies a:

a) LOGICAL BLOCK ADDRESS field that does not correspond to the first LBA of a physical block; or
b) TRANSFER LENGTH field that is not a multiple of the number of logical blocks per physical block (see table 91) and the number of logical blocks per physical block is greater than one.

The LPS Misalignment log page (see 6.4.6) reports misaligned write command information. The oldest reported misaligned write command is identified by a parameter code value of 0001h, with each successive time ordered misaligned write command identified by successively numbered parameter codes.

The maximum number of reportable LPS Misalignment log parameters supported by the device server is indicated in the MAX_LPSM field in the LPS Misalignment Count log parameter. If the parameter code value for the newest LPS Misalignment log parameter (i.e. parameter code with the greatest value) is equal to the value of the MAX_LPSM field in the LPS Misalignment Count log parameter (i.e. the log is full), then the device server shall not append additional LPS Misalignment log parameters to the LPS Misalignment log page.

The MWR field (see 6.5.10) specifies how misaligned write commands are processed. If the device server processes a misaligned write command, and the MWR field is set to:

a) 00b (i.e. DISABLED), then the device server shall process that write command without adding a log parameter in the LPS Misalignment log page (see 6.4.6);
b) 01b (i.e. ENABLED), then the device server shall:
   1) process that write command;
   2) add a log parameter in the LPS Misalignment log page, if the log is not full; and
   3) if that write command completes without error, complete the command with GOOD status with the sense key set to COMPLETED and the additional sense code set to MISALIGNED WRITE COMMAND;

   or

c) 10b (i.e. TERMINATE), then the device server shall:
   1) add a log parameter in the LPS Misalignment log page, if the page is not full; and
   2) terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to MISALIGNED WRITE COMMAND.

4.7 Logical block provisioning

4.7.1 Logical block provisioning overview

Each LBA in a logical unit is either mapped or unmapped. For LBAs that are mapped, there is a known relationship between the LBA and one or more physical blocks that contain logical block data. For LBAs that are unmapped, the relationship between the LBA and a physical block is not defined. Figure 6 shows two examples of the relationship between mapped and unmapped LBAs and physical blocks in a logical unit. One example shows one LBA per physical block and one example shows two LBAs per physical block. The LOGICAL BLOCKS PER PHYSICAL BLOCK EXPONENT field is defined in the READ CAPACITY (16) parameter data (see 5.21.2).

LOGICAL BLOCKS PER PHYSICAL BLOCK EXPONENT field set to 0h (i.e., 2^0 logical blocks per physical block):

LBA 0	LBA 1	LBA 2	LBA 3	LBA 4	LBA 5	LBA 6	LBA 7
PB	PB	Unmapped	PB	Unmapped	Unmapped	PB	PB

LOGICAL BLOCKS PER PHYSICAL BLOCK EXPONENT field set to 1h (i.e., 2^1 logical blocks per physical block):

LBA 0	LBA 1	LBA 2	LBA 3	LBA 4	LBA 5	LBA 6	LBA 7
PB		Unmapped		PB		Unmapped	

Key:
LBA n = logical block with LBA n
PB = physical block
Unmapped = the relationship between the LBA(s) and a physical block is not defined

Figure 6 — Examples of the relationship between mapped and unmapped LBAs and physical blocks

Each unmapped LBA is either anchored or deallocated. Anchored and deallocated are states in the LBP state machine (see 4.7.4) that have the following properties:

a) a write command that specifies an anchored LBA does not require allocation of additional LBA mapping resources for that LBA; and
b) a write command that specifies a deallocated LBA may require allocation of LBA mapping resources.

Depending on the logical block provisioning types (see table 5), the quantity of LBA mapping resources available to a logical unit may be greater than, equal to, or less than the quantity required to store logical block data for every LBA.

Table 5 list the logical block provisioning states supported by each type of logical block provisioning.

Table 5 — Logical block provisioning states supported by logical block provisioning type

Type	Logical block provisioning states			Reference
	Mapped	Unmapped		
		Anchored	Deallocated	
Fully provisioned	Mandatory	Prohibited	Prohibited	4.7.2
Resource provisioned	Mandatory	Optional	Optional	4.7.3.2
Thin provisioned	Mandatory	Optional	Mandatory	4.7.3.3

4.7.2 Fully provisioned logical unit

The device server shall map every LBA in a fully provisioned logical unit. A fully provisioned logical unit shall provide enough LBA mapping resources to contain all logical blocks for the logical unit’s capacity as reported in the READ CAPACITY (10) parameter data (see 5.20.2) and the READ CAPACITY (16) parameter data (see 5.21.2). The device server shall not cause any LBA on a fully provisioned logical unit to become unmapped (i.e., anchored or deallocated).

A fully provisioned logical unit does not support logical block provisioning management (see 4.7.3). A fully provisioned logical unit may support the GET LBA STATUS (16) command (see 5.6) and the GET LBA STATUS (32) command (see 5.7).

The device server in a fully provisioned logical unit shall set the LBPME bit to zero in the READ CAPACITY (16) parameter data (see 5.21.2).

4.7.3 Logical block provisioning management

4.7.3.1 Logical block provisioning management overview

A logical unit that supports logical block provisioning management (i.e., implements unmapped LBAs, unmap operations, and related actions) shall be either:

a) resource provisioned (see 4.7.3.2); or
b) thin provisioned (see 4.7.3.3).

A logical unit that supports logical block provisioning management may change from resource provisioned to thin provisioned as resources become unavailable. If the logical unit transitions from resource provisioned to thin provisioned, then the logical unit shall change the PROVISIONING TYPE field to 010b (i.e., the logical unit is thin provisioned) in the Logical Block Provisioning VPD page (see 6.6.9) and establish a unit attention condition with the additional sense code set to INQUIRY DATA HAS CHANGED as described in SPC-6.

A logical unit that supports logical block provisioning management shall implement the LBP state machine (see 4.7.4) for each LBA.

The device server in a logical unit that supports logical block provisioning management:

a) shall support the Logical Block Provisioning VPD page (see 6.6.9);
b) may supply a provisioning group designation descriptor as defined in the Logical Block Provisioning VPD page;
c) may support logical block provisioning thresholds (see 4.7.3.7.1);
d) may support the GET LBA STATUS (16) command (see 5.6);
e) may support the GET LBA STATUS (32) command (see 5.7)
f) should support the Block Limits VPD page (see 6.6.4); and
g) shall support at least one of the following unmap mechanisms:
    A) the UNMAP command (see 5.35);
    B) the UNMAP bit in the WRITE SAME (10) command (see 5.52);
    C) the UNMAP bit in the WRITE SAME (16) command (see 5.53); or
    D) the UNMAP bit in the WRITE SAME (32) command (see 5.54).

If the device server supports:

a) the UNMAP bit in the WRITE SAME (10) command or in the WRITE SAME (16) command; and
b) the WRITE SAME (32) command (see 5.54),

then the device server shall support the UNMAP bit in the WRITE SAME (32) command.

If a device server supports the UNMAP command and the Block Limits VPD page, then the device server shall:

a) set the MAXIMUM UNMAP LBA COUNT field (see 6.6.4) to a value greater than or equal to one; and
b) set the MAXIMUM UNMAP DESCRIPTOR COUNT field (see 6.6.4) to a value greater than or equal to one.

4.7.3.2 Resource provisioned logical unit

A resource provisioned logical unit shall support logical block provisioning management (see 4.7.3.1).

The device server shall map, anchor, or deallocate each LBA in a resource provisioned logical unit. A resource provisioned logical unit shall provide LBA mapping resources sufficient to map all LBAs for the logical unit’s capacity as indicated in the RETURNED LOGICAL BLOCK ADDRESS field of the READ CAPACITY (10) parameter data (see 5.20.2) and the READ CAPACITY (16) parameter data (see 5.21.2). A resource provisioned logical unit may provide resources in excess of this requirement.

The device server in a resource provisioned logical unit:

a) shall set the LBPME bit to one in the READ CAPACITY (16) parameter data (see 5.21.2);
b) shall set the PROVISIONING TYPE field to 001b (i.e., resource provisioned) in the Logical Block Provisioning VPD page (see 6.6.9); and
c) may set the ANC_SUP bit to one in the Logical Block Provisioning VPD page.

The initial condition of every LBA in a resource provisioned logical unit is anchored (see 4.7.4.1) or dellocated.

4.7.3.3 Thin provisioned logical unit

A thin provisioned logical unit shall support logical block provisioning management (see 4.7.3.1).

The device server in a thin provisioned logical unit may indicate a larger capacity in the RETURNED LOGICAL BLOCK ADDRESS field in the READ CAPACITY (10) parameter data (see 5.20.2) and the READ CAPACITY (16) parameter data (see 5.21.2) than the number of LBA mapping resources available for mapping LBAs in the logical unit.

The device server shall map, anchor, or deallocate each LBA in a thin provisioned logical unit (see table 5). A thin provisioned logical unit is not required to provide LBA mapping resources sufficient to map all LBAs for the logical unit’s capacity as indicated in the RETURNED LOGICAL BLOCK ADDRESS field of the READ CAPACITY (10) parameter data (see 5.20.2) and the READ CAPACITY (16) parameter data (see 5.21.2).

If the logical unit does not support anchored LBAs (i.e, the ANC_SUP bit is set to zero in the Logical Block Provisioning VPD page (see 6.6.9)), then:

a) every unmapped LBA in the logical unit shall be deallocated; and
b) the device server shall terminate every command that specifies anchoring an LBA (e.g., a WRITE SAME command with the ANCHOR bit set to one (see 5.52)).

The device server in a thin provisioned logical unit shall set:

a) the LBPME bit to one in the READ CAPACITY (16) parameter data (see 5.21.2); and
b) the PROVISIONING TYPE field to 010b (i.e., thin provisioned) in the Logical Block Provisioning VPD page (see 6.6.9).

The initial condition of every LBA in a thin provisioned logical unit is deallocated (see 4.7.4.1).

4.7.3.4 Unmapping LBAs

4.7.3.4.1 Unmapping overview

A logical unit that supports logical block provisioning management shall support unmapping of LBAs.

The logical block provisioning state of an LBA may change to unmapped (i.e., deallocated (see 4.7.4.6) or anchored (see 4.7.4.7)) as a result of:

a) an unmap request (see 4.7.3.4.2);
b) an autonomous LBA transition (see 4.7.3.5) (e.g., following a FORMAT UNIT command (see 5.4) or a write command that sets the logical block data to zero); or
c) other commands that result in LBAs being initialized to unmapped (e.g., a SANITIZE command (see 5.30)).

4.7.3.4.2 Processing unmap requests

Application clients use unmap commands (see 4.2.2) to request that LBAs be unmapped. For each LBA that is requested to be unmapped, the device server shall:

a) perform an unmap operation (see 4.7.3.4.3) on the LBA; or
b) make no change to the logical block provisioning state of the LBA.

The application client determines the logical block provisioning state of LBAs using the GET LBA STATUS (16) command (see 5.6) or the GET LBA STATUS (32) command (see 5.7).

Application clients should not rely on an UNMAP command (see 5.35) to cause specific data (e.g., zeros) to be returned by subsequent read operations on the specified LBAs. To produce consistent results for subsequent read operations, a write command (e.g., the WRITE SAME command) should be used to write user data.

EXAMPLE - To ensure that subsequent read operations return all zeros in a logical block, the application client should use the WRITE SAME (16) command with the NDOB bit set to one. If the UNMAP bit is set to one, then the device server may unmap the logical blocks specified by the WRITE SAME (16) command as described in 4.7.3.4.4.

4.7.3.4.3 Unmap operations

An unmap operation:

a) results in a single LBA becoming either deallocated or anchored;
b) may change the relationship between one LBA and one or more physical blocks; and
c) may change the logical block data that is returned in response to a subsequent read command specifying that LBA.

The data in all other mapped LBAs on the medium shall be preserved. Performing an unmap operation (e.g., to change from anchored to deallocated, or remain in the same logical block provisioning state) on an unmapped LBA shall not be considered an error.

An unmap operation may or may not release LBA mapping resources.

An application client may use an unmap command (see 4.2.2) to request that the device server perform an unmap operation on each specified LBA. A single unmap command may result in zero or more unmap operations.

4.7.3.4.4 WRITE SAME command and unmap operations

A WRITE SAME command (see 5.52, 5.53, and 5.54) may be used to request unmap operations that deallocate or anchor the specified LBAs. If unmap operations are requested in a WRITE SAME command, then for each specified LBA:

a) if the Data-Out Buffer of the WRITE SAME command is the same as the logical block data returned by a read operation from that LBA while in the unmapped state (see 4.7.4.4), then:
   1) the device server performs the actions described in table 6; and
   2) if an unmap operation is not performed in step 1), then the device server shall perform the specified write operation to that LBA;

   or

b) if the Data-Out Buffer of the WRITE SAME command is not the same as the logical block data returned by a read operation from that LBA while in the unmapped state (see 4.7.4.4), then the device server shall perform the specified write operation to that LBA.

Table 6 — WRITE SAME command and unmap operations

Logical block provisioning management type	Unmap operations that request to deallocate the specified LBA	Unmap operations that request to anchor the specified LBA
Thin provisioned logical unit	a) should perform an unmap operation to deallocate the LBA (see 4.7.4.6.1); or b) may perform an unmap operation to anchor the LBA (see 4.7.4.7.1)	should perform an unmap operation to anchor the LBA
Resource provisioned logical unit	a) should perform an unmap operation to anchor the LBA; or b) may perform an unmap operation to deallocate the LBA	should perform an unmap operation to anchor the LBA

A WRITE SAME command shall not cause an LBA to become unmapped if unmapping that LBA creates a case in which a subsequent read of that unmapped LBA is able to return logical block data that differs from the Data-Out Buffer for that WRITE SAME command (see 4.7.4.4).

If the device server does not support allowing a WRITE SAME command to request unmap operations, then the device server shall:

a) perform the write operations specified by the WRITE SAME command; and
b) not perform any unmap operations.

The device server shall perform the write operations specified by a WRITE SAME command and shall not perform any unmap operations if the device server sets the LBPRZ field to xx1b (see 6.6.9), and:

a) any bit in the user data transferred from the Data-Out Buffer is not zero; or
b) the protection information, if any, transferred from the Data-Out Buffer is not set to FFFF_FFFF_FFFF_FFFFh.

The device server shall perform the write operations specified by a WRITE SAME command and shall not perform any unmap operations, if the device server sets the LBPRZ field to 010b and:

a) the user data transferred from the Data-Out Buffer is not set to the provisioning initialization pattern; or

b) the protection information, if any, transferred from the Data-Out Buffer is not set to FFFF_FFFF_FFFF_FFFFh.

4.7.3.5 Autonomous LBA transitions

A device server may perform the following actions at any time:

a) transition any deallocated LBA to mapped;
b) transition any anchored LBA to mapped; or
c) transition any deallocated LBA to anchored.

If the LBPRZ field (see 6.6.9) is set to:

a) xx1b, and a mapped LBA references a logical block that contains:
   A) user data with all bits set to zero; and
   B) protection information, if any with the:
      a) LOGICAL BLOCK GUARD field set to FFFFh or set to 0000h;
      b) LOGICAL BLOCK APPLICATION TAG field set to FFFFh; and
      c) LOGICAL BLOCK REFERENCE TAG field set to FFFF_FFFFh;

or

b) 010b, and a mapped LBA references a logical block that contains:
   A) user data set to the provisioning initialization pattern; and
   B) protection information, if any, with the:
      a) LOGICAL BLOCK GUARD field set to FFFFh or set to the CRC for the provisioning initialization pattern;
      b) LOGICAL BLOCK APPLICATION TAG field set to FFFFh; and
      c) LOGICAL BLOCK REFERENCE TAG field set to FFFF_FFFFh,

then the device server may transition that mapped LBA to anchored or deallocated at any time.

The logical block provisioning state machine (see 4.7.4) specifies additional requirements for the transitions specified in this subclause.

4.7.3.6 Logical unit resource exhaustion considerations

4.7.3.6.1 Thin provisioned logical unit resource exhaustion considerations

If:

a) a write operation is requested by an application client, and a temporary lack of LBA mapping resources prevents the logical unit from performing the write operation; or
b) an unmap operation is requested by an application client to transition an LBA to the anchored state and a temporary lack of LBA mapping resources prevents the logical unit from anchoring the LBA,

then the device server shall terminate the command requesting the operation with CHECK CONDITION status with the sense key set to NOT READY and the additional sense code set to LOGICAL UNIT NOT READY, SPACE ALLOCATION IN PROGRESS. In this case, the application client should resend the command.

If:

a) a write operation is requested by an application client, and a persistent lack of LBA mapping resources prevents the logical unit from performing the write operation; or
b) an unmap operation is requested by an application client to transition an LBA to the anchored state and a persistent lack of LBA mapping resources prevents the logical unit from anchoring the LBA,

then the device server shall terminate the command requesting the unmap operation with CHECK CONDITION status with the sense key set to DATA PROTECT and the additional sense code set to SPACE ALLOCATION FAILED WRITE PROTECT. This condition shall not cause the device server to set the WP bit in the DEVICE-SPECIFIC PARAMETER field of the mode parameter header to one (see 6.5.1). In this case, recovery actions by the application client are not defined by this standard.

A logical block provisioning threshold may be available to monitor the availability of LBA mapping resources (see 4.7.3.7). A logical block provisioning log parameter that reports available LBA mapping resources may be available in the Logical Block Provisioning log page (see 6.4.5).

4.7.3.6.2 Resource provisioned logical unit resource exhaustion considerations

If:

a) a write operation is requested by an application client, and a temporary lack of LBA mapping resources prevents the logical unit from performing the write operation; or
b) an unmap operation is requested by an application client to transition an LBA to the anchored state and a temporary lack of LBA mapping resources prevents the logical unit from anchoring the LBA,

then the device server shall terminate the command requesting the operation with CHECK CONDITION status with the sense key set to NOT READY and the additional sense code set to LOGICAL UNIT NOT READY, SPACE ALLOCATION IN PROGRESS. In this case, the application client should resend the command.

In a resource provisioned logical unit a write operation requested by an application client shall not result in a persistent lack of LBA mapping resources that prevents the logical unit from performing the write operation. Therefore, a resource provisioned logical unit shall not terminate any command with CHECK CONDITION status with the sense key set to DATA PROTECT and the additional sense code set to SPACE ALLOCATION FAILED WRITE PROTECT.

A logical block provisioning threshold may be available to monitor the availability of LBA mapping resources (see 4.7.3.7). A parameter that reports available LBA mapping resources may be available in the page (see 6.4.5).

4.7.3.7 Logical block provisioning thresholds

4.7.3.7.1 Logical block provisioning thresholds overview

Logical block provisioning thresholds provide a mechanism for the device server to establish a unit attention condition to notify application clients when thresholds related to logical block provisioning are crossed. Logical block provisioning thresholds may operate on an armed increasing basis or an armed decreasing basis.

If a device server supports logical block provisioning thresholds, then the device server:

a) shall support the Logical Block Provisioning mode page (see 6.5.9); and
b) may support the Logical Block Provisioning log page (see 6.4.5).

Logical block provisioning thresholds may be based on threshold sets (see 4.7.3.7.2) or percentages (see 4.7.3.7.3).

4.7.3.7.2 Threshold sets

The end points of the range over which a logical block provisioning threshold operates are defined as follows:

$$\text{threshold minimum} = ((\text{threshold count} \times \text{threshold set size}) - (\text{threshold set size} \times 0.5))$$

$$\text{threshold maximum} = ((\text{threshold count} \times \text{threshold set size}) + (\text{threshold set size} \times 0.5))$$

where:

threshold minimum	is the lowest number of LBAs in the range for this threshold;
threshold maximum	is the highest number of LBAs in the range for this threshold;
threshold count	is the center of the threshold range for this threshold (i.e., the threshold count value as specified in the threshold descriptor in the Logical Block Provisioning mode page); and

threshold set size is the number of LBAs in each threshold set (i.e., $2^{(\text{threshold exponent})}$ LBAs where the threshold exponent is indicated in the Logical Block Provisioning VPD page (see 6.6.9)).

Table 7 defines the meaning of the combinations of values for the THRESHOLD RESOURCE field, the THRESHOLD TYPE field, and the THRESHOLD ARMING field that are used for logical block provisioning thresholds. See the Logical Block Provisioning mode page (see 6.5.9) for the definition of these fields.

Table 7 — Threshold resource value, threshold type value, and threshold arming value for logical block provisioning thresholds

Threshold resource value	Threshold type value	Threshold arming value	Description
01h	000b	000b	The device server applies the threshold to the availability of LBA mapping resources and performs notifications as the availability of those resources decreases. [a]
02h	000b	001b	The device server applies the threshold to the usage of LBA mapping resources and performs notifications as the usage of those resources increases.
All other combinations			Reserved

[a] The point when availability of LBA mapping resources reaches zero corresponds to the persistent lack of LBA mapping resources described in 4.7.3.6.1.

4.7.3.7.3 Threshold percentages

The end points of the range over which a logical block provisioning threshold percentages operates are defined as follows:

threshold minimum = (threshold count – (threshold percentage size × 0.5))

threshold maximum = (threshold count + (threshold percentage size × 0.5))

where:

threshold minimum is the lowest percentage of device resources available for allocation to logical blocks in the range for this threshold;

threshold maximum is the highest percentage of device resources available for allocation to logical blocks in the range for this threshold;

threshold count is the center of the threshold range for this threshold (i.e., the threshold count value as specified in the threshold descriptor in the Logical Block Provisioning mode page); and

threshold set size is the percentage of allocation resources in each threshold percentage (i.e., the percentage indicated by the THRESHOLD PERCENTAGE field (see 6.6.9)).

Table 8 defines the meaning of the combinations of values for the THRESHOLD RESOURCE field, the THRESHOLD TYPE field, and the THRESHOLD ARMING field that are used for logical block provisioning threshold percentages. See the Logical Block Provisioning mode page (see 6.5.9) for the definition of these fields.

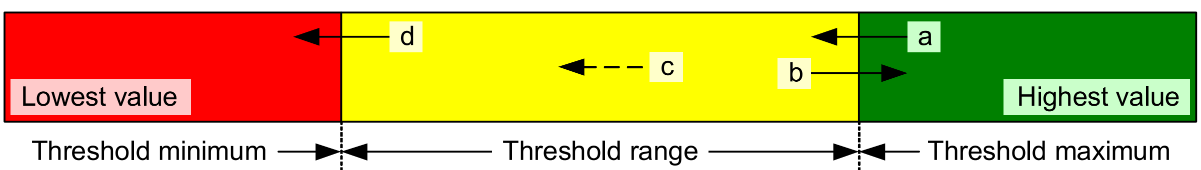
Table 8 — Threshold resource value, threshold type value, and threshold arming value for logical block provisioning percentages

Threshold resource value	Threshold type value	Threshold arming value	Description
03h	001b	000b	The device server applies the threshold to the availability of LBA mapping resources and performs notifications as the availability of those resources decreases.
All other combinations			Reserved

4.7.3.7.4 Logical block provisioning armed decreasing thresholds

Figure 7 shows the operation of a logical block provisioning armed decreasing threshold. Figure 7 represents the entire range of possible values over which the threshold is being applied (e.g., for an available resource, the lowest value represents zero available resources and the highest value represents the maximum possible number of available resources).

If enabled, reporting of armed decreasing threshold events (i.e., the THRESHOLD ARMING field is set to 000b in the threshold descriptor in the Logical Block Provisioning mode page (see 6.5.9)) operates as shown in figure 7.



Notes:

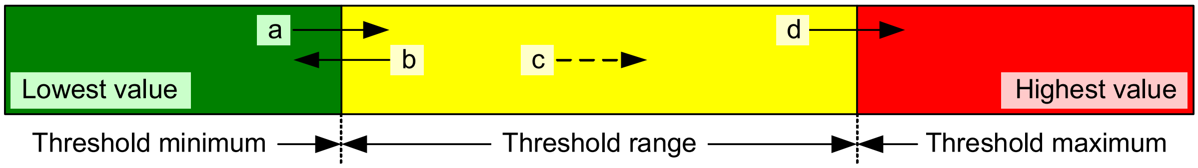
a) if the value to which the threshold is being applied drops below the threshold maximum for the threshold range, then the notification trigger shall be enabled;
b) if the value to which the threshold is being applied increases above the threshold maximum for the threshold range, then the notification trigger shall be disabled;
c) if the notification trigger is enabled, then the device server may disable the notification trigger and perform logical block provisioning threshold notification (see 4.7.3.7.6); and
d) if the notification trigger is enabled and the value to which the threshold is being applied drops below the threshold minimum for the threshold range, then the device server shall disable the notification trigger and perform logical block provisioning threshold notification as defined in 4.7.3.7.6.

Figure 7 — Armed decreasing threshold operation

4.7.3.7.5 Logical block provisioning armed increasing thresholds

Figure 8 shows the operation of a logical block provisioning armed increasing threshold. Figure 8 represents the entire range of possible values over which the threshold is being applied (e.g., for tracking usage of a resource, the lowest value represents zero resources being used and the highest value represents the maximum possible number of resources being used).

If enabled, reporting of armed increasing threshold events (i.e., the THRESHOLD ARMING field is set to 001b in the threshold descriptor in the Logical Block Provisioning mode page (see 6.5.9)) operates as shown in figure 8.



Notes:

a) if the value to which the threshold is being applied increases above the threshold minimum for the threshold range, then the notification trigger shall be enabled;
b) if the value to which the threshold is being applied decreases below the threshold minimum for the threshold range, then the notification trigger shall be disabled;
c) if the notification trigger is enabled, then the device server may disable the notification trigger and perform logical block provisioning threshold notification (see 4.7.3.7.6); and
d) if the notification trigger is enabled and the value to which the threshold is being applied increases above the threshold maximum for the threshold range, then the device server shall disable the notification trigger and perform logical block provisioning threshold notification as defined in 4.7.3.7.6.

Figure 8 — Armed increasing threshold operation

4.7.3.7.6 Logical block provisioning threshold notification

If the LBPERE bit is set to one in the Read-Write Error Recovery mode page (see 6.5.10), then logical block provisioning threshold notification is enabled and the device server shall perform notification for thresholds with the THRESHOLD TYPE field set to 000b in the threshold descriptor in the Logical Block Provisioning mode page (see 6.5.9) as follows:

a) if the SITUA bit is set to one in the Logical Block Provisioning mode page, then:
   A) if the device server has not established a unit attention condition as a result of this threshold being crossed since the last logical unit reset (see SAM-6) and a command through which the device server is able to report a unit attention condition arrives on any I_T nexus, then the device server shall establish a unit attention condition with the additional sense code set to THIN PROVISIONING SOFT THRESHOLD REACHED for only the SCSI initiator port associated with the I_T nexus on which that command was received before processing that command; or
   B) if the device server has established a unit attention condition as a result of this threshold being crossed since the last logical unit reset and a command through which the device server is able to report a unit attention condition arrives on any I_T nexus, then the device server should establish a unit attention condition with the additional sense code set to THIN PROVISIONING SOFT THRESHOLD REACHED for only the SCSI initiator port associated with the I_T nexus on which that command was received before processing that command unless establishment of the unit attention condition causes a vendor specific frequency of unit attention conditions for this threshold to be exceeded;

   or
b) if the SITUA bit is set to zero, then:
   A) if the device server has not established a unit attention condition for the SCSI initiator port associated with all I_T nexuses as a result of this threshold being crossed since the last logical unit reset (see SAM-6), then the device server shall establish a unit attention condition with the additional sense code set to THIN PROVISIONING SOFT THRESHOLD REACHED for the SCSI initiator port associated with every I_T nexus; or
   B) if the device server has established a unit attention condition for the SCSI initiator ports associated with all I_T nexuses as a result of this threshold being crossed since the last logical unit reset, then the device server should establish a unit attention condition with the additional

sense code set to THIN PROVISIONING SOFT THRESHOLD REACHED for the SCSI initiator port associated with every I_T nexus, unless establishment of the unit attention condition causes a vendor specific frequency of unit attention conditions for this threshold to be exceeded.

If a unit attention condition is established as described in this subclause, then the device server shall report the following value in the INFORMATION field in the sense data (see SPC-6):

a) the byte offset in the Logical Block Provisioning mode page of the first byte of the threshold descriptor to which this threshold notification applies.

If a unit attention condition with the additional sense code set to THIN PROVISIONING SOFT THRESHOLD REACHED is received by the application client, then the application client should reissue the command and take further recovery actions (e.g., administrator notification or other administrator actions). These recovery actions are not defined by this standard.

If the LBPERE bit is set to zero, then logical block provisioning threshold notification is disabled and the device server shall not establish any unit attention condition with the additional sense code set to THIN PROVISIONING SOFT THRESHOLD REACHED.

An additional sense code set to THIN PROVISIONING SOFT THRESHOLD REACHED is applicable to both thin provisioned logical units (see 4.7.3.3) and resource provisioned logical units (see 4.7.3.2).

4.7.4 LBP (logical block provisioning) state machine

4.7.4.1 LBP state machine overview

The LBP (logical block provisioning) state machine describes the mapping and unmapping of a single LBA by the device server for a thin provisioned logical unit (see 4.7.3.3) or a resource provisioned logical unit (see 4.7.3.2). This state machine does not apply to fully provisioned logical units (see 4.7.2).

There is one instance of this state machine for each LBA. If a command requests mapping or unmapping of more than one LBA, then there may be an independent transition in each instance of the state machine (e.g., each LBA may individually transition from mapped to deallocated or from anchored to deallocated).

4.7.4.2 LBP state machine for logical units supporting anchored LBAs

If the logical unit supports anchored LBAs (i.e., the ANC_SUP bit is set to one) and deallocated LBAs (i.e., is a thin provisioned logical unit, or a resource provisioned logical unit), then this state machine consists of the following states:

a) LBP1:Mapped state (see 4.7.4.5);
b) LBP2:Deallocated state (see 4.7.4.6) (initial state for thin provisioned logical units); and
c) LBP3:Anchored state (see 4.7.4.7) (initial state for resource provisioned logical units supporting anchored LBAs).

For thin provisioned logical units the initial state of the state machine associated with each LBA is the LBP2:Deallocated state.

For resource provisioned logical units supporting anchored LBAs the initial state of the state machine associated with each LBA is the LBP3:Anchored state.

Figure 9 describes the LBP state machine for a logical unit that supports anchored LBAs and deallocated LBAs.

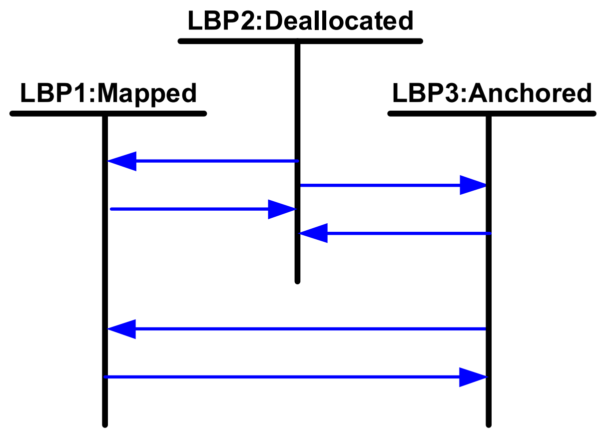


Figure 9 — LBP state machine (anchored LBAs supported and deallocated LBAs supported)

4.7.4.3 LBP state machine for logical units not supporting anchored LBAs

If the logical unit does not support anchored LBAs (i.e., is a thin provisioned logical unit and the ANC_SUP bit is set to zero or a resource provisioned logical unit and the ANC_SUP bit is set to zero), then this state machine consists of the following states:

a) LBP1:Mapped state(see 4.7.4.5); and
b) LBP2:Deallocated state (see 4.7.4.6).

The initial state of the state machine associated with each LBA is the LBP2:Deallocated state.

Figure 10 describes the LBP state machine for a logical unit that does not support anchored LBAs.

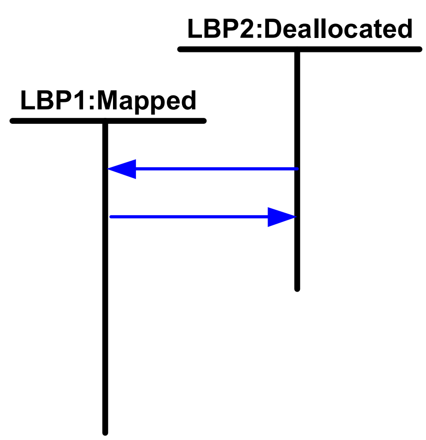


Figure 10 — LBP state machine (anchored LBAs not supported)

4.7.4.4 Performing read operations with respect to logical block provisioning

Table 9 defines the logical block data that a read operation shall return for a mapped LBA.

Table 9 — Logical block data returned by a read operation from a mapped LBA

Condition	Logical block data returned
The LBA became mapped as the result of a format operation or sanitize operation and no write command has specified that LBA since the LBA became mapped.	The logical block data that was written to that LBA by the format operation or the sanitize operation.
The LBA became mapped as the result of a write command and no additional write command has specified that LBA since the LBA was mapped.	The logical block data that was written to that LBA by that write command.
The LBA became mapped as the result of an autonomous transition, and no write command has specified that LBA since the LBA was mapped.	The logical block data returned is the same as if that autonomous transition had not occurred and the LBA had remained unmapped (see table 10).
A write command has specified that LBA since that LBA was mapped.	The logical block data that was most recently written to that LBA.

Table 10 defines the logical block data that a read operation shall return for an unmapped LBA.

Table 10 — Logical block data returned by a read operation from an unmapped LBA

LBPRZ field [a]	Method used to unmap the LBA	Logical block data returned
000b	see [b]	The value specified to be written if the write operation had been performed.
	see [c]	Logical block data containing: a) user data set to a vendor-specific value that is not obtained from any other LBA; and b) protection information, if any, set to FFFF_FFFF_FFFF_FFFFh.
xx1b	Any	Logical block data containing: a) user data set to zero; and b) protection information, if any, with the: A) LOGICAL BLOCK GUARD field set to FFFFh or set to 0000h (i.e., the valid CRC for user data in which all bits are set to zero); B) LOGICAL BLOCK APPLICATION TAG field set to FFFFh; and C) LOGICAL BLOCK REFERENCE TAG field set to FFFF_FFFFh.
010b	Any	Logical block data containing: a) user data set to the provisioning initialization pattern; and b) protection information, if any, with the: A) LOGICAL BLOCK GUARD field set to FFFFh or set to the CRC for the provisioning initialization pattern; B) LOGICAL BLOCK APPLICATION TAG field set to FFFFh; and C) LOGICAL BLOCK REFERENCE TAG field set to FFFF_FFFFh.

[a] The LBPRZ field is in the Logical Block Provisioning VPD page (see 6.6.9).

[b] A command for which the device server is allowed to perform either:

a) a write with specified logical block data; or
b) an unmap operation if the logical block data returned by read operations from unmapped LBAs matches the logical block data specified for the command that resulted in the unmap operation.

These commands are:

a) a FORMAT UNIT command specifying an initialization pattern;
b) a SANITIZE command specifying a sanitize overwrite operation; and
c) a WRITE SAME command with the UNMAP bit set to one.

[c] These methods include but are not limited to:

a) a FORMAT UNIT command not specifying an initialization pattern;
b) a REASSIGN BLOCKS command;
c) a SANITIZE command specifying a sanitize block erase operation or a sanitize cryptographic erase operation; and
d) an UNMAP command.

After a read operation returns a value for an LBA, subsequent read operations from that LBA shall return the same value until a subsequent command alters the logical block data in that LBA (e.g., a write command or an unmap command (see table 34)).

4.7.4.5 LBP1:Mapped state

4.7.4.5.1 LBP1:Mapped state description

Upon entry into this state, the relationship between the LBA and the physical block(s) that contains the logical block for that LBA shall be established.

If this state was entered from the LBP2:Deallocated state (see 4.7.4.6), then the device server shall allocate LBA mapping resources, if any, required to map the LBA.

If this state was entered from the LBP3:Anchored state (see 4.7.4.7), then:

a) the device server shall not allocate LBA mapping resources; and
b) the resource exhaustion conditions described in 4.7.3.6.1 shall not occur.

4.7.4.5.2 Transition LBP1:Mapped to LBP2:Deallocated

This transition shall occur after:

a) an unmap operation that results in the deallocation of the LBA that was mapped.

This transition may occur at any time if the LBPRZ field (see 6.6.9) is set to:

a) xx1b, and the mapped LBA references a logical block that contains:
   A) user data with all bits set to zero; and
   B) protection information, if any, with the:
      a) LOGICAL BLOCK GUARD field set to FFFFh or set to 0000h;
      b) LOGICAL BLOCK APPLICATION TAG field set to FFFFh; and
      c) LOGICAL BLOCK REFERENCE TAG field set to FFFF_FFFFh;

   or
b) 010b, and a mapped LBA references a logical block that contains;
   A) user data set to the provisioning initialization pattern; and
   B) protection information, if any,with the:
      a) LOGICAL BLOCK GUARD field set to FFFFh or set to the CRC for the provisioning initialization pattern;
      b) LOGICAL BLOCK APPLICATION TAG field set to FFFFh; and
      c) LOGICAL BLOCK REFERENCE TAG field set to FFFF_FFFFh.

4.7.4.5.3 Transition LBP1:Mapped to LBP3:Anchored

This transition shall occur after:

a) an unmap operation that results in the anchoring of the LBA that was mapped.

This transition may occur at any time if the LBPRZ field (see 6.6.9) is set to:

a) xx1b, and the mapped LBA references a logical block that contains:
   A) user data with all bits set to zero; and
   B) protection information, if any, with the:
      a) LOGICAL BLOCK GUARD field set to FFFFh or set to 0000h;
      b) LOGICAL BLOCK APPLICATION TAG field set to FFFFh; and
      c) LOGICAL BLOCK REFERENCE TAG field set to FFFF_FFFFh;

   or
b) 010b, and a mapped LBA references a logical block that contains;
   A) user data set to the provisioning initialization pattern; and
   B) protection information, if any, with the:
      a) LOGICAL BLOCK GUARD field set to FFFFh or set to the CRC for the provisioning initialization pattern;
      b) LOGICAL BLOCK APPLICATION TAG field set to FFFFh; and
      c) LOGICAL BLOCK REFERENCE TAG field set to FFFF_FFFFh.

4.7.4.6 LBP2:Deallocated state

4.7.4.6.1 LBP2:Deallocated state description

While in this state:

a) there shall be no relationship between the LBA and any physical block(s); and

b) the device server should deallocate LBA mapping resources after they are no longer in use.

For an LBA in this state, an unmap operation that specifies deallocation of the LBA shall not cause a transition from this state.

The device server shall process a read command specifying an LBA in this state (i.e., a deallocated LBA) as described in table 10.

4.7.4.6.2 Transition LBP2:Deallocated to LBP1:Mapped

This transition:

a) shall occur after a write operation to the LBA that was deallocated; or
b) may occur at any time for reasons not defined by this standard.

4.7.4.6.3 Transition LBP2:Deallocated to LBP3:Anchored

This transition:

a) shall occur after an unmap operation that results in anchoring of the LBA that was deallocated; or
b) may occur at any time for reasons not defined by this standard.

4.7.4.7 LBP3:Anchored state

4.7.4.7.1 LBP3:Anchored state description

Upon entry into this state:

a) LBA mapping resources shall be associated with the LBA; and
b) there may or may not be a relationship between the LBA and physical block(s).

If this state was entered from the LBP2:Deallocated state, then the device server shall allocate LBA mapping resources, if any, required to anchor the LBA.

If this state was entered from the LBP1:Mapped state, then:

a) the device server shall not allocate LBA mapping resources;
b) the device server relies on LBA mapping resources already allocated to the LBA; and
c) the resource exhaustion conditions described in 4.7.3.6.1 shall not occur.

For an LBA in this state, an unmap operation that specifies anchoring of the LBA shall not cause a transition from of this state.

The device server shall process a read command specifying an LBA in this state (i.e., an anchored LBA) as described in table 10.

4.7.4.7.2 Transition LBP3:Anchored to LBP1:Mapped

This transition:

a) shall occur after a write operation to the LBA that was anchored; or
b) may occur at any time for reasons not defined by this standard.

4.7.4.7.3 Transition LBP3:Anchored to LBP2:Deallocated

This transition shall occur after:

a) an unmap operation that results in the deallocation of the LBA that was anchored.

4.8 Data de-duplication

Data de-duplication is the ability of a device server to recognize redundant or duplicate data and reduce the number of duplicate or redundant copies of the data while maintaining the application client supplied LBAs of the duplicate or redundant copies of the data. De-duplication shall not affect protection information, if any. Logical units that support data de-duplication may report a count of LBA resources that have been made available as a result of being de-duplicated (see 6.4.5.5).

Data de-duplication may impact the number of LBA Mapping resources indicated in the Logical Block Provisioning log page (see 6.4.5) (e.g., the value indicated in the Available LBA Mapping Resource Count log parameter (see 6.4.5.2) and the Used LBA Mapping Resource Count log parameter (see 6.4.5.3) may not change as a result of successful write operations that contain data that is de-duplicated or successful unmap operations on LBAs that contain data that has been de-duplicated).

4.9 Ready state

A direct access block device is ready when the device server is capable of processing logical block access commands that require access to the medium (see 4.2.2).

A direct access block device using a removable medium (see 4.4) is not ready until a volume is mounted and other conditions are met (see 4.3). While a direct access block device is not ready the device server shall terminate logical block access commands with CHECK CONDITION status with the sense key set to NOT READY and the appropriate additional sense code for the condition.

Some direct access block devices may be switched from being ready to being not ready by using the START STOP UNIT command (see 5.31).

To make a direct access block device ready, an application client may be required to issue a START STOP UNIT command:

a) with a START bit set to one and the POWER CONDITION field set to 0h (i.e., START_VALID); or
b) with the POWER CONDITION field set to 1h (i.e., ACTIVE).

4.10 Initialization

A direct access block device may require initialization of its medium prior to processing logical block access commands. This initialization is requested by an application client using a FORMAT UNIT command (see 5.4). Parameters related to the format (e.g., logical block length) may be set with a MODE SELECT command (see SPC-6 and 6.5.2) prior to the format operation. Some direct access block devices are initialized by means not defined by this standard. The time when the initialization occurs is vendor specific.

Direct access block devices using a non-volatile medium may save the parameters related to the format and only require initialization once. However, some mode parameters may require initialization after each logical unit reset. A catastrophic failure of the direct access block device may require that an application client send a FORMAT UNIT command to recover from the failure.

Direct access block devices that use a volatile medium may require initialization after each logical unit reset prior to the processing of logical block access commands (see 4.2.2). Mode parameters may also require initialization after logical unit resets.

NOTE 2 - It is possible that mode parameter block descriptors read with a MODE SENSE command before a FORMAT UNIT completes contain information not reflecting the true state of the medium.

A direct access block device may become format corrupt after processing a MODE SELECT command that changes parameters (e.g., the logical block length) related to the medium format. During this time, the device server may terminate logical block access commands with CHECK CONDITION status with the sense key set to NOT READY and the appropriate additional sense code for the condition.

Any time the READ CAPACITY (10) parameter data (see 5.20.2) or the READ CAPACITY (16) parameter data (see 5.21.2) changes (e.g., when a FORMAT UNIT command or a MODE SELECT command causes a change to the logical block length or protection information, or when a vendor specific mechanism causes a change), then the device server shall establish a unit attention condition for the SCSI initiator port (see SAM-6) associated with each I_T nexus, except the I_T nexus on which the command causing the change was received with the additional sense code set to CAPACITY DATA HAS CHANGED.

NOTE 3 - Logical units compliant with SBC were not required to establish a unit attention condition with the additional sense code set to CAPACITY DATA HAS CHANGED.

4.11 Sanitize operations

4.11.1 Sanitize operations overview

A sanitize operation causes the device server to:

a) affect the information on the logical unit's medium such that recovery of logical block data from the medium is not possible;
b) affect the information in the cache by a method that is not defined by this standard such that previously existing data in cache is unable to be accessed; and
c) prevent future access by an application client to cache or medium where the device server is unable to alter the information.

One of the following sanitize operations may be requested on the logical unit's medium:

a) a sanitize overwrite operation that causes the device server to alter information by writing a data pattern to the medium one or more times;
b) a sanitize block erase operation that causes the device server to alter information by setting the physical blocks to a vendor specific value; or
c) a sanitize cryptographic erase operation that causes the device server to change encryption keys to prevent correct decryption of previously stored information, which may cause protection information, if any, to be indeterminate.

For zoned block devices:

a) the ZNR bit (see 5.30) requests specific processing for each write pointer zone (see ZBC-2) upon successful completion of a sanitize operation (see 4.11.4); and
b) ZBC-2 describes additional requirements to the way in which sanitize operations are performed and completed (i.e., extensions to the descriptions in 4.11.3).

An application client may request that a sanitize operation be performed in the restricted completion mode or the unrestricted completion mode (see 4.11.4) using the AUSE bit (see 5.30).

For both completion modes, the failure of a sanitize operation causes the device server to enter a failed sanitize condition that lasts until the processing described in 4.11.4 is completed. While in the failed sanitize condition, the device server terminates all commands, except SANITIZE commands and commands allowed during a sanitize operation (see 4.11.2) with CHECK CONDITION status with the sense key set to MEDIUM ERROR and the additional sense code set to SANITIZE COMMAND FAILED.

While in a sanitize failure condition in the restricted completion mode:

a) if a SANITIZE command requests the unrestricted completion mode (i.e., the AUSE bit set to one), then the device server shall terminate that SANITIZE command as described in 5.30.1; and
b) the steps that exit that failed sanitize condition are:
   1) the application client sends a SANITIZE command to request another sanitize operation; and
   2) that sanitize operation completes without error to cause the device server to exit the failed sanitize condition.

While in a sanitize failure condition in the unrestricted completion mode:

a) a SANITIZE command may request an additional sanitize operation with that sanitize operation:
   A) staying in unrestricted completion mode; or
   B) changing to restricted completion mode;

and

a SANITIZE command with the EXIT FAILURE MODE service action causes the device server to exit the failed sanitize condition as described in 4.11.4.All sanitize operations shall be performed on:

a) the medium that is being used to store logical block data;
b) the medium that is not being used to store logical block data (e.g., areas previously used to store logical block data, areas available for allocation, and physical blocks that have become inaccessible); and
c) all cache.

If a sanitize operation is unable to sanitize a storage element that is depopulated, the device shall set the Restoration Allowed attribute (see 4.36.2) to false for that storage element.

An application client requests that the device server perform a sanitize operation using the SANITIZE command. While the medium is write protected (see 4.12) the device server shall terminate a SANITIZE command with CHECK CONDITION status with the sense key set to DATA PROTECT and the appropriate additional sense code for the condition.

4.11.2 Commands allowed during a sanitize operation

Those of the following commands that the device server supports while not performing a sanitize operation are allowed during a sanitize operation:

a) INQUIRY commands;
b) LOG SENSE commands that specify the Temperature log page (see SPC-6);
c) MODE SENSE commands (see SPC-6) that specify:
   A) the Informational Exceptions Control mode page;
   B) the Caching mode page;
   C) the Control mode page;
   D) the Protocol Specific Port mode page; or
   E) the Protocol Specific Logical Unit mode page;
d) READ CAPACITY (10) commands (see 5.20);
e) READ CAPACITY (16) commands (see 5.21);
f) REPORT LUNS commands (see SPC-6);
g) REPORT SUPPORTED OPERATION CODES commands (see SPC-6);
h) REPORT SUPPORTED TASK MANAGEMENT FUNCTIONS commands (see SPC-6);
i) REPORT ZONES commands (see ZBC-2) with:
   A) the ZONE START LBA field set to zero;
   B) the REPORTING OPTIONS field set to 3Fh;
   C) the PARTIAL bit set to one; and
   D) the ALLOCATION LENGTH field set to a value less than or equal to 64;

   and

j) REQUEST SENSE commands.

4.11.3 Performing a sanitize operation

Before performing a sanitize operation, the device server shall:

a) terminate all commands in all task sets except commands allowed during a sanitize operation (see 4.11.2) with CHECK CONDITION status with the sense key set to NOT READY, the additional sense code set to LOGICAL UNIT NOT READY, SANITIZE IN PROGRESS, and the PROGRESS INDICATION field in the sense data set to indicate that the sanitize operation is beginning;
b) stop all enabled power condition timers (see SPC-6);

c) stop all timers for enabled background scan operations (see 4.23);
d) stop all timers or counters enabled for device specific background functions (see SPC-6);
e) discard partially downloaded microcode, if any; and
f) close open streams (see 4.32), if any.

While performing a sanitize operation, the device server shall:

a) process commands allowed during a sanitize operation (see 4.11.2) and terminate all other commands with CHECK CONDITION status with the sense key set to NOT READY, the additional sense code set to LOGICAL UNIT NOT READY, SANITIZE IN PROGRESS, and the PROGRESS INDICATION field in the sense data set to indicate the progress of the sanitize operation;
b) provide pollable sense data (see SPC-6) with the sense key set to NOT READY, the additional sense code set to LOGICAL UNIT NOT READY, SANITIZE IN PROGRESS, and the PROGRESS INDICATION field set to indicate the progress of the sanitize operation;
c) suspend the sanitize operation while processing the following conditions (see SAM-6):
   A) a power on;
   B) a hard reset;
   C) a logical unit reset; or
   D) a power loss expected;
d) not suspend the sanitize operation while processing an I_T nexus loss;
e) resume performing the sanitize operation after processing:
   A) a logical unit reset; or
   B) a power loss expected condition in which no power loss occurs within constraints defined by the applicable SCSI transport protocol standard (e.g., power loss timeout in SPL-5);
f) process task management functions without affecting the processing of the sanitize operation (e.g., an ABORT TASK task management function aborts the SANITIZE command and has no effect on performing the sanitize operation);
g) not alter mode data, INQUIRY data, or READ CAPACITY (16) parameter data (e.g., the number of logical blocks, logical block length, or protection information settings for the logical unit); and
h) identify inaccessible physical blocks and in a vendor specific manner prevent future access to these blocks following a successful sanitize operation.

4.11.4 Completing a sanitize operation

If a sanitize operation completes without error, and logical block provisioning management (see 4.7.3) is supported, then:

a) the initial condition for every LBA should be anchored (see 4.7.3.2) or deallocated (see 4.7.3.3); and
b) read operations and write operations should complete without error.

If a sanitize operation completes without error and logical block provisioning management is not supported, then:

a) read commands are processed as described in 5.30.2.2, 5.30.2.3, 5.30.2.4, and 5.30.2.5; and
b) write operations should complete without error.

If a sanitize operation completes without error on a zoned block device, then the ZNR bit (see 5.30) requests specific processing for each write pointer zone (see ZBC-2).

If the sanitize operation completes with an error in restricted completion mode, then the device server shall:

1) terminate the SANITIZE command being performed, if any (e.g., the IMMED bit was set to zero in the CDB, and the failure occurs before status is returned for the command), with CHECK CONDITION status with the sense key set to MEDIUM ERROR and the additional sense code set to SANITIZE COMMAND FAILED; and
2) enter the failed sanitize condition as described in 4.11.1.

If a sanitize operation completes with an error in unrestricted completion mode:

1) if the logical unit is a zoned block device and the ZNR bit in the CDB is set to:
   A) one then, for any write pointer zone where the write pointer was not reset as part of that sanitize operation, the device server shall not modify the write pointer (see ZBC-2); and

   B) zero then the device server shall:
      a) perform the equivalent of a RESET WRITE POINTER command (see ZBC-2) with the ALL bit set to one; and
      b) for each write pointer zone, if the reset write pointer operation is not successful, then set the Zone Condition (see ZBC-2) to OFFLINE;
2) the device server shall terminate the SANITIZE command being performed, if any (e.g., the IMMED bit was set to zero in the CDB, and the failure occurs before status is returned for the command), with CHECK CONDITION status with the sense key set to MEDIUM ERROR and the additional sense code set to SANITIZE COMMAND FAILED; and
3) the device server shall enter the failed sanitize condition as described in 4.11.1.

A sanitize operation that completed with error and was cleared with a SANITIZE command with the service action of EXIT FAILURE MODE may have not performed a complete sanitize operation (e.g., this action may enable the recovery of logical block data from the cache and medium for those logical blocks that were not sanitized).

After the sanitize operation completes the device server shall:

a) initialize and start all enabled timers and counters except power condition timers; and
b) initialize and start all operational (see SPC-6) power condition timers.

4.12 Write protection

Write protection prevents the alteration of the medium by logical block access commands issued to the device server. Write protection is controlled by:

a) the user of the medium through manual intervention (e.g., a mechanical lock on the SCSI target device);
b) hardware controls (e.g., tabs on the medium's housing); or
c) software write protection.

All sources of write protection are independent. If present, any write protection shall cause otherwise valid logical block access commands that request alteration of the medium to be terminated by the device server with CHECK CONDITION status with the sense key set to DATA PROTECT and the appropriate additional sense code for the condition. Only when all write protections are disabled shall the device server process logical block access commands that request alteration of the medium.

Hardware write protection results when a physical attribute of the SCSI target device or its medium is changed to specify that writing shall be prohibited. Changing the state of the hardware write protection requires physical intervention, either with the SCSI target device or its medium. If allowed by the SCSI target device, then changing the hardware write protection while the medium is mounted results in vendor specific behavior that may include the writing of previously buffered data (e.g., data in cache).

Software write protection results when the device server is marked as write protected by the application client using the SWP bit in the Control mode page (see SPC-6). Changing the state of software write protection shall not prevent previously accepted logical block data (e.g., logical block data in cache) from being written to the medium.

The device server reports the status of write protection in the device server and on the medium with the DEVICE-SPECIFIC PARAMETER field (see 6.5.1).

4.13 Medium defects

4.13.1 Medium defects overview

Any medium has the potential for medium defects that cause data to be lost. Therefore, physical blocks and/or logical blocks may contain additional information that allows the detection of changes to the logical block data caused by medium defect or other phenomena. The additional information may also allow the logical block data to be reconstructed following the detection of such a change (e.g., ECC bytes).

A medium defect causes:

a) a recovered error if the device server is able to read or write a logical block within the logical unit's recovery limits; or
b) an unrecovered error if the device server is unable to read or write a logical block within the logical unit's recovery limits,

where the logical unit's recovery limits are:

a) specified in the Read-Write Error Recovery mode page (see 6.5.10);
b) specified in the Verify Error Recovery mode page (see 6.5.11); or
c) vendor specific, if the device server does not implement the Read-Write Error Recovery mode page or the Verify Error Recovery mode page.

Direct access block devices may allow an application client to use the features of the WRITE LONG commands (see 5.50 and 5.51) to create a pseudo uncorrectable error. Processing and clearing pseudo uncorrectable errors is described in 4.18.2.

The device server maintains the defect lists shown in table 11.

Table 11 — Defect lists (i.e., PLIST and GLIST)

<table>
<tr><th>Defect list</th><th>Source</th><th>Content</th></tr>
<tr><td>PLIST
(i.e., primary defect list)</td><td>Manufacturer</td><td>Address descriptors (see 6.2) for physical blocks that contain permanent medium defects and never contain logical block data</td></tr>
<tr><td rowspan="5">GLIST
(i.e., grown defect list)</td><td rowspan="2">FORMAT UNIT commands (see 5.4)</td><td>Address descriptors for physical blocks detected by the device server to have medium defects during an optional certification process performed during a format operation</td></tr>
<tr><td>Address descriptors for physical blocks specified in the FORMAT UNIT parameter list (see 5.4.2)</td></tr>
<tr><td>REASSIGN BLOCKS commands (see 5.24)</td><td>Address descriptors for physical blocks referenced by the LBAs specified in the reassign LBA list (see 5.24.2)</td></tr>
<tr><td>Read medium operations</td><td>Address descriptors for physical blocks that have been reassigned as the result of automatic read reassignment</td></tr>
<tr><td>Write medium operations</td><td>Address descriptors for physical blocks that have been reassigned as the result of automatic write reassignment</td></tr>
</table>

The READ DEFECT DATA commands (see 5.22 and 5.23) allow an application client to request that the device server return the PLIST and/or the GLIST.

The FORMAT UNIT command allows an application client to request that the device server clear the GLIST.

During a format operation, the device server shall not assign LBAs to any physical block in:

a) the PLIST, if the PLIST is specified to be used; or
b) the GLIST, if the GLIST is specified to be used.

A device server performs automatic reassignment of defects as specified by the settings in the Read-Write Error Recovery mode page (see 6.5.10).

The device server does not perform automatic read reassignment for an LBA referencing a logical block on which an unrecovered error has occurred. If the application client is notified by the device server that an unrecovered error occurred (e.g., as indicated by a read command being terminated with CHECK

CONDITION status with the sense key set to MEDIUM ERROR and the additional sense code set to UNRECOVERED READ ERROR) and:

a) the application client is able to regenerate the logical block data for the LBA (e.g., in a redundancy group, the application client regenerates logical block data from the logical block data on the other logical units in the redundancy group) and the AWRE bit is set to one in the Read-Write Error Recovery mode page, then the application client may send a write command with that regenerated logical block data to trigger automatic write reassignment;
b) the application client is able to regenerate the logical block data for the LBA and the AWRE bit is set to zero in the Read-Write Error Recovery mode page, then the application client may:
    1) send a REASSIGN BLOCKS command to perform a reassign operation on the LBA; and
    2) send a write command with that regenerated logical block data;

    or

c) the application client is unable to regenerate the logical block data for the LBA, then the application client may send a REASSIGN BLOCKS command to request that the device server perform a reassign operation on the LBA.

4.13.2 Generation of defect lists

This standard defines address descriptor formats for describing defects (see 6.2). Table 12 lists the defects that each address descriptor format is capable of describing.

Table 12 — Address descriptor formats

Format	Single physical block	Multiple sequential physical blocks		Reference
		Entire track	Range	
Short block format	yes	no	no	6.2.2
Extended bytes from index format	yes [a]	yes [e]	yes [i]	6.2.3
Extended physical sector format	yes [b]	yes [f]	yes [i]	6.2.4
Long block format	yes	no	no	6.2.5
Bytes from index format	yes [c]	yes [g]	no	6.2.6
Physical sector format	yes [d]	yes [h]	no	6.2.7

[a] Describes a single physical block with the MADS bit set to zero and the BYTES FROM INDEX field set to a value other than FFF_FFFFh.
[b] Describes a single physical block with the MADS bit set to zero and the SECTOR NUMBER field set to a value other than FFF_FFFFh.
[c] Describes a single physical block with the BYTES FROM INDEX field set to a value other than FFFF_FFFFh.
[d] Describes a single physical block with the SECTOR NUMBER field set to a value other than FFFF_FFFFh.
[e] Describes an entire track with the BYTES FROM INDEX field set to FFF_FFFFh.
[f] Describes an entire track with the SECTOR NUMBER field set to FFF_FFFFh.
[g] Describes an entire track with the BYTES FROM INDEX field set to FFFF_FFFFh.
[h] Describes an entire track with the SECTOR NUMBER field set to FFFF_FFFFh.
[i] Describes a range with a pair of address descriptors using the same address descriptor format in which:
a) the first address descriptor describes the starting location and has the MADS bit set to one;
b) the second address descriptor describes the ending location and has the MADS bit set to zero; and
c) the ending location is after the starting location.

For a direct access block device using rotating media (see 4.3.2), to represent two or more sequential physical blocks on the same track using a pair of address descriptors:

a) the MADS bit shall be set to one in the first address descriptor;
b) the MADS bit shall be set to zero in the second address descriptor;
c) the CYLINDER NUMBER field in the first address descriptor shall be equal to the CYLINDER NUMBER field in the second address descriptor;
d) the HEAD NUMBER field in the first address descriptor shall be equal to the HEAD NUMBER field in the second address descriptor;
e) for a pair of extended bytes from index format address descriptors, the BYTES FROM INDEX field in the first address descriptor shall be less than the BYTES FROM INDEX field in the second address descriptor; and
f) for a pair of extended physical sector format address descriptors, the SECTOR NUMBER field in the first address descriptor shall be less than the SECTOR NUMBER field in the second address descriptor.

For a direct access block device using rotating media, to represent two or more sequential tracks on the same head using a pair of address descriptors:

a) the MADS bit shall be set to one in the first address descriptor;
b) the MADS bit shall be set to zero in the second address descriptor;

c) the CYLINDER NUMBER field in the first address descriptor shall be less than the CYLINDER NUMBER field in the second address descriptor;
d) the HEAD NUMBER field in the first address descriptor shall be equal to the HEAD NUMBER field in the second address descriptor;
e) for a pair of extended bytes from index format address descriptors, the BYTES FROM INDEX field in the first address descriptor and the second address descriptor shall be equal to FFF_FFFFh; and
f) for a pair of extended physical sector format address descriptors, the SECTOR NUMBER field in the first address descriptor and the second address descriptor shall be equal to FFF_FFFFh.

4.14 Write and unmap failures

If any write command that is not an atomic write command, does not complete successfully (e.g., the command completed with CHECK CONDITION status, or the command was being processed at the time of a power loss or an incorrect demount of a removable medium), then any data in the logical blocks referenced by the LBAs specified by that command is indeterminate. Before sending a read command or verify command specifying any LBAs that were specified by a write command that did not complete successfully, the application client should resend that write command. If an application client sends a read command or verify command specifying any LBAs that were specified by a write command that did not complete successfully before resending that write command, then the device server may return old data, new data, vendor-specific data, or any combination thereof for the logical blocks referenced by the specified LBAs.

If logical block provisioning (see 4.7) is supported and one or more unmap commands have not completed when a power loss, medium error, or hardware error occurs, then the logical block provisioning state of the LBAs requested to be unmapped by any of those commands may or may not have changed. The application client should resend that unmap command.

4.15 Caches

4.15.1 Caches overview

Direct access block devices may implement caches. A cache is an area of temporary storage in the direct access block device (e.g., to enhance performance) separate from the medium that is not directly accessible by the application client.

A cache stores logical block data.

A cache may be volatile or non-volatile. A volatile cache does not retain data through power cycles. A non-volatile cache retains data through power cycles. There may be a limit on the amount of time a non-volatile cache is able to retain data without power (see 4.15.10).

4.15.2 Cache segments

The cache may be divided into cache segments. The device server may allow application client control over cache segments using the following in the Caching mode page (see 6.5.6):

a) the IC bit;
b) the NUMBER OF CACHE SEGMENTS field;
c) the CACHE SEGMENT SIZE field; and
d) the SIZE bit.

Cache segments may be grouped where each group is identified by a cache ID. An application client may request that a specific group of cache segments be used for an IO through the use of IO advice hints (see 4.30). The GROUP NUMBER field in each command specifies the cache ID of the cache segments to use during the processing of that command.The algorithms used to manage the data in each cache segment is not defined by this standard. How cache segments are associated with cache IDs is not defined by this standard.

4.15.3 Read caching

While processing read commands and verify commands, the device server may use the cache to store logical blocks that the application client may request at some future time. The algorithm used to manage the cache is not part of this standard. However, parameters are provided (see 6.5.6) to advise the device server about future requests, or to restrict the use of cache for a particular request.

4.15.4 Write caching

While processing write commands, the device server may perform a write cache operation to store logical block data that is to be written to the medium at a later time with a write medium operation. This is called writeback caching. A write command may complete prior to logical blocks being written to the medium. As a result of using writeback caching there is a period of time during which the logical block data may be lost if:

a) power to the SCSI target device is lost and a volatile cache is being used; or
b) a hardware failure occurs.

There is also the possibility of an error occurring during the subsequent write medium operation. If an error occurs during the write medium operation, then the error may be reported as a deferred error on a later command. The application client may request that writeback caching be disabled with the Caching mode page (see 6.5.6) to prevent detected write errors from being reported as deferred errors. Even with writeback caching disabled, undetected write errors may occur. Verify commands (e.g., VERIFY and WRITE AND VERIFY) may be used to detect those errors.

If processing a write command results in logical block data in cache that is different from the logical block data on the medium, then the device server shall retain that logical block data in cache until a write medium operation is performed using that logical block data. After the write medium operation is complete, the device server may retain that logical block data in cache.

4.15.5 Command interactions with caches

The application client may affect behavior of the cache with:

a) the PRE-FETCH commands (see 5.13 and 5.14);
b) the SYNCHRONIZE CACHE commands (see 5.33 and 5.34); and
c) the Caching mode page (see 6.5.6).

When the cache becomes full of logical block data, the device server may replace the logical block data in the cache with new logical block data. The disable page out (DPO) bit in the CDBs of read commands, verify commands, and write commands allows the application client to influence the replacement of logical block data in the cache. A read command, verify command, or a write command with a DPO bit set to one is a hint to the device server that the logical blocks specified by that command are not likely to be accessed in the near future and should not be put in the cache or retained by the cache.

Application clients may use the force unit access (FUA) bit in the CDBs of read commands (e.g., see 5.16) or write commands (e.g., see 5.40) to specify that the device server shall access:

a) the medium;
b) non-volatile cache; or
c) the specified data pattern for that LBA (e.g., the data pattern for unmapped data (see 4.7.4.4)).

Setting the DPO bit to one (e.g., see 5.16) and the FUA bit to one in all read commands and all write commands has the same effect as bypassing the volatile cache.

4.15.6 Write operation and write medium operation interactions with caches

For each LBA accessed by a write operation:

1) if a cache contains more recent logical block data for that LBA than the medium, then the device server shall:
   A) perform a write cache operation to that LBA to update the logical block data in the cache;

B) invalidate that LBA in the cache and perform a write medium operation to that LBA; or
C) perform a write cache operation to that LBA to update the logical block data in the cache and perform a write medium operation to that LBA.

For each LBA accessed by a write medium operation that is not part of a write operation:

1) if a cache contains more recent logical block data for that LBA than the medium, then the device server shall:
   A) perform a write cache operation to that LBA to update the logical block data in the cache; or
   B) invalidate that LBA in the cache, before the device server performs the write medium operation to that LBA.

4.15.7 Read operation and read medium operation interactions with caches

For each LBA accessed by a read operation:

1) if a cache contains more recent logical block data for that LBA than the medium, then the device server shall perform a read cache operation from that LBA; or
2) the device server shall perform a read medium operation from that LBA.

For each LBA accessed by a read medium operation that is not part of a read operation:

1) if a cache contains more recent logical block data for the LBA than the medium, then the device server shall perform a write medium operation to that LBA; and
2) the device server may invalidate that LBA in the cache, before the device server performs the read medium operation from that LBA.

4.15.8 Verify medium operation interactions with caches

For each LBA accessed by a verify medium operation:

1) if a cache contains more recent logical block data for the LBA than the medium, then the device server shall perform a write medium operation to that LBA;
2) the device server may invalidate that LBA in the cache; and
3) before the device server performs the verify medium operation from that LBA.

4.15.9 Unmap operation interactions with caches

During an unmap operation, the device server changes any logical block data in the cache for the LBA unmapped by the operation so that any logical block data transferred by the device server to the Data-In Buffer while processing a subsequent read command reflects the results of the unmap operation (see 4.7.4.6.1 and 4.7.4.7.1).

4.15.10 Power loss effects on caches

The power, if any, needed to maintain a non-volatile cache may decrease to the point that the device server is unable to ensure the non-volatility of the cache for a vendor specific interval of time (e.g., the battery voltage becomes too low too sustain cache contents beyond a vendor specific time). If this occurs and the Extended INQUIRY Data VPD page (see SPC-6) indicates that the device server contains non-volatile cache (i.e., NV_SUP bit set to one), then:

a) if the reporting of informational exceptions control warnings is enabled (i.e., the EWASC bit is set to one in the Information Exceptions Control mode page (see 6.5.8)), then the device server shall report the degraded non-volatile cache as specified in the Information Exceptions Control mode page with an additional sense code set to WARNING - DEGRADED POWER TO NON-VOLATILE CACHE; or
b) if the reporting of informational exceptions control warnings is disabled (i.e., the EWASC bit is set to zero in the Information Exceptions Control mode page), then the device server shall establish a unit attention condition (see SAM-6) for the SCSI initiator port associated with every I_T nexus with the additional sense code set to WARNING - DEGRADED POWER TO NON-VOLATILE CACHE.

Non-volatile caches may become volatile (e.g., battery voltage becomes too low to sustain cache contents when power is lost). If non-volatile caches become volatile, then logical block data transferred for read commands or write commands in which the force unit access (FUA) bit in the CDB is set to one may bypass the cache.

If a non-volatile cache becomes volatile, then the device server shall set the REMAINING NON-VOLATILE TIME field to zero in the Non-volatile Cache log page (see 6.4.7).

If non-volatile cache becomes volatile and the Extended INQUIRY Data VPD page (see SPC-6) indicates that the device server contains non-volatile cache (i.e., the NV_SUP bit is set to one), then:

a) if the reporting of informational exceptions control warnings is enabled (i.e., the EWASC bit is set to one in the Information Exceptions Control mode page (see 6.5.8)), then the device server shall report the change in the cache as specified in the Information Exceptions Control mode page with the additional sense code set to WARNING - NON-VOLATILE CACHE NOW VOLATILE; or
b) if the reporting of informational exceptions control warnings is disabled (i.e., the EWASC bit is set to zero in the Information Exceptions Control mode page), then the device server shall establish a unit attention condition (see SAM-6) for the SCSI initiator port associated with every I_T nexus with the additional sense code set to WARNING - NON-VOLATILE CACHE NOW VOLATILE.

If:

a) a power on or hard reset occurs;
b) the Extended INQUIRY Data VPD page indicates that the device server contains a non-volatile cache (i.e., the NV_SUP bit is set to one); and
c) the non-volatile cache is currently volatile,

then the device server shall establish a unit attention condition for the SCSI initiator port associated with every I_T nexus with the additional sense code set to WARNING - NON-VOLATILE CACHE NOW VOLATILE.

4.16 Implicit head of queue command processing

Each of the following commands defined by this standard may be processed by the task manager as if it has a HEAD OF QUEUE task attribute (see SAM-6), even if the command is received with a SIMPLE task attribute or an ORDERED task attribute:

a) the READ CAPACITY (10) command (see 5.20); and
b) the READ CAPACITY (16) command (see 5.21).

The following command defined by this standard shall be processed by the task manager as if it has a HEAD OF QUEUE task attribute, even if the command is received with a SIMPLE task attribute or an ORDERED task attribute:

a) the SANITIZE command (see 5.30).

See SPC-6 for additional commands subject to implicit HEAD OF QUEUE command processing. See SAM-6 for additional rules on implicit head of queue processing.

4.17 Reservations

Reservation restrictions are placed on commands as a result of access qualifiers associated with the type of reservation. See SPC-6 for a description of reservations. The details of commands that are allowed under what types of reservations are described in table 13.

Commands from I_T nexuses holding a reservation should complete normally. Table 13 specifies the behavior of commands from registered I_T nexuses when a registrants only or all registrants type persistent reservation is present.

For each command, this standard or SPC-6 defines the conditions that result in the device server completing the command with RESERVATION CONFLICT status.

Table 13 — SBC-5 commands that are allowed in the presence of various reservations (part 1 of 3)

Command	Addressed logical unit has this type of persistent reservation held by another I_T nexus				
	From any I_T nexus		From registered I_T nexus (RR all types)	From I_T nexus not registered	
	Write Exclusive	Exclusive Access		Write Exclusive - RR	Exclusive Access - RR
BACKGROUND CONTROL	Conflict	Conflict	Allowed	Conflict	Conflict
COMPARE AND WRITE	Conflict	Conflict	Allowed	Conflict	Conflict
FORMAT UNIT	Conflict	Conflict	Allowed	Conflict	Conflict
FORMAT WITH PRESET	Conflict	Conflict	Allowed	Conflict	Conflict
GET LBA STATUS	Allowed	Conflict	Allowed	Allowed	Conflict
GET PHYSICAL ELEMENT STATUS	Allowed	Conflict	Allowed	Allowed	Conflict
GET STREAM STATUS	Allowed	Conflict	Allowed	Allowed	Conflict
ORWRITE	Conflict	Conflict	Allowed	Conflict	Conflict
POPULATE TOKEN	Allowed	Conflict	Allowed	Allowed	Conflict
PRE-FETCH	Allowed	Conflict	Allowed	Allowed	Conflict

Key:
RR = Registrants Only or All Registrants
Allowed = Commands received from I_T nexuses not holding the reservation or from I_T nexuses not registered when a registrants only or all registrants type persistent reservation is present should complete normally.
Conflict = Commands received from I_T nexuses not holding the reservation or from I_T nexuses not registered when a registrants only or all registrants type persistent reservation is present shall not be performed, and the device server shall complete the command with RESERVATION CONFLICT status.

[a] The device server in logical units claiming compliance with SBC-2 may complete the command with RESERVATION CONFLICT status. Device servers may report whether certain commands are allowed in the PERSISTENT RESERVE IN command REPORT CAPABILITIES service action parameter data ALLOW COMMANDS field (see SPC-6).

Table 13 — SBC-5 commands that are allowed in the presence of various reservations (part 2 of 3)

Command	Addressed logical unit has this type of persistent reservation held by another I_T nexus				
	From any I_T nexus		From registered I_T nexus (RR all types)	From I_T nexus not registered	
	Write Exclusive	Exclusive Access		Write Exclusive - RR	Exclusive Access - RR
PREVENT ALLOW MEDIUM REMOVAL (Prevent=0)	Allowed	Allowed	Allowed	Allowed	Allowed
PREVENT ALLOW MEDIUM REMOVAL (Prevent ≠ 0)	Conflict	Conflict	Allowed	Conflict	Conflict
READ	Allowed	Conflict	Allowed	Allowed	Conflict
READ CAPACITY	Allowed	Allowed	Allowed	Allowed	Allowed
READ DEFECT DATA	Allowed [a]	Conflict	Allowed	Allowed [a]	Conflict
REASSIGN BLOCKS	Conflict	Conflict	Allowed	Conflict	Conflict
REMOVE ELEMENT AND TRUNCATE	Conflict	Conflict	Allowed	Conflict	Conflict
REPORT REFERRALS	Allowed	Allowed	Allowed	Allowed	Allowed
REPORT PROVISIONING INITIALIZATION PATTERN	Allowed	Allowed	Allowed	Allowed	Allowed
RESTORE ELEMENTS AND REBUILD	Conflict	Conflict	Allowed	Conflict	Conflict
SANITIZE	Conflict	Conflict	Allowed	Conflict	Conflict
START STOP UNIT with START bit set to one and POWER CONDITION field set to 0h	Allowed	Allowed	Allowed	Allowed	Allowed
START STOP UNIT with START bit set to zero or POWER CONDITION field set to a value other than 0h	Conflict	Conflict	Allowed	Conflict	Conflict
STREAM CONTROL	Conflict	Conflict	Allowed	Conflict	Conflict
SYNCHRONIZE CACHE	Conflict	Conflict	Allowed	Conflict	Conflict
UNMAP	Conflict	Conflict	Allowed	Conflict	Conflict
VERIFY	Allowed	Conflict	Allowed	Allowed	Conflict
WRITE	Conflict	Conflict	Allowed	Conflict	Conflict

Key:
RR = Registrants Only or All Registrants
Allowed = Commands received from I_T nexuses not holding the reservation or from I_T nexuses not registered when a registrants only or all registrants type persistent reservation is present should complete normally.
Conflict = Commands received from I_T nexuses not holding the reservation or from I_T nexuses not registered when a registrants only or all registrants type persistent reservation is present shall not be performed, and the device server shall complete the command with RESERVATION CONFLICT status.

[a] The device server in logical units claiming compliance with SBC-2 may complete the command with RESERVATION CONFLICT status. Device servers may report whether certain commands are allowed in the PERSISTENT RESERVE IN command REPORT CAPABILITIES service action parameter data ALLOW COMMANDS field (see SPC-6).

Table 13 — SBC-5 commands that are allowed in the presence of various reservations (part 3 of 3)

Command	Addressed logical unit has this type of persistent reservation held by another I_T nexus				
	From any I_T nexus		From registered I_T nexus (RR all types)	From I_T nexus not registered	
	Write Exclusive	Exclusive Access		Write Exclusive - RR	Exclusive Access - RR
WRITE AND VERIFY	Conflict	Conflict	Allowed	Conflict	Conflict
WRITE ATOMIC	Conflict	Conflict	Allowed	Conflict	Conflict
WRITE LONG	Conflict	Conflict	Allowed	Conflict	Conflict
WRITE SAME	Conflict	Conflict	Allowed	Conflict	Conflict
WRITE SCATTERED	Conflict	Conflict	Allowed	Conflict	Conflict
WRITE STREAM	Conflict	Conflict	Allowed	Conflict	Conflict
WRITE USING TOKEN	Conflict	Conflict	Allowed	Conflict	Conflict

Key:
RR = Registrants Only or All Registrants
Allowed = Commands received from I_T nexuses not holding the reservation or from I_T nexuses not registered when a registrants only or all registrants type persistent reservation is present should complete normally.
Conflict = Commands received from I_T nexuses not holding the reservation or from I_T nexuses not registered when a registrants only or all registrants type persistent reservation is present shall not be performed, and the device server shall complete the command with RESERVATION CONFLICT status.

[a] The device server in logical units claiming compliance with SBC-2 may complete the command with RESERVATION CONFLICT status. Device servers may report whether certain commands are allowed in the PERSISTENT RESERVE IN command REPORT CAPABILITIES service action parameter data ALLOW COMMANDS field (see SPC-6).

4.18 Error reporting

4.18.1 Error reporting overview

If any of the conditions listed in table 14 occur during the processing of a command, then the device server shall terminate the command with CHECK CONDITION status with the sense key set to the specified value and the additional sense code set to the appropriate value for the condition. Some errors may occur after the completion status has already been reported. For such errors, SPC-6 defines a deferred error reporting mechanism. Table 14 lists some error conditions and the applicable sense keys. The list does not provide a complete list of all conditions that may cause CHECK CONDITION status.

Table 14 — 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
Logical block provisioning threshold notification	UNIT ATTENTION
Self diagnostic failed	HARDWARE ERROR
Unrecovered error	MEDIUM ERROR or HARDWARE ERROR
Recovered read error	RECOVERED ERROR
Pseudo unrecovered error	MEDIUM ERROR
Over-run or other error that may be resolved by repeating the command	ABORTED COMMAND
Attempt to write on write-protected medium	DATA PROTECT

Direct access block devices compliant with this standard shall support both the fixed and descriptor formats of sense data (see SPC-6).

Table 15 summarizes use of the sense data fields.

Table 15 — Sense data field usage for direct access block devices

Field	Usage	Reference
INFORMATION field [a]		
	REASSIGN BLOCKS command	5.24
	Any command that accesses the medium, based on the Read-Write Error Recovery mode page	4.19.3 and 6.5.10
	Any command that accesses the medium, based on the Verify Error Recovery mode page	6.5.11
	Any command that is terminated with a logical block provisioning threshold notification	4.7.3.7.6
	COMPARE AND WRITE command	5.3
COMMAND-SPECIFIC INFORMATION field	EXTENDED COPY command	SPC-6
	REASSIGN BLOCKS command	5.24
	WRITE SCATTERED (16) command	4.35.3 and 5.55
	WRITE SCATTERED (32) command	4.35.3 and 5.56
	If rebuild assist mode is enabled (see 4.19), then any command that accesses the medium, based on the Read-Write Error Recovery mode page [b]	4.19.3

[a] See SPC-6 for a description of how the VALID bit interacts with the INFORMATION field.
[b] If fixed format sense data is used but the value to be placed in the COMMAND-SPECIFIC INFORMATION field is greater than FFFF_FFFFh (e.g., an 8-byte LBA), then the device server shall report no value in the INFORMATION field (see SPC-6) and shall report no value in the COMMAND-SPECIFIC INFORMATION field (see SPC-6).

If a command attempts to access or reference an invalid LBA, then the device server shall report the first invalid LBA (e.g., lowest numbered LBA) in the INFORMATION field of the sense data (see SPC-6).

If a recovered read error is reported, then the device server shall report the last LBA (e.g., highest numbered LBA) on which a recovered read error occurred for the command in the INFORMATION field of the sense data.

If an unrecovered error is reported, then the device server shall report the LBA of the logical block on which an unrecovered error occurred in the INFORMATION field of the sense data.

4.18.2 Processing pseudo unrecovered errors

If a pseudo unrecovered error with correction disabled is encountered on a logical block (e.g., by a command, a background scan(see 4.23.1), or a background self-test (see SPC-6)), then the device server shall:

a) perform no error recovery on the affected logical blocks, including any read error recovery enabled by the Read-Write Error Recovery mode page (see 6.5.10) or the Verify Error Recovery mode page (see 6.5.11);
b) perform no automatic read reassignment or automatic write reassignment for the affected logical blocks, regardless of the settings of the AWRE bit and the ARRE bit in the Read-Write Error Recovery mode page;
c) not consider errors on the affected logical blocks to be informational exception conditions as defined in the Information Exceptions Control mode page (see 6.5.8);

d) not log errors on the affected logical blocks in any log page that contain error counters (see SPC-6); and
e) in any information returned for the error (e.g., in sense data or in the Background Scan Results log page (see 6.4.2)), set the sense key to MEDIUM ERROR and either:
   A) should set the additional sense code to READ ERROR – LBA MARKED BAD BY APPLICATION CLIENT; or
   B) may set the additional sense code to UNRECOVERABLE READ ERROR.

The logical unit shall clear a pseudo unrecovered error if it processes or performs one of the following for that LBA:

a) a format operation;
b) a reassign operation;
c) a sanitize overwrite operation;
d) a sanitize block erase operation; or
e) a write command that is not a WRITE LONG command specifying a pseudo unrecovered error.

The logical unit may clear a pseudo unrecovered error if it processes or performs one of the following for that LBA:

a) a sanitize cryptographic erase operation;
b) an unmap operation;
c) a MODE SELECT command that uses the mode parameter block descriptor (see 6.5.2) to change the capacity to be lower than that LBA;
d) a depopulate operation; or
e) a depopulation revocation operation.

The logical unit shall not clear a pseudo unrecovered error if it processes one of the following for that LBA:

a) a read command.

4.18.3 Block commands sense data descriptor

Table 16 defines the block commands sense data descriptor used in descriptor format sense data (see SPC-6) for direct access block devices.

Table 16 — Block commands sense data descriptor format

Byte \ Bit	7	6	5	4	3	2	1	0
0	DESCRIPTOR TYPE (05h)							
1	ADDITIONAL LENGTH (02h)							
2	Reserved							
3	Reserved		Obsolete	Reserved				

The DESCRIPTOR TYPE field and the ADDITIONAL LENGTH field are defined in SPC-6 and shall be set to the values shown in table 16 for the block commands sense data descriptor.

4.18.4 User data segment referral sense data descriptor

Table 17 defines the user data segment referral sense data descriptor used in descriptor format sense data for direct access block devices. The user data segment referral sense data descriptor contains descriptors indicating the user data segment(s) on the logical unit and the SCSI target port groups through which those user data segments may be accessed (see 4.26).

Table 17 — User data segment referral sense data descriptor format

Byte \ Bit	7	6	5	4	3	2	1	0
0	DESCRIPTOR TYPE (0Bh)							
1	ADDITIONAL LENGTH (y -1)							
2	Reserved							NOT_ALL_R
3	Reserved							
User data segment referral descriptor list								
4 ••• 4 + n	User data segment referral descriptor [first]							
⋮								
y - m ••• y	User data segment referral descriptor [last]							

The DESCRIPTOR TYPE field is defined in SPC-6 and shall be set to the value shown in table 17 for the user data segment referral sense data descriptor.

The ADDITIONAL LENGTH field indicates the number of bytes that follow in the logical block referrals sense data descriptor.

A not all referrals (NOT_ALL_R) bit set to zero indicates that the list of user data segment referral descriptors is a complete list of user data segments. A NOT_ALL_R bit set to one indicates that there are more user data segments than are able to be indicated by the user data segment referral sense data.

Each user data segment referral descriptor (see table 18) indicates information identifying:

a) a user data segment that is accessible through the SCSI target port groups indicated by this descriptor; and
b) one or more SCSI target port groups through which the user data segment indicated by this descriptor is able to be accessed.

User data segment referral descriptors shall be listed in ascending LBA order. If a user data segment referral descriptor describes the last user data segment (i.e., points to the largest LBA) and the preceding user data segment descriptors do not represent the complete list of user data segments, then the next user data segment referral descriptor, if any, shall describe the first user data segment (i.e., the user data segments may wrap).

Table 18 defines the user data segment referral descriptor.

Table 18 — User data segment referral descriptor format

Byte \ Bit	7	6	5	4	3	2	1	0
0 ••• 2	Reserved							
3	NUMBER OF TARGET PORT GROUP DESCRIPTORS							
4 ••• 11	(MSB) FIRST USER DATA SEGMENT LBA (LSB)							
12 ••• 19	(MSB) LAST USER DATA SEGMENT LBA (LSB)							
	Target port group descriptor list							
20 ••• 23	Target port group descriptor [first]							
	⋮							
m-3 ••• m	Target port group descriptor [last]							

The NUMBER OF TARGET PORT GROUP DESCRIPTORS field indicates the number of target port group descriptors that follow.

The FIRST USER DATA SEGMENT LBA field indicates the first LBA of the first user data segment (see 4.26) indicated by this descriptor.

The LAST USER DATA SEGMENT LBA field indicates the last LBA of the last user data segment (see 4.26) indicated by this descriptor.

The target port group descriptor (see table 19) specifies the target port group and the asymmetric access state of the target port group (see SPC-6). The device server shall return one target port group descriptor for each target port group in a target port asymmetric access state of active/optimized, active/non-optimized, or transitioning. The device server may return one target port group descriptor for each target port group in a target port asymmetric access state of unavailable.

Table 19 — Target port group descriptor

Byte \ Bit	7	6	5	4	3	2	1	0
0	Reserved				ASYMMETRIC ACCESS STATE			
1	Reserved							
2	(MSB)	TARGET PORT GROUP						
3								(LSB)

The ASYMMETRIC ACCESS STATE field (see SPC-6) contains the asymmetric access state of the user data segment(s) specified by this descriptor that may be accessed through this target port group.

The TARGET PORT GROUP field specifies a target port group (see SPC-6) that the application client uses when issuing commands associated with the user data segments specified by this descriptor.

4.18.5 Direct access block device sense data descriptor

Table 20 defines the direct access block device sense data descriptor, which may be used in descriptor format sense data (see SPC-6) instead of any of the following sense data descriptors:

a) information (see SPC-6);
b) command-specific information (see SPC-6);
c) sense key specific (see SPC-6);
d) field replaceable unit (see SPC-6); and
e) block commands (see 4.18.3).

If the device server includes the direct access block device sense data descriptor in a sense data descriptor list, then it shall not include any of those sense data descriptors in the same sense data descriptor list.

Table 20 — Direct access block device sense data descriptor format

Byte \ Bit	7	6	5	4	3	2	1	0
0	DESCRIPTOR TYPE (0Dh)							
1	ADDITIONAL LENGTH (16h)							
2	VALID	Reserved	Obsolete	Reserved				
3	Reserved							
4	SKSV	Sense key specific information						
•••								
6								
7	FIELD REPLACEABLE UNIT CODE							
8	(MSB)	INFORMATION						
•••								
15								(LSB)
16	(MSB)	COMMAND-SPECIFIC INFORMATION						
•••								
23								(LSB)

The DESCRIPTOR TYPE field is described in SPC-6, and shall be set as shown in table 20 for the direct access block device sense data descriptor.

A VALID bit set to zero indicates that the INFORMATION field is not defined in this standard or any other command standard. A VALID bit set to one indicates the INFORMATION field contains valid information as defined in this standard or a command standard.

A sense-key specific valid (SKSV) bit set to one indicates the sense key specific information contains valid information as defined in SPC-6. An SKSV bit set to zero indicates that the sense key specific information is not as defined by SPC-6.

The sense key specific information is described in the sense key specific sense data descriptor (see SPC-6).

The FIELD REPLACEABLE UNIT CODE field is described in the field replaceable unit sense data descriptor (see SPC-6).

The INFORMATION field is described in the information sense data descriptor (see SPC-6).

The COMMAND-SPECIFIC INFORMATION field is described in the command-specific information sense data descriptor (see SPC-6). The COMMAND-SPECIFIC INFORMATION field should be ignored in sense data for a command or operation for which the COMMAND-SPECIFIC INFORMATION field is not defined and in sense data that is not related to a command or operation (e.g., pollable sense data).

4.19 Rebuild assist mode

4.19.1 Rebuild assist mode overview

The rebuild assist mode provides a method for a SCSI storage array device (see SCC-2) to read recovered logical blocks from a failed logical unit in a storage array instead of rebuilding the logical blocks from other logical units in the storage array. This mode allows the failed logical unit to report logical blocks that are unreadable without requiring the SCSI storage array device to read every LBA in the failed logical unit to determine the unrecovered logical blocks. The SCSI storage array device then copies the logical blocks recovered from the failed logical unit to a replacement logical unit and only rebuilds the failed logical blocks.

Enabling the rebuild assist mode:

a) may cause the device server to initiate a self test to identify the scope of failures, if any;
b) modifies READ command recovery behavior by the device server based on the setting of the RARC bit (see 4.19.3 and 5.16); and
c) may cause sense data to be returned by the device server that indicates the location of multiple failing logical blocks on read commands and write commands.

The self-test operations performed by the device server while rebuild assist mode is enabled may result in detection of failed physical elements. A predicted unrecovered error is an unrecovered error that is the result of an attempt by the device server to access an LBA associated with a failed physical element. An unpredicted unrecovered error is an unrecovered error that is the result of a device server accessing an LBA that is not associated with a failed physical element.

4.19.2 Enabling rebuild assist mode

An application client should enable rebuild assist mode after the application client determines that a rebuild is required. The application client enables the rebuild assist mode by setting the ENABLED bit to one and setting the DISABLED PHYSICAL ELEMENT field to all zeros in the Rebuild Assist Output diagnostic page (see 6.3.3).

If a SEND DIAGNOSTIC command requests the enabling of the rebuild assist mode, then the device server:

1) shall enable the rebuild assist mode;
2) may perform a diagnostic test of the physical elements contained within the logical unit; and
3) should disable any physical elements that are not functional if a diagnostic test of the physical elements is performed.

The application client may verify that rebuild assist mode is enabled by verifying that the ENABLED bit is set to one in the Rebuild Assist Input diagnostic page (see 6.3.2).

4.19.3 Using the rebuild assist mode

4.19.3.1 Using rebuild assist mode overview

After rebuild assist mode is enabled, the application client should issue read commands to read the available logical block data from the failed logical unit. If the device server does not detect an unrecovered error while processing a read command, then the device server should continue processing the read command.

The rebuild assist mode allows the device server to report unrecovered read errors or unrecovered write errors that are either predicted (i.e., predicted unrecovered errors) or unpredicted (i.e., unpredicted unrecovered errors).

4.19.3.2 Unpredicted unrecovered read error

If a device server receives a read command with the RARC bit set to one, then rebuild assist mode shall not affect processing of the read command.

If rebuild assist mode is enabled and a device server receives a read command with the RARC bit set to zero and the device server detects an unpredicted unrecovered error that is not a pseudo unrecovered read error (see 4.18.2), then the device server:

1) shall perform limited read recovery that is vendor specific;
2) shall transfer the data for all recovered logical blocks, if any, from the logical block referenced by the starting LBA of the failed read command up to the first unrecovered logical block (i.e., the lowest numbered LBA) to the Data-In Buffer;
3) shall terminate the command with CHECK CONDITION status with the sense key set to MEDIUM ERROR, the additional sense code set to UNRECOVERED READ ERROR, and report the LBA referencing the unrecovered logical block in the INFORMATION field (see SPC-6); and
4) may use this failure in a vendor specific manner to predict other logical blocks that may be unrecovered.

If the application client receives sense data with the sense key set to MEDIUM ERROR, the additional sense code set to UNRECOVERED READ ERROR, and the INFORMATION field indicating a valid LBA (see SPC-6), then the application client should issue the next read command with the starting LBA set to the contents of the INFORMATION field plus one.

4.19.3.3 Predicted unrecovered read error

If the device server receives a read command with the RARC bit set to one, then rebuild assist mode shall not affect the processing of the read command.

If rebuild assist mode is enabled and the device server receives a read command with the RARC bit set to zero, and the device server detects a predicted unrecovered error, then the device server:

1) shall perform limited read recovery that is vendor specific;
2) shall transfer the data for all recovered logical blocks, if any, from the logical block referenced by the starting LBA of the failed read command up to the first predicted unrecovered LBA (i.e., the lowest numbered LBA) to the Data-In Buffer; and
3) shall terminate the read command with CHECK CONDITION status with the sense key set to ABORTED COMMAND, the additional sense code set to MULTIPLE READ ERRORS, and:
   A) report the following value in the INFORMATION field (see SPC-6):
      a) the LBA referencing the first unrecovered logical block (i.e., the lowest numbered LBA);

      and

   B) report the following value in the COMMAND-SPECIFIC INFORMATION field (see SPC-6):

a) the LBA referencing the last unrecovered logical block (i.e., the highest numbered LBA) in a sequence of contiguous unrecovered logical blocks that started with the LBA indicated in the INFORMATION field.

If the application client receives sense data with the sense key set to ABORTED COMMAND, the additional sense code set to MULTIPLE READ ERRORS, and the INFORMATION field indicating a valid LBA (see SPC-6), then the application client should issue the next read command with the starting LBA set to the contents of the COMMAND-SPECIFIC INFORMATION field plus one to continue recovering data from the logical unit. This process should be repeated until all of the LBAs have been scanned.

4.19.3.4 Unpredicted unrecovered write error

If rebuild assist mode is enabled and the device server detects an unpredicted unrecovered error while processing a write command, then the device server shall terminate the command with CHECK CONDITION status with the sense key set to MEDIUM ERROR, the additional sense code set to WRITE ERROR, and report the LBA referencing the first logical block (i.e., the lowest numbered LBA) in error in the INFORMATION field (see SPC-6).

4.19.3.5 Predicted unrecovered write error

If rebuild assist mode is enabled and the device server detects a predicted unrecovered error while processing a write command, then the device server:

1) transfers the write data from the Data-Out Buffer;
2) writes the transferred data up to the logical block referenced by the failing LBA; and
3) shall terminate the write command with CHECK CONDITION status with the sense key set to ABORTED COMMAND, the additional sense code set to MULTIPLE WRITE ERRORS, and:
   A) report the following value in the INFORMATION field (see SPC-6):
      a) the LBA referencing the first logical block (i.e., the lowest numbered LBA) in error;

   and

   B) report the following value in the COMMAND-SPECIFIC INFORMATION field (see SPC-6):
      a) the LBA referencing the last logical block (i.e., the highest numbered LBA) in error in a sequence of contiguous logical blocks that started with the LBA indicated in the INFORMATION field.

If the application client receives sense data with the sense key set to ABORTED COMMAND, the additional sense code set to MULTIPLE WRITE ERRORS, and the INFORMATION field indicating a valid LBA (see SPC-6), then the application client should issue the next write command with the starting LBA set to the contents of the COMMAND-SPECIFIC INFORMATION field plus one to continue writing to the logical unit.

4.19.4 Disabling the rebuild assist mode

Rebuild assist mode shall be disabled after a power on.

Rebuild assist mode shall not be affected by a hard reset, an I_T nexus loss, or any task management functions (see SAM-6).

The application client disables rebuild assist mode by setting the ENABLED bit to zero in the Rebuild Assist Output diagnostic page (see 6.3.3).

4.19.5 Testing rebuild assist mode

The Rebuild Assist Output diagnostic page (see 6.3.3) provides a method to test the application client's rebuild process.

An application client places a logical unit into a simulated failing condition by setting the ENABLED bit to one and setting one or more bits in the DISABLED PHYSICAL ELEMENT field to one in the Rebuild Assist Output diagnostic page.

Each bit in the DISABLED PHYSICAL ELEMENT field represents a physical element that is associated with a group of LBAs that are treated as having predicted unrecovered read errors. The correlation of bits in the DISABLED PHYSICAL ELEMENT field to LBAs in the logical unit is vendor specific.

An application client ends a test by disabling the rebuild assist mode (see 4.19.4).

4.20 START STOP UNIT and power conditions

4.20.1 START STOP UNIT and power conditions overview

The START STOP UNIT command (see 5.31) allows an application client to control the power condition of a logical unit. This method includes specifying that the logical unit transition to a specific 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-6). 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 the power condition specified by any START STOP UNIT command shall override the Power Condition mode page's power control.

If the device server processes a START STOP UNIT command that specifies a POWER CONDITION field set to 1h (i.e., ACTIVE), 2h (i.e., IDLE), or 3h (i.e., STANDBY), then the device server halts all idle power condition timers and all standby power condition timers as described in SPC-6 until the device server processes a START STOP UNIT command that results in those timers being initialized and started. If the device server processes a START STOP UNIT command that specifies a POWER CONDITION field set to 7h (i.e., LU_CONTROL), Ah (i.e., FORCE_IDLE_0), or Bh (i.e., FORCE_STANDBY_0), then the device server starts idle power condition timers and standby power condition timers as described in 5.31.

The device server may change any power conditions established during the processing of a START STOP UNIT command (e.g., halted power condition timers) while processing a Logical Unit Reset event (see SAM-6).

4.20.2 Processing of concurrent START STOP UNIT commands

If a START STOP UNIT command is being processed by the device server, and a subsequent START STOP UNIT command for which the CDB is validated requests that the logical unit change to a different power condition than was specified by the START STOP UNIT command being processed, then the device server shall terminate the subsequent START STOP UNIT command with CHECK CONDITION status with the sense key set to NOT READY and the additional sense code set to LOGICAL UNIT NOT READY, START STOP UNIT COMMAND IN PROGRESS.

The constraints on concurrent START STOP UNIT commands apply only to commands that have the IMMED bit set to zero. The effects of concurrent power condition changes requested by START STOP UNIT commands with the IMMED bit set to one are vendor specific.

4.20.3 Managing logical block access commands during a change to the active power condition

Application clients may minimize the return of BUSY status or TASK SET FULL status during a change to the active power condition by:

a) polling the power condition using the REQUEST SENSE command (see SPC-6); or
b) sending a START STOP UNIT command with the IMMED bit set to zero and the START bit set to one and waiting for GOOD status to be returned.

4.20.4 Stopped power condition

In addition to the active power condition, idle power conditions, and standby power conditions described in SPC-6, this standard describes the stopped power condition.

While in the stopped power condition:

a) the device server shall terminate TEST UNIT READY commands and logical block access commands with CHECK CONDITION status with the sense key set to NOT READY and the additional sense code set to LOGICAL UNIT NOT READY, INITIALIZING COMMAND REQUIRED;
b) the power consumed by the SCSI target device while in the stopped power condition should be less than the power consumed when the logical unit is in the active power condition or any of the idle power conditions (e.g., for direct access block devices that have a rotating medium, the medium is stopped in the stopped power condition);
c) the peak power consumption during a change from the stopped power condition to the active power condition or an idle power condition is not limited by this standard; and
d) the peak power consumption during a change from the stopped power condition to a standby power condition shall be no more than the typical peak power consumption in the active power condition.

No power condition defined in this standard shall affect the supply of any power required for proper operation of a service delivery subsystem.

4.20.5 START STOP UNIT and power condition state machine

4.20.5.1 START STOP UNIT and power condition state machine overview

The SSU_PC (START STOP UNIT and power condition) state machine for logical units implementing the START STOP UNIT command describes the:

a) logical unit power states and transitions resulting from specifications in the START STOP UNIT command;
b) settings in the Power Condition mode page (see SPC-6); and
c) the processing of commands.

NOTE 4 - The SSU_PC state machine is an enhanced version of the PC (power condition) state machine described in SPC-6.

The SSU_PC state machine consists of the states shown in table 21.

Table 21 — Summary of states in the SSU_PC state machine

State	Reference	PC state machine state with additional definition (see SPC-6)
SSU_PC0:Powered_On [a]	4.20.5.2	PC0:Powered_On
SSU_PC1:Active	4.20.5.3	PC1:Active
SSU_PC2:Idle	4.20.5.4	PC2:Idle
SSU_PC3:Standby	4.20.5.5	PC3:Standby
SSU_PC4:Active_Wait	4.20.5.6	PC4:Active_Wait
SSU_PC5:Wait_Idle	4.20.5.7	PC5:Wait_Idle
SSU_PC6:Wait_Standby	4.20.5.8	PC6:Wait_Standby
SSU_PC7:Idle_Wait	4.20.5.9	n/a
SSU_PC8:Stopped	4.20.5.10	n/a
SSU_PC9:Standby_Wait	4.20.5.11	n/a
SSU_PC10:Wait_Stopped	4.20.5.12	n/a
[a] SSU_PC0:Powered_On is the initial state.		

While in the following SSU_PC states, the logical unit may be increasing power usage to enter a higher power condition:

a) SSU_PC4:Active_Wait;
b) SSU_PC7:Idle_Wait; and
c) SSU_PC9:Standby_Wait.

While in the following SSU_PC states, the logical unit may be decreasing power usage to enter a lower power condition:

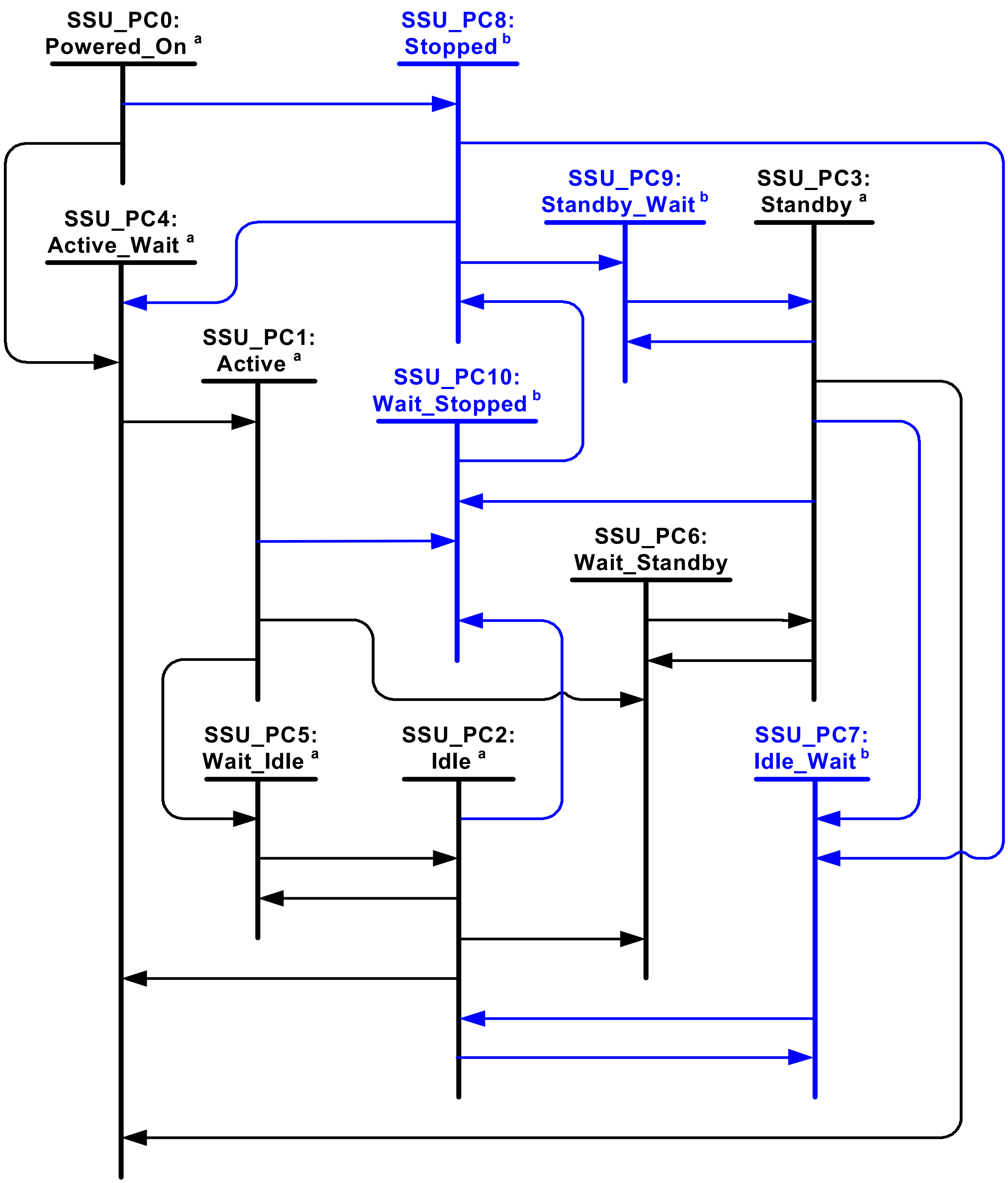
a) SSU_PC5:Wait_Idle;
b) SSU_PC6:Wait_Standby; and
c) SSU_PC10:Wait_Stopped.

Any command causing a state machine transition (e.g., a START STOP UNIT command with the IMMED bit set to zero) shall not complete with GOOD status until this state machine reaches the state (i.e., power condition) required or specified by the command.

The SSU_PC state machine shall start in the SSU_PC0:Powered_On state after power on. The SSU_PC state machine shall be configured to transition to the SSU_PC4:Active_Wait state or the SSU_PC8:Stopped state after power on by a mechanism not defined by this standard.

This state machine references timers controlled by the Power Condition mode page (see SPC-6) and refers to the START STOP UNIT command (see 5.31).

Figure 11 describes the SSU_PC state machine.



[a] This state or transition is also described in SPC-6, but may have additional characteristics unique to this standard (e.g., a transition to or from a state described in this standard).
[b] This state or transition is described in this standard.

Figure 11 — SSU_PC state machine

4.20.5.2 SSU_PC0:Powered_On state

4.20.5.2.1 SSU_PC0:Powered_On state description

See the PC0:Powered_On state in SPC-6 for details about this state.

4.20.5.2.2 Transition SSU_PC0:Powered_On to SSU_PC4:Active_Wait

This transition shall occur if:

a) the logical unit is ready to begin power on initialization; and
b) the logical unit has been configured to transition to the SSU_PC4:Active_Wait state.

The transition shall include a Transitioning From Powered On argument.

4.20.5.2.3 Transition SSU_PC0:Powered_On to SSU_PC8:Stopped

This transition shall occur if:

a) the logical unit has been configured to transition to the SSU_PC8:Stopped state.

The transition shall include a Transitioning From Powered On argument.

4.20.5.3 SSU_PC1:Active state

4.20.5.3.1 SSU_PC1:Active state description

See the PC1:Active state in SPC-6 for details about this state.

4.20.5.3.2 Transition SSU_PC1:Active to SSU_PC5:Wait_Idle

This transition shall occur if:

a) the device server processes a START STOP UNIT command (see 5.31) with the POWER CONDITION field set to 2h (i.e., IDLE); or
b) an idle condition timer (see SPC-6) is enabled, and that timer has expired.

The transition shall include a:

a) Transitioning To Idle_a argument, if:
    A) the highest priority timer that expired is the idle_a condition timer; or
    B) the START STOP UNIT command being processed has the POWER CONDITION MODIFIER field set to 0h (i.e., idle_a power condition);
b) Transitioning To Idle_b argument, if:
    A) the highest priority timer that expired is the idle_b condition timer; or
    B) the START STOP UNIT command being processed has the POWER CONDITION MODIFIER field set to 1h (i.e., idle_b power condition);

    or
c) Transitioning To Idle_c argument, if:
    A) the highest priority timer that expired is the idle_c condition timer; or
    B) the START STOP UNIT command being processed has the POWER CONDITION MODIFIER field set to 2h (i.e., idle_c power condition).

4.20.5.3.3 Transition SSU_PC1:Active to SSU_PC6:Wait_Standby

This transition shall occur if:

a) the device server processes a START STOP UNIT command (see 5.31) with the POWER CONDITION field set to 3h (i.e., STANDBY); or
b) a standby condition timer (see SPC-6) is enabled and that timer has expired.

The transition shall include a:

a) Transitioning To Standby_z argument, if:
   A) the highest priority timer that expired is the standby_z condition timer; or
   B) the START STOP UNIT command being processed has the POWER CONDITION MODIFIER field set to 0h (i.e., standby_z power condition);

   or

b) Transitioning To Standby_y argument, if:
   A) the highest priority timer that expired is the standby_y condition timer; or
   B) the START STOP UNIT command being processed has the POWER CONDITION MODIFIER field set to 1h (i.e., standby_y power condition).

4.20.5.3.4 Transition SSU_PC1:Active to SSU_PC10:Wait_Stopped

This transition shall occur if:

a) the device server processes a START STOP UNIT command (see 5.31) with the START bit set to zero and the POWER CONDITION field set to 0h (i.e., START_VALID).

4.20.5.4 SSU_PC2:Idle state

4.20.5.4.1 SSU_PC2:Idle state description

See the PC2:Idle state in SPC-6 for details about this state.

4.20.5.4.2 Transition SSU_PC2:Idle to SSU_PC4:Active_Wait

This transition shall occur if:

a) the device server processes a START STOP UNIT command (see 5.31) with the START bit set to one and the POWER CONDITION field set to 0h (i.e., START_VALID);
b) the device server processes a START STOP UNIT command with the POWER CONDITION field set to 1h (i.e., ACTIVE); or
c) the device server processes a command that requires the logical unit to be in the SSU_PC1:Active state to continue processing that command.

The transition shall include a:

a) Transitioning From Idle argument; and
b) Transitioning From Idle_c argument if the current power condition is the idle_c power condition.

4.20.5.4.3 Transition SSU_PC2:Idle to SSU_PC5:Wait_Idle

This transition shall occur if:

a) the following occur:
   A) an idle condition timer is enabled and that idle condition timer has expired; and
   B) the priority of that idle condition timer is greater than the priority of the idle condition timer associated with the current idle power condition (see SPC-6);

   or

b) the device server processes a START STOP UNIT command (see 5.31) with the POWER CONDITION field set to 2h (i.e., IDLE) and the POWER CONDITION MODIFIER field set to a value that specifies that the logical unit transition to a lower idle power condition.

The transition shall include a:

a) Transitioning To Idle_b argument, if:
   A) the highest priority timer that expired is the idle_b condition timer; or

B) the START STOP UNIT command being processed has the POWER CONDITION MODIFIER field set to 1h (i.e., idle_b power condition);

or

b) Transitioning To Idle_c argument, if:
   A) the highest priority timer that expired is the idle_c condition timer; or
   B) the START STOP UNIT command being processed has the POWER CONDITION MODIFIER field set to 2h (i.e., idle_c power condition).

4.20.5.4.4 Transition SSU_PC2:Idle to SSU_PC6:Wait_Standby

This transition shall occur if:

a) the device server processes a START STOP UNIT command (see 5.31) with the POWER CONDITION field set to 3h (i.e., STANDBY); or
b) a standby condition timer is enabled and that timer has expired.

The transition shall include a:

a) Transitioning To Standby_z argument, if:
   A) the highest priority timer that expired is the standby_z condition timer; or
   B) the START STOP UNIT command being processed has the POWER CONDITION MODIFIER field set to 0h (i.e., standby_z power condition);

   or

b) Transitioning To Standby_y argument, if:
   A) the highest priority timer that expired is the standby_y condition timer; or
   B) the START STOP UNIT command being processed has the POWER CONDITION MODIFIER field set to 1h (i.e., standby_y power condition).

4.20.5.4.5 Transition SSU_PC2:Idle to SSU_PC7:Idle_Wait

This transition shall occur if:

a) the device server processes a START STOP UNIT command (see 5.31) with the POWER CONDITION field set to 2h (i.e., IDLE) and the POWER CONDITION MODIFIER field set to a value that specifies that the logical unit transition to a higher idle power condition.

The transition shall include Transitioning From Idle argument and a:

a) Transitioning To Idle_a argument, if the START STOP UNIT command being processed has the POWER CONDITION MODIFIER field set to 0h (i.e., idle_a power condition); or
b) Transitioning To Idle_b argument, if the START STOP UNIT command being processed has the POWER CONDITION MODIFIER field set to 1h (i.e., idle_b power condition).

4.20.5.4.6 Transition SSU_PC2:Idle to SSU_PC10:Wait_Stopped

This transition shall occur if:

a) the device server processes a START STOP UNIT command (see 5.31) with the START bit set to zero and the POWER CONDITION field set to 0h (i.e., START_VALID).

4.20.5.5 SSU_PC3:Standby state

4.20.5.5.1 SSU_PC3:Standby state description

See the PC3:Standby state in SPC-6 for details about this state.

4.20.5.5.2 Transition SSU_PC3:Standby to SSU_PC4:Active_Wait

This transition shall occur if:

a) the device server processes a START STOP UNIT command (see 5.31) with the START bit set to one and the POWER CONDITION field set to 0h (i.e., START_VALID);
b) the device server processes a START STOP UNIT command with the POWER CONDITION field set to 1h (i.e., ACTIVE); or
c) the device server processes a command that requires the logical unit to be in the SSU_PC1:Active state to continue processing that command.

The transition shall include a Transitioning From Standby argument.

4.20.5.5.3 Transition SSU_PC3:Standby to SSU_PC6:Wait_Standby

This transition shall occur if:

a) the following occur:
    A) the standby_z condition timer is enabled and that timer expires; and
    B) the priority of that standby condition timer is greater than the priority of the standby condition timer associated with the current standby power condition (see SPC-6);

    or

b) the device server processes a START STOP UNIT command with the POWER CONDITION field set to 3h (i.e., STANDBY) and the POWER CONDITION MODIFIER field set to a value that specifies that the logical unit transition to a lower standby power condition.

The transition shall include Transitioning To Standby_z argument.

4.20.5.5.4 Transition SSU_PC3:Standby to SSU_PC7:Idle_Wait

This transition shall occur if:

a) the device server processes a START STOP UNIT command (see 5.31) with the POWER CONDITION field set to 2h (i.e., IDLE); or
b) the device server processes a command and determines that the device server is capable of continuing the processing of that command, when the logical unit is in the SSU_PC2:Idle state.

The transition shall include a Transitioning From Standby argument and a:

a) Transitioning To Idle_a argument, if:
    A) the device server processes a command and determines that the device server is capable of continuing the processing of that command, when the logical unit is in the idle_a power condition; or
    B) the device server processes a START STOP UNIT command with the POWER CONDITION MODIFIER field set to 0h (i.e., idle_a power condition);
b) Transitioning To Idle_b argument, if:
    A) the device server processes a command and determines that the device server is capable of continuing the processing of that command, when the logical unit is in the idle_b power condition; or
    B) the device server processes a START STOP UNIT command with the POWER CONDITION MODIFIER field set to 1h (i.e., idle_b power condition);

    or

c) Transitioning To_Idle_c argument, if:
    A) the device server processes a command and determines that the device server is capable of continuing the processing of that command, when the logical unit is in the idle_c power condition; or
    B) the device server processes a START STOP UNIT command with the POWER CONDITION MODIFIER field set to 2h (i.e., idle_c power condition).

4.20.5.5.5 Transition SSU_PC3:Standby to SSU_PC9:Standby_Wait

This transition shall occur if:

a) the device server processes a START STOP UNIT command (see 5.31) with the POWER CONDITION field set to 3h (i.e., STANDBY) and the POWER CONDITION MODIFIER field set to a value that specifies that the logical unit transition to a higher standby power condition.

The transition shall include a Transitioning To Standby_y argument.

4.20.5.5.6 Transition SSU_PC3:Standby to SSU_PC10:Wait_Stopped

This transition shall occur if:

a) the device server processes a START STOP UNIT command (see 5.31) with the START bit set to zero and the POWER CONDITION field set to 0h (i.e., START_VALID).

4.20.5.6 SSU_PC4:Active_Wait state

4.20.5.6.1 SSU_PC4:Active_Wait state description

While in this state:

a) each idle condition timer that is operational (see SPC-6) and not expired is running;
b) each standby condition timer that is operational and not expired is running;
c) the device server shall provide power management pollable sense data (see SPC-6) with the sense key set to NO SENSE and the additional sense code set to LOGICAL UNIT TRANSITIONING TO ANOTHER POWER CONDITION; and
d) the logical unit is performing the operations required for it to be in the SSU_PC1:Active state (e.g., a disk drive spins up its medium).

If this state was entered with a Transitioning From Idle argument, then:

a) the device server is capable of processing and completing the same commands, except START STOP UNIT commands with the IMMED bit set to zero (see 5.31), that the device server is able to process and complete while in the SSU_PC2:Idle state;
b) the peak power consumed in this state shall be no more than the typical peak power consumed in the SSU_PC1:Active state; and
c) if:
   A) this state was entered with a Transitioning From Idle_c argument; and
   B) the CCF IDLE field in the Power Condition mode page (see SPC-6) is set to 10b (i.e., enabled),

   then the device server shall terminate any command, except a START STOP UNIT command, that requires the logical unit be in the SSU_PC1:Active state to continue processing, with CHECK CONDITION status, with the sense key set to NOT READY and the additional sense code set to LOGICAL UNIT IS IN PROCESS OF BECOMING READY.

If this state was entered with a Transitioning From Standby argument, then:

a) the device server is capable of processing and completing the same commands, except START STOP UNIT commands with the IMMED bit set to zero, that the device server is able to process and complete while in the SSU_PC3:Standby state;
b) the peak power consumption in this state is not limited by this standard; and
c) if the CCF STANDBY field in the Power Condition mode page (see SPC-6) is set to 10b (i.e., enabled), then the device server shall terminate any command, except a START STOP UNIT command, that requires the logical unit be in the SSU_PC1:Active state or SSU_PC2:Idle state to continue processing, with CHECK CONDITION status, with the sense key set to NOT READY and the additional sense code set to LOGICAL UNIT IS IN PROCESS OF BECOMING READY.

If this state was entered with a Transitioning From Stopped argument, then:

a) the device server is capable of processing and completing the same commands, except START STOP UNIT commands with the IMMED bit set to zero, that the device server is able to process and complete while in the SSU_PC8:Stopped state;
b) the peak power consumption in this state is not limited by this standard; and
c) if the CCF STOPPED field in the Power Condition mode page (see SPC-6) is set to 10b (i.e., enabled), then the device server shall terminate any TEST UNIT READY command or logical block access command, with CHECK CONDITION status, with the sense key set to NOT READY and the additional sense code set to LOGICAL UNIT IS IN PROCESS OF BECOMING READY.

If this state was entered with a Transitioning From Powered On argument, then:

a) the device server is capable of processing and completing the same commands, except START STOP UNIT commands with the IMMED bit set to zero or TEST UNIT READY command, that the device server is able to process and complete while in the SSU_PC8:Stopped state;
b) the peak power consumption in this state is not limited by this standard; and
c) the device server shall terminate any TEST UNIT READY command or logical block access command with CHECK CONDITION status with the sense key set to NOT READY and the additional sense code set to LOGICAL UNIT IS IN PROCESS OF BECOMING READY.

If an idle condition timer or a standby condition timer is enabled and expires, then that timer is ignored in this state.

4.20.5.6.2 Transition SSU_PC4:Active_Wait to SSU_PC1:Active

See the PC4:Active_Wait to PC1:Active transition in SPC-6 for details about this transition.

4.20.5.7 SSU_PC5:Wait_Idle state

4.20.5.7.1 SSU_PC5:Wait_Idle state description

See the PC5:Wait_Idle state in SPC-6 for details about this state.

4.20.5.7.2 Transition SSU_PC5:Wait_Idle to SSU_PC2:Idle

See the PC5:Wait_Idle to PC2:Idle transition in SPC-6 for details about this transition.

4.20.5.8 SSU_PC6:Wait_Standby state

4.20.5.8.1 SSU_PC6:Wait_Standby state description

See the PC6:Wait_Standby state in SPC-6 for details about this state.

4.20.5.8.2 Transition SSU_PC6:Wait_Standby to SSU_PC3:Standby

See the PC6:Wait_Standby to PC3:Standby transition in SPC-6 for details about this transition.

4.20.5.9 SSU_PC7:Idle_Wait state

4.20.5.9.1 SSU_PC7:Idle_Wait state description

While in this state:

a) each idle condition timer that is operational (see SPC-6) and not expired is running;
b) each standby condition timer that is operational and not expired is running;
c) the device server shall provide power management pollable sense data (see SPC-6) with the sense key set to NO SENSE and the additional sense code set to LOGICAL UNIT TRANSITIONING TO ANOTHER POWER CONDITION; and

d) the logical unit is performing the operations required for it to be in the SSU_PC2:Idle state (e.g., a disk drive spins up its medium).

If this state was entered with a Transitioning From Idle argument, then:

a) the device server is capable of processing and completing the same commands, except START STOP UNIT commands with the IMMED bit set to zero (see 5.31), that the device server is able to process and complete while in the SSU_PC2:Idle state; and
b) the peak power consumed in this state shall be no more than the typical peak power consumed in the SSU_PC1:Active state.

If this state was entered with a Transitioning From Standby argument, then:

a) the device server is capable of processing and completing the same commands, except START STOP UNIT commands with the IMMED bit set to zero, that the device server is able to process and complete while in the SSU_PC3:Standby state;
b) the peak power consumption in this state is not limited by this standard; and
c) the CCF STANDBY field in the Power Condition mode page (see SPC-6) is set to 10b (i.e., enabled), then the device server shall terminate any command, except a START STOP UNIT command, that requires the logical unit be in the SSU_PC1:Active state or SSU_PC2:Idle state to continue processing, with CHECK CONDITION status, with the sense key set to NOT READY and the additional sense code set to LOGICAL UNIT IS IN PROCESS OF BECOMING READY.

If this state was entered with a Transitioning From Stopped argument, then:

a) the device server is capable of processing and completing the same commands, except START STOP UNIT commands with the IMMED bit set to zero, that the device server is able to process and complete while in the SSU_PC8:Stopped state;
b) the peak power consumption in this state is not limited by this standard; and
c) if the CCF STOPPED field in the Power Condition mode page (see SPC-6) is set to 10b (i.e., enabled), then the device server shall terminate any TEST UNIT READY command or logical block access command, with CHECK CONDITION status, with the sense key set to NOT READY and the additional sense code set to LOGICAL UNIT IS IN PROCESS OF BECOMING READY.

If an idle condition timer or a standby condition timer is enabled and expires, then that timer is ignored in this state.

4.20.5.9.2 Transition SSU_PC7:Idle_Wait to SSU_PC2:Idle

This transition shall occur when the logical unit meets the requirements for being in the:

a) idle_a power condition, if this state was entered with a Transitioning To Idle_a argument;
b) idle_b power condition, if this state was entered with a Transitioning To Idle_b argument; or
c) idle_c power condition, if this state was entered with a Transitioning To Idle_c argument.

4.20.5.10 SSU_PC8:Stopped state

4.20.5.10.1 SSU_PC8:Stopped state description

While in this state:

a) the logical unit is in the stopped power condition (see 4.20.4);
b) the idle condition timers and the standby condition timers are halted;
c) the device server shall provide power management pollable sense data (see SPC-6); and
d) the device server terminates each logical block access command or TEST UNIT READY command (see SPC-6) as described in 4.20.4.

4.20.5.10.2 Transition SSU_PC8:Stopped to SSU_PC4:Active_Wait

This transition shall occur if:

a) the device server processes a START STOP UNIT command (see 5.31) with the START bit set to one and the POWER CONDITION field set to 0h (i.e., START_VALID); or
b) the device server processes a START STOP UNIT command with the POWER CONDITION field set to 1h (i.e., ACTIVE).

The transition shall include a Transitioning From Stopped argument.

4.20.5.10.3 Transition SSU_PC8:Stopped to SSU_PC7:Idle_Wait

This transition shall occur if:

a) the device server processes a START STOP UNIT command (see 5.31) with the POWER CONDITION field set to 2h (i.e., IDLE).

The transition shall include a Transitioning From Stopped argument and a:

a) Transitioning To Idle_a argument, if the START STOP UNIT command being processed has the POWER CONDITION MODIFIER field set to 0h (i.e., idle_a power condition);
b) Transitioning To Idle_b argument, if the START STOP UNIT command being processed has the POWER CONDITION MODIFIER field set to 1h (i.e., idle_b power condition); or
c) Transitioning To Idle_c argument, if the START STOP UNIT command being processed has the POWER CONDITION MODIFIER field set to 2h (i.e., idle_c power condition).

4.20.5.10.4 Transition SSU_PC8:Stopped to SSU_PC9:Standby_Wait

This transition shall occur if:

a) the device server processes a START STOP UNIT command (see 5.31) with the POWER CONDITION field set to 3h (i.e., STANDBY).

The transition shall include a Transitioning From Stopped argument and a:

a) Transitioning To Standby_z argument, if the START STOP UNIT command being processed has the POWER CONDITION MODIFIER field set to 0h (i.e., standby_z power condition); or
b) Transitioning To Standby_y argument, if the START STOP UNIT command being processed has the POWER CONDITION MODIFIER field set to 1h (i.e., standby_y power condition).

4.20.5.11 SSU_PC9:Standby_Wait state

4.20.5.11.1 SSU_PC9:Standby_Wait state description

While in this state:

a) the device server shall provide power management pollable sense data (see SPC-6) with the sense key set to NO SENSE and the additional sense code set to LOGICAL UNIT TRANSITIONING TO ANOTHER POWER CONDITION;
b) the peak power consumed in this state shall be no more than the typical peak power consumed in the SSU_PC1:Active state; and
c) the logical unit is performing the operations required for it to be in the SSU_PC3:Standby state ((e.g., a direct access block device is activating circuitry).

If this state was entered with a Transitioning From Standby argument, then the device server is capable of processing and completing the same commands, except START STOP UNIT commands with the IMMED bit set to zero (see 5.31), that the device server is able to process and complete in the SSU_PC3:Standby state.

If this state was entered with a Transitioning From Stopped argument, then:

a) the device server is capable of processing and completing the same commands, except START STOP UNIT commands with the IMMED bit set to zero, that the device server is able to process and complete while in the SSU_PC8:Stopped state; and
b) if the CCF STOPPED field in the Power Condition mode page (see SPC-6) is set to 10b (i.e., enabled), then the device server terminates any TEST UNIT READY command or logical block access command, with CHECK CONDITION status, with the sense key set to NOT READY and the additional sense code set to LOGICAL UNIT IS IN PROCESS OF BECOMING READY.

4.20.5.11.2 Transition SSU_PC9:Standby_Wait to SSU_PC3:Standby

This transition shall occur when the logical unit meets the requirements for being in the:

a) standby_y power condition, if this state was entered with a Transitioning To Standby_y argument; or
b) standby_z power condition, if this state was entered with a Transitioning To Standby_z argument.

4.20.5.12 SSU_PC10:Wait_Stopped state

4.20.5.12.1 SSU_PC10:Wait_Stopped state description

While in this state:

a) the device server shall provide power management pollable sense data (see SPC-6) with the sense key set to NO SENSE and the additional sense code set to LOGICAL UNIT TRANSITIONING TO ANOTHER POWER CONDITION;
b) the device server is capable of processing and completing the same commands, except START STOP UNIT commands with the IMMED bit set to zero (see 5.31), that the device server is able to process and complete in the SSU_PC8:Stopped state;
c) the logical unit is performing the operations required for it to be in the SSU_PC8:Stopped state (e.g., a disk drive spins down its medium); and
d) the device server shall terminate any TEST UNIT READY command or logical block access command with CHECK CONDITION status with the sense key set to NOT READY and the additional sense code set to LOGICAL UNIT NOT READY, INITIALIZING COMMAND REQUIRED.

4.20.5.12.2 Transition SSU_PC10:Wait_Stopped to SSU_PC8:Stopped

This transition shall occur when:

a) the logical unit meets the requirements for being in the SSU_PC8:Stopped state.

4.21 Protection information model

4.21.1 Protection information overview

The protection information model provides for protection of user data while user data is being transferred between a sender and a receiver. Protection information is generated at the application layer and may be checked by any object associated with the I_T_L nexus (see SAM-6). Once received, protection information is retained (e.g., written to the medium, stored in non-volatile memory, or recalculated on read back) by the device server until overwritten. Power loss, hard reset, logical unit reset, and I_T nexus loss shall 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-6).

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-6), then checking of the logical block reference tag within a service delivery subsystem without accounting for modified data pointers and data alignments may cause false errors when logical blocks are transmitted out of order.

Protection information is also referred to as the data integrity field (DIF).

4.21.2 Protection types

4.21.2.1 Protection types overview

The content of protection information is dependent on the type of protection to which a logical unit has been formatted.

The type of protection supported by the logical unit shall be indicated in the SPT field in the Extended INQUIRY Data VPD page (see SPC-6). The current protection type shall be indicated in the P_TYPE field (see 5.21.2).

An application client may format the logical unit to a specific type of protection using the FMTPINFO field and the PROTECTION FIELD USAGE field (see 5.4).

An application client may format the logical unit to place protection information at intervals other than on logical block boundaries using the PROTECTION INTERVAL EXPONENT field.

A medium access command is processed in a different manner by a device server depending on the type of protection in effect. When used in relation to types of protection, the term “medium access command” is defined as any one of the following commands:

a) COMPARE AND WRITE;
b) ORWRITE (16);
c) ORWRITE (32);
d) READ (10);
e) READ (12);
f) READ (16);
g) READ (32);
h) VERIFY (10);
i) VERIFY (12);
j) VERIFY (16);
k) VERIFY (32);
l) WRITE (10);
m) WRITE (12);
n) WRITE (16);
o) WRITE (32);
p) WRITE AND VERIFY (10);
q) WRITE AND VERIFY (12);
r) WRITE AND VERIFY (16);
s) WRITE AND VERIFY (32);
t) WRITE ATOMIC (16);
u) WRITE ATOMIC (32);
v) WRITE SAME (10);
w) WRITE SAME (16);
x) WRITE SAME (32);
y) WRITE SCATTERED (16);
z) WIRITE SCATTERED (32);
aa) WRITE STREAM (16); and
ab) WRITE STREAM (32).

4.21.2.2 Type 0 protection

Type 0 protection defines no protection over that which is defined within the transport protocol.

A logical unit that has been formatted with protection information disabled (see 5.3) or a logical unit that does not support protection information (i.e., the PROTECT bit set to zero in the standard INQUIRY data (see SPC-6)) has type 0 protection.

If type 0 protection is enabled and the RDPROTECT field, the WRPROTECT field, the VRPROTECT field, or the ORPROTECT field is set to a non-zero value, then medium access commands are invalid and may be terminated by the device server with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.

If type 0 protection is enabled, then the following medium access commands are invalid and shall be terminated by the device server with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID COMMAND OPERATION CODE:

a) READ (32);
b) VERIFY (32);
c) WRITE (32);
d) WRITE AND VERIFY (32);
e) WRITE ATOMIC (32);
f) WRITE SAME (32);
g) WRITE SCATTERED (32); and
h) WRITE STREAM (32).

4.21.2.3 Type 1 protection

Type 1 protection:

a) defines the content of each LOGICAL BLOCK GUARD field;
b) does not define the content of any LOGICAL BLOCK APPLICATION TAG field; and
c) defines the content of each LOGICAL BLOCK REFERENCE TAG field.

If type 1 protection is enabled, then the following medium access commands are invalid and shall be terminated by the device server with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID COMMAND OPERATION CODE:

a) READ (32);
b) VERIFY (32);
c) WRITE (32);
d) WRITE AND VERIFY (32);
e) WRITE ATOMIC (32);
f) WRITE SAME (32);
g) WRITE SCATTERED (32); and
h) WRITE STREAM (32).

For valid medium access commands in which the RDPROTECT field, the WRPROTECT field, the VRPROTECT field, or the ORPROTECT field is set to:

a) zero, the Data-In Buffer and/or Data-Out Buffer associated with those commands shall consist of logical block data containing only user data; or
b) a non-zero value, the Data-In Buffer and/or Data-Out Buffer shall consist of logical block data containing both user data and protection information.

4.21.2.4 Type 2 protection

Type 2 protection:

a) defines the content of each LOGICAL BLOCK GUARD field;
b) does not define the content of any LOGICAL BLOCK APPLICATION TAG field; and
c) defines, except for the first logical block addressed by the command, the content of each LOGICAL BLOCK REFERENCE TAG field.

If type 2 protection is enabled and the RDPROTECT field, the WRPROTECT field, the VRPROTECT field, or the ORPROTECT field is set to a non-zero value, then the following medium access commands are invalid and shall be terminated by the device server with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID COMMAND OPERATION CODE:

a) COMPARE AND WRITE;
b) ORWRITE (16);
c) ORWRITE (32);
d) READ (10);
e) READ (12);
f) READ (16);
g) VERIFY (10);
h) VERIFY (12);
i) VERIFY (16);
j) WRITE (10);
k) WRITE (12);
l) WRITE (16);
m) WRITE AND VERIFY (10);
n) WRITE AND VERIFY (12);
o) WRITE AND VERIFY (16);
p) WRITE ATOMIC (16);
q) WRITE SAME (10);
r) WRITE SAME (16);
s) WRITE SCATTERED (16); and
t) WRITE STREAM (16).

For valid medium access commands in which the RDPROTECT field, the WRPROTECT field, the VRPROTECT field, or the ORPROTECT field is set to:

a) zero, the Data-In Buffer and/or Data-Out Buffer associated with those commands shall consist of logical block data containing only user data; or
b) a non-zero value, the Data-In Buffer and/or Data-Out Buffer shall consist of logical block data containing both user data and protection information.

4.21.2.5 Type 3 protection

Type 3 protection:

a) defines the content of each LOGICAL BLOCK GUARD field;
b) does not define the content of any LOGICAL BLOCK APPLICATION TAG field; and
c) does not define the content of any LOGICAL BLOCK REFERENCE TAG field.

If type 3 protection is enabled, then the following medium access commands are invalid and shall be terminated by the device server with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID COMMAND OPERATION CODE:

a) READ (32);
b) VERIFY (32);
c) WRITE (32);
d) WRITE AND VERIFY (32);
e) WRITE ATOMIC (32);
f) WRITE SAME (32);
g) WRITE SCATTERED (32); and
h) WRITE STREAM (32).

For valid medium access commands in which the RDPROTECT field, the WRPROTECT field, the VRPROTECT field, or the ORPROTECT field is set to:

a) zero, the Data-In Buffer and/or Data-Out Buffer associated with those commands shall consist of logical block data containing only user data; or
b) a non-zero value, the Data-In Buffer and/or Data-Out Buffer shall consist of logical block data containing both user data and protection information.

4.21.3 Protection information format

Table 22 defines the placement of protection information in a logical block with a single protection information interval (i.e., the PROTECTION INTERVAL EXPONENT field is set to zero in the parameter list header for a FORMAT UNIT command (see 5.4.2.2))

.

Table 22 — Logical block data format with a single protection information interval

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)

Table 23 shows an example of the placement of protection information in a logical block with more than one protection information interval (i.e., the PROTECTION INTERVAL EXPONENT field is set to a non-zero value in the parameter list header for a FORMAT UNIT command (see 5.4.2.2)).

Table 23 — An example of the logical block data for a logical block with more than one protection information interval

Byte \ Bit	7	6	5	4	3	2	1	0
0	USER DATA [first]							
•••								
n - 1								
n	(MSB)	LOGICAL BLOCK GUARD [first]						
n + 1								(LSB)
n + 2	(MSB)	LOGICAL BLOCK APPLICATION TAG [first]						
n + 3								(LSB)
n + 4	(MSB)	LOGICAL BLOCK REFERENCE TAG [first]						
•••								
n + 7								(LSB)
n + 8	USER DATA [second]							
•••								
m - 1								
m	(MSB)	LOGICAL BLOCK GUARD [second]						
m + 1								(LSB)
m + 2	(MSB)	LOGICAL BLOCK APPLICATION TAG [second]						
m + 3								(LSB)
m + 4	(MSB)	LOGICAL BLOCK REFERENCE TAG [second]						
•••								
m + 7								(LSB)
	⋮							
	USER DATA [last]							
•••								
z - 1								
z	(MSB)	LOGICAL BLOCK GUARD [last]						
z + 1								(LSB)
z + 2	(MSB)	LOGICAL BLOCK APPLICATION TAG [last]						
z + 3								(LSB)
z + 4	(MSB)	LOGICAL BLOCK REFERENCE TAG [last]						
•••								
z + 7								(LSB)

Each USER DATA field shall contain user data.

Each LOGICAL BLOCK GUARD field contains a CRC (see 4.21.4). Only the contents of the USER DATA field immediately preceding THE LOGICAL BLOCK GUARD field (i.e., the user data between the preceding logical block reference tag, if any, and the current logical block guard) shall be used to generate and check the CRC contained in the LOGICAL BLOCK GUARD field.

Each LOGICAL BLOCK APPLICATION TAG field is set by the application client. If the device server detects a:

a) LOGICAL BLOCK APPLICATION TAG field set to FFFFh and type 1 protection (see 4.21.2.3) is enabled;
b) LOGICAL BLOCK APPLICATION TAG field set to FFFFh and type 2 protection (see 4.21.2.4) is enabled; or
c) LOGICAL BLOCK APPLICATION TAG field set to FFFFh, LOGICAL BLOCK REFERENCE TAG field set to FFFF_FFFFh, and type 3 protection (see 4.21.2.5) is enabled,

then the device server disables checking of all protection information for the associated protection information interval when performing:

a) a read operation; and
b) a write operation if the NO_PI_CHK bit is set to one in the Extended INQUIRY Data VPD page (see SPC-6).

Otherwise, if the ATMPE bit in the Control mode page (see SPC-6) is:

a) set to one, then the logical block application tags are defined by the Application Tag mode page (see 6.5.3); or
b) set to zero, then the logical block application tags 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-6). If the ATO bit is set to one in the Control mode page, then the device server shall not modify the LOGICAL BLOCK APPLICATION TAG field.

The contents of a LOGICAL BLOCK APPLICATION TAG field shall not be used to generate or check the CRC contained in the LOGICAL BLOCK GUARD field.

Table 24 indicates the value that shall be contained in the first LOGICAL BLOCK REFERENCE TAG field of the first logical block:

a) in the Data-In Buffer and/or Data-Out Buffer for commands other than WRITE SCATTERED commands;
b) in the Data-Out Buffer for each LBA range (see 4.35) for WRITE SCATTERED (16) commands (see 5.55); or
c) in the Data-Out Buffer for each LBA range (see 4.35) for WRITE SCATTERED (32) commands (see 5.56).

.

Table 24 — Content of the first LOGICAL BLOCK REFERENCE TAG field

Protection Type	Content of the first LOGICAL BLOCK REFERENCE TAG field [a]
Type 1 [b] protection (see 4.21.2.3)	The least significant four bytes of the LBA contained in the LOGICAL BLOCK ADDRESS field of the CDB for medium access commands (see 4.21.2.1) other than WRITE SCATTERED (16) commands (see 5.55). The least significant four bytes of the LBA contained in the LOGICAL BLOCK ADDRESS field of the associated LBA range descriptor for WRITE SCATTERED (16) commands..
Type 2 protection (see 4.21.2.4)	The value in the EXPECTED INITIAL LOGICAL BLOCK REFERENCE TAG field of the CDB for medium access commands other than WRITE SCATTERED (32) commands (see 5.56). The value in the EXPECTED INITIAL LOGICAL BLOCK REFERENCE TAG field of the associated LBA range descriptor for WRITE SCATTERED (32) commands.
Type 3 protection (see 4.21.2.5)	Not defined in this standard. If the ATO bit is set to zero in the Control mode page (see SPC-6), then this field may be modified by the device server. If the ATO bit is set to one in the Control mode page, then the device server shall not modify this field.

[a] The first logical block in the Data-In Buffer and/or Data-Out Buffer for media access commands other than WRITE SCATTERED commands or the first logical block in each LBA range in the Data-Out Buffer for WRITE SCATTERED commands

[b] The length of the protection information interval is equal to the logical block length (see 5.4.2).

Subsequent LOGICAL BLOCK REFERENCE TAG fields for a logical block in the Data-In Buffer and/or Data-Out Buffer shall be set as specified in table 25.

Table 25 — Content of subsequent LOGICAL BLOCK REFERENCE TAG fields for a logical block in the Data-In Buffer and/or Data-Out Buffer

Protection Type	The content of subsequent LOGICAL BLOCK REFERENCE TAG fields in the Data-In Buffer and/or Data-Out Buffer
Type 1 protection (see 4.21.2.3) and Type 2 protection (see 4.21.2.4)	The previous logical block reference tag plus one. If the contents of the previous LOGICAL BLOCK REFERENCE TAG field is FFFF_FFFFh, then the contents of the subsequent LOGICAL BLOCK REFERENCE TAG field is 0000_0000h.
Type 3 protection (see 4.21.2.5)	Not defined in this standard. If the ATO bit is set to zero in the Control mode page (see SPC-6), then this field may be modified by the device server. If the ATO bit is set to one in the Control mode page, then the device server shall not modify this field.

The contents of a LOGICAL BLOCK REFERENCE TAG field shall not be used to generate or check the CRC contained in the LOGICAL BLOCK GUARD field.

4.21.4 Logical block guard

4.21.4.1 Logical block guard overview

A LOGICAL BLOCK GUARD field shall contain a CRC that is generated from the contents of only the USER DATA field immediately preceding the LOGICAL BLOCK GUARD field.

Table 26 defines the CRC polynomials used to generate the logical block guard from the contents of the USER DATA field.

Table 26 — CRC polynomials

Function	Definition
F(x)	A polynomial representing the transmitted USER DATA field, which is 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 and the coefficient of the lowest order term shall be bit zero of the last byte of the USER DATA field.
F'(x)	A polynomial representing the received 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., in finite field notation G(x) = 1_8BB7h)
R(x)	The remainder polynomial calculated during CRC generation by the transmitter, representing the transmitted LOGICAL BLOCK GUARD field.
R'(x)	A polynomial representing the received LOGICAL BLOCK GUARD field.
RB(x)	The remainder polynomial calculated during CRC checking by the receiver. RB(x) = 0 indicates no error was detected.
RC(x)	The remainder polynomial calculated during CRC checking by the receiver. RC(x) = 0 indicates no error was detected.
QA(x)	The quotient polynomial calculated during CRC generation by the transmitter. The value of QA(x) is not used.
QB(x)	The quotient polynomial calculated during CRC checking by the receiver. The value of QB(x) is not used.
QC(x)	The quotient polynomial calculated during CRC checking by the receiver. The value of QC(x) is not used.
M(x)	A polynomial representing the transmitted USER DATA field followed by the transmitted LOGICAL BLOCK GUARD field.
M'(x)	A polynomial representing the received USER DATA field followed by the received LOGICAL BLOCK GUARD field.

4.21.4.2 CRC generation

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

The transmitter shall calculate the CRC by appending 16 zeros to F(x) and dividing by G(x) to obtain the remainder R(x):

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

R(x) is the CRC value, and is transmitted in the LOGICAL BLOCK GUARD field.

M(x) is the polynomial representing the USER DATA field followed by the LOGICAL BLOCK GUARD field (i.e., F(x) followed by R(x)):

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

4.21.4.3 CRC checking

M'(x) (i.e., the polynomial representing the received USER DATA field followed by the received LOGICAL BLOCK GUARD field) may differ from M(x) (i.e., the polynomial representing the transmitted USER DATA field followed by the transmitted LOGICAL BLOCK GUARD field) if there are transmission errors.

The receiver may check M'(x) validity by appending 16 zeros to F'(x) and dividing by G(x) and comparing the calculated remainder RB(x) to the received CRC value R'(x):

$$\frac{(x^{16} \times F'(x))}{G(x)} = QB(x) + \frac{RB(x)}{G(x)}$$

In the absence of errors in F'(x) and R'(x), the remainder RB(x) is equal to R'(x).

The receiver may check M'(x) validity by dividing M'(x) by G(x) and comparing the calculated remainder RC(x) to zero:

$$\frac{M'(x)}{G(x)} = QC(x) + \frac{RC(x)}{G(x)}$$

In the absence of errors in F'(x) and R'(x), the remainder RC(x) is equal to zero.

Both methods of checking M'(x) validity are mathematically equivalent.

4.21.4.4 CRC test cases

Several CRC test cases are shown in table 27.

Table 27 — 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

4.21.5 Application of protection information

Before an application client transmits or receives logical block data with protection information, the application client:

1) determines if a logical unit supports protection information using the INQUIRY command (see the PROTECT bit in the standard INQUIRY data in SPC-6);
2) if protection information is supported, then determines if the logical unit is formatted to accept protection information using the READ CAPACITY (16) command (e.g., see the PROT_EN bit and the P_TYPE field (see 5.21.2)); and
3) if the logical unit supports protection information and is not formatted to accept protection information, then formats the logical unit (see 5.4) with protection information enabled.

If the logical unit supports protection information and is formatted to accept protection information, then the application client may use read commands that support protection information and should use verify commands and write commands that support protection information.

4.21.6 Protection information and commands

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

The commands that are affected while protection information is enabled are listed in table 34.

Commands that cause a device server to return the length in bytes of each logical block (e.g., the MODE SENSE (10) command and the READ CAPACITY (16) command) shall cause the device server to return the combined length of the USER DATA field(s) contained in the logical block, not including the length of any protection information (i.e., the LOGICAL BLOCK GUARD field(s), the LOGICAL BLOCK APPLICATION TAG field(s), and the LOGICAL BLOCK REFERENCE TAG field(s)) (e.g., if the user data plus the protection information is equal to 520 bytes and there is one protection information interval, then 512 is returned).

4.22 Grouping function

4.22.1 Grouping function overview

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). Groups are identified with the GROUP NUMBER field in the CDBs of certain commands (e.g., the PRE-FETCH (10) command (see 5.13)).

Support for the grouping function is indicated in the GROUP_SUP bit in the Extended INQUIRY Data VPD page (see SPC-6). The ways that the grouping function uses the GROUP NUMBER field are defined by the IO Advice Hints Grouping mode page (see 6.5.7) and the Group Statistics and Performance (1 to 31) log pages (see SPC-6).

EXAMPLE - In a SCSI domain in which two applications are using a subsystem where one application streams data and another accesses data randomly, if:

a) the streaming application groups all of its commands with one group number (e.g., x); and
b) the random application groups all of its commands with another group number (e.g., y),

then the applications use those group numbers (e.g., x and y) to collect separate performance metrics for each application.

A management application then reads the performance metrics and determines if the performance of a specific group is acceptable.

Device servers that support the grouping function shall support at least group 0 to group 31.

4.22.2 Grouping function extensions for IO advice hints

The IO Advice Hints Grouping mode page (see 6.5.7) may be used to ~~enable or disable the collection of information, and define~~ manage the IO advice hints associated with each group. The IO Advice Hints Grouping mode page contains 64 IO advice hints group descriptors (i.e., one descriptor for each possible group number) (see table 245) that are in ascending order of group number. The IO advice hints group number associated with each IO advice hints group descriptor is calculated as follows:

$$\text{IO advice hints group number} = (\text{offset} \div 16) - 1$$

where:

offset the offset in bytes of the IO advice hints group descriptor from the beginning of the IO Advice Hints Grouping mode page

If the GROUP_SUP bit in the Extended INQUIRY Data VPD page is set to one, then for all commands that contain a GROUP NUMBER field, ~~the~~ that GROUP NUMBER field specifies:

a) the IO advice hints group number that the device server shall use to associate IO advice hints with all operations associated with that command; ~~and~~
b) the cache ID (see 4.15.2) that the device server shall use for all operations associated with that command, if the IC_ENABLE bit is set to one; and
c) the stream identifier for that command, if the ST_ENBLE bit is set to one.

In the IO advice hints group descriptor for the specified group number in the IO Advice Hints Grouping mode page, if the IO ADVICE HINTS MODE field (see table 246) is set to:

a) 00b, then the device server shall use the IO advice hints in the IO advice hints group descriptor specified by the IO advice hints group number to associate IO advice hints (see 4.30); and
b) 01b, then the device server may associate vendor specific IO advice hints,

with each operation associated with the command.

If the ST_ENBLE bit is set to one in the IO advice hints group descriptor in the IO Advice Hints Grouping mode page, then the device server should use the contents of GROUP NUMBER field in a write stream command (see 4.32.2) as the stream identifier for the processing of that command.

If the CS_ENBLE bit is set to one in the IO advice hints group descriptor in the IO Advice Hints Grouping mode page, then the device server should use the cache segments associated with the ~~specified~~ cache ID specified by the contents of the GROUP NUMBER field.

If the IC_ENABLE bit is set to one in the IO advice hints group descriptor in the IO Advice Hints Grouping mode page, then the device server shall collect information about attributes associated with commands (i.e., information about commands with the same group value are collected into the specified group)~~perform the information collection function as described in 4.22.1~~.

Device servers that support the IO Advice Hints Grouping mode page ~~shall~~ should support group 0 to group 63.

If a MODE SELECT command specifies a combination of the IO ADVICE HINTS MODE field, the ST_ENBLE bit, the CS_ENBLE bit, and the IC_ENABLE bit that the device server does not support, then the device server may terminate that command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

4.23 Background scan operations

4.23.1 Background scan overview

A background scan operation is either a background pre-scan operation (see 4.23.2) or a background medium scan operation (see 4.23.3).

During a background scan operation, the device server performs read medium operations for the purpose of:

a) identifying logical blocks that are difficult to read (i.e., recoverable) or unreadable (i.e., unrecoverable);
b) logging problems encountered during the background scan operation; and
c) when allowed, taking a vendor specific action to repair recoverable logical blocks or perform automatic read reallocation of recoverable logical blocks.

During a background scan operation, if a read medium operation encounters a recovered error (i.e., a logical block is readable but requires extra actions (e.g., retries or application of a correction algorithm) to be read), then the device server may resolve the problem using vendor specific means. The value of the ARRE bit (see 6.5.10) determines whether or not the device server performs automatic read reassignment.

During a background scan operation, if a read medium operation encounters an unrecovered error (i.e., a logical block is unreadable), then the device server may mark the logical block unrecoverable. The value of the AWRE bit (see 6.5.10) determines whether or not the device server performs automatic write reassignment. If the AWRE bit is set to one, then the device server performs automatic write reassignment at the start of the next write medium operation accessing that logical block.

During a background scan operation, the device server:

a) may scan the logical blocks in any order (e.g., based on physical block layout);
b) should not retain any data from logical blocks in cache memory after the logical blocks are read;
c) shall ignore pseudo unrecovered errors with correction disabled (see 4.18.2); and
d) shall process pseudo unrecovered errors with correction enabled.

4.23.2 Background pre-scan operations

4.23.2.1 Enabling background pre-scan operations

A background pre-scan operation is enabled after:

1) the EN_PS bit (see 6.5.4) is set to zero:
2) the EN_PS bit is set to one; and
3) the SCSI device is power cycled if;
   A) the S_L_FULL bit (see 6.5.4)is:
      a) set to zero; or
      b) set to one and the Background Scan log parameters (see 6.4.2) are not all used;

   and

   B) the saved value of the EN_PS bit is set to one.

After a background pre-scan operation is enabled, the device server shall:

a) initialize the Background Pre-scan Time Limit timer to the time specified in the BACKGROUND PRE-SCAN TIME LIMIT field (see 6.5.4) and start the timer;
b) initialize the Background Medium Scan Interval timer to the time specified in the BACKGROUND MEDIUM SCAN INTERVAL TIME field (see 6.5.4) and start the timer; and
c) begin the background pre-scan operation (i.e., begin scanning the medium).

4.23.2.2 Suspending and resuming background pre-scan operations

A background pre-scan operation shall be suspended when any of the following occurs:

a) a command or task management function is processed that requires the background pre-scan operation to be suspended;
b) a SCSI event (e.g., a hard reset) (see SAM-6) is processed that requires the background pre-scan operation to be suspended;
c) a power condition timer expires (see the Power Condition mode page in SPC-6), and the PM_BG_PRECEDENCE field in the Power Condition mode page is set to 10b; or
d) the S_L_FULL bit (see 6.5.4) is set to one, and the Background Scan log parameters (see 6.4.2) are all used.

If a command is received that requires a background pre-scan operation to be suspended, then the following should occur within the time specified in the MAXIMUM TIME TO SUSPEND BACKGROUND SCAN field (see 6.5.4):

a) the logical unit suspends the background medium scan operation; and
b) the device server begins processing the command.

If a background pre-scan operation is suspended, then the device server shall not stop:

a) the Background Pre-scan Time Limit timer;

b) the Background Medium Scan Interval timer; and
c) any process that results in an event that causes a background function to occur (e.g., not stop any timers or counters associated with background functions).

While a background pre-scan operation is suspended and not halted (see 4.23.3.2), the device server shall convert each write operation accessing a logical block that has not been scanned during the background pre-scan operation into a write medium operation followed by a verify medium operation in order to verify that the logical block data just written was read back without error. If a write medium operation accesses a logical block that has already been scanned during the background pre-scan operation, then the device server shall not perform the additional verify medium operation.

A background pre-scan operation shall be resumed from where the operation was suspended when:

a) there are no commands in any task set to be processed;
b) there are no task management functions to be processed;
c) there are no SCSI events to be processed;
d) no ACA condition exists;
e) the PM_BG_PRECEDENCE field in the Power Condition mode page is set to 10b (see SPC-6), but no power condition timer defined in the Power Condition mode page has expired;
f) the S_L_FULL bit is set to zero (see 6.5.4), or the Background Medium Scan log parameters (see 6.4.2)are not all used;
g) the logical unit has been idle for the time specified in the MINIMUM IDLE TIME BEFORE BACKGROUND SCAN field (see 6.5.4); and
h) the background pre-scan operation has not been halted (see 4.23.3.2).

4.23.2.3 Halting background pre-scan operations

The device server shall halt a background pre-scan operation if any of the following occurs:

a) the background pre-scan operation completes scanning all logical blocks on the medium;
b) an application client sets the EN_PS bit to zero in the Background Control mode page (see 6.5.4);
c) the Background Pre-scan Time Limit timer expires;
d) the device server detects a fatal error;
e) the device server detects a vendor specific pattern of errors;
f) the device server detects a medium formatted without a PLIST (see 4.13); or
g) the device server detects temperature out of range.

After a background pre-scan operation has been halted, the device server shall not enable a background operation until the conditions in 4.23.2.1 are met.

4.23.3 Background medium scan

4.23.3.1 Enabling background medium scan operations

Background medium scan operations are enabled if:

a) a background pre-scan operation (see 4.23.2) is not in progress;
b) the S_L_FULL bit (see 6.5.4) is:
    A) set to zero; or
    B) set to one and the Background Scan log parameters (see 6.4.2) are not all used;

    and

c) the EN_BMS bit (see 6.5.4)is set to one.

If background medium scan operations are enabled, then the device server shall begin a background medium scan operation (i.e., begin scanning the medium) when:

a) the Background Medium Scan Interval timer has expired; and
b) the logical unit has been idle for the time specified in the MINIMUM IDLE TIME BEFORE BACKGROUND SCAN field (see 6.5.4).

After power on, if background pre-scan operations are not enabled (see 4.23.2.1), then the device server shall set the Background Medium Scan Interval timer to zero (i.e., expired).

Whenever a background medium scan operation begins, the device server shall set the Background Medium Scan Interval timer to the time specified in the BACKGROUND MEDIUM SCAN INTERVAL TIME field (see 6.5.4) and start the timer.

4.23.3.2 Suspending and resuming background medium scan operations

The logical unit shall suspend a background medium scan operation if any of the following occurs:

a) a command or task management function is processed that requires the background medium scan operation to be suspended;
b) a SCSI event (e.g., a hard reset) (see SAM-6) is processed that requires the background medium scan operation to be suspended;
c) a power condition timer expires (see the Power Condition mode page in SPC-6), and the PM_BG_PRECEDENCE field in the Power Condition mode page is set to 10b;
d) the S_L_FULL bit (see 6.5.4) is set to one, and the Background Scan log parameters (see 6.4.2) are all used; or
e) an application client sets the EN_BMS bit to zero (see 6.5.4).

If a command is received that requires a background medium scan operation to be suspended, then the following should occur within the time specified in the MAXIMUM TIME TO SUSPEND BACKGROUND SCAN field (see 6.5.4):

a) the logical unit suspends the background medium scan operation; and
b) and the device server begins processing the command.

If a background pre-scan operation is suspended, then the device server shall not stop:

a) the Background Medium Scan Interval timer; and
b) any process that results in an event that causes a background function to occur (e.g., not stop any timers or counters associated with background functions).

The logical unit shall resume a suspended background medium scan operation from where the operation was suspended when:

a) there are no commands in any task set to be processed;
b) there are no task management functions to be processed;
c) there are no SCSI events to be processed;
d) the PM_BG_PRECEDENCE field in the Power Condition mode page is set to 10b (see SPC-6), but no power condition timer defined in the Power Condition mode page has expired;
e) the S_L_FULL bit (see 6.5.4)is set to zero , or the Background Medium Scan log parameters in the Background Scan Results log page are not all used;
f) the EN_BMS (see 6.5.4)is set to one; and
g) the logical unit has been idle for the time specified in the MINIMUM IDLE TIME BEFORE BACKGROUND SCAN field (see 6.5.4).

4.23.3.3 Halting background medium scan operations

The device server shall halt background medium scan operations if any of the following occurs:

a) the background medium scan operation completes scanning all logical blocks on the medium;
b) the device server detects a fatal error;
c) the device server detects a vendor specific pattern of errors;
d) the device server detects a medium formatted without a PLIST (see 4.13); or
e) the device server detects temperature out of range.

After background medium scan operations have been halted, the device server shall not enable a background medium scan operation until the conditions in 4.23.3.1 are met.

4.23.4 Interpreting the logged background scan results

An application client may:

a) poll the Background Scan Results log page (see 6.4.2) to get information about background pre-scan and background medium scan activity; or
b) use the EBACKERR bit and the MRIE field (see 6.5.8) to select a method of indicating that a medium error was detected.

If the EBACKERR bit is set to one and a medium error was detected, then the device server shall return the following additional sense codes using the method defined by the value in the MRIE field:

a) WARNING - BACKGROUND PRE-SCAN DETECTED MEDIUM ERROR, if the failure occurs during a background pre-scan operation; or
b) WARNING - BACKGROUND MEDIUM SCAN DETECTED MEDIUM ERROR, if the failure occurs during a background medium scan operation.

The Background Scan Status log parameter (see 6.4.2.2) in the Background Scan Results log page (see 6.4.2) indicates:

a) whether or not a background scan operation is active or halted;
b) the number of background scan operations that have been performed on the medium; and
c) the progress of a background scan operation, if active.

This information may be used by an application client to monitor the background scan operations and should be used by an application client after notification via an informational exception (see 6.5.8).

The Background Scan Results log parameters (see 6.4.2.3), if any, in the Background Scan Results log page describe the LBA and the reassignment status of each logical block that generated recovered errors or unrecovered errors during the background scan's read medium operations.

After an application client analyzes the Background Scan Results log parameters and has completed actions, if any, to repair any of the indicated LBAs, the application client may delete all Background Scan Results log parameters by issuing a LOG SELECT command (e.g., with the PCR bit set to one in the CDB or with the PC field set to 11b and the PARAMETER LIST LENGTH field set to zero in the CDB) (see SPC-6).

A background medium scan operation may continue to run during log page accesses. To ensure that the values in the Background Scan Results log page do not change during a sequence of accesses, the application client:

1) sets the EN_BMS bit to zero in the Background Control mode page in order to suspend the background medium scan operation;
2) reads the Background Scan Results log page with a LOG SENSE command;
3) processes the Background Scan Results log page;
4) deletes the Background Scan Results log page entries with the LOG SELECT command (e.g., with the PCR bit set to one in the CDB); and
5) sets the EN_BMS bit to one in the Background Control mode page in order to re-enable the background scan operation.

4.24 Deferred microcode activation

After receiving a FORMAT UNIT command (see 5.4) or a START STOP UNIT command (see 5.31), a device server shall, prior to processing the command, activate any deferred microcode that has been downloaded as a result of a WRITE BUFFER command with the MODE field set to 0Eh (see SPC-6).

4.25 Model for uninterrupted sequences on LBA ranges

Direct access block devices may perform commands that require an uninterrupted sequence of actions to be performed on a specified range of LBAs. The uninterrupted sequence requirements are described in table 28. The uninterrupted sequences do not impact the processing of commands that access logical blocks other than those specified in the command requiring an uninterrupted sequence. The task attribute (see SAM-6) controls interactions between multiple commands. Commands with uninterrupted sequences on LBA ranges are shown in table 28.

Table 28 — Commands that require uninterrupted sequences

Command	Consistency enforcement	Reference
ORWRITE (16) ORWRITE (32)	The device server shall not perform any operations requested by any other command in the task set on logical blocks in the range specified by the command that requires an uninterrupted sequence of actions while performing the specified uninterrupted sequence of actions.	4.27
COMPARE AND WRITE	The device server shall not perform: a) any operations requested by any COMPARE AND WRITE command in the task set on logical blocks in the range specified by the command that requires an uninterrupted sequence of actions while performing the specified uninterrupted sequence of actions; b) any write operations to or unmap operations on logical blocks in the range specified by the command that requires an uninterrupted sequence of actions while performing the read operations specified in the uninterrupted sequence of actions; and c) any read operations or verify operations from logical blocks in the range specified by the command that requires an uninterrupted sequence of actions while performing the write operations specified in the uninterrupted sequence of actions.	5.3

4.26 Referrals

4.26.1 Referrals overview

Referrals allow a logical unit to inform an application client that one or more user data segments (i.e., ranges of logical blocks) are accessible through target port group(s).

Support for referrals is indicated by the device server setting the R_SUP bit to one in the Extended INQUIRY Data VPD page (see SPC-6).

An application client may determine information on referrals by:

a) issuing commands; or
b) monitoring sense data returned as part of a completed command or a terminated command.

Figure 12 shows an example of how a logical unit informs an application client that one or more user data segments are accessible through target port groups.

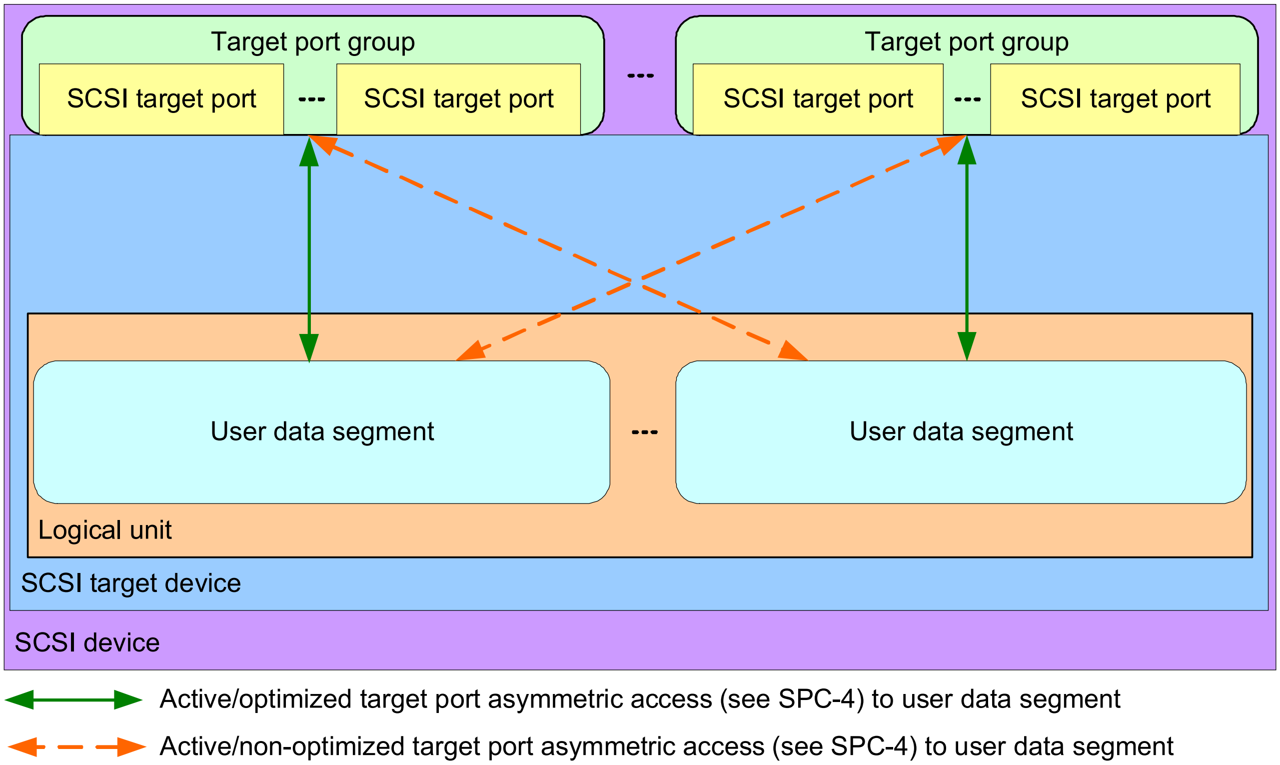


Figure 12 — Referrals

4.26.2 Discovering referrals

An application client may determine referrals information on a logical unit by:

1) determining if the R_SUP bit is set to one (i.e., the logical unit supports referrals) in the Extended INQUIRY Data VPD page (see SPC-6);
2) requesting the user data segment information from the Referrals VPD page (see 6.6.10);
3) requesting a list of target port groups by issuing a REPORT TARGET PORT GROUPS command (see SPC-6); and
4) either:
   A) requesting referrals information by issuing a REPORT REFERRALS command (see 5.28); or
   B) monitoring for referral information in sense data returned by the device server (see 4.26.3).

The following calculation is used to determine the first LBA for each user data segment within the range of LBAs indicated by the user data segment referral descriptors (see table 18) returned in:

a) the REPORT REFERRALS parameter data (see table 109); or
b) the user data segment referrals sense data descriptor (see 4.18.4):

first LBA of the current user data segment = first LBA + (segment size × segment multiplier)

where:

first LBA	the initial value is the first user data segment LBA specified in the user data segment referral descriptor (see table 18). Subsequent values, if any, are the first LBA of the previous user data segment;
segment size	the content of the USER DATA SEGMENT SIZE field (see 6.6.10); and
segment multiplier	the content of the USER DATA SEGMENT MULTIPLIER field (see 6.6.10).

If the content of the USER DATA SEGMENT SIZE field is greater than zero, and the content of the USER DATA SEGMENT MULTIPLIER field is greater than zero, then the following calculation may be used to determine the last LBA for each user data segment within the range of LBAs indicated by the user data segment referral descriptors (see table 18) returned in:

a) the REPORT REFERRALS parameter data (see table 109); or
b) the user data segment referrals sense data descriptor (see 4.18.4):

last LBA of the current user data segment = first LBA + (segment size – 1)

where:

first LBA	the first LBA of the current user data segment;
segment size	the content of the USER DATA SEGMENT SIZE field (see 6.6.10).

If the content of the USER DATA SEGMENT SIZE field is zero, then there is only one user data segment, and the last LBA of that user data segment is equal to the last LBA specified in the last USER DATA SEGMENT LBA field (see table 18).

See annex G for examples for discovering referrals.

4.26.3 Referrals in sense data

Returning referral information in sense data is enabled if the:

a) R_SUP bit in the Extended INQUIRY Data VPD page is set to one (i.e., the logical unit supports referrals) (see SPC-6); and
b) D_SENSE bit in the Control mode page is set to one (i.e., returning descriptor formatted sense data is enabled) (see SPC-6).

If reporting of referrals in sense data is enabled, a command completes without error, no other sense data is available within the logical unit, and the device server has an alternate I_T_L nexus that an application client should use to access at least one of the specified logical blocks, then the device server shall complete the command with GOOD status with the sense key set to COMPLETED, the additional sense code set to INSPECT REFERRALS SENSE DESCRIPTORS, and a user data segment referrals sense data descriptor (see 4.18.4).

The user data segment referral sense data descriptor (see 4.18.4) shall define the description of as many complete user data segments (i.e., one user data segment referral descriptor contains one complete user data segment) that fit in the maximum number of bytes allowed for sense data (i.e., 244 bytes or the maximum supported sense data length indicated in the Extended INQUIRY Data VPD page (see SPC-6)). If all the user data segments do not fit within the maximum number of bytes allowed for sense data, then:

a) the device server shall set the NOT_ALL_R bit to one in the user data segment referral sense data descriptor (see 4.18.4); and
b) the selection of which user data segments to include in the user data segment referral sense data descriptor is vendor specific.

Each user data segment referral sense data descriptor (see 4.18.4) contains information on alternate I_T_L nexuses to user data segments that the application client should use to access LBAs within the LBA range(s) indicated by the user data segments.

If reporting of referrals in sense data is enabled, the device server receives a command for which the device server is not able to access user data associated with the requested command, and the inaccessible user data is accessible through another target port group, then the device server shall terminate the command with CHECK CONDITION status with the sense key set to ABORTED COMMAND, the additional sense code set to INSPECT REFERRALS SENSE DESCRIPTORS, and a user data segment referral sense data descriptor. The user data segment referral sense data descriptor shall, at a minimum, indicate the user data segment that contains the LBA of the first inaccessible logical block. Any other type of error that occurs while processing the command shall take precedence and be reported as described in this standard. If any other type of error occurs while the device server is processing the command, then processing that error shall take precedence over processing the command, and the device server shall report the error as described in this standard.

If reporting of referrals in sense data is disabled (see 4.26.1), the device server receives a command for which the device server is not able to access user data associated with the requested command, and the inaccessible user data is accessible through another target port group, then the device server shall terminate the command with CHECK CONDITION status with the sense key set to HARDWARE ERROR and the additional sense code set to INTERNAL TARGET FAILURE.

4.27 ORWRITE commands

4.27.1 ORWRITE commands overview

The ORWRITE commands (see 5.10 and 5.11) provide a mechanism for an application client to manipulate bitmap structures on direct access block devices.

An ORWRITE command shall be processed by the device server performing the following as an uninterrupted sequence of actions (see 4.25):

1) perform read operations from the LBAs specified by this command;
2) transfer the specified number of logical blocks from the Data-Out Buffer;
3) perform the specified Boolean arithmetic function on:
   A) the user data contained in the logical blocks from read operations; and
   B) the user data contained in the logical blocks transferred from the Data-Out Buffer;
4) store the results of the Boolean arithmetic function in a bitmap buffer;
5) generate new protection information, if any, from the stored results;
6) store the generated protection information, if any, into the bitmap buffer; and
7) perform write operations using the updated logical block data from the bitmap buffer.

If the check of the protection information from the read operations is successful (see table 69), and the check of the protection information transferred from the Data-Out Buffer is successful (see table 70), then the device server shall generate the new protection information (see 4.21) as follows:

a) set the LOGICAL BLOCK GUARD field to the CRC (see 4.21.4) generated from the bitmap buffer by the device server;
b) set the LOGICAL BLOCK REFERENCE TAG field to the LOGICAL BLOCK REFERENCE TAG field received from the Data-Out Buffer; and
c) set the LOGICAL BLOCK APPLICATION TAG field to the LOGICAL BLOCK APPLICATION TAG field received from the Data-Out Buffer.

In order to support the manipulation of bitmap structures:

a) the ORWRITE (16) command supports the set operation (see 4.27.4); and
b) the ORWRITE (32) command supports:
   A) the set operation; and
   B) the change generation and clear operation (see 4.27.3).

4.27.2 ORWgeneration code

4.27.2.1 ORWgeneration code overview

The ORWRITE commands use a generation code for synchronization. The device server shall establish and maintain the following generation codes:

a) a current ORWgeneration code; and
b) a previous ORWgeneration code.

Subsequent ORWRITE command processing by the device server is dependent on comparisons involving the ORWgeneration codes. Changes in these ORWgeneration codes define a synchronization point in the management of the bitmap.

4.27.2.2 ORWgeneration code processing

The device server shall maintain at least one current ORWRITE processing policy. The device server may support more than one ORWRITE processing policy (see table 29 in 4.27.4).

When processing an ORWRITE (32) command (see 5.11), the device server compares the value in the EXPECTED ORWGENERATION field in the CDB to the current ORWgeneration code (see 4.27.2.1), and:

a) if the two values are equal, then the device server continues processing the ORWRITE (32) command as described in table 29 for the set operation and as described in 4.27.3 for the change generation and clear operation; or
b) if the two values are not equal, then:
   A) for a set operation, the current ORWRITE processing policy (see table 29) determines how the device server continues processing the ORWRITE (32) command; and
   B) for a change generation and clear operation, the device server terminates the ORWRITE (32) command (see 4.27.3).

If the device server supports both the ORWRITE (16) command (see 5.10) and the ORWRITE (32) command, then the device server shall process all ORWRITE (16) commands as if they contained an EXPECTED ORWGENERATION field set to zero.

4.27.3 Change generation and clear operation

The change generation portion of the change generation and clear operation is used to establish a point of synchronization. The clear portion of the change generation and clear operation is used to set zero or more bits in the bitmap structure to zero.

The device server performs a change generation and clear operation if:

a) the BMOP field (see 5.11) is set to 001b; and
b) the value in the EXPECTED ORWGENERATION field is equal to the current ORWgeneration code in the device server.

If the device server performs a change generation and clear operation, then the device server shall perform the following as an uninterrupted sequence:

1) perform read operations from the LBAs specified by this command;
2) transfer the specified logical blocks from the Data-Out Buffer;
3) perform an AND operation (see 3.1.4) on the user data contained in the logical blocks from the read operations and the user data contained in the logical blocks transferred from the Data-Out Buffer;
4) store the results of the AND operation in a bitmap buffer;
5) generate new protection information, if any, from the stored results;
6) store the generated protection information, if any, into the bitmap buffer;
7) perform write operations using the updated logical block data from the bitmap buffer;
8) set the current ORWRITE processing policy to the value in the PREVIOUS GENERATION PROCESSING field (see 5.11);
9) set the previous ORWgeneration code (see 4.27.2) to the current ORWgeneration code in the device server; and
10) set the current ORWgeneration code (see 4.27.2) to the value in the NEW ORWGENERATION field(see 5.11).

If the value in the EXPECTED ORWGENERATION field is not equal to the current ORWgeneration code, then the device server shall terminate the ORWRITE (32) command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to ORWRITE GENERATION DOES NOT MATCH.

If a power on or hard reset condition occurs, then the device server shall set:

a) the current ORWgeneration code to zero;
b) the previous ORWgeneration code to zero; and
c) the current ORWRITE processing policy to 7h.

The device server shall preserve the following across a logical unit reset:

a) the current ORWgeneration code;
b) the previous ORWgeneration code; and
c) the current ORWRITE processing policy.

4.27.4 Set operation

The set operation is used to set zero or more bits in the bitmap structure to one.

The device server performs a set operation for an ORWRITE command (see 5.10 and 5.11) if the BMOP field in the CDB is set to 000b.

The device server shall perform a set operation by performing the actions specified in table 29, which shows the current ORWgeneration code, the previous ORWgeneration code, and the device server's current ORWRITE processing policy (see 4.27.3).

Table 29 — Performing an ORWRITE set operation

Current ORWRITE processing policy	The value in the EXPECTED ORWGENERATION field matches		
	Current ORWgeneration code	Previous ORWgeneration code	Any other value
0h	PA	PA	CCG
1h	Reserved		
2h	PA	DN	CCG
3h	PA	PA	PA
4h	Reserved		
5h	PA	DN	DN
6h	Reserved		
7h	PA	CCG	CCG
8h to Fh	Reserved		

Key:

PA = the device server shall perform the following as an uninterrupted sequence:

1) perform read operations from the LBAs specified by the command;
2) transfer the specified logical blocks from the Data-Out Buffer;
3) perform an OR operation on the logical blocks from the read operations and the user data contained in the logical blocks transferred from the Data-Out Buffer;
4) store the results of the OR operation in a bitmap buffer;
5) generate new protection information, if any, from the stored results;
6) store the generated protection information, if any, into the bitmap buffer; and
7) perform write operations using the updated logical block data (i.e., those containing the user data resulting from the OR operation and the generated protection information, if any) from the bitmap buffer.

DN = the device server shall discard the contents of the Data-Out Buffer and shall complete the command with GOOD status.

CCG = 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 ORWRITE GENERATION DOES NOT MATCH

4.28 Block device ROD token operations

4.28.1 Block device ROD token operations overview

Application clients request that block device ROD token operations (see SPC-6) be performed using the commands summarized in this subclause or the commands specified in SPC-6.

Copy managers (see SPC-6) that implement the POPULATE TOKEN command (see 5.12) or the WRITE USING TOKEN command (see 5.59) shall support the following:

a) the POPULATE TOKEN command;
b) the WRITE USING TOKEN command;
c) the RECEIVE COPY STATUS (LID4) command (see SPC-6);
d) the RECEIVE ROD TOKEN INFORMATION command (see SPC-6 and 5.25); and
e) the Third-party Copy VPD page (see 6.6.11) containing at least one Block Device ROD token limits descriptor (see 6.6.11.3).

The POPULATE TOKEN command may cause the copy manager to create zero or one point in time ROD tokens. If the POPULATE TOKEN command causes one point in time ROD token to be created, then this point in time ROD token may be retrieved by an application client using the RECEIVE ROD TOKEN INFORMATION command.

The WRITE USING TOKEN command causes the copy manager to transfer the data represented by the specified ROD token (i.e., the data represented by the ROD token retrieved using the RECEIVE ROD TOKEN INFORMATION command or the data represented by the block device zero ROD token).

The copy manager manages the point in time ROD token.

After the copy manager begins processing a POPULATE TOKEN command or a WRITE USING TOKEN command, the copy manager shall preserve information for return in response to a RECEIVE ROD TOKEN INFORMATION command as defined in SPC-6.

Block device range descriptor lists (see 5.12.3) contain non-overlapping block device range descriptors and are used by the application client to specify:

a) the logical blocks to include in the data represented by the ROD token;
b) the sequence of the logical blocks in the data represented by the ROD token (e.g., the first logical block represented by the LBA described in the first block device range descriptor is placed at the beginning of the data represented by the ROD token, and the first logical block represented by the LBA described in the second block device range descriptor is placed in the data represented by the ROD token immediately following the last logical block represented by the LBA described in the first block device range descriptor);
c) the logical blocks to be written from the data represented by the ROD token; and
d) the sequence of the logical blocks written from the data represented by the ROD token (e.g., the first logical block represented by the LBA described in the first block device range descriptor is written from the beginning of the data represented by the ROD token, and the first logical block represented by the LBA described in the second block device range descriptor is written from data represented by the ROD token immediately following the data written to the last logical block represented by the LBA described in the first block device range descriptor).

If the copy manager uses out of order transfers to create the representation of data for the ROD token, then the TRANSFER COUNT field in the parameter data returned in the response to a RECEIVE ROD TOKEN INFORMATION command with a list identifier that specifies a POPULATE TOKEN command (see 5.25.2) shall be based only on the contiguous transfers that complete without error starting at the first LBA specified by the first block device range descriptor (i.e., any transfers completed without error beyond the first incomplete or unsuccessful transfer shall not contribute to the computation of the value in the TRANSFER COUNT field).

If the copy manager uses out of order transfers to write from the data represented by the ROD token, then the TRANSFER COUNT field in the parameter data returned in response to a RECEIVE ROD TOKEN INFORMATION command with a list identifier that specifies a WRITE USING TOKEN command (see 5.25.3) shall be based only on the contiguous transfers that complete without error starting at the first LBA specified by the first block device range descriptor (i.e., any transfers completed without error beyond the first incomplete or unsuccessful transfer shall not contribute to the computation of the value in the TRANSFER COUNT field).

4.28.2 POPULATE TOKEN command and WRITE USING TOKEN command completion

As part of completing a block device token operation originated by a POPULATE TOKEN command (see 5.12) or a WRITE USING TOKEN command (see 5.59), the copy manager shall compute the residual by subtracting the sum of the contents of the NUMBER OF LOGICAL BLOCK fields in all of the complete block device range descriptors of the parameter list (see 5.12.3) from the TRANSFER COUNT field in the parameter data returned in response to a RECEIVE ROD TOKEN INFORMATION command (see 5.25.2 and 5.25.3).

If the POPULATE TOKEN command was received with the IMMED bit set to zero, and the residual is negative, then the copy manager shall:

a) terminate the command with CHECK CONDITION status, with the additional sense code set to COPY TARGET DEVICE DATA UNDERRUN, the sense key set to:
   A) COPY ABORTED, if the transfer count is not zero; or
   B) ILLEGAL REQUEST, if the transfer count is zero,

   and report the transfer count in the INFORMATION field (see SPC-6); or
b) complete the command with GOOD status and return parameter data for the RECEIVE ROD TOKEN INFORMATION command received on the same I_T nexus with a matching LIST IDENTIFIER field with:
   A) the COPY OPERATION STATUS field set to 03h or set to 60h (see SPC-6);
   B) the EXTENDED COPY COMPLETION STATUS field set to CHECK CONDITION (see SAM-6); and
   C) the SENSE DATA field with the additional sense code set to COPY TARGET DEVICE DATA UNDERRUN, the sense key set to:
      a) COPY ABORTED, if the transfer count is not zero; or
      b) ILLEGAL REQUEST, if the transfer count is zero,

   and report the transfer count in the INFORMATION field (see SPC-6).

If the WRITE USING TOKEN command was received with the IMMED bit set to zero, and the residual is negative, then the copy manager shall terminate the command with CHECK CONDITION status, the additional sense code set to COPY TARGET DEVICE DATA UNDERRUN, the sense key set to:

a) COPY ABORTED, if the transfer count is not zero; or
b) ILLEGAL REQUEST, if the transfer count is zero,

and report the transfer count in the INFORMATION field (see SPC-6).

If the POPULATE TOKEN command or WRITE USING TOKEN command was received with the IMMED bit set to one, and the residual is negative, then the copy manager shall return parameter data for the RECEIVE ROD TOKEN INFORMATION command received on the same I_T nexus with a matching LIST IDENTIFIER field with:

a) the COPY OPERATION STATUS field set to 03h or 60h (see SPC-6);
b) the EXTENDED COPY COMPLETION STATUS field set to CHECK CONDITION status (see SAM-6); and
c) the SENSE DATA field with the additional sense code set to COPY TARGET DEVICE DATA UNDERRUN, the sense key set to:
   A) COPY ABORTED, if the transfer count is not zero; or
   B) ILLEGAL REQUEST, if the transfer count is zero,

   and report the transfer count in the INFORMATION field (see SPC-6).

4.28.3 Block device specific ROD tokens

Block device specific ROD token types (see SPC-6) are shown in table 30.

Table 30 — ROD token type values

ROD token type	Description	Reference
FF00_0000h to FFFF_0000h	Reserved	
FFFF_0001h	Block device zero ROD token	4.28.4
FFFF_0002h to FFFF_FFEFh	Reserved	

4.28.4 Block device zero ROD token

The block device zero ROD token represents user data in which all bits are set to zero and protection information, if any, with the:

a) LOGICAL BLOCK GUARD field set to FFFFh or set to 0000h (i.e., the valid CRC for user data in which all bits are set to zero);
b) LOGICAL BLOCK APPLICATION TAG field set to FFFFh; and
c) LOGICAL BLOCK REFERENCE TAG field set to FFFF_FFFFh.

If user data with all bits set to zero and protection information, if any, with the:

a) LOGICAL BLOCK GUARD field set to FFFFh or set to 0000h;
b) LOGICAL BLOCK APPLICATION TAG field set to FFFFh; and
c) LOGICAL BLOCK REFERENCE TAG field set to FFFF_FFFFh,

is represented by a ROD token, then, in response to a RECEIVE ROD TOKEN INFORMATION command in which the LIST IDENTIFIER field specifies a POPULATE TOKEN command, the copy manager may or may not return a ROD token that is the block device zero ROD token. The block device zero ROD token format is shown in table 31.

Table 31 — Block device zero ROD token format

Byte \ Bit	7	6	5	4	3	2	1	0
0	(MSB)							
•••	ROD TOKEN TYPE (FFFF_0001h)							
3								(LSB)
4	Reserved							
5								
6	(MSB)							
7	ROD TOKEN LENGTH (01F8h)							(LSB)
8								
•••	Reserved							
63								
64	Reserved							CRC VALID
65								
•••	Reserved							
511								

The logical block length associated with the block device zero ROD token is that of the direct access block device to which the data is being written (e.g., if a block device zero ROD token is used to write to a logical unit that has 642-byte logical blocks, then the logical block length of the block device zero ROD token is 642 bytes).

The ROD TOKEN TYPE field is defined in SPC-6 and shall be set to the value shown in table 31 for the block device zero ROD token.

The ROD TOKEN LENGTH field is defined in SPC-6 and shall be set to the value shown in table 31 for the block device zero ROD token.

A CRC VALID bit set to one indicates that the token represents user data in which the protection information, if any, has the LOGICAL BLOCK GUARD field in the protection information set to 0000h. A CRC VALID bit set to zero indicates that the token represents user data in which the protection information, if any, has the LOGICAL BLOCK GUARD field in the protection information set to FFFFh.

4.28.5 ROD token device type specific data

The device type specific data for ROD tokens (see SPC-6) created by a copy manager for a direct access block devices (see 6.7):

a) provides information about the logical unit at the time that the ROD token was created; and
b) is a subset of the parameter data returned by the READ CAPACITY (16) command (see 5.21) for the logical unit that contains the copy manager that created the ROD token.

If the READ CAPACITY (16) parameter data changes so that the copy manager that created the ROD token is no longer able to access the data represented by the ROD token, then that copy manager shall invalidate the ROD token.

4.29 Atomic writes

4.29.1 Atomic writes overview

An atomic write command performs one or more atomic write operations. The following write commands are atomic write commands:

a) WRITE ATOMIC (16) (see 5.48); and
b) WRITE ATOMIC (32) (see 5.49).

To perform an atomic write operation, the device server:

a) ensures that any subsequent read operation returns either all of the write data or none of the write data for the atomic write operation as described in 4.29.2;
b) performs operations as described in 4.29.3; and
c) processes ACA conditions as described in 4.29.4.

The MAXIMUM ATOMIC TRANSFER LENGTH field (see 6.6.4) indicates the maximum transfer length for atomic write commands that specify an atomic boundary set to zero. If an atomic write command specifies an atomic boundary set to zero and a transfer length greater than the maximum atomic transfer length, 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, and the field pointer pointing to the TRANSFER LENGTH field in the CDB.

The MAXIMUM ATOMIC TRANSFER LENGTH WITH ATOMIC BOUNDARY field (see 6.6.4) indicates the maximum transfer length for atomic write commands that specify an atomic boundary set to a non-zero value. If an atomic write command specifies a non-zero atomic boundary and the transfer length is greater than the maximum atomic transfer length with atomic boundary, then the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST, the additional sense code set to INVALID FIELD IN CDB, and the field pointer pointing to the TRANSFER LENGTH field in the CDB.

The MAXIMUM ATOMIC BOUNDARY SIZE field (see 6.6.4) indicates the maximum atomic boundary for atomic write commands. If an atomic write command specifies a non-zero atomic boundary that is greater than the maximum atomic boundary size, then the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST, the additional sense code set to INVALID FIELD IN CDB, and the field pointer pointing to the ATOMIC BOUNDARY field in the CDB.

If the starting LBA of an atomic write command does not meet the requirements of the ATOMIC ALIGNMENT field (see 6.6.4), then the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.

If the ATOMIC TRANSFER LENGTH GRANULARITY field (see 6.6.4) contains a non-zero value and:

a) the TRANSFER LENGTH field in an atomic write command is not a multiple of the atomic transfer length granularity; or
b) the ATOMIC BOUNDARY field in an atomic write command contains a non-zero value that is not a multiple of the atomic transfer length granularity,

then the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.

If an atomic write command specifies a non-zero atomic boundary that is less than or equal to the maximum atomic boundary size and a transfer length that is less than or equal to the maximum atomic transfer length with atomic boundary, then the device server shall perform one or more atomic write operations each of which is a size that is a multiple of the size specified in the ATOMIC BOUNDARY field.

If multiple atomic write operations are performed as a result of a non-zero atomic boundary, then:

a) the atomic write operations may be performed in any order; and
b) if the device server is unable to complete all atomic write operations, then the device server shall terminate the command with the sense key set to ABORTED COMMAND and the additional sense code set to INCOMPLETE MULTIPLE ATOMIC WRITE OPERATIONS.

If the device server supports atomic write commands, then the device server shall process commands as described in SAM-6 with the additional restrictions described in 4.29.

4.29.2 Atomic write operations that do not complete

If the device server is not able to successfully complete an atomic write operation (e.g., the command is terminated or aborted), then the device server shall ensure that none of the LBAs specified by the atomic write operation have been altered by any logical block data from the atomic write operation (i.e., the specified LBAs return logical block data as if the atomic write operation had not occurred).

If a power loss causes loss of logical block data from an atomic write operation in a volatile write cache that has not yet been stored on the medium, then the device server shall ensure that none of the LBAs specified by the atomic write operation have been altered by any logical block data from the atomic write operation (i.e., the specified LBAs return logical block data as if the atomic write operation had not occurred and writes from the cache to the medium preserve the specified atomicity).

EXAMPLE - If the device server processes an atomic write command that specifies a non-zero atomic boundary using five atomic write operations in such a way that:

1) two atomic write operations are performed successfully;
2) an atomic write operation encounters an error; and
3) the remaining two atomic write operations are not performed,

then subsequent read commands of the LBAs written by the two successful atomic write operations return new logical block data and subsequent read commands of the LBAs specified by the other three atomic write operations return old logical block data.

4.29.3 Performing operations with respect to atomic write operations

4.29.3.1 Performing operations before and after an atomic write operation

Before an atomic write operation completes, the device server shall not use any logical block data from the atomic write operation for any other operations (e.g., read operations return old logical block data until the atomic write operation completes).

After an atomic write operation completes, the device server shall use all of the logical block data from the atomic write operation as the basis for subsequent operations (e.g., read operations return logical block data from the atomic write operation).

4.29.3.2 Performing operations during an atomic write operation

Table 32 defines the device server requirements for starting to perform an atomic write operation that has an overlapping LBA with an operation that the device server is already performing.

Table 32 — Performing atomic write operations with overlapping LBAs during current operations

<table>
<tr><th>Operation A that is currently being processed by the device server is a</th><th>Device server is requested to perform an atomic write operation B that accesses an LBA that is being accessed by operation A</th></tr>
<tr><td>read or verify</td><td rowspan="2">The device server shall perform one of the following:
a) should complete operation A before starting operation B; or
b) may terminate operation A with CHECK CONDITION status with a sense key of ABORTED COMMAND and an additional sense code set to OVERLAPPING ATOMIC COMMAND IN PROGRESS before starting operation B.</td></tr>
<tr><td>write or unmap</td></tr>
<tr><td>format or sanitize</td><td>The device server shall terminate operation B as specified in 4.11.2 and 4.33.2</td></tr>
<tr><td>Uninterruptible sequence of operations</td><td rowspan="2">The device server shall wait for operation A to complete before it starts performing operation B.</td></tr>
<tr><td>atomic write</td></tr>
</table>

If the device server is performing an atomic write operation and is requested to perform an operation that has an LBA that overlaps with that atomic write operation, then the device server shall complete that atomic write operation before performing the requested operation.

EXAMPLE 1 - While the device server is performing an atomic write operation that accesses LBAs 0 to 15, it may perform a second atomic write operation accessing LBAs 16 to 31 and shall not perform a third atomic write operation accessing LBAs 7 to 15. After the first atomic write operation completes, the device server may perform the third atomic write operation.

EXAMPLE 2 -While the device server is performing an atomic write operation that accesses LBAs 0 to 15, if a second write operation accessing LBAs 7 to 31 is available for performing, then the device server may perform the write operation’s accesses to LBAs 16 to 31 and shall not perform the write operation’s accesses to LBAs 7 to 15. After the atomic write operation completes, the device server may perform the write operation’s accesses to LBAs 7 to 15.

Performing an atomic write operation does not prevent the device server from performing, at any time, other operations that only contain non-overlapping LBAs.

4.29.4 Processing ACA conditions during atomic write commands

If another command causes an ACA condition while an atomic write command is being processed, then the device server:

a) shall ensure that none of the LBAs specified by the atomic write operation have been altered by any logical block data from the atomic write operation (i.e., the specified LBAs return logical block data, as if the atomic write operation had not occurred); and
b) after the ACA condition is cleared:
    A) may terminate the atomic write command with CHECK CONDITION status with a sense key of ABORTED COMMAND and an additional sense code set to ATOMIC COMMAND ABORTED DUE TO ACA; or
    B) may process the atomic write operation.

4.30 IO advice hints

4.30.1 IO advice hints overview

IO advice hints allow application clients to provide device servers with information about anticipated LBA usage patterns. A device server uses IO advice hints to provide access to logical blocks that is appropriate to the specified usage patterns and optimize other aspects of the direct access block device (e.g., power consumption and performance). The degree to which the device server optimizes the direct access block device based on anticipated LBA usage patterns may not be measurable by the application client.

EXAMPLE - A device server that implements the IO advice hints features described in this subclause may provide improved performance for cases where the access pattern matches what was specified by the applicable IO advice hints (e.g., LBAs that are read in a sequential manner may exhibit better performance when the IO advice hints indicates sequential access).

IO advice hints are associated with individual commands that contain a GROUP NUMBER field (see 4.30.2). The device server is not required to retain this association in a way that produces repeatable results (e.g., recorded metadata associated with each LBA specified by the command). Device servers may provide the features described in this subclause by other means (e.g., a device server may direct the transferred data and the associated LBAs to an area of the media that has properties that match the specified IO advice hints).

4.30.2 Specifying IO advice hints

An application client may specify an IO advice hints by applying the IO advice hints extension of the grouping function using the GROUP NUMBER field in the CDB of certain commands. The GROUP NUMBER field in each command determines the IO advice hints used during the processing of that command. At the time a command that specifies an IO advice hints is received, the device server should process that command in a manner that produces the effects described in 4.30.1. This standard does not define a method for the device server to save or return IO advice hints after a command that specifies an IO advice hints has been processed.

IO advice hints are provided in most data transfer commands. If a device server supports IO advice hints, then the device server shall implement the:

a) IO Advice Hints Grouping mode page (see 6.5.7); and
b) grouping function (see 4.22.1) and grouping function extensions for IO advice hints (see 4.22.2).

4.31 Background operation control

Background operation control is managed by information in the Background Control Mode page (see 6.5.4). A device server may have requirements to perform advanced background operations if the percentage of device resources available for allocation reaches the minimum percentage indicated in the MINIMUM PERCENTAGE field (see 6.6.9). The device server may provide an indication that these resources have reached a value (e.g., minimum percentage plus 10%) that is specified by an armed decreasing threshold percentage (see 4.7.3.7.3). If the device server does not process a BACKGROUND CONTROL command that requests advanced background operations and the percentage of device resources available for allocation reach the minimum percentage indicated in the MINIMUM PERCENTAGE field (see 6.6.9), then the device server may perform advanced background operations without a request from the application client.

EXAMPLE - Advanced background operation may include NAND block erase operations, media read operations, and media write operations (e.g., garbage collection), which may impact response time for normal read requests or write requests from the application client.

If the application client is able to predict idle time when there are few read requests and few write requests to the device server, then the application client may notify the device server about this idle time so that the device server may perform advanced background operations. As a result, advanced background operations are minimally overlapped with normal read commands and normal write commands from the application client. The BACKGROUND CONTROL command (see 5.2) provides the mechanism for the application client to communicate this information to a logical unit.

The logical block provisioning thresholds (see 4.7.3.7) provides a mechanism for the device server to establish a unit attention condition to notify application clients when a related logical block provisioning threshold is crossed.

A device server that supports background operation control as described in this subclause:

a) shall be a resource provisioned device as described in 4.7.3.2 or a thin provisioned device as described in 4.7.3.3;
b) shall set the BOCS bit to one in the Block Device Characteristics VPD page (see 6.6.2);
c) shall support the Logical Block Provisioning VPD page (see 6.6.9);
d) should support logical block provisioning thresholds with the THRESHOLD TYPE field set to 001b and threshold resource value set to 01h (see 4.7.3.7.3);
e) shall support the Background Operation log page (see 6.4.3);
f) shall support the Background Control mode page (see 6.5.4); and
g) shall support the BACKGROUND CONTROL command (see 5.2).

The device server performs notification to the application client of decreasing provisioning resource percentage using logical block provisioning thresholds (see 4.7.3.7.3). Performing this notification informs the application client that the application client should determine a time to perform advanced background operations and then request the device server to perform these advanced background operations using the BACKGROUND CONTROL command.

If the device server reaches the minimum percentage of resources available as indicated in the MINIMUM PERCENTAGE field (see 6.6.9), then the device server may begin performing device server initiated advanced background operations. As a result, the application client should set the threshold percentage sufficiently high to allow the application client to receive a notification and specify to the device server to perform host initiated advanced background operations before the device reaches the minimum percentage.

If the device server processes a BACKGROUND CONTROL command with the BO_CTL field set to 01b (i.e., start host initiated background control operations), then the device server shall:

1) initialize the bo_time timer to the value specified in the BO_TIME field (see 5.2) and start the bo_time timer; and
2) perform host initiated advanced background operations until:
    A) the bo_time timer expires;
    B) the device server determines that all necessary advanced background operations are completed; or
    C) the device server processes a BACKGROUND CONTROL command with the BO_CTL field set to 10b (i.e., stop host initiated background control operations).

If the device server processes a BACKGROUND CONTROL command with the BO_CTL field set to 10b (i.e., stop host initiated background control operations) and the device server is not performing host initiated advanced background operations, then the device server shall not considered this an error.

If the device server processes a BACKGROUND CONTROL command with the BO_CTL field set to 10b (i.e., stop host initiated background control operations) and the device server is performing host initiated advanced background operations, then the device server shall stop performing host initiated advanced background operations.

If the device server processes a BACKGROUND CONTROL command with the BO_CTL field set to 00b, then the device server shall:

a) not change any host initiated advanced background operations;
b) ignore the value in the BO_TIME field; and
c) not considered this an error.

4.32 Stream control

4.32.1 Stream control overview

Stream control is limited by the MAXIMUM NUMBER OF STREAMS field, the OPTIMAL STREAM WRITE SIZE field, and the STREAM GRANULARITY SIZE field (see 6.6.5). Logical units that support stream control shall be resource provisioned logical units as described in 4.7.3.2 or thin provisioned logical units as described in 4.7.3.3.

Logical units that implement the stream control model are comprised of physical blocks that are arranged in a manner such that background operations to prepare resources for future allocations to LBAs are performed on those physical blocks. Writestream commands (see 4.32.2) with a transfer length that matches the value specified in the OPTIMAL STREAM WRITE SIZE field (see 6.6.5), and with an initial LBA that is either zero or a multiple of the value specified in the OPTIMAL STREAM WRITE SIZE field provide optimal performance from the device.

See annex E for an example method in which application clients may use alignment information to determine optimal performance for stream writes.

A device server that supports stream control as described in 4.32 shall:

a) support the GET STREAM STATUS command (see 5.9);
b) support the STREAM CONTROL command (see 5.32), unless the device server supports only permanent streams as described in 4.32.4;
c) support one or more write stream commands as described in 4.32.2; and
d) support the Block Limits Extension VPD page (see 6.6.5).

The number of optimal stream write size blocks that are prepared as a unit for future allocation to LBAs is indicated in the STREAM GRANULARITY SIZE field (see 6.6.5). If, at the time a stream is closed, the total number of blocks written to that stream is not a multiple of the stream granularity size, then the device server should prevent write commands from storing user data in the partially completed stream block.~~The application client should write an integer multiple of stream blocks to the logical unit before closing a stream.~~

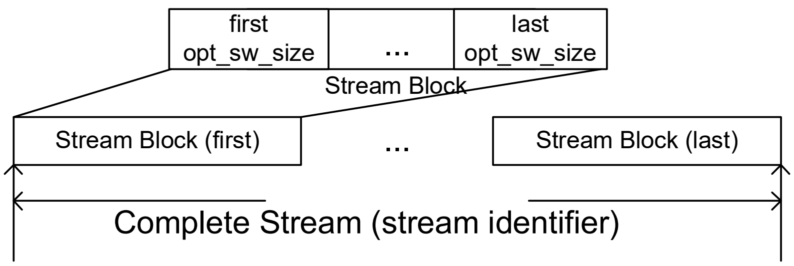
Stream control provides control of streams, each of which is comprised of user data associated with a single data structure (e.g., a single file or a complete database) or data that has the same lifetime. ~~An~~ If supported by the device server, an application client may open~~s~~ a new stream using the STREAM CONTROL command.~~ and then uses the stream identifier provided by the device server for all subsequent writes associated with that stream~~. Permanent streams(see 4.32.4) are always open.

Each stream is associated with a stream identifier.~~ ~~Each stream identifier ~~that~~ is associated with the stream from ~~when~~ the time that the stream is opened (e.g., after a power on for a permanent stream) until the time that the stream is closed. The GET STREAM STATUS command returns the stream identifier for each open stream. The application client specifies the stream identifier to be associated with the user data being written in write stream commands. Valid stream identifiers are 0001h to FFFFh. The device server may or may not use all valid stream identifiers.

A stream allows the device server to place the data associated with a stream identifier into locations that:

a) are part of one or more stream blocks; and
b) do not contain any data not associated with that stream identifier.

For optimal performance, writes to the stream should be aligned to and a multiple of the length specified by the OPTIMAL STREAM WRITE SIZE field indicated in the Block Limits Extension VPD page (see 6.6.5). Figure 13 shows the relationship of data blocks of optimal stream write size within a stream block, and multiple stream blocks within a complete stream.

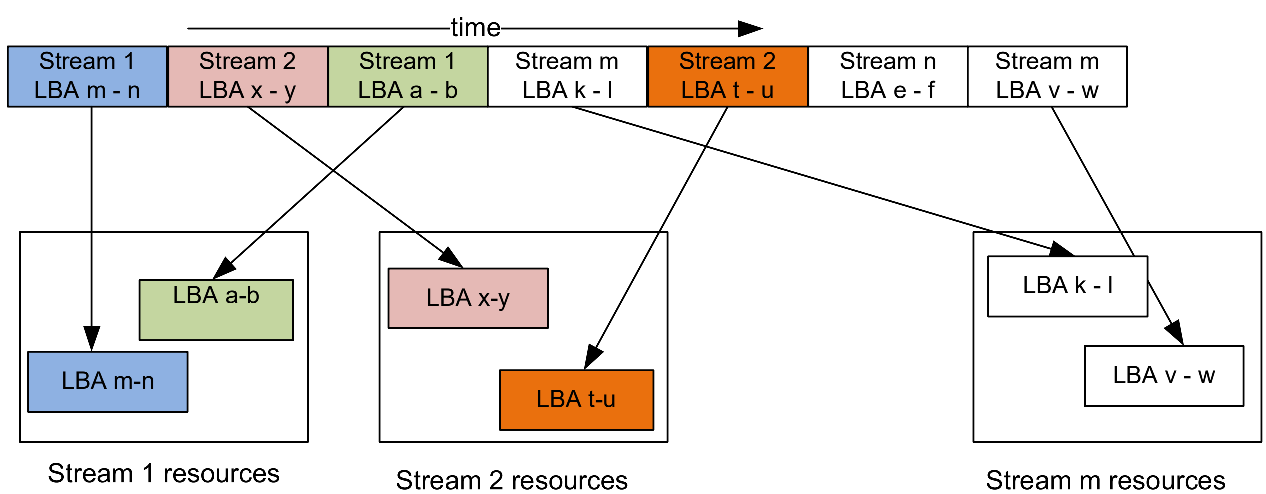


Key:
opt_sw_size is the optimal stream write size

Figure 13 — Stream Block Relationships

Figure 14 shows an example assignment of data blocks received for four different streams to associated device server resources where:

a) Stream 1 LBA m - n is one optimal stream write size in length;
b) Stream 2 LBA x - y is one optimal stream write size in length;
c) Stream 1 LBA a - b is two optimal stream write sizes in length;
d) Stream 2 LBA t - u is one optimal stream write size in length;
e) Stream m LBA k - l is one optimal stream write size in length;
f) Stream m LBA v - w is one optimal stream write size in length; and
g) Stream n mapping is not shown.



Key:
opt_sw_size is the optimal stream write size

Figure 14 — Multiple streams example

A STREAM CONTROL command (see 5.32) with the STR_CTL field set to 01b (i.e., open) shall result in the relative lifetime (see 5.9.2.2) being set to zero (i.e., no relative lifetime is applicable).

~~The application client opens and closes streams using the STREAM CONTROL command (see 5.32) and requests write operations to a stream using the WRITE STREAM (16) command (see 5.57) or the WRITE STREAM (32) command (see 5.58).~~ To use a stream that is not a permanent stream for user data the application client:

1) sends a STREAM CONTROL command with the STR_CTL field set to 01b (i.e., open);
2) waits for the stream identifier value returned in the ASSIGNED_STR_ID field ~~(see 5.32)~~;
3) sends one or more ~~WRITE STREAM~~ write stream commands with ~~the STR_ID field set to~~ the associated stream identifier specified in the STR_ID field or the GROUP NUMBER field as described in 4.32.2;
4) waits for responses to all ~~WRITE STREAM~~ write stream commands for the stream associated with the stream identifier; and
5) sends a STREAM CONTROL command with the STR_CTL field set to 10b (i.e., close) and the STR_ID field set to the value of the ~~associated~~ stream identifier.

To use a permanent stream for user data, the application client sends one or more write stream commands with the associated stream identifier specified in the STR_ID field or the GROUP NUMBER field as described in 4.32.2.

If the device server processes a ~~WRITE STREAM~~ write stream command ~~with the STR_ID field set to~~ that specifies a stream identifier of a stream that is not open, then the device server shall terminate ~~the WRITE STREAM~~ that write stream command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to STREAM NOT OPEN.

If the device server processes a STREAM CONTROL command with the STR_CTL field set to 10b (i.e., close) and the STR_ID field set to the stream identifier of a stream that is not open, then the device server shall terminate the STREAM CONTROL command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to STREAM NOT OPEN.

If the device server processes a STREAM CONTROL command with the STR_CTL field set to 01b (i.e., open) and the maximum number of streams as defined by the MAXIMUM NUMBER OF STREAMS field (see 6.6.5) are already open, then the device server shall terminate the STREAM CONTROL command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to MAXIMUM NUMBER OF STREAMS OPEN.

The application client may discover the state of open streams using the GET STREAM STATUS command (see 5.9).

Except for permanent streams ~~(see 4.32.4)~~, resources (e.g., stream identifier) allocated to maintaining a specific stream are released if:

a) the device server processes a STREAM CONTROL command (see 5.32) with the stream identifier of the specified stream and the STR_CTL field set to 10b (i.e., close);
b) a hard reset occurs;
c) a logical unit reset occurs;
d) a power on occurs;
e) a FORMAT UNIT command is processed;
f) a FORMAT WITH PRESETS command is processed; or
g) a sanitize operation is processed.

~~If physical blocks that are part of a stream block do not contain logical block data when the associated stream is closed, then logical block data should not be written to those physical blocks until all LBAs in that stream block are unmapped.~~

4.32.2 Write stream commands

Write stream commands specify the stream identifier (see 4.32.1) with:

a) an explicit stream identifier (i.e., the STR_ID field), as is present in:

A) the WRITE STREAM (16) command (see 5.57); and
B) the WRITE STREAM (32) command (see 5.58);

or

b) a per group stream identifier contained in the GROUP NUMBER field of a write command (e.g., in the WRITE (10) command (see 5.40)) as described in 4.22.2, if the ST_ENBLE bit is set to one in the associated IO advice hints group descriptor (see 6.5.7).

If the GET STREAM STATUS command (see 5.9) is supported and the RSCS bit (see 6.6.5) is:

a) set to zero, then the device server:
   A) shall support at least one of the explicit stream identifier write stream commands; and
   B) may support per group stream identifier write commands;

and

b) set to one, then the device server:
   A) may support one or more of the explicit stream identifier write stream commands; and
   B) shall support per group stream identifier usage of the GROUP NUMBER field in all the write commands containing a GROUP NUMBER field that the device server supports.

If at least one per group stream identifier write command is supported, then the device server shall support the IO Advice Hints Grouping mode page (see 6.5.7).

4.32.3 Reduced stream control

Reduced stream control:

a) reduces the maximum number of streams that the device server supports; and
b) increases the number of write commands that are able to specify a stream to be written in any write command that contains the GROUP NUMBER field in its CDB.

If the RSCS bit (see 6.6.5) is set to one, then the device server shall:

a) support per group stream identifier usage as described in 4.32.2;
b) support the IO Advice Hints Grouping mode page (see 6.5.7); and
c) set the MAXIMUM NUMBER OF STREAMS field (see 6.6.5) to a value that is less than 64.

Device servers that set the RSCS bit to one may support other features (e.g., permanent streams (see 4.32.4)).

4.32.4 Permanent streams

A permanent stream is a stream for which the device server does not allow closing or otherwise modifying the configuration of that stream. The PERM bit (see 5.9.2.3) indicates whether a stream is a permanent stream.

~~If a STREAM CONTROL command (see 5.32) specifies the closing of a permanent stream, the device server terminates that command with CHECK CONDITION status instead of closing the specified stream.~~

If a STREAM CONTROL command (see 5.32) specifies that a permanent stream be closed (i.e., the STR_CTL field set to 10b), then the device server shall terminate that 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 permanent stream is always an open stream.

Device severs should assign the lowest numbered stream identifiers to permanent streams.

4.33 Format operations

4.33.1 Format operations overview

A format operation results in the device server:

a) configuring the logical block length and number of logical blocks of the logical unit as specified by the block descriptor (see 6.5.2); and
b) performing the following as specified by:
   A) a FORMAT UNIT command (see 5.4):
      a) configure protection information;
      b) perform defect management;
      c) initialize LBAs; and
      d) vendor specific medium certification;
      or
   B) a FORMAT WITH PRESET command (see 5.5):
      a) change LBA configuration for a new device type;
      b) update peripheral device type;
      c) update VPD pages;
      d) perform medium defect management; and
      e) initialize LBAs as described in 4.33.3.

The degree that the medium is altered by a format operation is vendor specific. A format operation is requested by a FORMAT UNIT command or a FORMAT WITH PRESET command.

4.33.2 Performing a format operation

Before performing a format operation, the device server shall stop all:

a) enabled power condition timers (see SPC-6);
b) timers for enabled background scan operations (see 4.23); and
c) timers or counters enabled for device-specific background functions.

As the result of completing a format operation, the device server shall:

a) initialize and start all enabled timers and counters for background functions; and
b) initialize and start all operational (see SPC-6) power condition timers.

While performing a format operation, the device server shall:

a) process commands already in a task set when a FORMAT UNIT command or a FORMAT WITH PRESET command is received in a vendor specific manner;
b) process an INQUIRY command by returning parameter data based on the condition of the logical unit before beginning the FORMAT UNIT command or the FORMAT WITH PRESET command (i.e., INQUIRY data does not change until successful completion of a format operation):
c) process a REQUEST SENSE command by returning parameter data containing sense data with the sense key set to NOT READY, the additional sense code set to LOGICAL UNIT NOT READY, FORMAT IN PROGRESS, and the PROGRESS INDICATION field in the sense data (see SPC-6) set to indicate the progress of the format operation;
d) process REPORT LUNS commands;
e) terminate all commands, except INQUIRY commands, REPORT LUNS commands, and REQUEST SENSE commands, with CHECK CONDITION status with the sense key set to NOT READY and the additional sense code set to LOGICAL UNIT NOT READY, FORMAT IN PROGRESS;
f) remove all Background Scan Results log parameters (see 6.4.2.3) from the Background Scan Results log page, if supported; and
g) remove all Pending Defect log parameters (see 6.4.8.3) from the Pending Defects log page, if supported.

For a FORMAT UNIT command, the application client may specify:

a) that the device server clear the existing GLIST;

b) a list of address descriptors that the device server adds to the GLIST;
c) that the device server enable a certification operation that adds address descriptors for physical blocks with medium defects discovered during the certification operation to the GLIST; and
d) the behavior of the device server if it is not able to:
   A) access the PLIST or GLIST; or
   B) determine whether the PLIST or GLIST exists.

For a FORMAT WITH PRESET command, the device server manages all defect information. No information is specified by the application client (e.g., GLIST and PLIST).

4.33.3 Completing a format operation

4.33.3.1 Completing a format operation overview

If a format operation completes without error, then:

a) stream resources, if any, shall be released;
b) if the logical unit is a zoned block device, then all LBAs in the logical unit are as defined in ZBC-2;
c) if the logical unit is fully provisioned (i.e., the LBPME bit (see 5.21.2) is set to zero), then all LBAs in the logical unit are mapped (see 4.7.2); or
d) if the logical unit supports logical block provisioning management (i.e., the LBPME bit is set to one), then if the LBPRZ field (see 6.6.9) is set to:
   A) 000b, then each LBA in the logical unit shall be either:
      a) mapped, if an initialization pattern was specified that does not match the vendor-specific data returned by a read command for an unmapped LBA (see 4.7.4.4); or
      b) unmapped, if no initialization pattern was specified or an initialization pattern was specified that matches the vendor-specific data returned by a read command for an unmapped LBA (see 4.7.4.4);
   B) xx1b, then each LBA in the logical unit:
      a) shall be mapped, if the format operation did not initialize the user data to all zeroes for the logical block referenced by that LBA;
      b) shall be unmapped, if the format operation initialized the user data to all zeroes for the logical blocks referenced by all valid LBAs in the logical unit; or
      c) may be unmapped, if the format operation initialized the user data to all zeroes for the logical block referenced by that LBA, and the format operation did not initialize the user data to all zeroes for the logical blocks referenced by all valid LBAs in the logical unit;

      and
   C) 010b, then each LBA in the logical unit:
      a) shall be mapped, if an initialization pattern was specified that does not match the provisioning initialization pattern; or
      b) shall be unmapped, if no initialization pattern was specified or an initialization pattern was specified that matches the provisioning initialization pattern;

   and
e) if the format operation was performed for a FORMAT WITH PRESET command, then until Power On condition (see SAM-6) is detected, the device server shall:
   A) terminate all commands except REQUEST SENSE with CHECK CONDITION status, with the sense key set to NOT READY and the additional sense code set to LOGICAL UNIT NOT READY, POWER CYCLE REQUIRED; and
   B) for the REQUEST SENSE command, return sense data with the sense key set to NOT READY and the additional sense code set to LOGICAL UNIT NOT READY, POWER CYCLE REQUIRED.

If a format operation is aborted (e.g., by a power on condition or a hard reset condition (see SAM-6)) or completes with an error, then the logical unit may become format corrupt. Format corrupt may be cleared by a format operation that completes without error. If the logical unit is format corrupt, then the device server shall terminate any:

a) logical block access command other than commands that may correct the format corrupt condition (e.g., FORMAT UNIT command); and
b) any STREAM CONTROL command with the STR_CTRL field set to 01b (i.e., open stream),

with CHECK CONDITION status, with the sense key set to MEDIUM ERROR and the additional sense code set to MEDIUM FORMAT CORRUPTED.

4.33.3.2 Completing read commands after a successful format operation

4.33.3.2.1 Completing read commands overview

Following a successful format operation and before a write operation to an LBA, a read command or verify command that specifies that LBA shall be processed by the device server as described in:

a) 4.33.3.2.2, 4.33.3.2.3, and 4.33.3.2.4 for a mapped LBA; and
b) 4.7.4.4 for an unmapped LBA.

4.33.3.2.2 With FFMT field set to 00b

If the FFMT field (see table 40) was set to 00b in the most recent successful FORMAT UNIT command, then subsequent read commands or verify commands that complete without error are processed using:

a) the user data set as specified by:
    A) the initialization pattern, if any;
    B) the provisioning initialization pattern, if applicable; or
    C) the manufacturer's default initialization pattern;

    and
b) the protection information, if any, set to FFFF_FFFF_FFFF_FFFFh.

4.33.3.2.3 With FFMT field set to 01b

If the FFMT field (see table 40) was set to 01b in the most recent successful FORMAT UNIT command, then subsequent read commands or verify commands:

a) with unrecovered medium errors are processed as described in 4.18.1;
b) with pseudo unrecovered errors are processed as described in 4.18.2;
c) should be processed using unspecified logical block data and complete without error, if protection information is disabled; or
d) if protection information is enabled, then:
    A) may be processed using unspecified logical block data and complete without error; or
    B) may terminate with CHECK CONDITION status with sense data that indicates that the protection information check fails as defined in 5.16 or 5.36.

4.33.3.2.4 With FFMT field set to 10b

If the FFMT field (see table 40) was set to 10b in the most recent successful FORMAT UNIT command, then the device server may:

a) return unspecified logical block data and complete subsequent read commands without error;
b) complete subsequent verify commands without error; or
c) terminate subsequent read commands or verify commands with CHECK CONDITION status with the sense key set to HARDWARE ERROR, MEDIUM ERROR, or ABORTED COMMAND.

4.34 Transfer limits

Devices servers may have limits on the amount of logical block data that is able to be transferred in a single command. The Block Limits VPD page and the Block Limits Extension VPD page report those transfer limits. Table 33 describes the fields that specify transfer size and the VPD page field that limits the transfer size.

Table 33 — Transfer limits for commands

Command	Field that specifies the transfer size	VPD page field that indicates maximum limits	Additional sense code if the value in the specified field exceeds the maximum limit
Commands limited by fields in the Block Limits VPD page			
COMPARE AND WRITE	CDB NUMBER OF LOGICAL BLOCKS field	MAXIMUM COMPARE AND WRITE LENGTH field	INVALID FIELD IN CDB
ORWRITE	CDB TRANSFER LENGTH field	MAXIMUM TRANSFER LENGTH field	
PRE-FETCH	CDB PREFETCH LENGTH field	MAXIMUM PREFETCH LENGTH field	
READ	CDB TRANSFER LENGTH field	MAXIMUM TRANSFER LENGTH field	
VERIFY	CDB VERIFICATION LENGTH field		
WRITE	CDB TRANSFER LENGTH field		
WRITE AND VERIFY			
POPULATE TOKEN	NUMBER OF LOGICAL BLOCKS field in any Block device range descriptor (see 5.12.3)	MAXIMUM TRANSFER LENGTH field	INVALID FIELD IN PARAMETER LIST
WRITE USING TOKEN			
Commands limited by fields in the Block Limits Extension VPD page			
WRITE SCATTERED	CDB BUFFER TRANSFER LENGTH FIELD	MAXIMUM SCATTERED TRANSFER LENGTH field	INVALID FIELD IN CDB
	CDB NUMBER OF LBA RANGE DESCRIPTORS field	MAXIMUM SCATTERED LBA RANGE DESCRIPTOR COUNT field	INVALID FIELD IN CDB
	LBA range descriptor TRANSFER LENGTH field	MAXIMUM SCATTERED LBA RANGE TRANSFER LENGTH field	INVALID FIELD IN PARAMETER LIST

4.35 Scattered writes

4.35.1 Scattered writes overview

Scattered writes are requested by WRITE SCATTERED commands (see 5.55 and 5.56).

A WRITE SCATTERED command describes the ranges of LBAs using a set of LBA range descriptors in the Data-Out Buffer to describe the logical block address for the first LBA to be written and the number of logical blocks to be written (see figure 15). A WRITE SCATTERED command provides the logical block data to be written for each LBA range in the same Data-Out Buffer as the LBA range descriptors. The logical block data for each associated LBA range descriptor is provided in the same order as the LBA range descriptors and follows the LBA range descriptors and optional padding (see figure 15). The LOGICAL BLOCK DATA OFFSET field in the CDB specifies the start of the logical block data in the Data-Out Buffer.

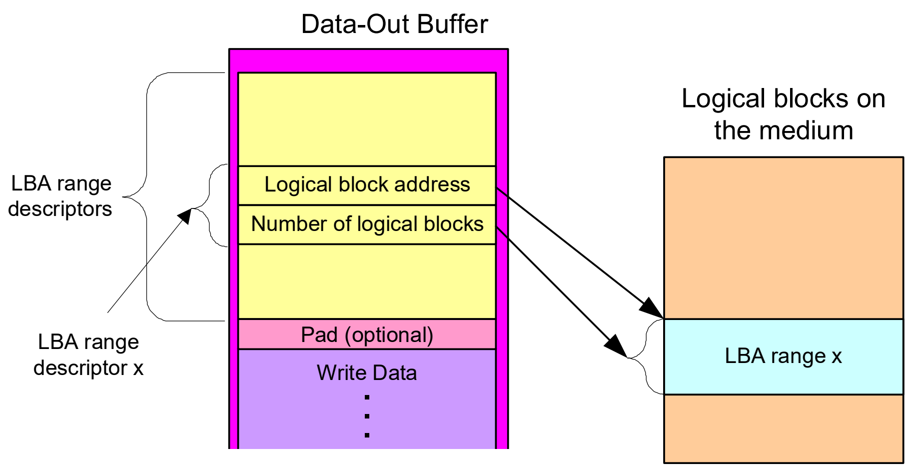


Figure 15 — LBA range descriptors

Scattered writes request one or more write operations to write data to specified LBA ranges. Each LBA range, specified by an LBA range descriptor identifies a set of contiguous LBAs associated with logical block data from the Data-Out Buffer (see figure 2).

To process the WRITE SCATTERED command the device server shall:

1) validate one or more LBA range descriptors as specified in 5.55; and
2) perform write operations for the validated descriptors as specified in 4.35.2.

The device server may:

a) validate one or more LBA range descriptors before performing write operations; and
b) perform write operations in any order.

4.35.2 Performing write operations for scattered writes

The logical block data in the Data-Out Buffer that is associated with the LBA range descriptors (see 4.35.1) is in the same order as the LBA range descriptors. The offset into the Data-Out Buffer for the logical block data associated with an LBA range descriptor is calculated by multiplying the sum of the NUMBER OF LOGICAL BLOCKS fields from each preceding LBA range descriptor by the length of a logical block (i.e., the length of the user data and protection information, if any, in one logical block) and adding the result to the offset of the logical block data from the CDB.

Figure 16 shows example write operations for a WRITE SCATTERED command.

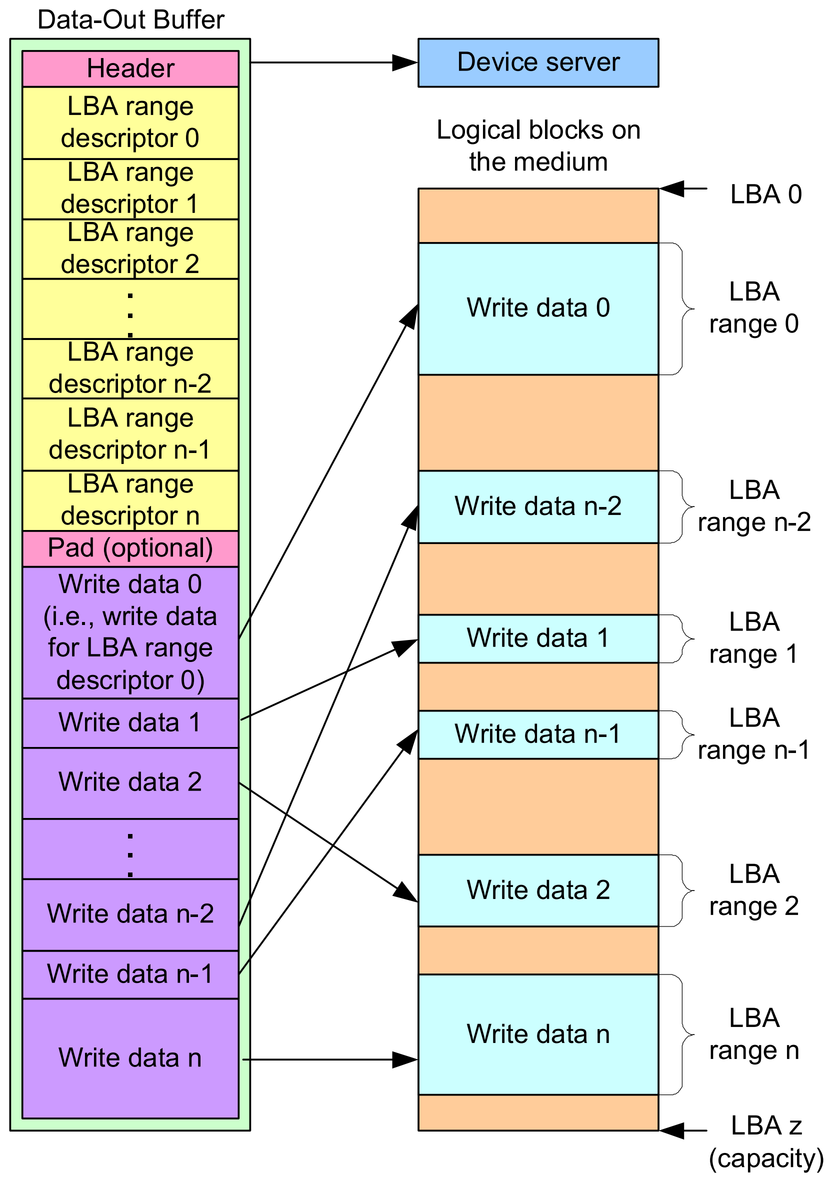


Figure 16 — Example write operations for WRITE SCATTERED commands

4.35.3 Scattered writes that encounter errors

If the device server encounters an error while performing write operations, then the device server:

a) may terminate processing of all LBA range descriptors; or
b) may continue processing range descriptors.

The device server shall terminate a WRITE SCATTERED command that encounters an error as specified in 4.18. If an unrecovered error is reported, then the INFORMATION field in the sense data should be set to a number indicating the lowest numbered LBA range descriptor for which the associated LBA range was not successfully written. LBA range descriptors shall be numbered starting with the number one indicating the first LBA range descriptor in the parameter data and incremented by one for each subsequent LBA range descriptor. An INFORMATION field set to zero indicates that the lowest numbered LBA range descriptor for which the associated LBA range was not successfully written is not reported.

The COMMAND-SPECIFIC INFORMATION field should be set:

a) as specified in 4.19.3.5, if rebuild assist mode is enabled and one or more predicted unrecovered write errors occurs;
b) to the count of LBA range descriptors for which the associated write operations completed without error; or
c) to zero if the device server does not support reporting a count of completed write operations.

4.36 Storage element depopulation and restoration

4.36.1 Overview

Storage element depopulation allows an application client to depopulate a storage element from a SCSI target device (i.e., making a specified storage element inaccessible for LBA resources and LBA mapping resources) upon successful completion of one of the following:

a) a REMOVE ELEMENT AND TRUNCATE command (see 5.26) that changes the associations between LBAs and physical blocks; and
b) a REMOVE ELEMENT AND MODIFY ZONES command (see ZBC-2) that does not change the associations between LBAs and physical blocks.

After a storage element has been depopulated, it may be restored to normal operation using a RESTORE ELEMENTS AND REBUILD command (see 5.29).

A storage element that has been depopulated provides:

a) no LBA mapping resources; and
b) no LBA resources.

The media in a SCSI device may consist of a number of storage elements. Each storage element has these characteristics (see 5.8.2.2):

a) an element identifier;
b) a Restoration Allowed attribute (see 4.36.2);
c) a health status; and
d) an associated capacity that is capable of providing LBA resources and LBA mapping resources.

A storage element is a type of physical element. Physical elements are associated with a unique element identifier that is assigned by the device server. The element identifier shall be non-zero. The association of element identifiers to physical elements shall persist through all events (e.g., across all resets) except microcode change (see SAM-6). The association of element identifiers to physical elements may persist through microcode change events.

The health status of a given physical element may become degraded (i.e., outside manufacturer’s specification limit). Such degradation may affect the overall performance of the SCSI device as seen by the application client.

4.36.2 Restoration Allowed attribute

A Restoration Allowed attribute set to:

a) true indicates that the storage element has been depopulated and is a candidate for being restored; and
b) false indicates that the storage element:
   A) has not been depopulated; or
   B) has been depopulated and is not a candidate for being restored.

4.36.3 Physical element status change notification

The device server may monitor the status of storage elements as a background operation. The device server may notify application clients that the status of one or more storage elements is not within expected manufacturer's specification limit (see 5.8.2.2). The specific mechanism for detection of this condition is not defined by this standard. If the reporting of informational exceptions control warnings is enabled (i.e., the EWASC bit is set to one (see 6.5.8)), then the device server shall report the change in condition as specified in the Informational Exceptions Control mode page with the additional sense code set to WARNING - PHYSICAL ELEMENT STATUS CHANGE.

Upon receipt of this notification the application client should examine physical element health (see 5.8).

4.36.4 Storage element depopulation

4.36.4.1 Overview

An application client may specify that a storage element be depopulated as described in 4.36.4. A device server that supports storage element depopulation shall support:

a) the REPORT SUPPORTED OPERATION CODES command (see SPC-6);
b) the GET PHYSICAL ELEMENT STATUS command (see 5.8); and
c) at least one of:
    A) the REMOVE ELEMENT AND TRUNCATE command (see 5.26); and
    B) the REMOVE ELEMENT AND MODIFY ZONES command (see ZBC-2).

A REMOVE ELEMENT AND TRUNCATE command specifies that the device server:

a) shall perform a depopulate operation (see 4.36.4.2);
b) shall perform a truncate operation (see 4.36.4.3); and
c) may perform an initialization.

The depopulate operation, truncate operation, and initialization operation, if any, may continue after the successful completion of the REMOVE ELEMENT AND TRUNCATE command.

If an initialization is not performed, then user data written before the depopulate operation may be readable in any accessible logical block.

The processing of a REMOVE ELEMENT AND TRUNCATE command shall not change:

a) the logical block length (see 4.5);
b) the lowest aligned logical block address (see 4.6.1); and
c) the protection type (see 4.21.2).

A REMOVE ELEMENT AND TRUNCATE command may be issued for each storage element that is to be removed from the current operating configuration.The effect of the processing of multiple REMOVE ELEMENT AND TRUNCATE commands shall be cumulative (see 5.26).

A device server may have a limit on the number of storage elements that may be depopulated. If the device server is requested to depopulate a storage element in excess of this limit, the device server may terminate that command with a sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.

Upon successful completion of the REMOVE ELEMENT AND TRUNCATE command; the depopulate operation, the truncate operation, and the initialization, if any, continue as background operations. While those operations are in progress, the device server shall:

a) provide pollable sense data (see SPC-6) with the sense key set to NOT READY, the additional sense code set to DEPOPULATION IN PROGRESS and the PROGRESS INDICATION field in the sense data set to indicate the progress of those operations; and
b) process other commands as described in 4.36.6.

Upon the completion of the depopulate operation, the truncate operation, and the initialization, if any, the contents of the user data area may have no relation to the contents of the user data area before the processing of the REMOVE ELEMENT AND TRUNCATE command.

Any depopulate operation, truncate operation, and initialization initiated by a REMOVE ELEMENT AND TRUNCATE command shall resume after any hard reset or logical unit reset. If any depopulate operation, truncate operation, and initialization requested by a REMOVE ELEMENT AND TRUNCATE command is interrupted by a power cycle then that operation shall be terminated and the logical unit may become format corrupt.

If a depopulate operation, a truncate operation, or an initialization initiated by the REMOVE ELEMENT AND TRUNCATE command does not complete successfully, then the logical unit may become format corrupt.

If a REMOVE ELEMENT AND TRUNCATE command requests~~the logical unit is format corrupt as a result of~~ a depopulate operation, a truncate operation, or an initialization and that requested action is terminated by a power cycle as described in 4.36.4.1(i.e., this subclause) with the result that the logical unit: ~~requested by a REMOVE ELEMENT AND TRUNCATE command~~

a) becomes format corrupt, then the device server shall terminate any medium access command with CHECK CONDITION status, with the sense key set to MEDIUM ERROR and the additional sense code set to DEPOPULATION FAILED; or
b) does not become format corrupt, then the device server should terminate any medium access command with CHECK CONDITION status, with the sense key set to NOT READY and the additional sense code set to DEPOPULATION INTERRUPTED.

4.36.4.2 Depopulate operations

A depopulate operation reduces the number of storage elements that are accessible to the logical unit's device server. To request a depopulate operation an application client issues a REMOVE ELEMENT AND TRUNCATE command.

Automatic reassignment of defects may occur during depopulate operations. Following a successful depopulate operation and before a write operation to an LBA, a read command or verify command that specifies that LBA that completes without error may return indeterminate results.

The device server shall set the Restoration Allowed attribute (see 4.36.2) of the removed storage element to:

a) true if the removed element is not defective; or
b) false if the removed element is defective.

4.36.4.3 Truncate operations

A truncate operation reduces the capacity of the media. The REMOVE ELEMENT AND TRUNCATE command specifies the capacity to which the media is truncated, and should be no larger than the capacity at the time the command begins processing minus the capacity associated with the storage element being depopulated (see 5.8.2.2).

As a result of successful completion of a truncate operation, the device server shall:

a) respond to successful READ CAPACITY commands (see 5.20 and 5.21) with parameter data reporting the resulting capacity;
b) respond to successful MODE SENSE commands (see SPC-6) with the mode parameter block descriptor (see 6.5.2) reporting the resulting capacity; and
c) establish a unit attention condition with the additional sense code set to CAPACITY DATA HAS CHANGED as described in 4.10.

4.36.5 Storage element restoration

4.36.5.1 Overview

An application client may specify that a previously depopulated storage element be restored as described in 4.36.5. A device server that supports storage element restoration shall support:

a) the REPORT SUPPORTED OPERATION CODES command (see SPC-6);
b) the GET PHYSICAL ELEMENT STATUS command (see 5.8);
c) the RESTORE ELEMENTS AND REBUILD command (see 5.29); and
d) at least one of:
   A) the REMOVE ELEMENT AND TRUNCATE command (see 5.26); and
   B) the REMOVE ELEMENT AND MODIFY ZONES command (see ZBC-2).

A RESTORE ELEMENTS AND REBUILD command specifies that the device server:

1) shall perform a depopulation revocation operation (see 4.36.5.2);
2) shall perform a rebuild operation (see 4.36.5.3); and
3) may perform an initialization.

If an initialization is not performed, user data written before the depopulation revocation operation may be readable in any accessible logical block.

The depopulation revocation operation, rebuild operation, and initialization operation, if any, may continue after the successful completion of the RESTORE ELEMENTS AND REBUILD command.

The processing of a RESTORE ELEMENTS AND REBUILD command shall not change:

a) the logical block length (see 4.5);
b) the lowest aligned logical block address (see 4.6.1); or
c) the protection type (see 4.21.2).

Upon successful completion of the RESTORE ELEMENTS AND REBUILD command; the depopulation revocation operation, rebuild operation, and the initialization, if any, shall continue as background operations. While those operations are in progress, the device server shall:

a) provide pollable sense data (see SPC-6) with the sense key set to NOT READY, the additional sense code set to DEPOPULATION RESTORATION IN PROGRESS and the PROGRESS INDICATION field in the sense data set to indicate the progress of those operations; and
b) process other commands as described in 4.36.6.

Upon the completion of the depopulation revocation operation, rebuild operation, and initialization, if any, the contents of the user data area may have no relation to the contents of the user data area before the processing of the RESTORE ELEMENTS AND REBUILD command.

Any depopulation revocation operation, rebuild operation, and initialization requested by a RESTORE ELEMENTS AND REBUILD command shall resume after any interruption hard reset or logical unit reset. If any depopulate operation, truncate operation, and initialization requested by a RESTORE ELEMENTS AND REBUILD command is interrupted by a power cycle then that operation shall be terminated and the logical unit may become format corrupt.

If a depopulation revocation operation, a rebuild operation, or an initialization initiated by the RESTORE ELEMENTS AND REBUILD command does not complete successfully, then the logical unit may become format corrupt.

If a RESTORE ELEMENTS AND REBUILD command requests~~the logical unit is format corrupt as a result of~~ a depopulation revocation operation, rebuild operation, or initialization and that requested action is terminated by a power cycle interruption as described in 4.36.5.1 (i.e., this subclause) with the result that the logical unit:~~requested by a RESTORE ELEMENTS AND REBUILD command~~

a) becomes format corrupt, then the device server shall terminate any medium access command with CHECK CONDITION status, with the sense key set to MEDIUM ERROR and the additional sense code set to DEPOPULATION RESTORATION FAILED; or

b) does not become format corrupt, then the device server should terminate any medium access command with CHECK CONDITION status, with the sense key set to NOT READY and the additional sense code set to DEPOPULATION RESTORATION INTERRUPTED.

4.36.5.2 Depopulation revocation operation

A depopulation revocation operation attempts to restore to operation every storage element that has the RALWD bit (see 5.8.2.2) set to one.

A depopulation revocation operation is requested by a RESTORE ELEMENTS AND REBUILD command (see 5.29).

Automatic reassignment of defects may occur during depopulation revocation operations. Following a successful depopulation revocation operation and before a write operation to an LBA, a read command or verify command that specifies that LBA that completes without error may return indeterminate logical block data.

4.36.5.3 Rebuild operation

A rebuild operation assigns LBA mapping resources for the storage elements that are being restored.

As a result of successful completion of a rebuild operation, the device server shall:

a) respond to successful READ CAPACITY commands (see 5.20 and 5.21) with parameter data reporting the resulting capacity;
b) respond to successful MODE SENSE commands (see SPC-6) with the mode parameter block descriptor (see 6.5.2) reporting the resulting capacity; and
c) establish a unit attention condition with the additional sense code set to CAPACITY DATA HAS CHANGED as described in 4.10.

4.36.6 Command processing during storage element depopulation and restoration

After a device server has started processing the operations associated with a REMOVE ELEMENT AND TRUNCATE command or a RESTORE ELEMENTS AND REBUILD command, and until the device server completes those operations, the device server shall terminate all commands other than:

a) GET PHYSICAL ELEMENT STATUS commands (see 5.8);
b) INQUIRY commands (see SPC-6);
c) LOG SENSE commands that specify the Temperature log page (see SPC-6);
d) MODE SENSE commands (see SPC-6) that specify:
   A) the Informational Exceptions Control mode page (see 6.5.8);
   B) the Caching mode page (see 6.5.6);
   C) the Control mode page (see SPC-6);
   D) the Protocol Specific Port mode page (see SPC-6); or
   E) the Protocol Specific Logical Unit mode page (see SPC-6);
e) READ CAPACITY (10) commands (see 5.20);
f) READ CAPACITY (16) commands (see 5.21);
g) REPORT LUNS commands (see SPC-6);
h) REPORT SUPPORTED OPERATION CODES commands (see SPC-6);
i) REPORT SUPPORTED TASK MANAGEMENT FUNCTIONS commands (see SPC-6);
j) REPORT ZONES commands (see ZBC-2) with:
   A) the ZONE START LBA field set to zero;
   B) the REPORTING OPTIONS field set to 3Fh;
   C) the PARTIAL bit set to one; and
   D) the ALLOCATION LENGTH field set to a value less than or equal to 64;

   and
k) REQUEST SENSE commands (see SPC-6) (e.g., to retrieve pollable sense data),

with CHECK CONDITION status, with the sense key set to NOT READY, the additional sense code set to:

a) DEPOPULATION IN PROGRESS, and the PROGRESS INDICATION field (see SPC-6) in the sense data set to indicate the progress of the operations associated with the REMOVE ELEMENT AND TRUNCATE command; or
b) DEPOPULATION RESTORATION IN PROGRESS and the PROGRESS INDICATION field in the sense data set to indicate the progress of the operations associated with the RESTORE ELEMENTS AND REBUILD command.

5 Commands for direct access block devices

5.1 Commands for direct access block devices overview

The commands for direct access block devices are listed in table 34.

Table 34 — Commands for direct access block devices (part 1 of 5)

Command	Operation code [a]	Type	LBACT	Reference
ATA PASS-THROUGH (12)	A1h	O	n/a	SAT-4
ATA PASS-THROUGH (16)	85h	O	n/a	SAT-4
BACKGROUND CONTROL	9Eh/15h	O	n/a	5.2
CHANGE ALIASES	A4h/0Bh	O	n/a	SPC-6
CLOSE ZONE	94h/01h	X	n/a	ZBC-2
COMPARE AND WRITE	89h	O	R, W	5.3
COPY OPERATION ABORT	83h/1Ch	O	n/a	SPC-6
EXTENDED COPY	83h/01h	O	n/a	SPC-6
FINISH ZONE	94h/02h	X	n/a	ZBC-2
FORMAT UNIT	04h	M	Z	5.4
FORMAT WITH PRESET	38h	O	Z	5.5
GET LBA STATUS (16)	9Eh/12h	O	n/a	5.6
GET LBA STATUS (32)	7Fh/0012h	O	n/a	5.7
GET PHYSICAL ELEMENT STATUS	9Eh/17h	O	n/a	5.8
GET STREAM STATUS	9Eh/16h	O	n/a	5.9
INQUIRY	12h	M	n/a	SPC-6
LOG SELECT	4Ch	O	n/a	SPC-6
LOG SENSE	4Dh	O	n/a	SPC-6
MAINTENANCE IN	A3h/00h to 04h A3h/06h to 09h	X	n/a	SPC-6 SCC-2
MAINTENANCE OUT	A4h/00h to 05h A4h/07h to 09h	X	n/a	SPC-6 SCC-2
MODE SELECT (10)	55h	O	n/a	SPC-6
MODE SENSE (10)	5Ah	O	n/a	SPC-6
OPEN ZONE	94h/03h	X	n/a	ZBC-2

Key:

O = optional
M = mandatory
X = implementation requirements are defined in the reference
R = read command
U = unmap command
V = verify command
W = write command
Z = other command
LBACT= logical block access command type (see 4.2.2)

[a] If a command is defined by a combination of operation code and service action, then the operation code value is shown preceding a slash and the service action value is shown after the slash.

Table 34 — Commands for direct access block devices (part 2 of 5)

Command	Operation code [a]	Type	LBACT	Reference
ORWRITE (16)	8Bh	O	R, W	5.10
ORWRITE (32)	7Fh/000Eh	O	R, W	5.11
PERSISTENT RESERVE IN	5Eh	O	n/a	SPC-6
PERSISTENT RESERVE OUT	5Fh	O	n/a	SPC-6
POPULATE TOKEN	83h/10h	O	n/a	5.12
PRE-FETCH (10)	34h	O	R	5.13
PRE-FETCH (16)	90h	O	R	5.14
PREVENT ALLOW MEDIUM REMOVAL	1Eh	O	n/a	5.15
READ (10)	28h	M	R	5.16
READ (12)	A8h	O	R	5.17
READ (16)	88h	M	R	5.18
READ (32)	7Fh/0009h	O	R	5.19
READ ATTRIBUTE	8Ch	O	n/a	SPC-6
READ BUFFER (10)	3Ch	O	n/a	SPC-6
READ BUFFER (16)	9Bh	O	n/a	SPC-6
READ CAPACITY (10)	25h	M	n/a	5.20
READ CAPACITY (16)	9Eh/10h	M	n/a	5.21
READ DEFECT DATA (10)	37h	O	n/a	5.22
READ DEFECT DATA (12)	B7h	O	n/a	5.23
REASSIGN BLOCKS	07h	O	Z	5.24
READ MEDIA SERIAL NUMBER	ABh/01h	O	n/a	SPC-6
RECEIVE COPY DATA	84h/06h	O	n/a	SPC-6
RECEIVE COPY STATUS	84h/05h	O	n/a	SPC-6
RECEIVE DIAGNOSTIC RESULTS	1Ch	X	n/a	SPC-6 6.3
RECEIVE ROD TOKEN INFORMATION	84h/07h	X	n/a	SPC-6 4.28 5.25
REDUNDANCY GROUP IN	BAh	X	n/a	SCC-2
REDUNDANCY GROUP OUT	BBh	X	n/a	SCC-2
REMOVE ELEMENT AND MODIFY ZONES	9Eh/1Ah	O	Z	ZBC-2
REMOVE ELEMENT AND TRUNCATE	9Eh/18h	O	Z	5.26

Key:

O = optional
M = mandatory
X = implementation requirements are defined in the reference
R = read command
U = unmap command
V = verify command
W = write command
Z = other command
LBACT= logical block access command type (see 4.2.2)

[a] If a command is defined by a combination of operation code and service action, then the operation code value is shown preceding a slash and the service action value is shown after the slash.

Table 34 — Commands for direct access block devices (part 3 of 5)

Command	Operation code [a]	Type	LBACT	Reference
REMOVE I_T NEXUS	A4h/0Ch	O	n/a	SPC-6
REPORT ALIASES	A3h/0Bh	O	n/a	SPC-6
REPORT ALL ROD TOKENS	84h/08h	O	n/a	SPC-6
REPORT IDENTIFYING INFORMATION	A3h/05h	O	n/a	SPC-6
REPORT LUNS	A0h	M	n/a	SPC-6
REPORT PRIORITY	A3h/0Eh	O	n/a	SPC-6
REPORT PROVISIONING INITIALIZATION PATTERN	A3h/1Dh	O	n/a	5.27
REPORT REALMS	95h/06h	X	n/a	ZBC-2
REPORT REFERRALS	9Eh/13h	O	n/a	5.28
REPORT SUPPORTED OPERATION CODES	A3h/0Ch	O	n/a	SPC-6
REPORT SUPPORTED TASK MANAGEMENT FUNCTIONS	A3h/0Dh	O	n/a	SPC-6
REPORT TARGET PORT GROUPS	A3h/0Ah	O	n/a	SPC-6
REPORT TIMESTAMP	A3h/0Fh	O	n/a	SPC-6
REPORT ZONE DOMAINS	95h/07h	X	n/a	ZBC-2
REPORT ZONES	95h/00h	X	n/a	ZBC-2
REQUEST SENSE	03h	M	n/a	SPC-6
RESET WRITE POINTER	94h/04h	X	n/a	ZBC
RESTORE ELEMENTS AND REBUILD	9Eh/19h	O	Z	5.29
SANITIZE	48h	O	Z	5.30
SECURITY PROTOCOL IN	A2h	O	n/a	SPC-6
SECURITY PROTOCOL OUT	B5h	O	n/a	SPC-6
SEND DIAGNOSTIC	1Dh	O	n/a	SPC-6
SET IDENTIFYING INFORMATION	A4h/06h	O	n/a	SPC-6
SET PRIORITY	A4h/0Eh	O	n/a	SPC-6
SET TARGET PORT GROUPS	A4h/0Ah	O	n/a	SPC-6
SET TIMESTAMP	A4h/0Fh	O	n/a	SPC-6
SPARE IN	BCh	X	n/a	SCC-2
SPARE OUT	BDh	X	n/a	SCC-2
START STOP UNIT	1Bh	O	n/a	5.31

Key:

O = optional
M = mandatory
X = implementation requirements are defined in the reference
R = read command
U = unmap command
V = verify command
W = write command
Z = other command
LBACT= logical block access command type (see 4.2.2)

[a] If a command is defined by a combination of operation code and service action, then the operation code value is shown preceding a slash and the service action value is shown after the slash.

Table 34 — Commands for direct access block devices (part 4 of 5)

Command	Operation code [a]	Type	LBACT	Reference
STREAM CONTROL	9Eh/14h	O	n/a	5.32
SYNCHRONIZE CACHE (10)	35h	O	W	5.33
SYNCHRONIZE CACHE (16)	91h	O	W	5.34
TEST UNIT READY	00h	M	n/a	SPC-6
UNMAP	42h	X	U	5.35 4.7
VERIFY (10)	2Fh	O	V, W	5.36
VERIFY (12)	AFh	O	V, W	5.37
VERIFY (16)	8Fh	O	V, W	5.38
VERIFY (32)	7Fh/000Ah	O	V, W	5.39
VOLUME SET IN	BEh	X	n/a	SCC-2
VOLUME SET OUT	BFh	X	n/a	SCC-2
WRITE (10)	2Ah	O	W	5.40
WRITE (12)	AAh	O	W	5.41
WRITE (16)	8Ah	O	W	5.42
WRITE (32)	7Fh/000Bh	O	W	5.43
WRITE AND VERIFY (10)	2Eh	O	V, W	5.44
WRITE AND VERIFY (12)	AEh	O	V, W	5.45
WRITE AND VERIFY (16)	8Eh	O	V, W	5.46
WRITE AND VERIFY (32)	7Fh/000Ch	O	V, W	5.47
WRITE ATOMIC (16)	9Ch	O	W	5.48
WRITE ATOMIC (32)	7Fh/000Fh	O	W	5.49
WRITE ATTRIBUTE	8Dh	O	n/a	SPC-6
WRITE BUFFER	3Bh	O	n/a	SPC-6
WRITE LONG (10)	3Fh	O	Z	5.50
WRITE LONG (16)	9Fh/11h	O	Z	5.51
WRITE SAME (10)	41h	X	U, W	5.52 4.7
WRITE SAME (16)	93h	X	U, W	5.53 4.7
WRITE SAME (32)	7Fh/000Dh	X	U, W	5.54 4.7
WRITE SCATTERED (16)	9Fh/12h	O	W	5.55

Key:

O = optional
M = mandatory
X = implementation requirements are defined in the reference
R = read command
U = unmap command
V = verify command
W = write command
Z = other command
LBACT= logical block access command type (see 4.2.2)

[a] If a command is defined by a combination of operation code and service action, then the operation code value is shown preceding a slash and the service action value is shown after the slash.

Table 34 — Commands for direct access block devices (part 5 of 5)

Command	Operation code [a]	Type	LBACT	Reference
WRITE SCATTERED (32)	7Fh/0011h	O	W	5.56
WRITE STREAM (16)	9Ah	O	W	5.57
WRITE STREAM (32)	7Fh/0010h	O	W	5.58
WRITE USING TOKEN	83h/11h	X	Z	5.59 4.28
ZONE ACTIVATE	95h/08h	X	n/a	ZBC-2
ZONE QUERY	95h/09h	X	n/a	ZBC-2

Note 1 - Operation codes that are obsolete are listed in Annex B, Annex C, and SPC-6.
Note 2 - 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 to FFh.
Note 3 - A complete summary of operation codes is available at http://www.t10.org/lists/2op.htm. The summary includes information about obsolete commands.

Key:

O = optional
M = mandatory
X = implementation requirements are defined in the reference
R = read command
U = unmap command
V = verify command
W = write command
Z = other command
LBACT= logical block access command type (see 4.2.2)

[a] If a command is defined by a combination of operation code and service action, then the operation code value is shown preceding a slash and the service action value is shown after the slash.

5.2 BACKGROUND CONTROL command

5.2.1 BACKGROUND CONTROL command overview

The BACKGROUND CONTROL command (see table 35) is used to request that the device server start or stop host initiated advanced background operations (see 4.31), if any.

This command uses the SERVICE ACTION IN (16) CDB format (see clause B.2).

Table 35 — BACKGROUND CONTROL command

Byte \ Bit	7	6	5	4	3	2	1	0
0	OPERATION CODE (9Eh)							
1	Reserved			SERVICE ACTION (15h)				
2	BO_CTL		Reserved					
3	BO_TIME							
4 ••• 14	Reserved							
15	CONTROL							

The OPERATION CODE field and SERVICE ACTION field are defined in SPC-6 and shall be set to the value shown in table 35 for the BACKGROUND CONTROL command.

The background operation control (BO_CTL) field specifies that the device server shall control host initiated advanced background operations as described in 4.31. The BO_CTL field is described in table 36.

Table 36 — BO_CTL field

Code	Description
00b	Do not change host initiated advanced background operations.
01b	Start host initiated advanced background operations.
10b	Stop host initiated advanced background operations.
11b	Reserved

The background operation time (BO_TIME) field specifies the maximum time that the device server shall have to perform host initiated advanced background operations in units of 100 ms (see 4.31). The BO_TIME field is ignored if the BO_CTL field is not set to 01b. A BO_TIME field set to 00h specifies that there is no limit to the time that the device server may perform host initiated advanced background operations.

The CONTROL byte is defined in SAM-6.

5.3 COMPARE AND WRITE command

The COMPARE AND WRITE command (see table 37) requests that the device server perform the following as an uninterrupted sequence of actions (see 4.25):

1) perform read operations from the specified LBAs;
2) perform a compare operation on only the user data (i.e., not on the protection information) from:
   A) the read operations; and
   B) the compare instance transferred from the Data-Out Buffer;
3) if the compare operation indicates a match, then perform the following operations:
   1) check the protection information, if any, in the write instance transferred from the Data-Out Buffer based on the contents of the WRPROTECT field as shown in table 135; and
   2) perform write operations to the LBAs specified by this command using the write instance;

   and

4) if the compare operation does not indicate a match, then terminate the command with CHECK CONDITION status with the sense key set to MISCOMPARE and the additional sense code set to MISCOMPARE DURING VERIFY OPERATION. In the sense data (see 4.18 and SPC-6) the offset from the start of the Data-Out Buffer to the first byte of data that was not equal shall be reported in the INFORMATION field.

The Data-Out Buffer contains two instances of logical block data:

1) the compare instance, in which:
   A) the user data is used for the compare operation; and
   B) the protection information, if any, is not used;

   and

2) the write instance, in which:
   A) the user data is used for the write operations; and
   B) the protection information, if any, is used for the write operations.

Table 37 — COMPARE AND WRITE command

Bit / Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (89h)							
1	WRPROTECT			DPO	FUA	Reserved		
2	(MSB)							
•••	LOGICAL BLOCK ADDRESS							
9								(LSB)
10								
•••	Reserved							
12								
13	NUMBER OF LOGICAL BLOCKS							
14	Reserved		GROUP NUMBER					
15	CONTROL							

The OPERATION CODE field is defined in SPC-6 and shall be set to the value shown in table 37 for the COMPARE AND WRITE command.

See the WRITE (10) command for the definition of the WRPROTECT field.

See the READ (10) command (see 5.16) for the definition of the DPO bit. See the READ (10) command (see 5.16) for the definition of the FUA bit specifying behavior for the read operations. See the WRITE (10) command (see 5.40) for the definition of the FUA bit specifying behavior for the write operations.

The LOGICAL BLOCK ADDRESS field specifies the LBA of the first logical block (see 4.5) to be accessed by the device server (e.g., the first LBA accessed by both a read operation and a write operation). If the specified LBA exceeds the capacity of the medium, then the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to LOGICAL BLOCK ADDRESS OUT OF RANGE.

The NUMBER OF LOGICAL BLOCKS field specifies:

a) the number of contiguous logical blocks on which read operations shall be performed, starting with the LBA specified by the LOGICAL BLOCK ADDRESS field;
b) the number of contiguous logical blocks that shall be transferred from the Data-Out Buffer for the compare operation; and
c) if the compare operation indicates a match, then the number of contiguous logical blocks that shall be transferred from the Data-Out Buffer and on which write operations shall be performed, starting with the LBA specified by the LOGICAL BLOCK ADDRESS field.

A NUMBER OF LOGICAL BLOCKS field set to zero specifies that no read operations shall be performed, no logical block data shall be transferred from the Data-Out Buffer, no compare operations shall be performed, and no write operations shall be performed. This condition shall not be considered an error.

If the specified LBA and the specified number of logical blocks exceed the capacity of the medium (see 4.5), then the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to LOGICAL BLOCK ADDRESS OUT OF RANGE.

If the number of logical blocks exceeds the value in the MAXIMUM COMPARE AND WRITE LENGTH field (see 6.6.4), 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.

See the PRE-FETCH (10) command (see 5.13) and 4.22 for the definition of the GROUP NUMBER field.

The CONTROL byte is defined in SAM-6.

5.4 FORMAT UNIT command

5.4.1 FORMAT UNIT command overview

The FORMAT UNIT command (see table 38) requests that the device server perform a format operation.

The device server shall handle any deferred microcode as specified in 4.24.

Table 38 defines the FORMAT UNIT command.

Table 38 — FORMAT UNIT command

Bit / Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (04h)							
1	FMTPINFO		LONGLIST	FMTDATA	CMPLST	DEFECT LIST FORMAT		
2	Vendor specific							
3	Reserved							
4	Reserved						FFMT	
5	CONTROL							

The OPERATION CODE field is defined in SPC-6 and shall be set to the value shown in table 38 for the FORMAT UNIT command.

The combination (see table 44) of the format protection information (FMTPINFO) field and the PROTECTION FIELD USAGE field (see 5.4.2.2) specifies whether or not the device server enables or disables the use of protection information.

A LONGLIST bit set to zero specifies that the parameter list, if any, contains a short parameter list header as shown in table 42. A LONGLIST bit set to one specifies that the parameter list, if any, contains a long parameter list header as shown in table 43. If the FMTDATA bit is set to zero, then the LONGLIST bit shall be ignored.

A format data (FMTDATA) bit set to one specifies that the FORMAT UNIT parameter list (see 5.4.2) shall be transferred from the Data-Out Buffer. A FMTDATA bit set to zero specifies that no parameter list be transferred from the Data-Out Buffer. If the FMTDATA bit is set to zero and the FMTPINFO field is not set to zero, 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.

Following a successful format operation, the PROT_EN bit and the P_TYPE field (see 5.21.2) indicate the type of protection currently in effect on the logical unit.

If protection information is written during a format operation (i.e., the FMTPINFO field is set to a value greater than zero), then protection information shall be written to a default value of FFFF_FFFF_FFFF_FFFFh.

A complete list (CMPLST) bit set to zero specifies that the device server shall add the defect list included in the FORMAT UNIT parameter list to the existing GLIST (see 4.13). A CMPLST bit set to one specifies that the device server shall replace the existing GLIST with the defect list, if any, included in the FORMAT UNIT parameter list.

If the FMTDATA bit is set to zero, then the CMPLST bit shall be ignored.

If the FMTDATA bit is set to one, then the DEFECT LIST FORMAT field specifies the format of the address descriptors in the defect list in the FORMAT UNIT parameter list.

Table 39 defines support requirements for address descriptors based on the combinations of the FMTDATA bit, the CMPLST bit, the DEFECT LIST FORMAT field, and the DEFECT LIST LENGTH field.

Table 39 — FORMAT UNIT command address descriptor support requirements

<table>
<tr><th colspan="3">Field in the FORMAT UNIT CDB</th><th rowspan="2">DEFECT LIST LENGTH [a]</th><th rowspan="2">Support</th><th rowspan="2">Comments</th></tr>
<tr><th>FMTDATA</th><th>CMPLST</th><th>DEFECT LIST FORMAT</th></tr>
<tr><td>0</td><td>any</td><td>000b</td><td>Not available</td><td>M</td><td>Vendor specific defect information</td></tr>
<tr><td rowspan="8">1</td><td>0</td><td rowspan="4">Either:
a) 000b (i.e., short block address descriptor)(see 6.2.2);
b) 001b (i.e., extended bytes from index address descriptor)(see 6.2.3);
c) 010b (i.e., extended physical sector address descriptor)(see 6.2.4);
d) 011b (i.e., long block address descriptor)(see 6.2.5);
e) 100b (i.e., bytes from index address descriptor)(see 6.2.6); or
f) 101b (i.e., physical sector address descriptor)(see 6.2.7)</td><td rowspan="2">Zero</td><td>O</td><td>See [b] and [d]</td></tr>
<tr><td>1</td><td>O</td><td>See [b] and [e]</td></tr>
<tr><td>0</td><td rowspan="2">Non-zero</td><td>O</td><td>See [c] and [d]</td></tr>
<tr><td>1</td><td>O</td><td>See [c] and [e]</td></tr>
<tr><td rowspan="2">0</td><td rowspan="4">110b (i.e., vendor specific)</td><td>Zero</td><td>O</td><td>See [b] and [d]</td></tr>
<tr><td>Non-zero</td><td>O</td><td>See [c] and [d]</td></tr>
<tr><td rowspan="2">1</td><td>Zero</td><td>O</td><td>See [b] and [e]</td></tr>
<tr><td>Non-zero</td><td>O</td><td>See [c] and [e]</td></tr>
<tr><td colspan="4">All other combinations</td><td colspan="2">Reserved</td></tr>
</table>

[a] This field is in the parameter list header.
[b] No defect list is included in the parameter list.
[c] A defect list is included in the parameter list.
[d] The device server retains the existing GLIST.
[e] The device server discards the existing GLIST.

The fast format (FFMT) field is described in table 40

Table 40 — FFMT field description

Code	Description	Support
00b	The device server initializes the medium (see 4.10) as specified in the CDB and parameter list before completing the format operation. After successful completion of the format operation, read commands and verify commands are processed as described in 4.33.3.2.1 and 4.33.3.2.2.	Mandatory
01b	The device server initializes the medium (see 4.10) without overwriting the medium (i.e., resources for managing medium access are initialized and the medium is not written) before completing the format operation. After successful completion of the format operation, read commands and verify commands are processed as described in 4.33.3.2.1 and 4.33.3.2.3. If the device server determines that the options specified in this FORMAT UNIT command are incompatible with the read command and verify command requirements described in 4.33.3.2.3, then the device server shall not perform the format operation and shall terminate the FORMAT UNIT command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FAST FORMAT.	Optional
10b	The device server initializes the medium (see 4.10) without overwriting the medium (i.e., resources for managing medium access are initialized and the medium is not written) before completing the format operation. After successful completion of the format operation, read commands and verify commands are processed as described in 4.33.3.2.1 and 4.33.3.2.4.	Optional
11b	Reserved	

The CONTROL byte is defined in SAM-6.

5.4.2 FORMAT UNIT parameter list

5.4.2.1 FORMAT UNIT parameter list overview

Table 41 defines the FORMAT UNIT parameter list.

Table 41 — FORMAT UNIT parameter list

Byte \ Bit	7	6	5	4	3	2	1	0
0	Parameter list header (see table 42 or table 43 in 5.4.2.2)							
	Initialization pattern descriptor (if any) (see table 45 in 5.4.2.3)							
	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 6.2) each specifying a location on the medium to which the device server shall not assign LBAs. The device server shall maintain the current logical block to physical block alignment (see 4.6) for logical blocks not specified in the defect list.

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. If the address descriptors are not in the required order, then the device server may terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

5.4.2.2 Parameter list header

The parameter list headers (see table 42 and table 43) provide several optional format control parameters. If the application client requests a function that is not implemented by the device server, then the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

If the LONGLIST bit is set to zero in the FORMAT UNIT CDB, then the short parameter list header (see table 42) is used.

Table 42 — Short parameter list header

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

If the LONGLIST bit is set to one in the FORMAT UNIT CDB, then the long parameter list header (see table 43) is used.

Table 43 — Long parameter list header

Byte \ Bit	7	6	5	4	3	2	1	0
0	Reserved					PROTECTION FIELD USAGE		
	FOV	Format options bits				Obsolete	IMMED	Vendor specific
1		DPRY	DCRT	STPF	IP			
2	Reserved							
3	P_I_INFORMATION				PROTECTION INTERVAL EXPONENT			
4	(MSB)	DEFECT LIST LENGTH						
•••								
7								(LSB)

The combination (see table 44) of the PROTECTION FIELD USAGE field and the FMTPINFO field (see 5.4.1) specifies the requested protection type (see 4.21.2).

Table 44 — FMTPINFO field and PROTECTION FIELD USAGE field (part 1 of 2)

Fields indicated by the device server		Fields specified by the application client		Description
SPT [a]	**PROTECT** [b]	**FMTPINFO**	**PROTECTION FIELD USAGE**	
xxxb	0	00b	000b	The logical unit shall be formatted to type 0 protection [c] (see 4.21.2.2) resulting in the PROT_EN bit [d] being set to zero and the P_TYPE field [d] being set to 000b.
			>000b	Illegal [e]
		01b	xxxb	Illegal [f]
		1xb	xxxb	Illegal [f]
xxxb	1	00b	000b	The logical unit shall be formatted to type 0 protection [c] (see 4.21.2.2) resulting in the PROT_EN bit [d] being set to zero and the P_TYPE field [d] being set to 000b.
			>000b	Illegal [e]
xxxb	1	01b	xxxb	Illegal [f]
000b 001b 011b 111b	1	10b	000b	The logical unit shall be formatted to type 1 protection [g] (see 4.21.2.3) resulting in the PROT_EN bit [d] being set to one and the P_TYPE field [d] being set to 000b.
			>000b	Illegal [e]
010b 100b 101b	1	10b	xxxb	Illegal [f]
000b	1	11b	xxxb	Illegal [f]

[a] See the Extended INQUIRY Data VPD page (see SPC-6) for the definition of the SPT field.
[b] See the standard INQUIRY data (see SPC-6) for the definition of the PROTECT bit.
[c] The device server shall format the medium to the logical block length specified in the mode parameter block descriptor of the mode parameter header (see SPC-6).
[d] See the READ CAPACITY (16) parameter data (see 5.21.2) for the definition of the PROT_EN bit and P_TYPE field.
[e] The device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN PARAMETER LIST.
[f] 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.
[g] The device server shall format the medium to the logical block length specified in the mode parameter block descriptor of the mode parameter header plus eight bytes for each protection information interval (e.g., if the logical block length is 2 048 and the PROTECTION INTERVAL EXPONENT field set to two, then there are four 512 byte protection information intervals each followed by eight bytes of protection information resulting in a formatted logical block length of 2 080 bytes). Following a successful format operation, the PROT_EN (see 5.21.2) indicates whether protection information (see 4.21) is enabled.

Table 44 — FMTPINFO field and PROTECTION FIELD USAGE field (part 2 of 2)

Fields indicated by the device server		Fields specified by the application client		Description
SPT [a]	PROTECT [b]	FMTPINFO	PROTECTION FIELD USAGE	
001b 010b 101b 111b	1	11b	000b	The logical unit shall be formatted to type 2 protection [g] (see 4.21.2.4) resulting in the PROT_EN bit [d] being set to one and the P_TYPE field [d] being set to 001b.
001b 010b	1	11b	>000b	Illegal [e]
011b 100b	1	11b	000b	Illegal [e]
011b 100b 101b 111b	1	11b	001b	The logical unit shall be formatted to type 3 protection. [g] (see 4.21.2.5) resulting in the PROT_EN bit [d] being set to one and the P_TYPE field [d] being set to 010b.
			>001b	Illegal [e]
110b	1	10b 11b	xxxb	Reserved

[a] See the Extended INQUIRY Data VPD page (see SPC-6) for the definition of the SPT field.
[b] See the standard INQUIRY data (see SPC-6) for the definition of the PROTECT bit.
[c] The device server shall format the medium to the logical block length specified in the mode parameter block descriptor of the mode parameter header (see SPC-6).
[d] See the READ CAPACITY (16) parameter data (see 5.21.2) for the definition of the PROT_EN bit and P_TYPE field.
[e] The device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN PARAMETER LIST.
[f] 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.
[g] The device server shall format the medium to the logical block length specified in the mode parameter block descriptor of the mode parameter header plus eight bytes for each protection information interval (e.g., if the logical block length is 2 048 and the PROTECTION INTERVAL EXPONENT field set to two, then there are four 512 byte protection information intervals each followed by eight bytes of protection information resulting in a formatted logical block length of 2 080 bytes). Following a successful format operation, the PROT_EN (see 5.21.2) indicates whether protection information (see 4.21) is enabled.

A format options valid (FOV) bit set to zero specifies that the device server shall use its default settings for the functionality represented by the DPRY bit, the DCRT bit, the STPF bit, and IP bit (i.e., the format options bits). If the FOV bit is set to zero, then the application client should set each of the format options bits to zero. If the FOV bit is set to zero, and any of the format options bits are not set to zero, then the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

A FOV bit set to one specifies that the device server shall process the format options bits as follows:

a) a disable primary (DPRY) bit:
   A) set to zero specifies that the device server shall not assign LBAs to parts of the medium identified as defective in the PLIST; or
   B) set to one specifies that the device server shall not use the PLIST to identify defective areas of the medium, and the PLIST shall not be deleted;
b) a disable certification (DCRT) bit:

   A) set to zero specifies that the device server shall perform a vendor specific medium certification operation and add address descriptors for defects that it detects during the certification operation to the GLIST; or
   B) set to one specifies that the device server shall not perform any vendor specific medium certification process or format verification operation;
c) the stop format (STPF) bit controls the behavior of the device server if 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 one or more of the following occurs:
   A) **list locate error**: the device server is not able to locate a specified defect list or determine whether a specified defect list exists; or
   B) **list access error**: the device server encounters an error while accessing a specified defect list;
d) a STPF bit set to zero specifies that:
   A) if a list locate error or a list access error occurs, then the device server shall continue to process the FORMAT UNIT command; and
   B) after the format operation is complete, if:
      a) a list locate error and a list access error both occurred, then the device server shall terminate the FORMAT UNIT command with CHECK CONDITION status at the completion of the command with the sense key set to RECOVERED ERROR and the additional sense code set to DEFECT LIST NOT FOUND or DEFECT LIST ERROR;
      b) a list locate error occurred and a list access error did not occur, then the device server shall terminate the FORMAT UNIT command with CHECK CONDITION status with the sense key set to RECOVERED ERROR with the additional sense code set to DEFECT LIST NOT FOUND; and
      c) a list access error occurred and a list locate error did not occur, then the device server shall terminate the FORMAT UNIT command with CHECK CONDITION status with the sense key set to RECOVERED ERROR with the additional sense code set to DEFECT LIST ERROR;
e) a STPF bit set to one specifies that:
   A) if a list locate error occurs, then the device server shall terminate the FORMAT UNIT command with CHECK CONDITION status with the sense key set to MEDIUM ERROR and the additional sense code set to either DEFECT LIST NOT FOUND; or
   B) if a list access error occurs, then the device server shall terminate the FORMAT UNIT command with CHECK CONDITION status with the sense key set to MEDIUM ERROR with the additional sense code set to DEFECT LIST ERROR;

   and

f) an initialization pattern (IP) bit:
   A) set to zero specifies that an initialization pattern descriptor (see 5.4.2.3) is not included and that the device server shall use its default initialization pattern; or
   B) set to one specifies that:
      a) an initialization pattern descriptor is included in the FORMAT UNIT parameter list following the parameter list header; and
      b) if the device server does not support initialization pattern descriptors, then the device server shall terminate the FORMAT UNIT command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

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

The P_I_INFORMATION field, if any (i.e., if the long parameter list header is used), should be set to 0h. If the P_I_INFORMATION field is not set to zero, then the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

For a type 0 or a type 1 protection information request, if the PROTECTION INTERVAL EXPONENT field, if any, is not set to 0h, then the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

For a type 2 protection or a type 3 protection format request, the protection interval exponent determines the length of user data to be transferred before protection information is transferred (i.e., the protection information interval).

The protection information interval is calculated as follows:

$$\text{protection information interval} = \text{logical block length} \div 2^{(\text{protection interval exponent})}$$

where:

logical block length	is the number of bytes of user data in a logical block (see 4.5)
protection interval exponent	is zero if the short parameter list header (see table 42) is used or the contents of the PROTECTION INTERVAL EXPONENT field if the long parameter list header (see table 43) is used

If the protection information interval calculates to a value that is not an even number (e.g., $520 \div 2^3 = 65$) or not a whole number (e.g., $520 \div 2^4 = 32.5$ and $520 \div 2^{10} = 0.508$), then the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

The DEFECT LIST LENGTH field specifies the total length in bytes of the defect list (i.e., the address descriptors) that follow and does not include the length of the initialization pattern descriptor, if any. The formats for the address descriptor(s) are shown in 6.2.

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 45) is transferred to the device server as part of the FORMAT UNIT parameter list.

Table 45 — Initialization pattern descriptor

<table>
<tr><th>Byte \ Bit</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td>0</td><td colspan="2">Obsolete</td><td>SI</td><td colspan="5">Reserved</td></tr>
<tr><td>1</td><td colspan="8">INITIALIZATION PATTERN TYPE</td></tr>
<tr><td>2</td><td colspan="8">(MSB) INITIALIZATION PATTERN LENGTH (n - 3)</td></tr>
<tr><td>3</td><td colspan="8">(LSB)</td></tr>
<tr><td>4</td><td colspan="8" rowspan="3">INITIALIZATION PATTERN</td></tr>
<tr><td>•••</td></tr>
<tr><td>n</td></tr>
</table>

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 specifies that the device server shall ignore:

a) the FMTPINFO field;
b) the CMPLIST bit;
c) the DEFECT LIST FORMAT field;
d) all the bits and fields in the parameter list header, except the IMMED bit; and
e) any defect list data.

The device server shall write the initialization pattern using a security erasure write technique. The security erasure write technique requirement and procedure is not defined by this standard. The device server is not required to write the initialization pattern over the header and other parts of the medium not previously accessible to the application client. If the device server is unable to write over any part 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), then the device server shall terminate the command with CHECK CONDITION status with the sense key set to MEDIUM ERROR and the additional sense code set to the appropriate value for the condition. The device server shall attempt to rewrite all remaining parts of the medium even if some parts are not able to be rewritten.

NOTE 5 - The intent of the security erasure write is to render any previous user data unrecoverable by any analog or digital technique.

NOTE 6 - Migration from the SI bit to the SANITIZE command (see 5.30) is recommended for all implementations.

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

The INITIALIZATION PATTERN TYPE field (see table 46) specifies the type of pattern the device server shall use to initialize each logical block within the application client accessible part of the medium. All bytes within a logical block shall be written with the initialization pattern.

Table 46 — INITIALIZATION PATTERN TYPE field

Code	Description
00h	Use a default initialization pattern [a]
01h	Repeat the pattern specified in the INITIALIZATION PATTERN field as required to fill the logical block [b]
02h to 7Fh	Reserved
80h to FFh	Vendor specific

[a] If the INITIALIZATION PATTERN LENGTH field is not set to zero, then the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN PARAMETER LIST.
[b] If the INITIALIZATION PATTERN LENGTH field is set to zero, then the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

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

The INITIALIZATION PATTERN field specifies the initialization pattern.

5.5 FORMAT WITH PRESET command

The FORMAT WITH PRESET command (see table 47) requests that the device server perform a format operation (see 4.33) using the parameters specified by the PRESET IDENTIFIER field.

If deferred microcode has been saved and not activated (see SPC-6), then the device server shall terminate this command with CHECK CONDITION status with the sense key set to NOT READY and the additional sense code set to LOGICAL UNIT NOT READY, MICROCODE ACTIVATION REQUIRED.

If the FORMAT WITH PRESET command is supported, the Format Presets VPD page (see 6.6.8) shall be supported.

The processing of a FORMAT WITH PRESET command may result in the device server providing a different logical unit (e.g., the peripheral device type may change). As a result, the successful completion of the format operation (see 4.33.3.1) requires a power cycle before most commands return a status other than CHECK CONDITION.

.

Table 47 — FORMAT WITH PRESET command

Bit / Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (38h)							
1	IMMED	FMTMAXLBA	Reserved					
2	(MSB)	PRESET IDENTIFIER						
•••								
5								(LSB)
6	Reserved							
•••								
8								
9	CONTROL							

The OPERATION CODE field is defined in SPC-6 and shall be set as shown in table 47 for the FORMAT WITH PRESET command.

An immediate (IMMED) bit set to zero specifies that the device server shall return status upon completion of the format operation. An IMMED bit set to one specifies that the device server shall return status after the PRESET IDENTIFIER field has been validated.

A format to maximum last logical block address (FMTMAXLBA) bit set to zero specifies that the format operation should result in the RETURNED LOGICAL BLOCK ADDRESS field returned by a subsequent READ CAPACITY (16) command (see 5.20) being equal to the value in the DESIGNED LAST LOGICAL BLOCK ADDRESS field in the specified format preset descriptor (see 6.6.8). A FMTMAXLBA bit set to one specifies that the format operation shall result in the RETURNED LOGICAL BLOCK ADDRESS field being set to the largest possible value based on the condition of the media being formatted.

The PRESET IDENTIFIER field specifies which format preset descriptor (see 6.6.8) is used. If the specified preset identifier is not equal to the contents of any PRESET IDENTIFIER field in the Format Presets VPD page (see 6.6.8), 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.

The CONTROL byte is defined in SAM-6.

5.6 GET LBA STATUS (16) command

5.6.1 GET LBA STATUS (16) command overview

The GET LBA STATUS (16) command (see table 48) requests that the device server transfer parameter data describing the logical block provisioning status (see 4.7) and additional status for the specified LBA and zero or more subsequent LBAs to the Data-In Buffer.

The device server may or may not process this command as an uninterrupted sequence of actions (e.g., if concurrent operations are occurring that affect the logical block provisioning status, then the returned parameter data may be inconsistent or out of date).

This command uses the SERVICE ACTION IN (16) CDB format (see clause B.2).

Table 48 — GET LBA STATUS (16) command

Byte \ Bit	7	6	5	4	3	2	1	0
0	OPERATION CODE (9Eh)							
1	Reserved			SERVICE ACTION (12h)				
2	(MSB)	STARTING LOGICAL BLOCK ADDRESS						
•••								
9								(LSB)
10	(MSB)	ALLOCATION LENGTH						
•••								
13								(LSB)
14	REPORT TYPE							
15	CONTROL							

The OPERATION CODE field is defined in SPC-6 and shall be set to the value shown in table 48 for the GET LBA STATUS (16) command.

The SERVICE ACTION field is defined in SPC-6 and shall be set to the value shown in table 48 for the GET LBA STATUS (16) command.

The STARTING LOGICAL BLOCK ADDRESS field specifies the LBA of the first logical block addressed by this command. If the specified starting LBA exceeds the capacity of the medium (see 4.5), then the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to LOGICAL BLOCK ADDRESS OUT OF RANGE.

The ALLOCATION LENGTH field is defined in SPC-6. In response to a GET LBA STATUS (16) command, the device server may send less data to the Data-In Buffer than is specified by the allocation length. If, in response to a single GET LBA STATUS (16) command, the device server does not send sufficient data to the Data-In Buffer to satisfy the requirement of the application client, then, to retrieve additional information, the application client may send additional GET LBA STATUS (16) commands with different starting LBA values.

The REPORT TYPE field specifies the type of LBA status descriptors to return as shown in table 49.

Table 49 — REPORT TYPE field

Code	Description
00h	Return descriptors for all LBAs
01h	Return descriptors for all LBAs using only non-zero provisioning status (see table 54)
02h	Return descriptors for all LBAs that are mapped (see 4.7.4.5)
03h	Return descriptors for all LBAs that are deallocated (see 4.7.4.6)
04h	Return descriptors for all LBAs that are anchored (see 4.7.4.7)
10h	Return descriptors for LBAs that may return an unrecovered error
All others	Reserved

The CONTROL byte is defined in SAM-6.

5.6.2 GET LBA STATUS parameter data

5.6.2.1 GET LBA STATUS parameter data overview

The GET LBA STATUS parameter data (see table 50) contains an eight-byte header followed by one or more LBA status descriptors.

Table 50 — GET LBA STATUS parameter data

Byte \ Bit	7	6	5	4	3	2	1	0
0	(MSB)							
•••	PARAMETER DATA LENGTH (n - 3)							
3								(LSB)
4								
•••	Reserved							
6								
7	Reserved				COMPLETION CONDITION			RTP
	LBA status descriptors							
8								
•••	LBA status descriptor [first] (see 5.6.2.2)							
23								
	⋮							
n - 15								
•••	LBA status descriptor [last] (see 5.6.2.2) (if any)							
n								

The PARAMETER DATA LENGTH field indicates the number of bytes of parameter data that follow. The value in the PARAMETER DATA LENGTH field shall be:

a) set to 4, if there are no status descriptors to report;

b) at least 20 (i.e., the available parameter data shall contain at least one LBA status descriptor), if there are any status descriptors to report; and
c) four added to a multiple of 16 (i.e., the available parameter data shall end on a boundary between LBA status descriptors).

As a result of processing considerations not defined by this standard, two GET LBA STATUS commands with identical values in all CDB fields may result in two different values in the PARAMETER DATA LENGTH field.

The relationship between the PARAMETER DATA LENGTH field and the ALLOCATION LENGTH field in the CDB is defined in SPC-6.

The COMPLETION CONDITION field indicates the condition that caused completion of the GET LBA STATUS command. The COMPLETION CONDITION field is described in table 51.

Table 51 — COMPLETION CONDITION field

Code	Description
000b	No indication of the completion condition.
001b	The command completed as a result of meeting the allocation length.
010b [a]	The command completed as a result of completing the scan length.
011b	The command completed as a result of reaching the capacity of the medium (see 4.5).
all others	Reserved
[a] This only applies to the GET LBA STATUS (32) command	

If the command completes by reaching the capacity of the medium and any other condition at the same time, then the COMPLETION CONDITION field shall be set to 011b.

The report type processed (RTP) bit indicates whether the value in the REPORT TYPE field was processed. If the RTP bit is set to zero, then the REPORT TYPE field was not processed (i.e., the GET LBA STATUS parameter data is returned as if the REPORT TYPE field was set to 00h). If the RTP bit is set to one, then the REPORT TYPE field was processed. If a device server supports the REPORT TYPE field set to a non-zero value, then the RTP bit shall be set to one.

5.6.2.2 LBA status descriptor

The LBA status descriptor (see table 52) contains LBA status information for one or more LBAs.

Table 52 — LBA status descriptor format

Byte \ Bit	7	6	5	4	3	2	1	0
0	(MSB)							
•••	LBA STATUS LOGICAL BLOCK ADDRESS							
7								(LSB)
8	(MSB)							
•••	NUMBER OF LOGICAL BLOCKS							
11								(LSB)
12	Reserved	LBA ACCESSIBILITY			PROVISIONING STATUS			
13	ADDITIONAL STATUS							
14	Reserved							
15								

The LBA STATUS LOGICAL BLOCK ADDRESS field contains the first LBA of the LBA extent for which this descriptor reports LBA status.

The NUMBER OF LOGICAL BLOCKS field contains the number of logical blocks in that LBA extent. The device server should return the largest possible value in the NUMBER OF LOGICAL BLOCKS field.

The LBA ACCESSIBILITY field is shown in table new.

Table 53 — LBA ACCESSIBILITY field

Code	Description
000b	LBA accessibility is not reported
001b	LBA extent is not able to be written and not able to be read
010b	LBA extent is read only
All others	Reserved

The PROVISIONING STATUS field is shown in table 54.

Table 54 — PROVISIONING STATUS field

Code	Description	Allowed for report types
0h	Each LBA in the LBA extent is mapped (see 4.7.4.5) or has an unknown state.	00h
1h [a]	Each LBA in the LBA extent is deallocated (see 4.7.4.6).	00h, 01h, 03h, 10h
2h [a]	Each LBA in the LBA extent is anchored (see 4.7.4.7).	00h, 01h, 04h, 10h
3h	Each LBA in the LBA extent is mapped.	01h, 02h, 10h
4h	Each LBA in the LBA extent has an unknown provisioning status.	01h, 10h
All others	Reserved	
[a] The LBA ACCESSIBILITY field shall not be set to 001b for this case.		

If the logical unit is fully provisioned (see 4.7.2), then the PROVISIONING STATUS field for all LBAs shall be set to:

a) 0h (i.e., mapped or unknown), if the RTP bit is set to zero;
b) 0h (i.e., mapped or unknown), if the REPORT TYPE field is set to 00h in the CDB and the RTP bit is set to one; or
c) 3h (i.e., mapped), if the REPORT TYPE field is not set to 00h in the CDB and the RTP bit is set to one.

The ADDITIONAL STATUS field shall be set to 00h if the REPORT TYPE field is set to 00h in the CDB and shall be as shown in table 55 for all other values of report type.

Table 55 — ADDITIONAL STATUS field

Code	Description
00h	No additional status to report.
01h	The device server has detected that each LBA in the LBA extent may return an unrecovered error.
All others	Reserved

5.6.2.3 LBA status descriptor relationships

The LBA STATUS LOGICAL BLOCK ADDRESS field in the first LBA status descriptor returned in the GET LBA STATUS parameter data shall contain the lowest numbered LBA that is greater than or equal to the starting logical block address specified in the CDB that meets the requirements for the specified report type. For subsequent LBA status descriptors, the contents of the LBA STATUS LOGICAL BLOCK ADDRESS field shall contain:

a) for a non-zero report type, the value of the lowest numbered LBA meeting the requirements for the specified report type that is greater than or equal to the sum of the values in:
   A) the LBA STATUS LOGICAL BLOCK ADDRESS field in the previous LBA status descriptor; and
   B) the NUMBER OF LOGICAL BLOCKS field in the previous LBA status descriptor;

   or

b) for report type 0h, the sum of the values in:
   A) the LBA STATUS LOGICAL BLOCK ADDRESS field in the previous LBA status descriptor; and
   B) the NUMBER OF LOGICAL BLOCKS field in the previous LBA status descriptor.

Adjacent LBA status descriptors may have the same values for the PROVISIONING STATUS field.

5.7 GET LBA STATUS (32) command

5.7.1 GET LBA STATUS (32) command overview

The GET LBA STATUS (32) command (see table 56) requests that the device server transfer parameter data describing the logical block provisioning status (see 4.7) and additional status for the specified LBA and zero or more subsequent LBAs to the Data-In Buffer.

The device server may or may not process this command as an uninterrupted sequence of actions (e.g., if concurrent operations are occurring that affect the logical block provisioning status, then the returned parameter data may be inconsistent or out of date).

This command uses the variable length CDB format (see clause B.1).

Table 56 — GET LBA STATUS (32) command

Byte \ Bit	7	6	5	4	3	2	1	0
0	OPERATION CODE (7Fh)							
1	CONTROL							
2 ••• 6	Reserved							
7	ADDITIONAL CDB LENGTH (18h)							
8	(MSB)	SERVICE ACTION (0012h)						
9								(LSB)
10	REPORT TYPE							
11	Reserved							
12	(MSB)	STARTING LOGICAL BLOCK ADDRESS						
••• 19								(LSB)
20	(MSB)	SCAN LENGTH						
••• 23								(LSB)
24 ••• 27	ELEMENT IDENTIFIER							
28	(MSB)	ALLOCATION LENGTH						
••• 31								(LSB)

The OPERATION CODE field, ADDITIONAL CDB LENGTH field, and SERVICE ACTION field are defined in SPC-6 and shall be set to the values shown in table 56 for the GET LBA STATUS (32) command.

The CONTROL byte is defined in SAM-6.

The SCAN LENGTH field specifies the maximum number of contiguous logical blocks to be scanned for logical blocks that meet the specified report type. A value of 0000_0000h in the SCAN LENGTH field specifies that there is no limit on the number of logical blocks to be scanned (e.g., scan to the end of the media, or scan until the allocation length is met).

The ELEMENT IDENTIFIER field specifies the element identifier of the physical element (see 4.36.1) for which LBAs shall be reported based on the value in the REPORT TYPE field. If the ELEMENT IDENTIFIER field is set to 0000_0000h, then LBAs for all physical elements shall be reported based on the value in the REPORT TYPE field.

See the GET LBA STATUS (16) command for the description of the STARTING LOGICAL BLOCK ADDESSS field, the ALLOCATION LENGTH field, the REPORT TYPE field, and the returned parameter data (see 5.6.2).

5.8 GET PHYSICAL ELEMENT STATUS command

5.8.1 GET PHYSICAL ELEMENT STATUS command overview

The GET PHYSICAL ELEMENT STATUS command (see table 57) requests that the device server return status information for physical elements within the logical unit.

This command uses the SERVICE ACTION IN (16) CDB format (see clause B.2).

Table 57 — GET PHYSICAL ELEMENT STATUS command

Byte \ Bit	7	6	5	4	3	2	1	0
0	OPERATION CODE (9Eh)							
1	Reserved			SERVICE ACTION (17h)				
2 ••• 5	Reserved							
6 ••• 9	(MSB) STARTING ELEMENT (LSB)							
10 ••• 13	(MSB) ALLOCATION LENGTH (LSB)							
14	FILTER		Reserved		REPORT TYPE			
15	CONTROL							

The OPERATION CODE field and the SERVICE ACTION field are defined in SPC-6 and shall be set to the values shown in table 57 for the GET PHYSICAL ELEMENT STATUS command.

The STARTING ELEMENT field specifies the element identifier of the first physical element addressed by this command.

The ALLOCATION LENGTH field is defined in SPC-6. In response to a GET PHYSICAL ELEMENT STATUS command, the device server may send less data to the Data-In Buffer than is specified by the allocation length. If, in response to a GET PHYSICAL ELEMENT STATUS command, the device server does not send sufficient data to the Data-In Buffer to satisfy the requirement of the application client, then, to retrieve additional information, the application client may send additional GET PHYSICAL ELEMENT STATUS commands with different starting element values.

The FILTER field restricts the physical element status descriptors to return, as shown in table 58.

Table 58 — FILTER field

Code	Description
00b	All physical status descriptors as specified by the other fields in the CDB.
01b	Only physical element status descriptors for which the value of the PHYSICAL ELEMENT HEALTH field (see 5.8.2.2) is: a) greater than or equal to 65h and less than or equal to CFh (i.e., outside manufacturer's specification limit); b) equal to FBh (i.e., all operations associated with depopulation revocation have completed and one or more completed with error); c) equal to FCh (i.e., an operation associated with depopulation revocation is in progress); d) equal to FDh (i.e., all operations associated with storage element depopulation have completed and one or more completed with error); e) equal to FEh (i.e., an operation associated with storage element depopulation is in progress); or f) equal to FFh (i.e., all operations associated with storage element depopulation have completed without error).
all others	Reserved

The REPORT TYPE field specifies the type of physical element status descriptors to return as shown in table 59.

Table 59 — REPORT TYPE field

Code	Description
0h	Return descriptors for physical elements, based on the FILTER field
1h	Return descriptors for storage elements, based on the FILTER field
all others	Reserved

The CONTROL byte is defined in SAM-6.

5.8.2 GET PHYSICAL ELEMENT STATUS parameter data

5.8.2.1 GET PHYSICAL ELEMENT STATUS parameter data overview

The GET PHYSICAL ELEMENT STATUS parameter data (see table 60) contains a 32-byte header followed by zero or more physical element status descriptors.

Table 60 — GET PHYSICAL ELEMENT STATUS parameter data

Byte \ Bit	7	6	5	4	3	2	1	0
0	(MSB)							
•••	NUMBER OF DESCRIPTORS							
3								(LSB)
4	(MSB)							
•••	NUMBER OF DESCRIPTORS RETURNED							
7								(LSB)
8	(MSB)							
•••	IDENTIFIER OF ELEMENT BEING DEPOPULATED							
11								(LSB)
12	(MSB)	CURRENT MAXIMUM NUMBER OF DEPOPULATED ELEMENTS						
13								(LSB)
14	(MSB)	CURRENT NUMBER OF DEPOPULATED ELEMENTS						
15								(LSB)
16								
•••	Reserved							
31								
Physical element status descriptor list								
32								
•••	Physical element status descriptor [first] (see 5.8.2.2)							
63								
⋮								
n - 32								
•••	Physical element status descriptor [last] (see 5.8.2.2)							
n								

The NUMBER OF DESCRIPTORS field shall contain the number of descriptors in the element descriptor~~s~~ list. The element descriptor~~s~~ list is a list of physical elements that:

a) meet the requirements of the REPORTING OPTIONS field;
b) meet the requirements of the FILTER field; and
c) have an element identifier that is greater than or equal to the element identifier specified by the STARTING ELEMENT field in the CDB.

The contents of the NUMBER OF DESCRIPTORS field are not altered based on the allocation length.

The NUMBER OF DESCRIPTORS RETURNED field contains the number of valid physical element status descriptors returned in the parameter data.

The IDENTIFIER OF ELEMENT BEING DEPOPULATED field contains the element identifier of the element that has a physical element health set to FEh (i.e., an operation associated with storage element depopulation is in progress). If the value of this field is set to zero, then no operation associated with storage element depopulation is in progress.

As a result of processing considerations not defined by this standard, two GET PHYSICAL ELEMENT STATUS commands with identical values in all CDB fields may result in two different values in the NUMBER OF DESCRIPTORS field.

The CURRENT MAXIMUM NUMBER OF DEPOPULATED ELEMENTS field indicates the limit on the number of physical elements that are able to be depopulated at the time that the device server processes the GET PHYSICAL ELEMENT STATUS command. If the CURRENT MAXIMUM NUMBER OF DEPOPULATED ELEMENTS field is set to zero, then the number of physical elements that are able to be depopulated is not reported.

The CURRENT NUMBER OF DEPOPULATED ELEMENTS field indicates the number of physical elements that are depopulated at the time the device server processes the GET PHYSICAL ELEMENT STATUS command.

The physical element status descriptors shall be sorted in ascending order of the element identifier.

5.8.2.2 Physical element status descriptor

The physical element status descriptor (see table 61) contains status information for a physical element.

Table 61 — Physical element status descriptor format

Byte \ Bit	7	6	5	4	3	2	1	0
0 ••• 3	Reserved							
4 ••• 7	(MSB) ELEMENT IDENTIFIER (LSB)							
8 ••• 12	Reserved							
13	Reserved							RALWD
14	PHYSICAL ELEMENT TYPE							
15	PHYSICAL ELEMENT HEALTH							
16 ••• 23	(MSB) ASSOCIATED CAPACITY (LSB)							
24 ••• 31	Reserved							

The ELEMENT IDENTIFIER field contains the non-zero identifier of the physical element (e.g., storage element) associated with this physical element status descriptor.

If the Restoration Allowed attribute (see 4.36.2) for this physical element is:

a) true, then the restoration allowed (RALWD) bit shall be set to one; and
b) false, then the RALWD bit shall be set to zero.

The PHYSICAL ELEMENT TYPE field indicates the type of the physical element associated with this physical element status descriptor, as described in table 62.

Table 62 — PHYSICAL ELEMENT TYPE field

Code	Description
01h	Storage element
all others	Reserved

The PHYSICAL ELEMENT HEALTH field indicates the health of the physical element associated with this physical element status descriptor, as described in table 63.

Table 63 — PHYSICAL ELEMENT HEALTH field

Code	Description
00h	Not reported.
01h to 63h [a]	The physical element health is within manufacturer's specification limits.
64h	The physical element health is at manufacturer's specification limit.
65h to CFh [a]	The physical element health is outside manufacturer's specification limit.
D0h to FAh	Reserved
FBh	All operations associated with depopulation revocation have completed and one or more completed with error.
FCh	An operation associated with depopulation revocation is in progress.
FDh	All operations associated with storage element depopulation have completed and one or more completed with error.
FEh	An operation associated with storage element depopulation is in progress.
FFh	All operations associated with storage element depopulation have completed without error.
[a] The device server may implement a subset of these values.	

The ASSOCIATED CAPACITY field indicates the the number of logical blocks by which the capacity of the device is reduced if the physical element associated with this physical element status descriptor becomes depopulated. A value of FFFF_FFFF_FFFF_FFFFh indicates that the number of logical blocks by which the capacity is reduced is not specified.

5.9 GET STREAM STATUS command

5.9.1 GET STREAM STATUS command overview

The GET STREAM STATUS command (see table 64) requests that the device server transfer parameter data describing the status of streams (see 4.32) for the logical unit to the Data-In Buffer.

The device server may or may not process this command as an uninterrupted sequence of actions (e.g., if concurrent operations are occurring that affect the status of streams, then the returned parameter data may be inconsistent or out of date).

This command uses the SERVICE ACTION IN (16) CDB format (see clause B.2).

Table 64 — GET STREAM STATUS command

Byte \ Bit	7	6	5	4	3	2	1	0
0	OPERATION CODE (9Eh)							
1	Reserved			SERVICE ACTION (16h)				
2	Reserved							
3								
4	STARTING STREAM IDENTIFIER							
5								
6	Reserved							
•••								
9								
10	(MSB)	ALLOCATION LENGTH						
•••								
13								(LSB
14	Reserved							
15	CONTROL							

The OPERATION CODE field and the SERVICE ACTION field are defined in SPC-6 and shall be set to the values shown in table 64 for the GET STREAM STATUS command.

The STARTING STREAM IDENTIFIER field specifies the stream identifier of the first stream addressed by this command (see 5.9.2.3). If the specified starting stream identifier exceeds the value indicated by the MAXIMUM NUMBER OF STREAMS field of the Block Limits VPD page (see 6.6.4), 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 ILLEGAL FIELD IN CDB.

The ALLOCATION LENGTH field is defined in SPC-6. If, in response to a single GET STREAM STATUS command, the device server does not send sufficient data to the Data-In Buffer to satisfy the requirement of the application client, then the application client may send additional GET STREAM STATUS commands with different starting stream identifier values to retrieve additional information.

The CONTROL byte is defined in SAM-6.

5.9.2 GET STREAM STATUS parameter data

5.9.2.1 GET STREAM STATUS parameter data overview

The GET STREAM STATUS parameter data (see table 65) contains an eight-byte header followed by zero or more stream status descriptors.

Table 65 — GET STREAM STATUS parameter data

Byte \ Bit	7	6	5	4	3	2	1	0
0 ••• 3	PARAMETER DATA LENGTH (n-7)							
4 5	Reserved							
6	(MSB)	NUMBER OF OPEN STREAMS						
7								(LSB)
	Stream status descriptors							
8 ••• 15	Stream status descriptor [first] (see 5.9.2.2)							
	⋮							
n-8 ••• n	Stream status descriptor [last] (see 5.9.2.2)							

The PARAMETER DATA LENGTH field shall contain the length in bytes of the stream list. The stream list length is the number of open streams in the logical unit multiplied by eight. The contents of the STREAM LIST LENGTH field are not altered based on the allocation length.

As a result of processing considerations not defined by this standard, two GET STREAM STATUS commands with identical values in all CDB fields may result in two different values in the PARAMETER DATA LENGTH field.

The relationship between the PARAMETER DATA LENGTH field and the ALLOCATION LENGTH field in the CDB is defined in SPC-6.

The NUMBER OF OPEN STREAMS field indicates the number of streams that are currently open in the logical unit.

5.9.2.2 Stream status descriptor

The stream status descriptor (see table 66) contains stream status information for one open stream.

Table 66 — Stream status descriptor~~ format~~

Byte \ Bit	7	6	5	4	3	2	1	0
0	PERM	Reserved						
1	Reserved							
2	(MSB)	STREAM IDENTIFIER						
3								(LSB)
4	Reserved	RELATIVE LIFETIME						
5 ••• 7	Reserved							

A permanent (PERM) bit set to one indicates that this steam status descriptor describes a permanent stream (see 4.32.4). A PERM bit set to zero indicates that stream status descriptor does not describe a permanent stream.

The STREAM IDENTIFIER field contains the stream identifier of an open stream.

The RELATIVE LIFETIME field (see table x1) indicates the length of time that data written using the stream described by this stream status descriptor is anticipated to remain on the medium as compared to data written by other streams (i.e., streams for which the RELATIVE LIFETIME field contains a different non-zero value).

TABLE 67 — RELATIVE LIFETIME field

Code	Relative lifetime
00h	no relative lifetime is applicable
01h	shortest relative lifetime
02h	second shortest relative lifetime
03h to 3Dh	intermediate relative lifetimes
3Eh	second longest relative lifetime
3Fh	longest relative lifetime

5.9.2.3 Stream status descriptor relationships

The STREAM IDENTIFIER field in the first stream status descriptor returned in the GET STREAM STATUS parameter data shall contain:

a) the value specified in the STARTING STREAM IDENTIFIER field of the CDB if that stream is open; or
b) the value of the next greater stream identifier of an open stream.

If the value specified in the STARTING STREAM IDENTIFIER field of the CDB is greater than the highest stream identifier of an open stream, then the device server shall not return any stream status descriptors.

For subsequent stream status descriptors, the contents of the STREAM IDENTIFIER field shall contain the value of the next greater stream identifier of an open stream.

5.10 ORWRITE (16) command

The ORWRITE (16) command (see table 68) requests that the device server perform an ORWRITE command operation (see 4.27.4).

This command uses the variable length CDB format (see clause B.1).

Table 68 — ORWRITE (16) command

Byte \ Bit	7	6	5	4	3	2	1	0
0	OPERATION CODE (8Bh)							
1	ORPROTECT			DPO	FUA	Reserved		
2	(MSB)							
•••	LOGICAL BLOCK ADDRESS							
9								(LSB)
10	(MSB)							
•••	TRANSFER LENGTH							
13								(LSB)
14	Reserved		GROUP NUMBER					
15	CONTROL							

The OPERATION CODE field is defined in SPC-6 and shall be set to the value shown in table 68 for the ORWRITE (16) command.

See the READ (10) command (see 5.16) for the definition of the FUA bit specifying behavior for read operations. See the WRITE (10) command (see 5.40) for the definition of the FUA bit specifying behavior for write operations. See the READ (10) command (see 5.16) for the definition of the DPO bit. See the PRE-FETCH (10) command (see 5.13) for the definition of the LOGICAL BLOCK ADDRESS field. See the PRE-FETCH (10) command (see 5.13) and 4.22 for the definition of the GROUP NUMBER field.

The TRANSFER LENGTH field specifies the number of contiguous logical blocks of data that are read, transferred from the Data-Out Buffer, and ORed into a bitmap buffer, starting with the logical block referenced by the LBA specified by the LOGICAL BLOCK ADDRESS field. If the specified LBA and the specified transfer length exceed the capacity of the medium (see 4.5), then the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to LOGICAL BLOCK ADDRESS OUT OF RANGE. The TRANSFER LENGTH field is constrained by the MAXIMUM TRANSFER LENGTH field (see 6.6.4).

The CONTROL byte is defined in SAM-6.

The device server shall:

a) check protection information from the read operations based on the ORPROTECT field as described in table 69; and
b) check protection information transferred from the Data-Out Buffer based on the ORPROTECT field as described in table 70.

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

The device server shall check the protection information from the read operations based on the ORPROTECT field as described in table 69. All footnotes for table 69 are at the end of the table.

Table 69 — ORPROTECT field - checking protection information from the read operations (part 1 of 3)

Code	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 [i j]	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.		
001b 101b [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 [h]	LOGICAL BLOCK REFERENCE TAG CHECK FAILED
			REF_CHK = 0	No check performed
	No	Error condition [a]		
010b [b]	Yes	LOGICAL BLOCK GUARD	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	Error condition [a]		

Table 69 — ORPROTECT field - checking protection information from the read operations (part 2 of 3)

Code	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	No check performed	
		LOGICAL BLOCK APPLICATION TAG	No check performed	
		LOGICAL BLOCK REFERENCE TAG	No check performed	
	No	Error condition [a]		
100b [b]	Yes	LOGICAL BLOCK GUARD	GRD_CHK = 1	LOGICAL BLOCK GUARD CHECK FAILED
			GRD_CHK = 0	No check performed
		LOGICAL BLOCK APPLICATION TAG	No check performed	
		LOGICAL BLOCK REFERENCE TAG	No check performed	
	No	Error condition [a]		

Table 69 — ORPROTECT field - checking protection information from the read operations (part 3 of 3)

Code	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
110b to 111b	Reserved			

[a] If the logical unit supports protection information (see 4.21) and has not been formatted with protection information, then the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.

[b] If the logical unit does not support protection information, then the device server should 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.

[c] If the device server has knowledge of the contents of the LOGICAL BLOCK APPLICATION TAG field, then the device server shall check the logical block application tag. If the ATO bit in the Control mode page (see SPC-6) is set to one, then this knowledge is acquired from:

a) the Application Tag mode page (see 6.5.3), if the ATMPE bit in the Control mode page (see SPC-6) is set to one; or.
b) a method not defined by this standard, if the ATMPE bit is set to zero.

[d] If the device server terminates the command with CHECK CONDITION status, then the device server shall set the sense key to ABORTED COMMAND.

[e] If multiple errors occur while the device server is processing the command, then the selection by the device server of which error to report is not defined by this standard.

[f] See the Extended INQUIRY Data VPD page (see SPC-6) for the definitions of the GRD_CHK bit, the APP_CHK bit, and the REF_CHK bit.

[g] If the device server detects:

a) a LOGICAL BLOCK APPLICATION TAG field set to FFFFh, and type 1 protection (see 4.21.2.3) is enabled; or
b) a LOGICAL BLOCK APPLICATION TAG field set to FFFFh, the LOGICAL BLOCK REFERENCE TAG field set to FFFF_FFFFh, and type 3 protection (see 4.21.2.5) is enabled,

then the device server shall not check any protection information in the associated protection information interval.

[h] If type 1 protection is enabled, then the device server shall check the logical block reference tag by comparing it to the lower four bytes of the LBA associated with the logical block. If type 3 protection is enabled, then the device server shall check each logical block reference tag only if the device server has knowledge of the contents of the LOGICAL BLOCK REFERENCE TAG field. The method for acquiring this knowledge is not defined by this standard.

[i] If the RWWP bit in the Control mode page (see SPC-6) is set to one, 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.

[j] If the DPICZ bit in the Control mode page (see SPC-6) is set to one, and the RWWP bit in the Control mode page is set to zero, then protection information shall not be checked.

The device server shall check the protection information transferred from the Data-Out Buffer based on the ORPROTECT field as described in table 70. All footnotes for table 70 are at the end of the table.

Table 70 — ORPROTECT field - checking protection information from the Data-Out Buffer (part 1 of 2)

Code	Logical unit formatted with protection information	Field in protection information	Device server check	If check fails [d h], additional sense code
000b	Yes [e f g]	No protection information in the Data-Out Buffer to check		
	No	No protection information in the Data-Out Buffer to check		
001b [b]	Yes	LOGICAL BLOCK GUARD	Shall	LOGICAL BLOCK GUARD CHECK FAILED
		LOGICAL BLOCK APPLICATION TAG	Dependent on RWWP [c]	LOGICAL BLOCK APPLICATION TAG CHECK FAILED
		LOGICAL BLOCK REFERENCE TAG	Shall (except for type 3) [i]	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	Dependent on RWWP [c]	LOGICAL BLOCK APPLICATION TAG CHECK FAILED
		LOGICAL BLOCK REFERENCE TAG	May [i]	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]		
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]		

Table 70 — ORPROTECT field - checking protection information from the Data-Out Buffer (part 2 of 2)

Code	Logical unit formatted with protection information	Field in protection information	Device server check	If check fails [d h], additional sense code
101b [b]	Yes	LOGICAL BLOCK GUARD	Shall	LOGICAL BLOCK GUARD CHECK FAILED
		LOGICAL BLOCK APPLICATION TAG	Dependent on RWWP [c]	LOGICAL BLOCK APPLICATION TAG CHECK FAILED
		LOGICAL BLOCK REFERENCE TAG	May [i]	LOGICAL BLOCK REFERENCE TAG CHECK FAILED
	No	Error condition [a]		
110b to 111b	Reserved			

[a] If a logical unit supports protection information (see 4.21) and has not been formatted with protection information, then the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.

[b] If the logical unit does not support protection information, then the device server should 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.

[c] If the ATO bit is set to one in the Control mode page (see SPC-6), and the device server has knowledge of the contents of the LOGICAL BLOCK APPLICATION TAG field, then the device server:
a) may check each logical block application tag if the RWWP bit is set to zero in the Control mode page (see SPC-6); and
b) shall check each logical block application tag if the RWWP bit is set to one in the Control mode page.
If the ATMPE bit in the Control mode page (see SPC-6) is set to one, then this knowledge is acquired from the Application Tag mode page. If the ATMPE bit is set to zero, then the method for acquiring this knowledge is not defined by this standard.

[d] If the device server terminates the command with CHECK CONDITION status, then the device server shall set the sense key to ABORTED COMMAND.

[e] The device server shall write a generated CRC (see 4.21.4.2) into each LOGICAL BLOCK GUARD field.

[f] If the RWWP bit in the Control mode page (see SPC-6) is set to one, then the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB. If the RWWP bit is set to zero, and:
a) type 1 protection is enabled, then 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; or
b) type 3 protection is enabled, then the device server shall write a value of FFFF_FFFFh into the LOGICAL BLOCK REFERENCE TAG field of each of the written logical blocks.

[g] If the ATO bit is set to one in the Control mode page (see SPC-6), then the device server shall write FFFFh into each LOGICAL BLOCK APPLICATION TAG field. If the ATO bit is set to zero, then the device server may write any value into each LOGICAL BLOCK APPLICATION TAG field.

[h] If multiple errors occur while the device server is processing the command, then the selection by the device server of which error to report is not defined by this standard.

[i] If type 1 protection is enabled, then the device server shall check the logical block reference tag by comparing it to the lower four bytes of the LBA associated with the logical block. If type 3 protection is enabled, then the device server may check each logical block reference tag if the ATO bit is set to one in the Control mode page (see SPC-6), and the device server has knowledge of the contents of the LOGICAL BLOCK REFERENCE TAG. The method for acquiring this knowledge is not defined by this standard.

5.11 ORWRITE (32) command

The ORWRITE (32) command (see table 71) requests that the device server perform one of the following ORWRITE command (see 4.27) operations:

a) a change generation and clear operation (see 4.27.3); or
b) a set operation (see 4.27.4).

Table 71 — ORWRITE (32) command

Byte \ Bit	7	6	5	4	3	2	1	0
0	OPERATION CODE (7Fh)							
1	CONTROL							
2	Reserved					BMOP		
3	Reserved				PREVIOUS GENERATION PROCESSING			
4	Reserved							
5								
6	Reserved		GROUP NUMBER					
7	ADDITIONAL CDB LENGTH (18h)							
8	(MSB)	SERVICE ACTION (000Eh)						
9								(LSB)
10	ORPROTECT			DPO	FUA	Reserved		
11	Reserved							
12	(MSB)	LOGICAL BLOCK ADDRESS						
•••								
19								(LSB)
20	(MSB)	EXPECTED ORWGENERATION						
•••								
23								(LSB)
24	(MSB)	NEW ORWGENERATION						
•••								
27								(LSB)
28	(MSB)	TRANSFER LENGTH						
•••								
31								(LSB)

The OPERATION CODE field, the ADDITIONAL CDB LENGTH field, and the SERVICE ACTION field are defined in SPC-6 and shall be set to the values shown in table 71 for the ORWRITE (32) command.

The CONTROL byte is defined in SAM-6.

The bitmap operation (BMOP) field specifies the operation as described in table 72.

Table 72 — BMOP field

Code	Description
000b	The device server shall perform a set operation (see 4.27.4), and the contents of the PREVIOUS GENERATION PROCESSING field and NEW ORWGENERATION field shall be ignored.
001b	The device server shall perform a change generation and clear operation (see 4.27.3).
All others	Reserved

The PREVIOUS GENERATION PROCESSING field specifies the policy for performing future set operations that is to be established in the device server by a successful change generation and clear operation (see 4.27.2.2).

See the ORWRITE (16) command (see 5.10) for the definitions of the FUA bit, the DPO bit, the ORPROTECT field, the LOGICAL BLOCK ADDRESS field, the TRANSFER LENGTH field, and the GROUP NUMBER field.

The EXPECTED ORWGENERATION field contains a code that is compared with generation codes established and maintained by the device server.

The NEW ORWGENERATION field specifies the current ORWgeneration code that is to be established in the device server by a successful change generation and clear operation (see 4.27.3).

The device server shall:

a) check protection information from the read operations based on the ORPROTECT field as described in table 69; and
b) check protection information transferred from the Data-Out Buffer based on the ORPROTECT field as described in table 70.

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

5.12 POPULATE TOKEN command

5.12.1 POPULATE TOKEN command overview

The POPULATE TOKEN command (see table 73) requests that the copy manager (see SPC-6) create a point in time ROD token that represents the specified logical blocks (see 4.28).

Each logical block represented by the point in time ROD token includes logical block data.

Table 73 — POPULATE TOKEN command

Byte \ Bit	7	6	5	4	3	2	1	0
0	OPERATION CODE (83h)							
1	Reserved			SERVICE ACTION (10h)				
2 ••• 5	Reserved							
6 ••• 9	(MSB) LIST IDENTIFIER (LSB)							
10 ••• 13	(MSB) PARAMETER LIST LENGTH (LSB)							
14	Reserved		GROUP NUMBER					
15	CONTROL							

The OPERATION CODE field and the SERVICE ACTION field are defined in SPC-6 and shall be set to the values shown in table 73 for the POPULATE TOKEN command.

The LIST IDENTIFIER field is defined in SPC-6. The list identifier shall be processed as if the LIST ID USAGE field in the parameter data for an EXTENDED COPY(LID4) command (see SPC-6) is set to 00b.

The PARAMETER LIST LENGTH field specifies the length in bytes of the parameter list that is available to be transferred from the Data-Out Buffer. If the parameter list length is greater than zero and less than 00000010h (i.e., 16), then the copy manager shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to PARAMETER LIST LENGTH ERROR. A PARAMETER LIST LENGTH field set to zero specifies that no data shall be transferred. This shall not be considered an error.

See the PRE-FETCH (10) command (see 5.13) and 4.22 for the definition of the GROUP NUMBER field.

The CONTROL byte is defined in SAM-6.

5.12.2 POPULATE TOKEN parameter list

The parameter list for the POPULATE TOKEN command is shown in table 74.

Table 74 — POPULATE TOKEN parameter list

Byte \ Bit	7	6	5	4	3	2	1	0
0	(MSB)	POPULATE TOKEN DATA LENGTH (n - 1)						
1								(LSB)
2	Reserved						RTV	IMMED
3	Reserved							
4	(MSB)	INACTIVITY TIMEOUT						
•••								
7								(LSB)
8	(MSB)	ROD TYPE						
•••								
11								(LSB)
12	Reserved							
13								
14	(MSB)	BLOCK DEVICE RANGE DESCRIPTOR LENGTH (n - 15)						
15								(LSB)
Block device range descriptor list								
16	Block device range descriptor [first] (see 5.12.3)							
•••								
31								
⋮								
n - 15	Block device range descriptor [last] (see 5.12.3) (if any)							
•••								
n								

The POPULATE TOKEN DATA LENGTH field specifies the length in bytes of the data that is available to be transferred from the Data-Out Buffer. The populate token data length does not include the number of bytes in the POPULATE TOKEN DATA LENGTH field. If the POPULATE TOKEN DATA LENGTH field is less than 001Eh (i.e., 30), then the copy manager shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

A ROD type valid (RTV) bit set to zero specifies that the copy manager may create a ROD token with any point in time copy ROD type and shall ignore the contents of the ROD TYPE field. An RTV bit set to one specifies that the copy manager shall use the contents of the ROD TYPE field to create the point in time copy ROD.

The immediate (IMMED) bit specifies when the copy manager shall return status for the POPULATE TOKEN command. If the IMMED bit is set to zero, then the copy manager shall process the POPULATE TOKEN command until all specified operations are complete or an error is detected. If the IMMED bit is set to one, then the copy manager:

1) shall validate the CDB (i.e., detect and report all errors in the CDB);
2) shall transfer all the parameter data to the copy manager;

3) may validate the parameter data;
4) shall complete the POPULATE TOKEN command with GOOD status; and
5) shall complete performing of all specified operations as a background operation (see SPC-6).

If the INACTIVITY TIMEOUT field is not set to zero, then the INACTIVITY TIMEOUT field specifies the number of seconds to use for the ROD token inactivity timeout (see SPC-6). If the INACTIVITY TIMEOUT field is set to a value larger than the value in the MAXIMUM INACTIVITY TIMEOUT field (see 6.6.11.3), then the copy manager shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

If the INACTIVITY TIMEOUT field is set to zero, then the DEFAULT INACTIVITY TIMEOUT field (see 6.6.11.3) specifies the number of seconds to use for the ROD token inactivity timeout (see SPC-6).

If the RTV bit is set to one, then the ROD TYPE field specifies the ROD type (see SPC-6) for creating the point in time copy ROD token. The copy manager shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN PARAMETER LIST, if:

a) the copy manager does not support the specified ROD type for use with the POPULATE TOKEN command; or
b) the ROD TYPE field specifies a ROD type (see SPC-6) that is not a point in time copy ROD.

The BLOCK DEVICE RANGE DESCRIPTOR LENGTH field specifies the length in bytes of the block device range descriptor list. The block device range descriptor list length should be a multiple of 16. If the block device range descriptor list length is not a multiple of 16, then the last block device range descriptor is incomplete and shall be ignored. If the block device range descriptor list length is less than 0010h (i.e.,16), then the copy manager shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

If any block device range descriptors in the block device range descriptor list are truncated as a result of the parameter list length in the CDB, then those block device range descriptors shall be ignored.

If the number of complete block device range descriptors is larger than the maximum range descriptors value in the block device ROD token limits descriptor (see 6.6.11.3), then the copy manager shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to TOO MANY SEGMENT DESCRIPTORS.

If the same LBA is included in more than one block device range descriptor, then the copy manager shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

If the number of bytes of user data represented by the sum of the contents of the NUMBER OF LOGICAL BLOCKS fields in all of the complete block device range descriptors is larger than:

a) the MAXIMUM BYTES IN BLOCK ROD field in the block ROD device type specific features descriptor in the ROD token features third-party copy descriptor in the Third-party Copy VPD page (see SPC-6) and that field is set to a non-zero value; or
b) the MAXIMUM TOKEN TRANSFER SIZE field (see 6.6.11.3) and that field is set to a non-zero value,

then the copy manager shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

5.12.3 Block device range descriptor

The block device range descriptor is shown in table 75.

Table 75 — Block device range descriptor

Byte \ Bit	7	6	5	4	3	2	1	0
0	(MSB)							
•••	LOGICAL BLOCK ADDRESS							
7								(LSB)
8	(MSB)							
•••	NUMBER OF LOGICAL BLOCKS							
11								(LSB)
12								
•••	Reserved							
15								

The LOGICAL BLOCK ADDRESS field specifies the first LBA on which the copy manager shall operate for this block device range descriptor.

The NUMBER OF LOGICAL BLOCKS field specifies the number of logical blocks on which the copy manager shall operate for this block device range descriptor beginning with the LBA specified by the LOGICAL BLOCK ADDRESS field.

Processing of block device range descriptors with a number of logical blocks that is not a multiple of the OPTIMAL BLOCK ROD LENGTH GRANULARITY field in the block ROD device type specific features descriptor in the ROD token features third-party copy descriptor in the Third-party Copy VPD page (see SPC-6) may incur delays in processing. If the OPTIMAL BLOCK ROD LENGTH GRANULARITY field in the block ROD device type specific features descriptor in the ROD token features third-party copy descriptor in the Third-party Copy VPD page is not reported, then the optimal transfer length granularity in the Block Limits VPD page (see 6.6.4) may indicate the granularity.

For a POPULATE TOKEN command, processing of block device range descriptors where the number of bytes of user data contained in the number of logical blocks exceeds the OPTIMAL BYTES TO TOKEN PER SEGMENT field in the block ROD device type specific features descriptor in the ROD token features third-party copy descriptor in the Third-party Copy VPD page may incur delays in processing.

For a WRITE USING TOKEN command (see 5.59), processing of block device range descriptors where the number of bytes of user data contained in the number of logical blocks exceeds the OPTIMAL BYTES FROM TOKEN PER SEGMENT field in the block ROD device type specific features descriptor in the ROD token features third-party copy descriptor in the Third-party Copy VPD page may incur delays in processing.

If the number of bytes of user data contained in the number of logical blocks is greater than:

1) the value in the MAXIMUM BYTES IN BLOCK ROD field in the block ROD device type specific features descriptor in the ROD token features third-party copy descriptor in the Third-party Copy VPD page, and the MAXIMUM BYTES IN BLOCK ROD field is set to a non-zero value; or
2) the value in the MAXIMUM TRANSFER LENGTH field (see 6.6.4), the MAXIMUM TRANSFER LENGTH field is set to a non-zero value, and the MAXIMUM BYTES IN BLOCK ROD field in the block ROD device type specific features descriptor is not reported,

then the copy manager shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

If the NUMBER OF LOGICAL BLOCKS field is set to zero, then the copy manager shall perform no operation for this block device range descriptor. This condition shall not be considered an error.

If the specified LBA and the specified number of logical blocks exceed the capacity of the medium (see 4.5), then the copy manager shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to LOGICAL BLOCK ADDRESS OUT OF RANGE.

5.13 PRE-FETCH (10) command

The PRE-FETCH (10) command (see table 76) requests that the device server:

a) for any mapped LBAs specified by the command that are not already contained in cache, perform read medium operations and write cache operations (see 4.15); and
b) for any unmapped LBAs specified by the command, update the volatile cache and/or non-volatile cache to prevent retrieval of stale data.

No data shall be transferred to the Data-In Buffer.

NOTE 7 - Migration from the PRE-FETCH (10) command to the PRE-FETCH (16) command is recommended for all implementations.

Table 76 — PRE-FETCH (10) command

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

The OPERATION CODE field is defined in SPC-6 and shall be set to the value shown in table 76 for the PRE-FETCH (10) command.

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 the device server shall:

a) validate the CDB;
b) if the cache has:
   A) sufficient capacity to accept all of the specified logical blocks, then complete the command with CONDITION MET status; or
   B) insufficient capacity to accept all of the specified logical blocks, then complete the command with GOOD status;

   and

c) if one or more of the specified logical blocks are not successfully transferred to the cache for reasons other than lack of cache capacity, then report a deferred error (see SPC-6).

The LOGICAL BLOCK ADDRESS field specifies the LBA of the first logical block (see 4.5) accessed by this command.

A GROUP NUMBER field set to a non-zero value specifies the group into which attributes associated with the command should be collected (see 4.22). A GROUP NUMBER field set to 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 that shall be pre-fetched (i.e., transferred to the cache from the medium), starting with the LBA specified by the LOGICAL BLOCK ADDRESS field. A PREFETCH LENGTH field set to zero specifies that all logical blocks starting with the LBA specified in the LOGICAL BLOCK ADDRESS field to the last logical block on the medium shall be pre-fetched. Any other value specifies the number of logical blocks that shall be pre-fetched. If the specified LBA and the specified prefetch length exceed the capacity of the medium (see 4.5), then the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to LOGICAL BLOCK ADDRESS OUT OF RANGE.

The CONTROL byte is defined in SAM-6.

If the IMMED bit is set to zero, and the specified logical blocks were transferred to the cache without error, then the device server shall complete the command with CONDITION MET status.

If the IMMED bit is set to zero and the cache does not have sufficient capacity to accept all of the specified logical blocks, then the device server shall transfer to the cache as many of the specified logical blocks that fit. If these logical blocks are transferred without error, then the device server shall complete the command with GOOD status.

5.14 PRE-FETCH (16) command

The PRE-FETCH (16) command (see table 77) requests that the device server perform the actions defined for the PRE-FETCH (10) command (see 5.13).

No data shall be transferred to the Data-In Buffer.

Table 77 — PRE-FETCH (16) command

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

The OPERATION CODE field is defined in SPC-6 and shall be set to the value shown in table 77 for the PRE-FETCH (16) command.

See the PRE-FETCH (10) command (see 5.13) for the definitions of the other fields in this command.

5.15 PREVENT ALLOW MEDIUM REMOVAL command

The PREVENT ALLOW MEDIUM REMOVAL command (see table 78) requests that the logical unit enable or disable the removal of the medium. If medium removal is prevented on any I_T nexus that has access to the logical unit, then the logical unit shall not allow medium removal.

Table 78 — PREVENT ALLOW MEDIUM REMOVAL command

Byte \ Bit	7	6	5	4	3	2	1	0
0	OPERATION CODE (1Eh)							
1	Reserved							
2	Reserved							
3	Reserved							
4	Reserved						PREVENT	
5	CONTROL							

The OPERATION CODE field is defined in SPC-6 and shall be set to the value shown in table 78 for the PREVENT ALLOW MEDIUM REMOVAL command.

Table 79 defines the PREVENT field.

Table 79 — PREVENT field

Value	Description
00b	Medium removal is allowed.
01b	Medium removal shall be prevented.
10b to 11b	Obsolete

The CONTROL byte is defined in SAM-6.

The prevention of medium removal shall begin when any application client issues a PREVENT ALLOW MEDIUM REMOVAL command with the PREVENT field set to 01b (i.e., medium removal prevented). The prevention of medium removal for the logical unit shall no longer be prevented after:

a) one of the following occurs for each I_T nexus through which medium removal had been prevented:
   A) receipt of a PREVENT ALLOW MEDIUM REMOVAL command with the PREVENT field set to 00b; or
   B) an I_T nexus loss;
b) a power on;
c) a hard reset; or
d) a logical unit reset.

If possible, the device server shall perform a synchronize cache operation before ending the prevention of medium removal.

If a persistent reservation or registration is being preempted by a PERSISTENT RESERVE OUT command with PREEMPT AND ABORT service action (see SPC-6) or PERSISTENT RESERVE OUT command with CLEAR service action (see SPC-6), then the equivalent of a PREVENT ALLOW MEDIUM REMOVAL command with the PREVENT field set to 00b shall be processed for each I_T nexuses associated with the persistent reservation or registrations being preempted allowing an application client to override the prevention of medium removal function for a SCSI initiator port (e.g., an initiator port is not operating correctly).

While a prevention of medium removal condition is in effect, the logical unit shall inhibit mechanisms that allow removal of the medium by an operator.

5.16 READ (10) command

The READ (10) command (see table 80) requests that the device server:

a) perform read operations from the specified LBAs: and
b) transfer the requested logical block data to the Data-In Buffer.

The logical block data transferred to the Data-In Buffer shall include protection information based on the value in the RDPROTECT field (see table 81) and the medium format.

NOTE 8 - Migration from the READ (10) command to the READ (16) command is recommended for all implementations.

Table 80 — READ (10) command

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

The OPERATION CODE field is defined in SPC-6 and shall be set to the value shown in table 80 for the READ (10) command.

The device server shall check the protection information from the read operations before returning status for the command based on the RDPROTECT field as described in table 81. All footnotes for table 81 are at the end of the table.

Table 81 — RDPROTECT field (part 1 of 3)

<table>
<tr><th>Code</th><th>Logical unit formatted with protection information</th><th>Shall device server transmit protection information?</th><th>Field in protection information [h]</th><th>Extended INQUIRY Data VPD page bit value [g]</th><th>If check fails [d] [f], additional sense code</th></tr>
<tr><td rowspan="7">000b</td><td rowspan="6">Yes [j]</td><td rowspan="7">No</td><td rowspan="2">LOGICAL BLOCK GUARD</td><td>GRD_CHK = 1</td><td>LOGICAL BLOCK GUARD CHECK FAILED</td></tr>
<tr><td>GRD_CHK = 0</td><td>No check performed</td></tr>
<tr><td rowspan="2">LOGICAL BLOCK APPLICATION TAG</td><td>APP_CHK = 1 [c]</td><td>LOGICAL BLOCK APPLICATION TAG CHECK FAILED</td></tr>
<tr><td>APP_CHK = 0</td><td>No check performed</td></tr>
<tr><td rowspan="2">LOGICAL BLOCK REFERENCE TAG</td><td>REF_CHK = 1 [i]</td><td>LOGICAL BLOCK REFERENCE TAG CHECK FAILED</td></tr>
<tr><td>REF_CHK = 0</td><td>No check performed</td></tr>
<tr><td>No</td><td colspan="3">No protection information available to check</td></tr>
<tr><td rowspan="7">001b
101b [b]</td><td rowspan="6">Yes</td><td rowspan="6">Yes [e]</td><td rowspan="2">LOGICAL BLOCK GUARD</td><td>GRD_CHK = 1</td><td>LOGICAL BLOCK GUARD CHECK FAILED</td></tr>
<tr><td>GRD_CHK = 0</td><td>No check performed</td></tr>
<tr><td rowspan="2">LOGICAL BLOCK APPLICATION TAG</td><td>APP_CHK = 1 [c]</td><td>LOGICAL BLOCK APPLICATION TAG CHECK FAILED</td></tr>
<tr><td>APP_CHK = 0</td><td>No check performed</td></tr>
<tr><td rowspan="2">LOGICAL BLOCK REFERENCE TAG</td><td>REF_CHK = 1 [i]</td><td>LOGICAL BLOCK REFERENCE TAG CHECK FAILED</td></tr>
<tr><td>REF_CHK = 0</td><td>No check performed</td></tr>
<tr><td>No [a]</td><td colspan="4">No protection information available to transmit to the Data-In Buffer or for checking</td></tr>
</table>

Table 81 — RDPROTECT field (part 2 of 3)

<table>
<tr><th>Code</th><th>Logical unit formatted with protection information</th><th>Shall device server transmit protection information?</th><th>Field in protection information [h]</th><th>Extended INQUIRY Data VPD page bit value [g]</th><th>If check fails [d f], additional sense code</th></tr>
<tr><td rowspan="6">010b [b]</td><td rowspan="5">Yes</td><td rowspan="5">Yes [e]</td><td>LOGICAL BLOCK GUARD</td><td colspan="2">No check performed</td></tr>
<tr><td rowspan="2">LOGICAL BLOCK APPLICATION TAG</td><td>APP_CHK = 1 [c]</td><td>LOGICAL BLOCK APPLICATION TAG CHECK FAILED</td></tr>
<tr><td>APP_CHK = 0</td><td>No check performed</td></tr>
<tr><td rowspan="2">LOGICAL BLOCK REFERENCE TAG</td><td>REF_CHK = 1 [i]</td><td>LOGICAL BLOCK REFERENCE TAG CHECK FAILED</td></tr>
<tr><td>REF_CHK = 0</td><td>No check performed</td></tr>
<tr><td>No [a]</td><td colspan="4">No protection information available to transmit to the Data-In Buffer or for checking</td></tr>
<tr><td rowspan="4">011b [b]</td><td rowspan="3">Yes</td><td rowspan="3">Yes [e]</td><td>LOGICAL BLOCK GUARD</td><td colspan="2">No check performed</td></tr>
<tr><td>LOGICAL BLOCK APPLICATION TAG</td><td colspan="2">No check performed</td></tr>
<tr><td>LOGICAL BLOCK REFERENCE TAG</td><td colspan="2">No check performed</td></tr>
<tr><td>No [a]</td><td colspan="4">No protection information available to transmit to the Data-In Buffer or for checking</td></tr>
<tr><td rowspan="5">100b [b]</td><td rowspan="4">Yes</td><td rowspan="4">Yes [e]</td><td rowspan="2">LOGICAL BLOCK GUARD</td><td>GRD_CHK = 1</td><td>LOGICAL BLOCK GUARD CHECK FAILED</td></tr>
<tr><td>GRD_CHK = 0</td><td>No check performed</td></tr>
<tr><td>LOGICAL BLOCK APPLICATION TAG</td><td colspan="2">No check performed</td></tr>
<tr><td>LOGICAL BLOCK REFERENCE TAG</td><td colspan="2">No check performed</td></tr>
<tr><td>No [a]</td><td colspan="4">No protection information available to transmit to the Data-In Buffer or for checking</td></tr>
<tr><td>110b to 111b</td><td colspan="5">Reserved</td></tr>
</table>

Table 81 — RDPROTECT field (part 3 of 3)

Code	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

[a] If the logical unit supports protection information (see 4.21) and has not been formatted with protection information, then the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.

[b] If the logical unit does not support protection information, then the device server should 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.

[c] If the device server has knowledge of the contents of the LOGICAL BLOCK APPLICATION TAG field, then the device server shall check each logical block application tag. If the ATO bit in the Control mode page (see SPC-6) is set to one, then this knowledge is acquired from:

a) the EXPECTED LOGICAL BLOCK APPLICATION TAG field and the LOGICAL BLOCK APPLICATION TAG MASK field in the CDB, if a READ (32) command (see 5.19) is received by the device server;
b) the Application Tag mode page (see 6.5.3), if a command other than READ (32) is received by the device server, and the ATMPE bit in the Control mode page (see SPC-6) is set to one; or
c) a method not defined by this standard, if a command other than READ (32) is received by the device server, and the ATMPE bit is set to zero.

[d] If the device server terminates the command with CHECK CONDITION status, then the device server shall set the sense key to ABORTED COMMAND.

[e] The device server shall transmit protection information to the Data-In Buffer.

[f] If multiple errors occur while the device server is processing the command, then the selection by the device server of which error to report is not defined by this standard.

[g] See the Extended INQUIRY Data VPD page (see SPC-6) for the definitions of the GRD_CHK bit, the APP_CHK bit, and the REF_CHK bit.

[h] If the device server detects:

a) a LOGICAL BLOCK APPLICATION TAG field set to FFFFh, and type 1 protection (see 4.21.2.3) or type 2 protection (see 4.21.2.4) is enabled; or
b) a LOGICAL BLOCK APPLICATION TAG field set to FFFFh, the LOGICAL BLOCK REFERENCE TAG field set to FFFF_FFFFh, and type 3 protection (see 4.21.2.5) is enabled,

then the device server shall not check any protection information in the associated protection information interval.

[i] If type 1 protection is enabled, then the device server shall check the logical block reference tag by comparing it to the lower four bytes of the LBA associated with the logical block. If type 2 protection or type 3 protection is enabled, and the device server has knowledge of the contents of the LOGICAL BLOCK REFERENCE TAG field, then the device server shall check each logical block reference tag. If type 2 protection is enabled, then this knowledge may be acquired through the EXPECTED INITIAL LOGICAL BLOCK REFERENCE TAG field in a READ (32) command (see 5.19). If type 3 protection is enabled, then the method for acquiring this knowledge is not defined by this standard.

[j] If the DPICZ bit in the Control mode page (see SPC-6) is set to one, then protection information shall not be checked.

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 6.5.6). 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 (see 4.15). 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 replacement strategy are not defined by this standard.

NOTE 9 - The DPO bit is used to control replacement of logical blocks in the cache when the application client has information on the future usage of the logical blocks. If the DPO bit is set to one, then 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 are not to be put in the cache nor retained by the cache. If the DPO bit is set to zero, then the application client is specifying that the logical blocks accessed by this command are likely to be accessed again in the near future.

A force unit access (FUA) bit set to one specifies that the device server shall read the logical blocks from:

a) the specified data pattern for that LBA (e.g., the data pattern for unmapped data (see 4.7.4.4));
b) the non-volatile cache, if any; or
c) the medium.

If the FUA bit is set to one and a volatile cache contains a more recent version of a logical block than the non-volatile cache, if any, or the medium, then, before reading the logical block, the device server shall write the logical block to:

a) the non-volatile cache, if any; or
b) the medium.

An FUA bit set to zero specifies that the device server may read the logical blocks from:

a) the volatile cache, if any;
b) the specified data pattern for that LBA (e.g., the data pattern for unmapped data (see 4.7.4.4))
c) the non-volatile cache, if any; or
d) the medium.

If rebuild assist mode (see 4.19) is supported and not enabled, then the device server shall ignore the rebuild assist recovery control (RARC) bit. If rebuild assist mode is supported and enabled, then the RARC bit specifies that the device server shall perform read medium operations as defined in 4.19.3.2 and 4.19.3.3.

If the rebuild assist mode is not supported and the RARC bit is set to one, then the device server should 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.

See the PRE-FETCH (10) command (see 5.13) for the definition of the LOGICAL BLOCK ADDRESS field.

See the PRE-FETCH (10) command and 4.22 for the definition of the GROUP NUMBER field.

The TRANSFER LENGTH field specifies the number of contiguous logical blocks of data that shall be read and transferred to the Data-In Buffer, starting with the logical block referenced by the LBA specified by the LOGICAL BLOCK ADDRESS field. A TRANSFER LENGTH field set to zero specifies that no logical blocks shall be read or transferred. This condition shall not be considered an error. Any other value specifies the number of logical blocks that shall be read and transferred. If the specified LBA and the specified transfer length exceed the capacity of the medium (see 4.5), then the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to LOGICAL BLOCK ADDRESS OUT OF RANGE. The TRANSFER LENGTH field is constrained by the MAXIMUM TRANSFER LENGTH field (see 6.6.4).

The CONTROL byte is defined in SAM-6.

5.17 READ (12) command

The READ (12) command (see table 82) requests that the device server perform the actions defined for the READ (10) command (see 5.16).

NOTE 10 - Migration from the READ (12) command to the READ (16) command is recommended for all implementations.

Table 82 — READ (12) command

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

The OPERATION CODE field is defined in SPC-6 and shall be set to the value shown in table 82 for the READ (12) command.

The CONTROL byte is defined in SAM-6.

See the READ (10) command (see 5.16) for the definitions of the other fields in this command.

5.18 READ (16) command

The READ (16) command (see table 83) requests that the device server perform the actions defined for the READ (10) command (see 5.16).

Table 83 — READ (16) command

Byte \ Bit	7	6	5	4	3	2	1	0
0	OPERATION CODE (88h)							
1	RDPROTECT			DPO	FUA	RARC	Obsolete	DLD2
2	(MSB)							
•••	LOGICAL BLOCK ADDRESS							
9								(LSB)
10	(MSB)							
•••	TRANSFER LENGTH							
13								(LSB)
14	DLD1	DLD0	GROUP NUMBER					
15	CONTROL							

The OPERATION CODE field is defined in SPC-6 and shall be set to the value shown in table 83 for the READ (16) command.

The command duration limit (see SAM-6) is indicated in the command duration limit descriptor (see SPC-6). Which command duration limit descriptor, if any, applies to this command is specified by the DLD2 bit, the DLD1 bit, and the DLD0 bit, as shown in table 84. The CDLP field and the RWCDLP bit in the REPORT SUPPORTED OPERATION CODES parameter data (see SPC-6) indicate that the command duration limit descriptor is in:

a) the Command Duration Limit A mode page (see SPC-6);
b) the Command Duration Limit B mode page (see SPC-6);
c) the Command Duration Limit T2A mode page (see SPC-6); or
d) the Command Duration Limit T2B mode page (see SPC-6).

Table 84 — Duration limit descriptor DLD bits

Duration limit descriptor bits			Description
DLD2	DLD1	DLD0	
0b	0b	0b	Command is not a duration limited command (see SAM-6)
0b	0b	1b	First command duration limit descriptor
0b	1b	0b	Second command duration limit descriptor
0b	1b	1b	Third command duration limit descriptor
1b	0b	0b	Fourth command duration limit descriptor
1b	0b	1b	Fifth command duration limit descriptor
1b	1b	0b	Sixth command duration limit descriptor
1b	1b	1b	Seventh command duration limit descriptor

The CONTROL byte is defined in SAM-6.

See the READ (10) command (see 5.16) for the definitions of the other fields in this command.

5.19 READ (32) command

The READ (32) command (see table 85) requests that the device server perform the actions defined for the READ (10) command (see 5.16).

The device server shall only process a READ (32) command if type 2 protection is enabled (see 4.21.2.4).

Table 85 — READ (32) command

Byte \ Bit	7	6	5	4	3	2	1	0
0	OPERATION CODE (7Fh)							
1	CONTROL							
2 ••• 5	Reserved							
6	Reserved		GROUP NUMBER					
7	ADDITIONAL CDB LENGTH (18h)							
8	(MSB)	SERVICE ACTION (0009h)						
9								(LSB)
10	RDPROTECT			DPO	FUA	RARC	Obsolete	Reserved
11	Reserved					DLD2	DLD1	DD0
12	(MSB)	LOGICAL BLOCK ADDRESS						
•••								
19								(LSB)
20	(MSB)	EXPECTED INITIAL LOGICAL BLOCK REFERENCE TAG						
•••								
23								(LSB)
24	(MSB)	EXPECTED LOGICAL BLOCK APPLICATION TAG						
25								(LSB)
26	(MSB)	LOGICAL BLOCK APPLICATION TAG MASK						
27								(LSB)
28	(MSB)	TRANSFER LENGTH						
•••								
31								(LSB)

The OPERATION CODE field, the ADDITIONAL CDB LENGTH field, and the SERVICE ACTION field are defined in SPC-6 and shall be set to the values shown in table 85 for the READ (32) command.

The CONTROL byte is defined in SAM-6.

See the READ (10) command (see 5.16) for the definitions of the GROUP NUMBER field, the RDPROTECT field, the DPO bit, the FUA bit, the RARC bit, the LOGICAL BLOCK ADDRESS field, and the TRANSFER LENGTH field.

See the READ (16) command (see 5.18) for the definitions of the DLD2 bit, DLD1 bit, and DLD0 bit.

If checking of the LOGICAL BLOCK REFERENCE TAG field is enabled (see table 81 in 5.16), then the EXPECTED INITIAL LOGICAL BLOCK REFERENCE TAG field contains the value of the LOGICAL BLOCK REFERENCE TAG field expected in the protection information of the first logical block accessed by the command instead of a value based on the LBA (see 4.21.3).

If the ATO bit is set to one in the Control mode page (see SPC-6), and checking of the LOGICAL BLOCK APPLICATION TAG field is enabled (see table 81 in 5.16), then the LOGICAL BLOCK APPLICATION TAG MASK field contains a value that is a bit mask for enabling the checking of the LOGICAL BLOCK APPLICATION TAG field in every instance of protection information for each logical block accessed by the command. A LOGICAL BLOCK APPLICATION TAG MASK field bit set to one enables the checking of the corresponding bit of the EXPECTED LOGICAL BLOCK APPLICATION TAG field with the corresponding bit of the LOGICAL BLOCK APPLICATION TAG field in every instance of protection information. A LOGICAL BLOCK APPLICATION TAG MASK field bit set to zero disables the checking of the corresponding bit of the EXPECTED LOGICAL BLOCK APPLICATION TAG field with the corresponding bit of the LOGICAL BLOCK APPLICATION TAG field in every instance of protection information.

If the ATO bit is set:

a) to zero; or
b) to one in the Control mode page (see SPC-6), and checking of the LOGICAL BLOCK APPLICATION TAG field is disabled (see table 81),

then the LOGICAL BLOCK APPLICATION TAG MASK field and the EXPECTED LOGICAL BLOCK APPLICATION TAG field shall be ignored

5.20 READ CAPACITY (10) command

5.20.1 READ CAPACITY (10) overview

The READ CAPACITY (10) command (see table 86) requests that the device server transfer eight bytes of parameter data describing the capacity and medium format of the direct access block device to the Data-In Buffer. This command may be processed as if it has a HEAD OF QUEUE task attribute (see 4.16). If the logical unit supports protection information (see 4.21) or logical block provisioning management (see 4.7), then the application client should use the READ CAPACITY (16) command (see 5.21) instead of the READ CAPACITY (10) command.

NOTE 11 - Migration from the READ CAPACITY (10) command to the READ CAPACITY (16) command is recommended for all implementations.

Table 86 — READ CAPACITY (10) command

Byte \ Bit	7	6	5	4	3	2	1	0
0	OPERATION CODE (25h)							
1	Reserved							Obsolete
2 ••• 5	Obsolete							
6 7	Reserved							
8	Reserved							Obsolete
9	CONTROL							

The OPERATION CODE field is defined in SPC-6 and shall be set to the value shown in table 86 for the READ CAPACITY (10) command.

The CONTROL byte is defined in SAM-6.

5.20.2 READ CAPACITY (10) parameter data

The READ CAPACITY (10) parameter data is shown in table 87. Any time the READ CAPACITY (10) parameter data changes, the device server should establish a unit attention condition as described in 4.10.

Table 87 — READ CAPACITY (10) parameter data

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

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 direct access block device; or
b) FFFF_FFFFh, if the LBA of the last logical block on the direct access block device is greater than the maximum value that is able to be specified in the RETURNED LOGICAL BLOCK ADDRESS field.

If the RETURNED LOGICAL BLOCK ADDRESS field is set to FFFF_FFFFh, then the application client should issue a READ CAPACITY (16) command (see 5.21) to request that the device server transfer the READ CAPACITY (16) parameter data to the Data-In Buffer.

The LOGICAL BLOCK LENGTH IN BYTES field contains the number of bytes of user data in a logical block.

5.21 READ CAPACITY (16) command

5.21.1 READ CAPACITY (16) command overview

The READ CAPACITY (16) command (see table 88) requests that the device server transfer parameter data describing the capacity and medium format of the direct access block device to the Data-In Buffer. This command is mandatory if the logical unit supports protection information (see 4.21) or logical block provisioning management (see 4.7) and is optional otherwise. This command may be processed as if it has a HEAD OF QUEUE task attribute (see 4.16).

This command uses the SERVICE ACTION IN (16) CDB format (see clause B.2).

Table 88 — READ CAPACITY (16) command

Bit / Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (9Eh)							
1	Reserved			SERVICE ACTION (10h)				
2 ••• 9	Obsolete							
10 ••• 13	(MSB) ALLOCATION LENGTH (LSB)							
14	Reserved							Obsolete
15	CONTROL							

The OPERATION CODE field and SERVICE ACTION field are defined in SPC-6 and shall be set to the values shown in table 88 for the READ CAPACITY (16) command.

The ALLOCATION LENGTH field is defined in SPC-6.

The CONTROL byte is defined in SAM-6.

5.21.2 READ CAPACITY (16) parameter data

The READ CAPACITY (16) parameter data is shown in table 89. Any time the READ CAPACITY (16) parameter data changes, the device server should establish a unit attention condition as described in 4.10.

Table 89 — READ CAPACITY (16) parameter data

Byte \ Bit	7	6	5	4	3	2	1	0
0	(MSB)							
•••	RETURNED LOGICAL BLOCK ADDRESS							
7								(LSB)
8	(MSB)							
•••	LOGICAL BLOCK LENGTH IN BYTES							
11								(LSB)
12	Reserved		Restricted for ZBC-2		P_TYPE			PROT_EN
13	P_I_EXPONENT				LOGICAL BLOCKS PER PHYSICAL BLOCK EXPONENT			
14	LBPME	LBPRZ	(MSB)	LOWEST ALIGNED LOGICAL BLOCK ADDRESS				
15								(LSB)
16								
•••	Reserved							
31								

The RETURNED LOGICAL BLOCK ADDRESS field and LOGICAL BLOCK LENGTH IN BYTES field of the READ CAPACITY (16) parameter data are defined in the READ CAPACITY (10) parameter data (see 5.20). The maximum value that shall be returned in the RETURNED LOGICAL BLOCK ADDRESS field is FFFF_FFFF_FFFF_FFFEh.

The protection type (P_TYPE) field and the protection enable (PROT_EN) bit (see table 90) indicate the logical unit's current type of protection.

Table 90 — P_TYPE field and PROT_EN bit

P_TYPE	PROT_EN	Description
n/a	0	The logical unit is formatted to type 0 protection (see 4.21.2.2).
000b	1	The logical unit is formatted to type 1 protection (see 4.21.2.3).
001b	1	The logical unit is formatted to type 2 protection (see 4.21.2.4).
010b	1	The logical unit is formatted to type 3 protection (see 4.21.2.5).
011b to 111b	1	Reserved

The P_I_EXPONENT field may be used to determine the number of protection information intervals placed within each logical block (see 5.4.2).

The number of protection information intervals is calculated as follows:

$$\text{number of protection information intervals} = 2^{(\text{p\_i exponent})}$$

where:

p_i exponent is the contents of the P_I EXPONENT field.

The LOGICAL BLOCKS PER PHYSICAL BLOCK EXPONENT field is shown in table 91.

Table 91 — LOGICAL BLOCKS PER PHYSICAL BLOCK EXPONENT field

Code	Description
0	One or more physical blocks per logical block [a]
n > 0	2^n logical blocks per physical block
[a] The number of physical blocks per logical block is not reported.	

A logical block provisioning management enabled (LBPME) bit set to one indicates that the logical unit implements logical block provisioning management (i.e., is resource provisioned or thin provisioned) (see 4.7.3). An LBPME bit set to zero indicates that the logical unit does not implement logical block provisioning management (e.g., is fully provisioned (see 4.7.2)).

The logical block provisioning read zeros (LBPRZ) bit shall be set to one if the LBPRZ field (see 6.6.9) is set to xx1b. The LBPRZ bit shall be set to zero if the LBPRZ field is not set to xx1b.

The LOWEST ALIGNED LOGICAL BLOCK ADDRESS field indicates the LBA of the first logical block that is located at the beginning of a physical block (see 4.6).

5.22 READ DEFECT DATA (10) command

5.22.1 READ DEFECT DATA (10) command overview

The READ DEFECT DATA (10) command (see table 92) requests that the device server transfer parameter data (see 5.22.2) containing a four-byte header, the PLIST, and/or the GLIST to the Data-In Buffer.

If the device server is unable to access a specified defect list as a result of a medium error, then the device server shall terminate the command with CHECK CONDITION status with the sense key set to MEDIUM ERROR and the additional sense code set to DEFECT LIST NOT FOUND.

If the device server is unable to access a specified defect list as a result of an error other than a medium error or because a specified defect list does not exist, then the device server shall either:

1) terminate the command with CHECK CONDITION status with the sense key set to NO SENSE and the additional sense code set to DEFECT LIST NOT FOUND; or
2) return only the READ DEFECT DATA parameter data header, with the DEFECT LIST LENGTH field set to zero.

Device servers may or may not return a defect list until after a successful completion of a FORMAT UNIT command (see 5.3).

NOTE 12 - Migration from the READ DEFECT DATA (10) command to the READ DEFECT DATA (12) command is recommended for all implementations.

Table 92 — 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)	ALLOCATION LENGTH						
8								(LSB)
9	CONTROL							

The OPERATION CODE field is defined in SPC-6 and shall be set to the value shown in table 92 for the READ DEFECT DATA (10) command.

Table 93 defines the request PLIST (REQ_PLIST) bit and the request GLIST (REQ_GLIST) bit.

Table 93 — REQ_PLIST bit and REQ_GLIST bit

REQ_PLIST	REQ_GLIST	Description
0	0	The device server shall return only the first four bytes of the READ DEFECT DATA parameter data (i.e., the parameter data header), with the DEFECT LIST LENGTH field set to zero.
	1	The device server shall return the READ DEFECT DATA parameter data header and include the GLIST, if any, in the defect list.
1	0	The device server shall return the READ DEFECT DATA parameter data header and include the PLIST, if any, in the defect list.
	1	The device server shall return the READ DEFECT DATA parameter data header and include the both the PLIST, if any, and the GLIST, if any, in the defect list. Whether the PLIST and GLIST are merged or not is vendor specific.

The DEFECT LIST FORMAT field specifies the address descriptor format type (see 6.2) that the device server should use for the defect list. A device server unable to return the requested address descriptor format shall return the address descriptors in their default format and indicate that format type in the DEFECT LIST FORMAT field in (see 5.22.2 and 5.23.2).

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

The ALLOCATION LENGTH field is defined in SPC-6. If the length of the address descriptors that the device server has to report is greater than the maximum value that is able to be specified by the ALLOCATION LENGTH field, then the device server shall transfer no data and shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.

The CONTROL byte is defined in SAM-6.

5.22.2 READ DEFECT DATA (10) parameter data

The READ DEFECT DATA (10) parameter data (see table 94) contains a four-byte header, followed by zero or more address descriptors.

Table 94 — READ DEFECT DATA (10) parameter data

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

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

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

The DEFECT LIST FORMAT field indicates the format of the address descriptors returned in the defect list. This field is defined in 6.2.

If the device server returns short block format address descriptors (see 6.2.2) or long block format address descriptors (see 6.2.5), then the address descriptors contain vendor-specific values.

The DEFECT LIST LENGTH field indicates the length in bytes of the defect list. 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 6.2).

The defect list contains address descriptors (see 6.2).

5.23 READ DEFECT DATA (12) command

5.23.1 READ DEFECT DATA (12) command overview

The READ DEFECT DATA (12) command (see table 95) requests that the device server transfer parameter data (see 5.23.2) containing a four-byte header, the PLIST, and/or the GLIST to the Data-In Buffer.

An application client determines the length of the defect list by sending a READ DEFECT DATA (12) command with an ALLOCATION LENGTH field set to eight and the ADDRESS DESCRIPTOR INDEX field set to 0000_0000h. The device server returns the defect list header that contains the length of the defect list.

Table 95 — 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	(MSB)							
•••	ADDRESS DESCRIPTOR INDEX							
5								(LSB)
6	(MSB)							
•••	ALLOCATION LENGTH							
9								(LSB)
10	Reserved							
11	CONTROL							

The OPERATION CODE field is defined in SPC-6 and shall be set to the value shown in table 95 for the READ DEFECT DATA (12) command.

See the READ DEFECT DATA (10) command (see 5.22) for the definitions of the REQ_PLIST bit, the REQ_GLIST bit, and the DEFECT LIST FORMAT field.

The ADDRESS DESCRIPTOR INDEX field specifies the index of the first address descriptor (see 6.2) in the defect list that the device server shall return. If the ADDRESS DESCRIPTOR INDEX field is set to:

a) a value less than the number of available address descriptors, then the device server shall transfer a defect list beginning with the address descriptor that is at the ADDRESS DESCRIPTOR INDEX field value multiplied by the size of the address descriptor; or
b) a value greater than or equal to the number of available address descriptors, then the device server shall return a zero length defect list.

The ALLOCATION LENGTH field is defined in SPC-6, however if the length of all the address descriptors that are available is greater than FFFF_FFFFh, then the device server shall transfer the length of address descriptors specified by the allocation length or the DEFECT LIST LENGTH field value plus eight, whichever is less, and complete the command with GOOD status.

The CONTROL byte is defined in SAM-6.

5.23.2 READ DEFECT DATA (12) parameter data

The READ DEFECT DATA (12) parameter data (see table 96) contains an eight-byte header, followed by zero or more address descriptors.

Table 96 — READ DEFECT DATA (12) parameter data

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

The GENERATION CODE field is a two-byte counter that shall be incremented by one by the device server every time the defect list is changed. A GENERATION CODE field set to 0000h indicates the generation code is not supported. If the GENERATION CODE field is supported, then the GENERATION CODE field shall be initialized to at least 0001h at power on and the device server shall wrap this field to 0001h as the next increment after reaching its maximum value (i.e., FFFFh).

Application clients that use the GENERATION CODE field should read this field often enough to ensure that the contents of this field do not increment a multiple of 65 535 times between readings.

The DEFECT LIST LENGTH field indicates the length in bytes of address descriptors from the beginning address descriptor specified by the ADDRESS DESCRIPTOR INDEX field to the last address descriptor available to be returned. A value of FFFF_FFFFh in the DEFECT LIST LENGTH field indicates that more than FFFF_FFFEh bytes are available.

See the READ DEFECT DATA (10) command (see 5.22) for the definitions of the other fields in the READ DEFECT DATA (12) parameter data.

5.24 REASSIGN BLOCKS command

5.24.1 REASSIGN BLOCKS command overview

The REASSIGN BLOCKS command (see table 97) requests that the device server perform a reassign operation on one or more LBAs (e.g., LBAs referencing logical blocks on which unrecovered read errors occurred) to another area on the medium set aside for this purpose and to add the physical blocks containing those logical blocks to the GLIST. This command shall not alter the contents of the PLIST (see 4.13).

The parameter list provided in the Data-Out Buffer contains a reassign LBA list that contains the LBAs of the logical blocks to be reassigned. The device server shall reassign the parts of the medium used for each logical block referenced by an LBA in the reassign LBA list. More than one physical block may be reassigned by each LBA. If the device server recovers logical block data from the original logical block, then the device server shall perform a write medium operation to that LBA using the recovered logical block data, which writes to the logical block referenced by the reassigned LBA.

The device server shall invalidate any of the specified LBAs that are in cache.

If the device server does not recover logical block data in a fully provisioned logical unit (see 4.7.2), then the device server shall:

a) write vendor specific data as the user data, if the RBWZ bit is set to zero;
b) write zeros as the user data, if the RBWZ bit is set to one; and
c) write a default value of FFFF_FFFF_FFFF_FFFFh as the protection information, if enabled (see 4.21.2).

If the device server does not recover logical block data in a resource provisioned logical unit (see 4.7.3.2) or a thin provisioned logical unit (see 4.7.3.3), then the device server shall, for each specified LBA, either:

a) unmap the specified LBA; or
b) perform the following operations:
    A) write vendor-specific data as the user data, if the RBWZ bit is set to zero;
    B) write zeros as the user data, if the RBWZ bit is set to one; and
    C) write a default value of FFFF_FFFF_FFFF_FFFFh as the protection information, if enabled.

The vendor-specific data written as user data may contain remnants of the original logical block (e.g., partially or fully recovered user data).

The data in all other logical blocks on the medium shall be preserved.

Specifying an LBA to be reassigned that previously has been reassigned causes the device server to reassign that LBA again.

If the device server terminates the REASSIGN BLOCKS command with CHECK CONDITION status, and the sense data COMMAND-SPECIFIC INFORMATION field contains a valid LBA, then the application client should remove all LBAs from the reassign LBA list prior to the one returned in the COMMAND-SPECIFIC INFORMATION field. If the sense key is set to MEDIUM ERROR and the INFORMATION field contains the valid LBA, then the application client should insert that LBA into the reassign LBA list and reissue the REASSIGN BLOCKS command with the new reassign LBA 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 reassign LBA list.

Table 97 — REASSIGN BLOCKS command

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

The OPERATION CODE field is defined in SPC-6 and shall be set to the value shown in table 97 for the REASSIGN BLOCKS command.

A long LBA (LONGLBA) bit set to zero specifies that the reassign LBA list in the REASSIGN BLOCKS parameter list (see 5.24.2) contains four-byte LBAs. A LONGLBA bit set to one specifies that the reassign LBA list in the REASSIGN BLOCKS parameter list contains eight-byte LBAs.

A long list (LONGLIST) bit set to zero specifies the REASSIGN BLOCKS short parameter list header (see table 99) is used. A LONGLIST bit set to one specifies the REASSIGN BLOCKS long parameter list header (see table 100) is used.

The CONTROL byte is defined in SAM-6.

5.24.2 REASSIGN BLOCKS parameter list

The REASSIGN BLOCKS parameter list (see table 98) contains a four-byte parameter list header followed by a reassign LBA list containing one or more LBAs.

Table 98 — REASSIGN BLOCKS parameter list

Byte \ Bit	7	6	5	4	3	2	1	0
0 ••• 3	Parameter list header (see table 99 or table 100)							
Reassign LBA list (if any)								
4 ••• 7 or 11	Reassign LBA [first] (see table 101 or table 102)							
n-4 or n-7 ••• n	Reassign LBA [last] (see table 101 or table 102)							

The REASSSIGN BLOCKS short parameter list header is shown in table 99.

Table 99 — REASSIGN BLOCKS short parameter list header

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

The REASSSIGN BLOCKS long parameter list header is shown in table 100.

Table 100 — REASSIGN BLOCKS long parameter list header

Byte \ Bit	7	6	5	4	3	2	1	0
0	(MSB)	REASSIGN LBA LENGTH						
•••								
3								(LSB)

The REASSIGN LBA LENGTH field specifies the total length in bytes of the reassign LBA list. The REASSIGN LBA LENGTH field does not include the parameter list header length and is equal to:

a) four times the number of LBAs, if the LONGLBA bit (see 5.24) is set to zero; or
b) eight times the number of LBAs, if the LONGLBA bit is set to one.

The REASSIGN LBA LIST field contains a list of LBAs to be reassigned. The LBAs shall be sorted in ascending order.

If the LONGLBA bit is set to zero, then table 101 defines the format of the reassigned LBA.

Table 101 — Reassign LBA if the LONGLBA bit is set to zero

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

The REASSIGN LOGICAL BLOCK ADDRESS field specifies an LBA to be reassigned.

If the LONGLBA bit is set to one, then table 102 defines the reassign LBA.

Table 102 — Reassign LBA if the LONGLBA bit is set to one

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

The REASSIGN LOGICAL BLOCK ADDRESS field specifies an LBA to be reassigned.

If a specified LBA exceeds the capacity of the medium (see 4.5), 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 should be set to LOGICAL BLOCK ADDRESS OUT OF RANGE or may be set to INVALID FIELD IN PARAMETER LIST.

If the direct access block device has insufficient capacity to reassign all of the specified LBAs, then the device server shall terminate the command with CHECK CONDITION status with the sense key set to HARDWARE ERROR and the additional sense code set to NO DEFECT SPARE LOCATION AVAILABLE.

If the direct access block device is unable to complete a REASSIGN BLOCKS command without error, then the device server shall terminate the command with CHECK CONDITION status with the appropriate sense data (see 4.18 and SPC-6).

If one or more LBAs are not reassigned, then the device server shall report the first LBA not reassigned in the COMMAND-SPECIFIC INFORMATION field of the sense data (see SPC-6). If:

a) information about the first LBA not reassigned is not available;
b) all the LBAs have been reassigned; or
c) the first LBA not reassigned does not fit in the COMMAND-SPECIFIC INFORMATION field, then the device server shall report the following value in the COMMAND-SPECIFIC INFORMATION field of the sense data (see SPC-6):
   A) FFFF_FFFFh if fixed format sense data is being used; or
   B) FFFF_FFFF_FFFF_FFFFh if descriptor format sense data is being used.

If the REASSIGN BLOCKS command failed as a result of an unexpected unrecovered read error that would cause the loss of data in a logical block not specified in the reassign LBA list, then the LBA of the logical block with the unrecovered read error is reported in the INFORMATION field of the sense data (see 4.18.1).

5.25 RECEIVE ROD TOKEN INFORMATION

5.25.1 RECEIVE ROD TOKEN INFORMATION overview

The RECEIVE ROD TOKEN INFORMATION command (see SPC-6) provides a method for an application client to receive information about the results of a previous or current block device ROD token operation. Table 103 shows the operations and a reference to the subclause where each topic is described.

Table 103 — RECEIVE ROD TOKEN INFORMATION reference

Command originating the operation	Command reference	RECEIVE ROD TOKEN INFORMATION returned parameter data reference
POPULATE TOKEN	5.12	5.25.2
WRITE USING TOKEN	5.59	5.25.3

5.25.2 RECEIVE ROD TOKEN INFORMATION parameter data for POPULATE TOKEN command

If a RECEIVE ROD TOKEN INFORMATION command (see SPC-6) specifies a list identifier that matches the list identifier specified in a previous POPULATE TOKEN command (see 5.12) received on the same I_T nexus, then table 104 shows the parameter data returned by the copy manager.

Table 104 — RECEIVE ROD TOKEN INFORMATION parameter data for POPULATE TOKEN

Byte \ Bit	7	6	5	4	3	2	1	0
0	(MSB)							
•••	AVAILABLE DATA (n - 3)							
3								(LSB)
4	Reserved			RESPONSE TO SERVICE ACTION (10h)				
5	Reserved	COPY OPERATION STATUS						
6	(MSB)	OPERATION COUNTER						
7								(LSB)
8	(MSB)							
•••	ESTIMATED STATUS UPDATE DELAY							
11								(LSB)
12	EXTENDED COPY COMPLETION STATUS							
13	LENGTH OF THE SENSE DATA FIELD (m - 31)							
14	SENSE DATA LENGTH							
15	TRANSFER COUNT UNITS (F1h)							
16	(MSB)							
•••	TRANSFER COUNT							
23								(LSB)
24	(MSB)	SEGMENTS PROCESSED (0000h)						
25								(LSB)
26								
•••	Reserved							
31								
32								
•••	SENSE DATA (if any)							
m								
m + 1	(MSB)							
•••	ROD TOKEN DESCRIPTOR LENGTH (n - (m + 4))							
m + 4								(LSB)
m + 5	Restricted (see SPC-6)							
m + 6								
m + 7								
•••	ROD TOKEN (if any)							
n								

The AVAILABLE DATA field, the COPY OPERATION STATUS field, the OPERATION COUNTER field, the ESTIMATED STATUS UPDATE DELAY field, the EXTENDED COPY COMPLETION STATUS field, the LENGTH OF THE SENSE DATA FIELD field, SENSE DATA LENGTH field, the SENSE DATA field, and the ROD TOKEN field are defined in SPC-6.

The RESPONSE TO SERVICE ACTION field is defined in SPC-6 and shall be set to the value shown in table 104 in response to a RECEIVE ROD TOKEN INFORMATION command in which the LIST IDENTIFIER field specifies a POPULATE TOKEN command.

The TRANSFER COUNT UNITS field is defined in SPC-6 and shall be set to the value shown in table 104 in response to a RECEIVE ROD TOKEN INFORMATION command in which the LIST IDENTIFIER field specifies a POPULATE TOKEN command.

The TRANSFER COUNT field indicates the number of contiguous logical blocks represented by the ROD token that were read without error starting at the LBA specified in the first block device range descriptor and including the LBAs described in all complete block device range descriptors of the POPULATE TOKEN command to which this response applies.

If the value in the TRANSFER COUNT field is not equal to the sum of the contents of the NUMBER OF LOGICAL BLOCKS fields in all of the complete block device range descriptors of the POPULATE TOKEN command to which this response applies, then the COPY OPERATION STATUS field shall be set to 3h. Other values in the COPY OPERATION STATUS field are defined in SPC-6.

The SEGMENTS PROCESSED field is defined in SPC-6 and shall be set to the value shown in table 104 in response to a RECEIVE ROD TOKEN INFORMATION command in which the LIST IDENTIFIER field specifies a POPULATE TOKEN command.

The ROD TOKEN DESCRIPTOR LENGTH field is defined in SPC-6 and shall be set to the size of the ROD TOKEN field plus two in response to a RECEIVE ROD TOKEN INFORMATION command in which the LIST IDENTIFIER field specifies a POPULATE TOKEN command.

5.25.3 RECEIVE ROD TOKEN INFORMATION parameter data for WRITE USING TOKEN command

If a RECEIVE ROD TOKEN INFORMATION command (see SPC-6) specifies a list identifier that matches the list identifier specified in a previous WRITE USING TOKEN command (see 5.59) received on the same I_T nexus, then table 105 shows the parameter data returned by the copy manager.

Table 105 — RECEIVE ROD TOKEN INFORMATION parameter data for WRITE USING TOKEN

Byte \ Bit	7	6	5	4	3	2	1	0
0	(MSB)	AVAILABLE DATA (n - 3)						
3								(LSB)
4	Reserved			RESPONSE TO SERVICE ACTION (11h)				
5	Reserved	COPY OPERATION STATUS						
6	(MSB)	OPERATION COUNTER						
7								(LSB)
8	(MSB)	ESTIMATED STATUS UPDATE DELAY						
•••								
11								(LSB)
12	EXTENDED COPY COMPLETION STATUS							
13	LENGTH OF THE SENSE DATA FIELD ((n - 4) - 31)							
14	SENSE DATA LENGTH							
15	TRANSFER COUNT UNITS (F1h)							
16	(MSB)	TRANSFER COUNT						
•••								
23								(LSB)
24	(MSB)	SEGMENTS PROCESSED (0000h)						
25								(LSB)
26	Reserved							
•••								
31								
32	SENSE DATA (if any)							
•••								
n - 4								
n - 3	Restricted (see SPC-6)							
n								

The AVAILABLE DATA field, the COPY OPERATION STATUS field, the OPERATION COUNTER field, the ESTIMATED STATUS UPDATE DELAY field, the EXTENDED COPY COMPLETION STATUS field, the LENGTH OF THE SENSE DATA FIELD field, the SENSE DATA LENGTH field, and the SENSE DATA field are defined in SPC-6.

The RESPONSE TO SERVICE ACTION field is defined in SPC-6 and shall be set to the value shown in table 105 in response to a RECEIVE ROD TOKEN INFORMATION command in which the LIST IDENTIFIER field specifies a WRITE USING TOKEN command.

The TRANSFER COUNT UNITS field is defined in SPC-6 and shall be set to the value shown in table 105 in response to a RECEIVE ROD TOKEN INFORMATION command in which the LIST IDENTIFIER field specifies a WRITE USING TOKEN command.

The TRANSFER COUNT field indicates the number of contiguous logical blocks that were written without error starting with the LBA specified in the first block device range descriptor and including the LBAs specified in all block device range descriptors of the WRITE USING TOKEN command to which this response applies.

If the value in the TRANSFER COUNT field is not equal to the sum of the contents of the NUMBER OF LOGICAL BLOCKS fields in all of the complete block device range descriptors of the WRITE USING TOKEN command to which this response applies, then the COPY OPERATION STATUS field shall be set to 3h. Other values in the COPY OPERATION STATUS field are defined in SPC-6.

The SEGMENTS PROCESSED field is defined in SPC-6 and shall be set to the value shown in table 105 in response to a RECEIVE ROD TOKEN INFORMATION command in which the LIST IDENTIFIER field specifies a POPULATE TOKEN command.

5.26 REMOVE ELEMENT AND TRUNCATE command

The REMOVE ELEMENT AND TRUNCATE command (see table 106) requests that the device server perform a storage element depopulation (see 4.36.4).

If deferred microcode has been saved and not activated (see SPC-6), then the device server shall terminate this command with CHECK CONDITION status with the sense key set to NOT READY and the additional sense code set to LOGICAL UNIT NOT READY, MICROCODE ACTIVATION REQUIRED.

This command uses the SERVICE ACTION IN (16) CDB format (see clause B.2).

Table 106 — REMOVE ELEMENT AND TRUNCATE command

Byte \ Bit	7	6	5	4	3	2	1	0
0	OPERATION CODE (9Eh)							
1	Reserved			SERVICE ACTION (18h)				
2	(MSB)							
•••	REQUESTED CAPACITY							
9								(LSB)
10	(MSB)							
•••	ELEMENT IDENTIFIER							
13								(LSB)
14	Reserved							
15	CONTROL							

The OPERATION CODE field and the SERVICE ACTION field are defined in SPC-6 and shall be set to the values shown in table 106 for the REMOVE ELEMENT AND TRUNCATE command.

The REQUESTED CAPACITY field specifies the capacity in logical blocks (i.e. one greater than the number of logical blocks returned by the READ CAPACITY command) of the media upon completion of the command. A value of zero specifies that the device server shall choose the resultant capacity of the media. If the device server is unable to set the capacity of the medium to the specified value, then the device server shall:

a) not change the capacity of the media; and
b) terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.

The ELEMENT IDENTIFIER field specifies the element identifier associated with the storage element (see 4.36.1) to be depopulated. If the ELEMENT IDENTIFIER field specifies a physical element that is not a storage element, (i.e. the PHYSICAL ELEMENT TYPE field (see table 62) is not set to 01h in the corresponding physical element status descriptor) or specifies a physical element not supported by the device (see 5.8.2.2), then the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.

If the ELEMENT IDENTIFIER field specifies an element identifier for which the device server does not support depopulation, then the device server shall terminate this command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.

If the ELEMENT IDENTIFIER field specifies an element identifier that is associated with a storage element that is depopulated, then the device server shall make no changes and not consider this an error.

The CONTROL byte is defined in SAM-6.

If a REMOVE ELEMENT AND TRUNCATE command requests that the device server make the number of physical elements that are depopulated greater than the non-zero limit on the number of physical elements that are able to be depopulated, then the device server shall terminate that command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST and the additional sense code set to INSUFFICIENT RESOURCES.

Before sending a REMOVE ELEMENT AND TRUNCATE command, an application client may use the parameter data returned by a GET PHYSICAL ELEMENT STATUS command (see 5.8) to discover:

a) the number of physical elements that are depopulated; and
b) the limit on the number of physical elements that are able to be depopulated.

After successful command completion , further processing may occur as described in 4.36.4.

5.27 REPORT PROVISIONING INITIALIZATION PATTERN command

The REPORT PROVISIONING INITIALIZATION PATTERN command (see table 107) requests that the device server transfer the provisioning initialization pattern to the Data-In Buffer.

Table 107 — REPORT PROVISIONING INITIALIZATION PATTERN command

Byte \ Bit	7	6	5	4	3	2	1	0
0	OPERATION CODE (A3h)							
1	Reserved				SERVICE ACTION (1Dh)			
2 ••• 5	Reserved							
6 ••• 9	(MSB) ALLOCATION LENGTH (LSB)							
10	Reserved							
11	CONTROL							

The OPERATION CODE field and SERVICE ACTION field are defined in SPC-6 and shall be set to the value shown in table 107 for the REPORT PROVISIONING INITIALIZATION PATTERN command.

The ALLOCATION LENGTH field is defined in SPC-6.

The CONTROL byte in defined in SAM-6.

5.28 REPORT REFERRALS command

5.28.1 REPORT REFERRALS command overview

The REPORT REFERRALS command (see table 108) requests that the device server transfer parameter data indicating the user data segment(s) on the logical unit and the SCSI target ports through which those user data segments may be accessed (see 4.26) to the Data-In Buffer. This command shall be supported by a logical unit that reports in the Extended INQUIRY Data VPD page (see SPC-6) that it supports referrals (i.e., the R_SUP bit set to one).

This command uses the SERVICE ACTION IN (16) CDB format (see clause B.2).

Table 108 — REPORT REFERRALS command

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

The OPERATION CODE field and SERVICE ACTION field are defined in SPC-6 and shall be set to the values shown in table 108 for the REPORT REFERRALS command.

The LOGICAL BLOCK ADDRESS field specifies an LBA in the first user data segment that the device server shall report in the REPORT REFERRALS parameter data. If the specified LBA exceeds the capacity of the medium (see 4.5), then the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to LOGICAL BLOCK ADDRESS OUT OF RANGE.

The ALLOCATION LENGTH field is defined in SPC-6.

A one segment (ONE_SEG) bit set to zero specifics that the device server shall return information on all user data segments starting with the user data segment that contains the LBA specified in the LOGICAL BLOCK ADDRESS field and ending with the user data segment that contains the last LBA of the logical unit. A ONE_SEG bit set to one specifies the device server shall only return information on the user data segment that contains the LBA specified in the LOGICAL BLOCK ADDRESS field.

The CONTROL byte is defined in SAM-6.

5.28.2 REPORT REFERRALS parameter data

The REPORT REFERRALS parameter data (see table 109) contains information indicating the user data segment(s) on the logical unit and the SCSI target port groups though which those user data segments may be accessed (see 4.26).

Table 109 — REPORT REFERRALS parameter data

Bit Byte	7	6	5	4	3	2	1	0
0	Reserved							
1								
2	(MSB)	USER DATA SEGMENT REFERRAL DESCRIPTOR LENGTH (y - 3)						
3								(LSB)
User data segment referral descriptor list								
4	User data segment referral descriptor [first] (if any)							
•••								
4 + n								
⋮								
y - m	User data segment referral descriptor [last] (if any)							
•••								
y								

The USER DATA SEGMENT REFERRAL DESCRIPTOR LENGTH field indicates the number of bytes that follow in the REPORT REFERRALS parameter data.

The user data segment referral descriptor (see table 18) is defined in the user data segment referral sense data descriptor (see 4.18.4).

5.29 RESTORE ELEMENTS AND REBUILD command

The RESTORE ELEMENTS AND REBUILD command (see table 110) requests that the device server perform a storage element restoration (see 4.36.5).

If deferred microcode has been saved and not activated (see SPC-6), then the device server shall terminate this command with CHECK CONDITION status with the sense key set to NOT READY and the additional sense code set to LOGICAL UNIT NOT READY, MICROCODE ACTIVATION REQUIRED.

Table 110 — RESTORE ELEMENTS AND REBUILD command

Byte \ Bit	7	6	5	4	3	2	1	0
0	OPERATION CODE (9Eh)							
1	Reserved			SERVICE ACTION (19h)				
2	Reserved							
•••								
14								
15	CONTROL							

The OPERATION CODE field and the SERVICE ACTION field are defined in SPC-6 and shall be set to the values shown in table 110 for the RESTORE ELEMENTS AND REBUILD command.

For a RESTORE ELEMENTS AND REBUILD command, 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 COMMAND SEQUENCE ERROR if:

a) there is at least one depopulated storage element (e.g., the PHYSICAL ELEMENT HEALTH field is set to FFh); and
b) all depopulated storage elements have the Restoration Allowed attribute (see 4.36.2) set to false.

For a RESTORE ELEMENTS AND REBUILD command, the following shall not be considered an error:

a) the device has no depopulated storage elements; or
b) at least one depopulated storage element has the Restoration Allowed attribute (see 4.36.2) set to true.

The CONTROL byte is defined in SAM-6.

After command completion without error, further processing may occur as described in 4.36.5.

5.30 SANITIZE command

5.30.1 SANITIZE command overview

The SANITIZE command (see table 111) requests that the device server perform a sanitize operation (see 4.11). This device server shall process this command as if it has a HEAD OF QUEUE task attribute (see 4.16).

Table 111 — SANITIZE command

Byte \ Bit	7	6	5	4	3	2	1	0
0	OPERATION CODE (48h)							
1	IMMED	ZNR	AUSE	SERVICE ACTION				
2 ••• 6	Reserved							
7	(MSB)	PARAMETER LIST LENGTH						
8								LSB)
9	CONTROL							

The OPERATION CODE field is defined in SPC-6 and shall be set to the value shown in table 111 for the SANITIZE command.

If the immediate (IMMED) bit is set to zero, then the device server shall return status after the sanitize operation is completed. If the IMMED bit set to one, then the device server shall return status as soon as the CDB and parameter data, if any, have been validated. The REQUEST SENSE command may be used to poll for progress of the sanitize operation regardless of the value of the IMMED bit. Use of the IMMED bit depends on the service action as described in table 112.

For a zoned block device, the zoned no reset (ZNR) bit requests the device server to perform specific actions upon successful completion of a sanitize operation (see 4.11). Conditions described in ZBC-2 (e.g., the Zone Condition becoming OFFLINE for a write pointer zone) may result in the device server not performing the actions requested by the ZNR bit. Unless otherwise defined, if the ZNR bit is set to:

a) zero, then a successful sanitize operation shall perform the equivalent of a RESET WRITE POINTER command (see ZBC-2) with the ALL bit set to one; and
b) one, then a successful sanitize operation shall:
    A) not modify the write pointer (see ZBC-2) for any write pointer zone that is sanitized by a sanitize cryptographic erase operation; and
    B) perform the equivalent of a FINISH ZONE command (see ZBC-2) with the ALL bit set to one for any write pointer zone that is sanitized by a sanitize overwrite operation or a sanitize block erase operation.

For a logical unit that is not a zoned block device the ZNR bit shall be ignored.

If the allow unrestricted sanitize exit (AUSE) bit is set to one, and the specified sanitize operation fails, then a subsequent EXIT FAILURE MODE service action requests that the device server exit the failed sanitize condition (see 4.11.1)..

If:

1) the AUSE bit is set to zero in the SANITIZE command that requested a sanitize operation;
2) that sanitize operation completes with an error (see 4.11.4); and
3) a subsequent SANITIZE command is received with:

A) the EXIT FAILURE MODE service action; or
B) any service action with the AUSE bit set to one,

then the device server shall terminate that subsequent SANITIZE command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.

The SERVICE ACTION field is defined in 5.30.2.

The PARAMETER LIST LENGTH field specifies the length in bytes of the parameter data that is available to be transferred from the Data-Out Buffer. A PARAMETER LIST LENGTH field set to zero specifies that no data shall be transferred.

The CONTROL byte is defined in SAM-6.

5.30.2 SANITIZE command service actions

5.30.2.1 SANITIZE command service actions overview

The SANITIZE command service actions are shown in table 112. At least one service action shall be supported if the SANITIZE command is supported. If the service action specified in the CDB is not supported, then the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.

If deferred microcode has been saved and not activated (see SPC-6), then the device server shall terminate this command with CHECK CONDITION status with the sense key set to NOT READY and the additional sense code set to LOGICAL UNIT NOT READY, MICROCODE ACTIVATION REQUIRED.

Table 112 — SANITIZE service action codes

Code	Name	Description	PARAMETER LIST LENGTH requirement [a]	IMMED bit usage	Reference
01h	OVERWRITE	Perform a sanitize overwrite operation	Set to > 0004h and < (logical block length + 5)	Yes [b]	5.30.2.2
02h	BLOCK ERASE	Perform a sanitize block erase operation	Set to 0000h	Yes [b]	5.30.2.3
03h	CRYPTOGRAPHIC ERASE	Perform a sanitize cryptographic erase operation	Set to 0000h	Yes [b]	5.30.2.4
1Fh	EXIT FAILURE MODE	Exit a failed sanitize condition (see 4.11.1)	Set to 0000h	No [c]	5.30.2.5
all others	Reserved				

[a] If the requirement is not met, then the SANITIZE command is terminated with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.
[b] The IMMED bit is used as defined in this subclause (i.e., 5.30.1) by this service action.
[c] The contents of IMMED bit should be ignored for this service action.

5.30.2.2 OVERWRITE service action

The OVERWRITE service action (see table 112) requests that the device server perform a sanitize overwrite operation (see 4.11).

While performing a sanitize overwrite operation, the device server shall remove all Background Scan Results log parameters (see 6.4.2.3) from the Background Scan Results log page, if supported, and remove all Pending Defect log parameters (see 6.4.8.3) from the Pending Defects log page, if supported.

The parameter list format for the OVERWRITE service action is shown in table 113.

Table 113 — OVERWRITE service action parameter list

Byte \ Bit	7	6	5	4	3	2	1	0
0	INVERT	TEST		OVERWRITE COUNT				
1	Reserved							
2	(MSB)	INITIALIZATION PATTERN LENGTH (n - 3)						
3								LSB)
4	INITIALIZATION PATTERN							
•••								
n								

If the INVERT bit is set to zero, then on each overwrite pass:

a) the user data shall be written as specified in the INITIALIZATION PATTERN field; and
b) the protection information, if any, shall be set to FFFF_FFFF_FFFF_FFFFh.

If the INVERT bit is set to one, then the user data and protection information bytes, if any, shall be inverted (i.e., each bit XORed with one) between consecutive overwrite passes.

The TEST field is shown in table 114.

Table 114 — TEST field

Code	Description
00b	Shall not cause any changes in the defined behavior of the SANITIZE command.
01b to 11b	Vendor specific [a]
[a] Setting the TEST field to one of these values may adversely affect security properties of the OVERWRITE service action.	

The OVERWRITE COUNT field specifies the number of overwrite passes to be performed. The value of 00h is reserved.

The INITIALIZATION PATTERN LENGTH field specifies the length in bytes of the INITIALIZATION PATTERN field. If the INITIALIZATION PATTERN LENGTH field is set to zero or a value greater than the logical block length, then the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

The INITIALIZATION PATTERN field specifies the data pattern to be used to write the user data. This data pattern is repeated as necessary to fill each logical block. For each logical block, the first byte of the user data shall begin with the first byte of the initialization pattern.

If the INVERT bit is set to one and:

a) the OVERWRITE COUNT field is set to an even number, then the pattern used for the first write pass shall consist of:
   A) the user data set to the inversion of the INITIALIZATION PATTERN data; and
   B) the protection information, if any, set to 0000_0000_0000_0000h;

   or

b) the OVERWRITE COUNT field is set to an odd number, then the pattern used for the first write pass shall consist of:
   A) the user data set to the INITIALIZATION PATTERN data; and
   B) the protection information, if any, set to FFFF_FFFF_FFFF_FFFFh.

After a sanitize overwrite operation completes without error:

a) the device server completes read commands for which no other error occurs during processing with GOOD status and read medium operations return the data written by the sanitize overwrite operation; and
b) protection information, if any, shall be set to FFFF_FFFF_FFFF_FFFFh in all logical blocks on the medium.

5.30.2.3 BLOCK ERASE service action

The BLOCK ERASE service action (see table 112) requests that the device server perform a sanitize block erase operation (see 4.11).

After a sanitize block erase operation completes without error:

a) the device server may terminate commands that request read operations specifying mapped LBAs (see 4.7.1) based on the setting of the WABEREQ field (see 6.6.2); and
b) if the logical unit is formatted with protection information, then:
   A) the protection information for each mapped LBA may be indeterminate; and
   B) if the device server terminates a command that requests read operations specifying mapped LBAs as a result of a protection information error, then the device server shall terminate that command with CHECK CONDITION status with the sense key set to ABORTED COMMAND and the appropriate additional sense code for the condition (e.g., for READ commands, the additional sense code shown in table 81).

5.30.2.4 CRYPTOGRAPHIC ERASE service action

The CRYPTOGRAPHIC ERASE service action (see table 112) requests that the device server perform a sanitize cryptographic erase operation (see 4.11).

After a sanitize cryptographic erase operation completes without error:

a) the device server may terminate commands that request read operations specifying mapped LBAs (see 4.7.1) based on the setting of the WACEREQ field (see 6.6.2); and
b) if the logical unit is formatted with protection information, then:
   A) the protection information for each mapped LBA may be indeterminate; and
   B) if the device server terminates a command that requests read operations specifying mapped LBAs as a result of a protection information error, then the device server shall terminate that command with CHECK CONDITION status with the sense key set to ABORTED COMMAND and the appropriate additional sense code for the condition (e.g., for READ commands, the additional sense code shown in table 81).

5.30.2.5 EXIT FAILURE MODE service action

If the unrestricted completion mode (see 4.11.1) is in effect, then a SANITIZE command with the EXIT FAILURE

MODE service action requests that the device server exit the failed sanitize condition (see 4.11.1).

If the most recent sanitize operation, if any, has completed without error, then the EXIT FAILURE MODE service action completes without error.

After successful completion of a SANITIZE command with the EXIT FAILURE MODE service action:

a) if any LBA is mapped (see 4.7.1), and the logical unit is formatted with protection information, then:
   A) the protection information for each mapped LBA may be indeterminate; and
   B) if the device server terminates a command that requests read operations specifying mapped LBAs as a result of a protection information error, then the device server shall terminate that command with CHECK CONDITION status with the sense key set to ABORTED COMMAND and the appropriate additional sense code for the condition (e.g., for READ commands, the additional sense code shown in table 81);

   and

b) the device server should complete reads to unmapped LBAs without error (see 4.7.4.6.1 and 4.7.4.7.1).

5.31 START STOP UNIT command

The START STOP UNIT command (see table 115) requests that the device server change the power condition of the logical unit (see 4.20) or load or eject the medium. This includes specifying that the device server enable or disable the direct access block device for medium access operations by controlling power conditions and timers.

The device server shall handle any deferred microcode as specified in 4.24.

Table 115 — START STOP UNIT command

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

The OPERATION CODE field is defined in SPC-6 and shall be set to the value shown in table 115 for the START STOP UNIT command.

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

The combinations of values in the POWER CONDITION field and the POWER CONDITION MODIFIER field are shown in table 116. If the POWER CONDITION field is supported and is set to a value other than 0h, then the device server shall ignore the START bit and the LOEJ bit.

Table 116 — POWER CONDITION and POWER CONDITION MODIFIER field (part 1 of 2)

POWER CONDITION value	POWER CONDITION name	POWER CONDITION MODIFIER value	Device server action
0h	START_ VALID	0h	Process the START bit and the LOEJ bit.
1h	ACTIVE	0h	Cause the logical unit to transition to the active power condition. [a]
2h	IDLE	0h	Cause the logical unit to transition to the idle_a power condition. [a,b]
		1h	Cause the logical unit to transition to the idle_b power condition. [a,b,c]
		2h	Cause the logical unit to transition to the idle_c power condition. [a,b,d]
3h	STANDBY	0h	Cause the logical unit to transition to the standby_z power condition. [a,b]
		1h	Cause the logical unit to transition to the standby_y power condition. [a,b]
5h	Obsolete	0h to Fh	Obsolete
7h	LU_ CONTROL	0h	Initialize and start all of the idle condition timers that are enabled (see SPC-6), and initialize and start all of the standby condition timers that are enabled (see SPC-6).

[a] Process the following actions:
1) the device server shall comply with any SCSI transport protocol specific power condition transition restrictions (e.g., the NOTIFY (ENABLE SPINUP) requirement (see SPL-5));
2) the logical unit shall transition to the specified power condition; and
3) the device server shall halt all of the idle condition timers that are enabled (see SPC-6) and halt all of the standby condition timers that are enabled (see SPC-6) until:
   A) a START STOP UNIT command is processed that returns control of the power condition to the logical unit as described in 4.20.1;or
   B) a logical unit reset occurs.

[b] If a timer for a background scan operation expires, or a device specific event occurs, then the logical unit shall not leave this power condition to perform the background function associated with the timer or event.

[c] The device server shall cause the direct access block device to increase its tolerance of external physical forces (e.g., causes a device that has movable read/write heads to move those heads to a safe position).

[d] The device server shall cause the direct access block device to increase its tolerance of external physical forces and reduce its power consumption to use less power than when the logical unit is in the idle_b power condition (e.g., cause a device that has rotating medium to rotate the medium at a lower revolutions per minute).

[e] If the specified timer is supported and enabled, then the device server shall:
1) cause the specified timer to expire (see SPC-6), which may cause the logical unit to transition to the specified power condition;
2) initialize and start all of the idle condition timers that are enabled (see SPC-6); and
3) initialize and start all of the standby condition timers that are enabled (see SPC-6),

otherwise the device server shall terminate the START STOP UNIT command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.

Table 116 — POWER CONDITION and POWER CONDITION MODIFIER field (part 2 of 2)

POWER CONDITION value	POWER CONDITION name	POWER CONDITION MODIFIER value	Device server action
Ah	FORCE_ IDLE_0	0h	Cause the idle_a condition timer to expire. [e]
		1h	Cause the idle_b condition timer to expire. [e]
		2h	Cause the idle_c condition timer to expire. [e]
Bh	FORCE_ STANDBY_0	0h	Cause the standby_z condition timer to expire. [e]
		1h	Cause the standby_y condition timer to expire. [e]
All other combinations			Reserved

[a] Process the following actions:
1) the device server shall comply with any SCSI transport protocol specific power condition transition restrictions (e.g., the NOTIFY (ENABLE SPINUP) requirement (see SPL-5));
2) the logical unit shall transition to the specified power condition; and
3) the device server shall halt all of the idle condition timers that are enabled (see SPC-6) and halt all of the standby condition timers that are enabled (see SPC-6) until:
   A) a START STOP UNIT command is processed that returns control of the power condition to the logical unit as described in 4.20.1;or
   B) a logical unit reset occurs.

[b] If a timer for a background scan operation expires, or a device specific event occurs, then the logical unit shall not leave this power condition to perform the background function associated with the timer or event.

[c] The device server shall cause the direct access block device to increase its tolerance of external physical forces (e.g., causes a device that has movable read/write heads to move those heads to a safe position).

[d] The device server shall cause the direct access block device to increase its tolerance of external physical forces and reduce its power consumption to use less power than when the logical unit is in the idle_b power condition (e.g., cause a device that has rotating medium to rotate the medium at a lower revolutions per minute).

[e] If the specified timer is supported and enabled, then the device server shall:
1) cause the specified timer to expire (see SPC-6), which may cause the logical unit to transition to the specified power condition;
2) initialize and start all of the idle condition timers that are enabled (see SPC-6); and
3) initialize and start all of the standby condition timers that are enabled (see SPC-6),

otherwise the device server shall terminate the START STOP UNIT command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.

If the START STOP UNIT command specifies a power condition that conflicts with an operation in progress then, after the START STOP UNIT command completes with GOOD status, the logical unit may not be in the power condition that was requested by the command.

It is not an error to specify that the logical unit transition to its current power condition.

If no START STOP UNIT command is being processed by the device server, then the device server shall process any received START STOP UNIT command.

If a START STOP UNIT command is being processed by the device server and a subsequent START STOP UNIT command for which the CDB is validated requests that the logical unit change to the same power condition that was specified by the START STOP UNIT command being processed, then the device server shall process the subsequent command.

If the NO_FLUSH bit is set to zero, then the device server shall write all logical block data in cache that is newer than logical block data on the medium to the medium (e.g., as if in response to a SYNCHRONIZE CACHE command (see 5.33 and 5.34) with the LOGICAL BLOCK ADDRESS field set to zero and the NUMBER OF LOGICAL BLOCKS field set to zero) prior to entering into any power condition that prevents accessing the medium (e.g., before the rotating medium spindle motor is stopped during transition to the stopped power condition). If the NO_FLUSH bit is set to one, then the device server should not write any cached logical blocks to the medium prior to entering into any power condition that prevents accessing the medium.

If the load eject (LOEJ) bit is set to zero and the POWER CONDITION field 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 and the POWER CONDITION field is set to zero, then the logical unit shall unload the medium if the START bit is set to zero. If the LOEJ bit is set to one, the POWER CONDITION field is set to zero, and the START bit is set to one, then the logical unit shall load the medium.

If the START bit is set to zero and the POWER CONDITION field is set to zero, then the device server shall:

a) cause the logical unit to transition to the stopped power condition;
b) halt any idle condition timer that is enabled (see SPC-6); and
c) halt any standby condition timer that is enabled (see SPC-6).

If the START bit set to one and the POWER CONDITION field is set to zero, then the device server shall:

1) comply with requirements defined in SCSI transport protocol standards (e.g., the NOTIFY (ENABLE SPINUP) requirement (see SPL-5));
2) cause the logical unit to transition to the active power condition;
3) initialize and start any idle condition timer that is enabled; and
4) initialize and start any standby condition timer that is enabled.

The CONTROL byte is defined in SAM-6.

5.32 STREAM CONTROL command

5.32.1 STREAM CONTROL command overview

The STREAM CONTROL command (see table 117) requests the device server to open a stream and return the stream identifier in the return parameter data or close the stream specified in the STR_ID field in the CDB.

This command uses the SERVICE ACTION IN (16) CDB format (see clause B.2).

Table 117 — STREAM CONTROL command

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

The OPERATION CODE field and SERVICE ACTION field are defined in SPC-6 and shall be set to the values shown in table 117 for the STREAM CONTROL command.

The stream control (STR_CTL) field specifies the operation to be performed as described in table 118.

Table 118 — STR_CTL field

Code	Description
01b	Open a stream and return the stream identifier in the ASSIGNED_STR_ID field in the returned parameter data.
10b	Close the stream associated with the STR_ID field.
all others	Reserved

If the STR_CTL field is set to 10b, then the stream identifier (STR_ID) field specifies the stream identifier associated with the requested operation. If the STR_CTL field is not set to 10b, then the device server shall ignore the STR_ID field.

The ALLOCATION LENGTH field is defined in SPC-6.

The CONTROL byte is defined in SAM-6.

~~A STREAM CONTROL command that opens a stream shall result in the relative lifetime (see 5.9.2.2) being set to zero.~~

~~If a STREAM CONTROL command specifies that a permanent stream (see 4.32.4) be closed, then the device server shall terminate that command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.~~

5.32.2 STREAM CONTROL parameter data

The STREAM CONTROL parameter data is shown in table 119.

Table 119 — STREAM CONTROL parameter data

Byte \ Bit	7	6	5	4	3	2	1	0
0	PARAMETER LENGTH (07h)							
1	Reserved							
•••								
3								
4	(MSB)	ASSIGNED_STR_ID						
5								(LSB)
6	Reserved							
7								

The PARAMETER LENGTH field indicates the length of the parameter data and shall be set as shown in table 119 for the STREAM CONTROL parameter data.

If the STR_CTL field was set to 01b (i.e., open) in the STREAM CONTROL command, then the device server shall set the ASSIGNED_STR_ID field to a non zero value that is not currently assigned to an open stream by the device server and open that stream. If the STR_CTL field was not set to 01b in the STREAM CONTROL command, then the ASSIGNED_STR_ID field is reserved.

5.33 SYNCHRONIZE CACHE (10) command

The SYNCHRONIZE CACHE (10) command (see table 120) requests that, for each logical block whose logical block data is in the volatile cache and has not already been written to the non-volatile cache, if any, or the medium, the device server either:

a) perform a write medium operation to the LBA using the logical block data in volatile cache; or
b) write the logical block to the non-volatile cache, if any.

NOTE 13 - Migration from the SYNCHRONIZE CACHE (10) command to the SYNCHRONIZE CACHE (16) command is recommended for all implementations.

Table 120 — SYNCHRONIZE CACHE (10) command

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

The OPERATION CODE field is defined in SPC-6 and shall be set to the value shown in table 120 for the SYNCHRONIZE CACHE (10) command.

See the PRE-FETCH (10) command (see 5.13) for the definition of the LOGICAL BLOCK ADDRESS field.

See the PRE-FETCH (10) command (see 5.13) and 4.22 for the definition of the GROUP NUMBER field.

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

If the IMMED bit is set to one and the synchronize cache operation has not completed, then the SYNC_PROG field (see 6.5.6) defines the device server behavior while the synchronize cache command is being processed.

A NUMBER OF LOGICAL BLOCKS field set to a non-zero value specifies the number of logical blocks that shall be synchronized, starting with the logical block referenced by the LBA specified by the LOGICAL BLOCK ADDRESS field. A NUMBER OF LOGICAL BLOCKS field set to zero specifies that all logical blocks starting with the one referenced by the LBA specified in the LOGICAL BLOCK ADDRESS field to the last logical block on the medium shall be synchronized. If the specified LBA and the specified number of logical blocks exceed the capacity of the medium (see 4.5), then the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to LOGICAL BLOCK ADDRESS OUT OF RANGE.

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

The CONTROL byte is defined in SAM-6.

5.34 SYNCHRONIZE CACHE (16) command

The SYNCHRONIZE CACHE (16) command (see table 121) requests that the device server perform the actions defined for the SYNCHRONIZE CACHE (10) command (see 5.33).

Table 121 — SYNCHRONIZE CACHE (16) command

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

The OPERATION CODE field is defined in SPC-6 and shall be set to the value shown in table 121 for the SYNCHRONIZE CACHE (16) command.

See the SYNCHRONIZE CACHE (10) command (see 5.33) for the definitions of the other fields in this command.

5.35 UNMAP command

5.35.1 UNMAP command overview

The UNMAP command (see table 122) requests that the device server cause one or more LBAs to be unmapped (see 4.7.3).

Table 122 — UNMAP command

Byte \ Bit	7	6	5	4	3	2	1	0
0	OPERATION CODE (42h)							
1	Reserved							ANCHOR
2 ••• 5	Reserved							
6	Reserved		GROUP NUMBER					
7	(MSB) PARAMETER LIST LENGTH							
8	PARAMETER LIST LENGTH (LSB)							
9	CONTROL							

The OPERATION CODE field is defined in SPC-6 and shall be set to the value shown in table 122 for the UNMAP command.

For a thin provisioned logical unit (see 4.7.3.3) with the ANC_SUP bit set to one in the Logical Block Provisioning VPD page (see 6.6.9):

a) if the ANCHOR bit is set to zero, then any LBA on which an unmap operation (see 4.7.3.4) is performed shall either become deallocated (see 4.7.4.6) or anchored (see 4.7.4.7) and should become deallocated; and
b) if the ANCHOR bit is set to one, then any LBA on which an unmap operation is performed shall become anchored.

For a thin provisioned logical unit (see 4.7.3.3) with the ANC_SUP bit set to zero in the Logical Block Provisioning VPD page:

a) if the ANCHOR bit is set to zero, then any LBA on which an unmap operation is performed shall become deallocated; and
b) if the ANCHOR bit is set to one, 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.

For a resource provisioned logical unit (see 4.7.3.2), the ANCHOR bit shall be ignored and any LBA on which an unmap operation is performed shall become anchored (i.e., the command is processed as if the ANCHOR bit is set to one).

For a thin provisioned logical unit or a resource provisioned logical unit any LBA on which an unmap operation is not performed does not change logical block provisioning state.

See the PRE-FETCH (10) command (see 5.13) and 4.22 for the definition of the GROUP NUMBER field.

The PARAMETER LIST LENGTH field specifies the length in bytes of the UNMAP parameter list that is available to be transferred from the Data-Out Buffer. If the parameter list length is greater than zero and less than eight, 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 PARAMETER LIST LENGTH ERROR. A PARAMETER LIST LENGTH set to zero specifies that no data shall be transferred.

The CONTROL byte is defined in SAM-6.

5.35.2 UNMAP parameter list

The UNMAP parameter list (see table 123) contains an UNMAP parameter list header and block descriptors. Each UNMAP block descriptor specifies a range of LBAs for which each LBA is processed as specified by the ANCHOR bit in the CDB (see 5.35.1).The LBAs specified in the block descriptors may contain overlapping LBA extents, and may be in any order.

Table 123 — UNMAP parameter list

Byte \ Bit	7	6	5	4	3	2	1	0
0	(MSB)	UNMAP DATA LENGTH (n - 1)						
1								(LSB)
2	(MSB)	UNMAP BLOCK DESCRIPTOR DATA LENGTH (n - 7)						
3								(LSB)
4 ••• 7	Reserved							
UNMAP block descriptor list (if any)								
8 ••• 23	UNMAP block descriptor [first] (see table 124) (if any)							
⋮								
n - 15 ••• n	UNMAP block descriptor [last] (see table 124) (if any)							

The UNMAP DATA LENGTH field specifies the length in bytes of the following data that is available to be transferred from the Data-Out Buffer. The unmap data length does not include the number of bytes in the UNMAP DATA LENGTH field.

The UNMAP BLOCK DESCRIPTOR DATA LENGTH field specifies the length in bytes of the UNMAP block descriptors that are available to be transferred from the Data-Out Buffer. The unmap block descriptor data length should be a multiple of 16. If the unmap block descriptor data length is not a multiple of 16, then the last unmap block descriptor is incomplete and shall be ignored. If the UNMAP BLOCK DESCRIPTOR DATA LENGTH field is set to zero, then no unmap block descriptors are included in the UNMAP parameter list. This condition shall not be considered an error.

If any UNMAP block descriptors in the UNMAP block descriptor list are truncated as a result of the parameter list length in the CDB, then that UNMAP block descriptor shall be ignored.

Table 124 defines the UNMAP block descriptor.

Table 124 — UNMAP block descriptor

Byte \ Bit	7	6	5	4	3	2	1	0
0	(MSB)							
•••	UNMAP LOGICAL BLOCK ADDRESS							
7								(LSB)
8	(MSB)							
•••	NUMBER OF LOGICAL BLOCKS							
11								(LSB)
12								
•••	Reserved							
15								

The UNMAP LOGICAL BLOCK ADDRESS field specifies the first LBA that is requested to be unmapped (see 4.7.3.4.2) for this UNMAP block descriptor.

The NUMBER OF LOGICAL BLOCKS field specifies the number of LBAs that are requested to be unmapped beginning with the LBA specified by the UNMAP LOGICAL BLOCK ADDRESS field.

If the NUMBER OF LOGICAL BLOCKS is set to zero, then no LBAs shall be unmapped for this UNMAP block descriptor. This condition shall not be considered an error.

If the specified LBA and the specified number of logical blocks exceeds the capacity of the medium (see 4.5), then the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to LOGICAL BLOCK ADDRESS OUT OF RANGE.

If the total number of logical blocks specified in the UNMAP block descriptor data exceeds the value indicated in the MAXIMUM UNMAP LBA COUNT field (see 6.6.4) or if the number of UNMAP block descriptors exceeds the value of the MAXIMUM UNMAP BLOCK DESCRIPTOR COUNT field (see 6.6.4), then the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

5.36 VERIFY (10) command

The VERIFY (10) command (see table 126) requests that the device server:

1) perform verify operations from the specified LBAs; and
2) if specified, perform a compare operation on:
   A) the logical block data transferred from the Data-Out Buffer; and
   B) the logical block data from the verify operations.

The device server may process the LBAs in any order but shall perform this sequence in the specified order for a given LBA.

The application client uses the BYTCHK field in the CDB to specify the contents of the Data-Out Buffer as shown in table 125.

Table 125 — Data-Out Buffer contents for the VERIFY (10) command

BYTCHK field	Data-Out Buffer contents
00b	Not used
01b	Logical block data for the number of logical blocks specified in the VERIFICATION LENGTH field
10b [a]	Not defined
11b	Logical block data for a single logical block
[a] A BYTCHK field set to 10b is reserved.	

NOTE 14 - Migration from the VERIFY (10) command to the VERIFY 16) command is recommended for all implementations.

Table 126 — 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)	LOGICAL BLOCK ADDRESS						
•••								
5								(LSB)
6	Restricted for MMC-6	Reserved	GROUP NUMBER					
7	(MSB)	VERIFICATION LENGTH						
8								(LSB)
9	CONTROL							

The OPERATION CODE field is defined in SPC-6 and shall be set to the value shown in table 126 for the VERIFY (10) command.

See the READ (10) command (see 5.16) for the definition of the DPO bit. See the PRE-FETCH (10) command (see 5.13) for the definition of the LOGICAL BLOCK ADDRESS field. See the PRE-FETCH (10) command (see 5.13) and 4.22 for the definition of the GROUP NUMBER field.

If the byte check (BYTCHK) field is set to 00b, then:

a) no Data-Out Buffer transfer shall occur;
b) for any mapped LBA specified by the command, the device server shall check the protection information from the verify operation based on the VRPROTECT field as shown in table 127; and
c) for any unmapped LBA specified by the command, the verify operation shall complete as if there is no verification error.

If:

a) the BYTCHK field is set to 01b or 11b;
b) the VBULS bit is set to zero in the Block Device Characteristics VPD page (see 6.6.2); and
c) any LBA specified by the command is unmapped (i.e., deallocated or anchored),

then the device server shall terminate the command with CHECK CONDITION status with the sense key set to MISCOMPARE and the additional sense code set to MISCOMPARE VERIFY OF UNMAPPED LBA.

If:

a) the BYTCHK field is set to 01b or 11b; and
b) either:
   A) the VBULS bit is set to one in the Block Device Characteristics VPD page; or
   B) all LBAs specified by the command are mapped,

then:

a) if the BYTCHK field is set to 01b, then the Data-Out Buffer transfer shall include the number of logical blocks specified by the VERIFICATION LENGTH field;
b) if the BYTCHK field is set to 11b, then:
   A) the Data-Out Buffer transfer shall include one logical block; and
   B) the device server shall:
      1) duplicate the single logical block, as described in the WRITE SAME command (see 5.52), the number of times required to satisfy the VERIFICATION LENGTH field; and
      2) place the duplicated data in the Data-Out Buffer;
c) the device server shall check the protection information transferred from the Data-Out Buffer based on the VRPROTECT field as shown in table 129;
d) for any mapped LBA specified by the command, the device server shall perform the verify operation and check the protection information from the verify operation based on the VRPROTECT field as shown in table 128;

   and

e) the device server shall perform:
   A) a compare operation of:
      a) user data from the verify operations; and
      b) user data from the Data-Out Buffer;

      and

   B) a compare operation based on the VRPROTECT field as shown in table 130 of:
      a) protection information from the verify operations; and
      b) protection information from the Data-Out Buffer.

The order of the user data and protection information checks and compare operations is vendor specific.

If a compare operation indicates a miscompare, then the device server shall terminate the command with CHECK CONDITION status with the sense key set to MISCOMPARE and the additional sense code set to the appropriate value for the condition.

The VERIFICATION LENGTH field specifies the number of contiguous logical blocks that shall be verified, starting with the logical block referenced by the LBA specified by the LOGICAL BLOCK ADDRESS field. A VERIFICATION LENGTH field set to zero specifies that no logical blocks shall be transferred or verified. This condition shall not be considered an error. If the specified LBA and the specified verification length exceed the capacity of the medium (see 4.5), then the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to LOGICAL BLOCK ADDRESS OUT OF RANGE. The VERIFICATION LENGTH field is constrained by the MAXIMUM TRANSFER LENGTH field (see 6.6.4).

If the BYTCHK field is set to 01b, then the VERIFICATION LENGTH field also specifies the number of logical blocks that the device server shall transfer from the Data-Out Buffer.

The CONTROL byte is defined in SAM-6.

If the BYTCHK field is set to 00b, then table 127 defines the checks that the device server shall perform on the protection information from the verify operations based on the VRPROTECT field. All footnotes for table 127 are at the end of the table.

Table 127 — VRPROTECT field with the BYTCHK field set to 00b – checking protection information from the verify operations (part 1 of 3)

Code	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 [i]	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.		
001b 101b [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 [h]	LOGICAL BLOCK REFERENCE TAG CHECK FAILED
			REF_CHK = 0	No check performed
	No	Error condition [a]		
010b [b]	Yes	LOGICAL BLOCK GUARD	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	Error condition [a]		

Table 127 — VRPROTECT field with the BYTCHK field set to 00b – checking protection information from the verify operations (part 2 of 3)

Code	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	No check performed	
		LOGICAL BLOCK APPLICATION TAG	No check performed	
		LOGICAL BLOCK REFERENCE TAG	No check performed	
	No	Error condition [a]		
100b [b]	Yes	LOGICAL BLOCK GUARD	GRD_CHK = 1	LOGICAL BLOCK GUARD CHECK FAILED
			GRD_CHK = 0	No check performed
		LOGICAL BLOCK APPLICATION TAG	No check performed	
		LOGICAL BLOCK REFERENCE TAG	No check performed	
	No	Error condition [a]		
110b to 111b	Reserved			

Table 127 — VRPROTECT field with the BYTCHK field set to 00b – checking protection information from the verify operations (part 3 of 3)

Code	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

[a] If the logical unit supports protection information (see 4.21) and has not been formatted with protection information, then the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.

[b] If the logical unit does not support protection information, then the device server should 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.

[c] If the device server has knowledge of the contents of the LOGICAL BLOCK APPLICATION TAG field, then the device server shall check each logical block application tag. If the ATO bit in the Control mode page (see SPC-6) is set to one, then this knowledge is acquired from:

a) the EXPECTED LOGICAL BLOCK APPLICATION TAG field and the LOGICAL BLOCK APPLICATION TAG MASK field in the CDB, if the command is a VERIFY (32) command (see 5.39);
b) the Application Tag mode page (see 6.5.3), if the command is not a VERIFY (32) command and the ATMPE bit in the Control mode page (see SPC-6) is set to one; or
c) a method not defined by this standard, if the command is not a VERIFY (32) command and the ATMPE bit is set to zero.

[d] If the device server terminates the command with CHECK CONDITION status, then the device server shall set the sense key to ABORTED COMMAND.

[e] If multiple errors occur while the device server is processing the command, then the selection by the device server of which error to report is not defined by this standard.

[f] See the Extended INQUIRY Data VPD page (see SPC-6) for the definitions of the GRD_CHK bit, the APP_CHK bit, and the REF_CHK bit.

[g] If the device server detects:

a) a LOGICAL BLOCK APPLICATION TAG field set to FFFFh and type 1 protection (see 4.21.2.3) or type 2 protection (see 4.21.2.4) is enabled; or
b) a LOGICAL BLOCK APPLICATION TAG field set to FFFFh, LOGICAL BLOCK REFERENCE TAG field set to FFFF_FFFFh, and type 3 protection (see 4.21.2.5) is enabled,

then the device server shall not check any protection information in the associated protection information interval.

[h] If type 1 protection is enabled, then the device server shall check each logical block reference tag by comparing it to the lower four bytes of the LBA associated with the logical block. If type 2 protection or type 3 protection is enabled, and the device server has knowledge of the contents of the LOGICAL BLOCK REFERENCE TAG field, then the device server shall check the logical block reference tag. If type 2 protection is enabled, then this knowledge may be acquired through the EXPECTED INITIAL LOGICAL BLOCK REFERENCE TAG field in a VERIFY (32) command (see 5.39). If type 3 protection is enabled, then the method for acquiring this knowledge is not defined by this standard.

[i] If the DPICZ bit in the Control mode page (see SPC-6) is set to one, then protection information shall not be checked.

If the BYTCHK field is set to 01b or 11b, then table 128 defines the checks that the device server shall perform on the protection information from the verify operations based on the VRPROTECT field. All footnotes for table 128 are at the end of the table.

Table 128 — VRPROTECT field with the BYTCHK field set to 01b or 11b – checking protection information from the verify operations (part 1 of 2)

<table>
<tr><th>Code</th><th>Logical unit formatted with protection information</th><th>Field in protection information [g]</th><th>Extended INQUIRY Data VPD page bit value [f]</th><th>If check fails [d] [e], additional sense code</th></tr>
<tr><td rowspan="7">000b</td><td rowspan="6">Yes [i]</td><td rowspan="2">LOGICAL BLOCK GUARD</td><td>GRD_CHK = 1</td><td>LOGICAL BLOCK GUARD CHECK FAILED</td></tr>
<tr><td>GRD_CHK = 0</td><td>No check performed</td></tr>
<tr><td rowspan="2">LOGICAL BLOCK APPLICATION TAG</td><td>APP_CHK = 1 [c] [g]</td><td>LOGICAL BLOCK APPLICATION TAG CHECK FAILED</td></tr>
<tr><td>APP_CHK = 0</td><td>No check performed</td></tr>
<tr><td rowspan="2">LOGICAL BLOCK REFERENCE TAG</td><td>REF_CHK = 1 [h]</td><td>LOGICAL BLOCK REFERENCE TAG CHECK FAILED</td></tr>
<tr><td>REF_CHK = 0</td><td>No check performed</td></tr>
<tr><td>No</td><td colspan="3">No protection information available to check</td></tr>
<tr><td rowspan="4">001b
010b
011b
100b
101b [b]</td><td rowspan="3">Yes</td><td>LOGICAL BLOCK GUARD</td><td colspan="2">No check performed</td></tr>
<tr><td>LOGICAL BLOCK APPLICATION TAG</td><td colspan="2">No check performed</td></tr>
<tr><td>LOGICAL BLOCK REFERENCE TAG</td><td colspan="2">No check performed</td></tr>
<tr><td>No</td><td colspan="3">Error condition [a]</td></tr>
<tr><td>110b to 111b</td><td colspan="4">Reserved</td></tr>
</table>

Table 128 — VRPROTECT field with the BYTCHK field set to 01b or 11b – checking protection information from the verify operations (part 2 of 2)

Code	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

[a] If the logical unit supports protection information (see 4.21) and has not been formatted with protection information, then the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.

[b] If the logical unit does not support protection information, then the device server should 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.

[c] If the device server has knowledge of the contents of the LOGICAL BLOCK APPLICATION TAG field, then the device server shall check each logical block application tag. If the ATO bit in the Control mode page (see SPC-6) is set to one, then this knowledge is acquired from:

a) the EXPECTED LOGICAL BLOCK APPLICATION TAG field and the LOGICAL BLOCK APPLICATION TAG MASK field in the CDB, if the command is a VERIFY (32) command (see 5.39);
b) the Application Tag mode page (see 6.5.3), if the command is not a VERIFY (32) command and the ATMPE bit in the Control mode page (see SPC-6) is set to one; or
c) a method not defined by this standard, if the command is not a VERIFY (32) command and the ATMPE bit is set to zero.

[d] If the device server terminates the command with CHECK CONDITION status, then the device server shall set the sense key to ABORTED COMMAND.

[e] If multiple errors occur while the device server is processing the command, then the selection by the device server of which error to report is not defined by this standard.

[f] See the Extended INQUIRY Data VPD page (see SPC-6) for the definitions of the GRD_CHK bit, the APP_CHK bit, and the REF_CHK bit.

[g] If the device server detects:

a) a LOGICAL BLOCK APPLICATION TAG field set to FFFFh and type 1 protection (see 4.21.2.3) or type 2 protection (see 4.21.2.4) is enabled; or
b) a LOGICAL BLOCK APPLICATION TAG field set to FFFFh, LOGICAL BLOCK REFERENCE TAG field set to FFFF_FFFFh, and type 3 protection (see 4.21.2.5) is enabled,

then the device server shall not check any protection information in the associated protection information interval.

[h] If type 1 protection is enabled, then the device server shall check each logical block reference tag by comparing it to the lower four bytes of the LBA associated with the logical block. If type 2 protection or type 3 protection is enabled, and the device server has knowledge of the contents of the LOGICAL BLOCK REFERENCE TAG field, then the device server shall check the logical block reference tag. If type 2 protection is enabled, then this knowledge may be acquired through the EXPECTED INITIAL LOGICAL BLOCK REFERENCE TAG field in a VERIFY (32) command (see 5.39). If type 3 protection is enabled, then the method for acquiring this knowledge is not defined by this standard.

[i] If the DPICZ bit in the Control mode page (see SPC-6) is set to one, then protection information shall not be checked.

If the BYTCHK field is set to 01b or 11b, then table 129 defines the checks that the device server shall perform on the protection information transferred from the Data-Out Buffer based on the VRPROTECT field. All footnotes for table 129 are at the end of the table.

Table 129 — VRPROTECT field with the BYTCHK field set to 01b or 11b – checking protection information from the Data-Out Buffer (part 1 of 2)

Code	Logical unit formatted with protection information	Field in protection information [g]	Device server check	If check fails [d e], additional sense code
000b	Yes	No protection information in the Data-Out Buffer to check		
	No	No protection information in the Data-Out Buffer to check		
001b [b]	Yes	LOGICAL BLOCK GUARD	Shall	LOGICAL BLOCK GUARD CHECK FAILED
		LOGICAL BLOCK APPLICATION TAG	May [c]	LOGICAL BLOCK APPLICATION TAG CHECK FAILED
		LOGICAL BLOCK REFERENCE TAG	Shall (except for type 3) [f]	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 [f]	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]		
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]		

Table 129 — VRPROTECT field with the BYTCHK field set to 01b or 11b – checking protection information from the Data-Out Buffer (part 2 of 2)

Code	Logical unit formatted with protection information	Field in protection information [g]	Device server check	If check fails [d e], additional sense code
101b [b]	Yes	LOGICAL BLOCK GUARD	Shall	LOGICAL BLOCK GUARD CHECK FAILED
		LOGICAL BLOCK APPLICATION TAG	May [c]	LOGICAL BLOCK APPLICATION TAG CHECK FAILED
		LOGICAL BLOCK REFERENCE TAG	May [f]	LOGICAL BLOCK REFERENCE TAG CHECK FAILED
	No	Error condition [a]		
110b to 111b	Reserved			

[a] If the logical unit supports protection information (see 4.21) and has not been formatted with protection information, then the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.

[b] If the logical unit does not support protection information, then the device server should 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.

[c] If the device server has knowledge of the contents of the LOGICAL BLOCK APPLICATION TAG field and the ATO bit is set to one in the Control mode page (see SPC-6), then the device server may check each logical block application tag. If the ATO bit is set to one, then this knowledge is acquired from:

a) the EXPECTED LOGICAL BLOCK APPLICATION TAG field and the LOGICAL BLOCK APPLICATION TAG MASK field in the CDB, if the command is a VERIFY (32) command;
b) the Application Tag mode page (see 6.5.3), if the command is not a VERIFY (32) command and the ATMPE bit in the Control mode page (see SPC-6) is set to one; or
c) a method not defined by this standard, if the command is not a VERIFY (32) command and the ATMPE bit is set to zero.

[d] If the device server terminates the command with CHECK CONDITION status, then the device server shall set the sense key to ABORTED COMMAND.

[e] If multiple errors occur while the device server is processing the command, then the selection by the device server of which error to report is not defined by this standard.

[f] If type 1 protection is enabled, then the device server shall check the logical block reference tag by comparing it to the lower four bytes of the LBA associated with the logical block. If type 2 protection is enabled and the device server has knowledge of the contents of the LOGICAL BLOCK REFERENCE TAG field, then the device server shall check each logical block reference tag. If type 2 protection is enabled, then this knowledge may be acquired through the EXPECTED INITIAL LOGICAL BLOCK REFERENCE TAG field in a VERIFY (32) command (see 5.39). If type 3 protection is enabled, the ATO bit is set to one in the Control mode page (see SPC-6), and the device server has knowledge of the contents of the LOGICAL BLOCK REFERENCE TAG field, then the device server may check each logical block reference tag. If type 3 protection is enabled, then the method for acquiring this knowledge is not defined by this standard.

[g] If the NO_PI_CHK bit is set to one in the Extended INQUIRY Data VPD page (see SPC-6) and the device server detects:

a) a LOGICAL BLOCK APPLICATION TAG field set to FFFFh and type 1 protection (see 4.21.2.3) or type 2 protection (see 4.21.2.4) is enabled; or
b) a LOGICAL BLOCK APPLICATION TAG field set to FFFFh, LOGICAL BLOCK REFERENCE TAG field set to FFFF_FFFFh, and type 3 protection (see 4.21.2.5) is enabled,

then the device server shall not check any protection information in the associated protection information interval.

If the BYTCHK field is set to 01b or 11b, then table 130 defines the processing by the device server of the protection information during the compare operation based on the VRPROTECT field. All footnotes for table 130 are at the end of the table.

Table 130 — VRPROTECT field with the BYTCHK field set to 01b or 11b – compare operation requirements (part 1 of 4)

<table>
<tr><th>Code</th><th>Logical unit formatted with protection information</th><th>Field in protection information [j]</th><th>Compare operation</th><th>If compare fails [c d], additional sense code</th></tr>
<tr><td rowspan="2">000b</td><td>Yes</td><td colspan="3">No protection information in the Data-Out Buffer to compare. Only user data is compared within each logical block.</td></tr>
<tr><td>No</td><td colspan="3">No protection information from the verify operations or in the Data-Out Buffer to compare. Only user data is compared within each logical block.</td></tr>
<tr><td rowspan="7">001b [b]</td><td rowspan="6">Yes</td><td>LOGICAL BLOCK GUARD</td><td>Shall</td><td>LOGICAL BLOCK GUARD CHECK FAILED</td></tr>
<tr><td>LOGICAL BLOCK APPLICATION TAG (ATO = 1) [e]</td><td>Shall [i]</td><td>LOGICAL BLOCK APPLICATION TAG CHECK FAILED</td></tr>
<tr><td>LOGICAL BLOCK APPLICATION TAG (ATO = 0) [f]</td><td>Shall not</td><td>No compare performed</td></tr>
<tr><td>LOGICAL BLOCK REFERENCE TAG (not type 3)</td><td>Shall [g]</td><td>LOGICAL BLOCK REFERENCE TAG CHECK FAILED</td></tr>
<tr><td>LOGICAL BLOCK REFERENCE TAG (type 3 and ATO = 1)</td><td>Shall [h]</td><td>LOGICAL BLOCK REFERENCE TAG CHECK FAILED</td></tr>
<tr><td>LOGICAL BLOCK REFERENCE TAG (type 3 and ATO = 0)</td><td>Shall not</td><td>No compare performed</td></tr>
<tr><td>No</td><td colspan="3">Error condition [a]</td></tr>
</table>

Table 130 — VRPROTECT field with the BYTCHK field set to 01b or 11b – compare operation requirements
(part 2 of 4)

Code	Logical unit formatted with protection information	Field in protection information [j]	Compare operation	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 [i]	LOGICAL BLOCK APPLICATION TAG CHECK FAILED
		LOGICAL BLOCK APPLICATION TAG (ATO = 0) [f]	Shall not	No compare performed
		LOGICAL BLOCK REFERENCE TAG (not type 3)	Shall [g]	LOGICAL BLOCK REFERENCE TAG CHECK FAILED
		LOGICAL BLOCK REFERENCE TAG (type 3 and ATO = 1)	Shall [h]	LOGICAL BLOCK REFERENCE TAG CHECK FAILED
		LOGICAL BLOCK REFERENCE TAG (type 3 and ATO = 0)	Shall not	No compare performed
	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 [i]	LOGICAL BLOCK APPLICATION TAG CHECK FAILED
		LOGICAL BLOCK APPLICATION TAG (ATO = 0) [f]	Shall not	No compare performed
		LOGICAL BLOCK REFERENCE TAG (not type 3)	Shall [g]	LOGICAL BLOCK REFERENCE TAG CHECK FAILED
		LOGICAL BLOCK REFERENCE TAG (type 3 and ATO = 1)	Shall [h]	LOGICAL BLOCK REFERENCE TAG CHECK FAILED
		LOGICAL BLOCK REFERENCE TAG (type 3 and ATO = 0)	Shall not	No compare performed
	No	Error condition [a]		

Table 130 — VRPROTECT field with the BYTCHK field set to 01b or 11b – compare operation requirements (part 3 of 4)

Code	Logical unit formatted with protection information	Field in protection information [j]	Compare operation	If compare fails [c d], additional sense code
101b [b]	Yes	LOGICAL BLOCK GUARD	Shall	LOGICAL BLOCK GUARD CHECK FAILED
		LOGICAL BLOCK APPLICATION TAG (ATO = 1) [e]	Shall [i]	LOGICAL BLOCK APPLICATION TAG CHECK FAILED
		LOGICAL BLOCK APPLICATION TAG (ATO = 0) [f]	Shall not	No compare performed
		LOGICAL BLOCK REFERENCE TAG	Shall not	No compare performed
	No	Error condition [a]		
110b to 111b	Reserved			

Table 130 — VRPROTECT field with the BYTCHK field set to 01b or 11b – compare operation requirements (part 4 of 4)

Code	Logical unit formatted with protection information	Field in protection information [j]	Compare operation	If compare fails [c d], additional sense code

[a] If the logical unit supports protection information (see 4.21) and has not been formatted with protection information, then the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.

[b] If the logical unit does not support protection information, then the device server should 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.

[c] If the device server terminates the command with CHECK CONDITION status, then the device server shall set the sense key to MISCOMPARE.

[d] If multiple errors occur while the device server is processing the command, then the selection by the device server 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-6), then the device server shall not modify the logical block application tag.

[f] If the ATO bit is set to zero in the Control mode page (see SPC-6), then the device server may modify any logical block application tag.

[g] If the BYTCHK field is set to 11b, then the device server shall compare the value from each LOGICAL BLOCK REFERENCE TAG field received in the single logical block data from the Data-Out Buffer with the corresponding LOGICAL BLOCK REFERENCE TAG field in the first logical block from the verify operations, and the device server shall compare the value of the previous LOGICAL BLOCK REFERENCE TAG field plus one with each of the subsequent LOGICAL BLOCK REFERENCE TAG fields (see 4.22.3).

[h] If the BYTCHK field is set to 11b, then the device server shall compare the value from each LOGICAL BLOCK REFERENCE TAG field received in the single logical block data from the Data-Out Buffer with the corresponding LOGICAL BLOCK REFERENCE TAG field in each logical block from the verify operations (see 4.21.3).

[i] If the device server has knowledge of the contents of the LOGICAL BLOCK APPLICATION TAG field and the ATO bit is set to one in the Control mode page (see SPC-6), then the device server shall compare each logical block application tag. If the ATO bit is set to one, then this knowledge is acquired from:

a) the EXPECTED LOGICAL BLOCK APPLICATION TAG field and the LOGICAL BLOCK APPLICATION TAG MASK field in the CDB, if the command is a VERIFY (32) command;
b) the Application Tag mode page (see 6.5.3), if the command is not a VERIFY (32) command and the ATMPE bit in the Control mode page (see SPC-6) is set to one; or
c) a method not defined by this standard, if the command is not a VERIFY (32) command and the ATMPE bit is set to zero.

[j] If the device server detects:

a) a LOGICAL BLOCK APPLICATION TAG field set to FFFFh and type 1 protection (see 4.22.2.3) or type 2 protection (see 4.22.2.4) is enabled; or
b) a LOGICAL BLOCK APPLICATION TAG field set to FFFFh, LOGICAL BLOCK REFERENCE TAG field set to FFFF_FFFFh, and type 3 protection (see 4.22.2.5) is enabled,

then the device server shall not compare any protection information in the associated protection information interval.

5.37 VERIFY (12) command

The VERIFY (12) command (see table 131) requests that the device server perform the actions defined for the VERIFY (10) command (see 5.36).

NOTE 15 - Migration from the VERIFY (12) command to the VERIFY (16) command is recommended for all implementations.

Table 131 — 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	Reserved		GROUP NUMBER					
11	CONTROL							

The OPERATION CODE field is defined in SPC-6 and shall be set to the value shown in table 131 for the VERIFY (12) command.

See the VERIFY (10) command (see 5.36) for the definitions of the other fields in this command.

5.38 VERIFY (16) command

The VERIFY (16) command (see table 132) requests that the device server perform the actions defined for the VERIFY (10) command (see 5.36).

Table 132 — 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	Reserved		GROUP NUMBER					
15	CONTROL							

The OPERATION CODE field is defined in SPC-6 and shall be set to the value shown in table 132 for the VERIFY (16) command.

See the VERIFY (10) command (see 5.36) for the definitions of the other fields in this command.

5.39 VERIFY (32) command

The VERIFY (32) command (see table 133) requests that the device server perform the actions defined for the VERIFY (10) command (see 5.36).

The device server shall process a VERIFY (32) command only if type 2 protection is enabled (see 4.21.2.4).

Table 133 — VERIFY (32) command

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

The OPERATION CODE field, the ADDITIONAL CDB LENGTH field, and the SERVICE ACTION field are defined in SPC-6 and shall be set to the values shown in table 133 for the VERIFY (32) command.

See the VERIFY (10) command (see 5.36) for the definitions of the CONTROL byte, the GROUP NUMBER field, the VRPROTECT field, the DPO bit, the BYTCHK field, the LOGICAL BLOCK ADDRESS field, and the VERIFICATION LENGTH field.

If checking of the LOGICAL BLOCK REFERENCE TAG field is enabled (see table 127, table 128, table 129, and table 130 in 5.36), then the EXPECTED INITIAL LOGICAL BLOCK REFERENCE TAG field contains the value of the LOGICAL BLOCK REFERENCE TAG field expected in the protection information of the first logical block accessed by the command instead of a value based on the LBA (see 4.21.3).

If the ATO bit is set to one in the Control mode page (see SPC-6) and checking of the LOGICAL BLOCK APPLICATION TAG field is enabled (see table 127, table 128, table 129, and table 130 in 5.36), then the LOGICAL BLOCK APPLICATION TAG MASK field contains a value that is a bit mask for enabling the checking of the LOGICAL BLOCK APPLICATION TAG field in every instance of the protection information for each logical block accessed by the command. A LOGICAL BLOCK APPLICATION TAG MASK bit set to one enables the checking of the corresponding bit of the EXPECTED LOGICAL BLOCK APPLICATION TAG field with the corresponding bit of the LOGICAL BLOCK APPLICATION TAG field in every instance of the protection information. A LOGICAL BLOCK APPLICATION TAG MASK field bit set to zero disables the checking of the corresponding bit of the EXPECTED LOGICAL BLOCK APPLICATION TAG field with the corresponding bit of the LOGICAL BLOCK APPLICATION TAG field in every instance of protection information.

The LOGICAL BLOCK APPLICATION TAG MASK field and the EXPECTED LOGICAL BLOCK APPLICATION TAG field shall be ignored if:

a) the ATO bit is set to zero; or
b) the ATO bit is set to one in the Control mode page (see SPC-6) and checking of the LOGICAL BLOCK APPLICATION TAG field is disabled (see table 127, table 128, table 129, and table 130 in 5.36).

5.40 WRITE (10) command

5.40.1 WRITE (10) command overview

The WRITE (10) command (see table 134) requests that the device server:

a) transfer the specified logical block data from the Data-Out Buffer; and
b) perform write operations to the specified LBAs using the transferred logical blocks.

NOTE 16 - Migration from the WRITE (10) command to the WRITE (16) command is recommended for all implementations.

Table 134 — WRITE (10) command

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

The OPERATION CODE field is defined in SPC-6 and shall be set to the value shown in table 134 for the WRITE (10) command.

The device server shall check the protection information, if any, transferred from the Data-Out Buffer based on the WRPROTECT field as described in table 135. All footnotes for table 135 are at the end of the table.

Table 135 — WRPROTECT field (part 1 of 2)

Code	Logical unit formatted with protection information	Field in protection information [k]	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]	Yes [e]	LOGICAL BLOCK GUARD	Shall	LOGICAL BLOCK GUARD CHECK FAILED
		LOGICAL BLOCK APPLICATION TAG	Dependent on RWWP [c]	LOGICAL BLOCK APPLICATION TAG CHECK FAILED
		LOGICAL BLOCK REFERENCE TAG	Shall (except for type 3) [j]	LOGICAL BLOCK REFERENCE TAG CHECK FAILED
	No [a]	No protection information available to check		
010b [b]	Yes [e]	LOGICAL BLOCK GUARD	Shall not	No check performed
		LOGICAL BLOCK APPLICATION TAG	Dependent on RWWP [c]	LOGICAL BLOCK APPLICATION TAG CHECK FAILED
		LOGICAL BLOCK REFERENCE TAG	May [j]	LOGICAL BLOCK REFERENCE TAG CHECK FAILED
	No [a]	No protection information available to check		
011b [b]	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]	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		

Table 135 — WRPROTECT field (part 2 of 2)

Code	Logical unit formatted with protection information	Field in protection information [k]	Device server check	If check fails [d i], additional sense code
101b [b]	Yes [e]	LOGICAL BLOCK GUARD	Shall	LOGICAL BLOCK GUARD CHECK FAILED
		LOGICAL BLOCK APPLICATION TAG	Dependent on RWWP [c]	LOGICAL BLOCK APPLICATION TAG CHECK FAILED
		LOGICAL BLOCK REFERENCE TAG	May [j]	LOGICAL BLOCK REFERENCE TAG CHECK FAILED
	No [a]	No protection information available to check		
110b to 111b	Reserved			

a If a logical unit supports protection information (see 4.21) and has not been formatted with protection information, then the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.

b If the logical unit does not support protection information, then the device server should terminate the requested command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.

c See 5.40.2

d If the device server terminates the command with CHECK CONDITION status, then the device server shall set the sense key to ABORTED COMMAND.

e The device server shall preserve the contents of protection information (e.g., write it to the medium or store it in non-volatile memory).

f The device server shall write a generated CRC (see 4.21.4.2) into each LOGICAL BLOCK GUARD field.

g If the RWWP bit in the Control mode page (see SPC-6) is set to one, then the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB. If the RWWP bit is set to zero and:
   a) type 1 protection is enabled, then 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; or
   b) type 2 protection or type 3 protection is enabled, then the device server shall write a value of FFFF_FFFFh 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-6), then the device server shall write FFFFh into each LOGICAL BLOCK APPLICATION TAG field. If the ATO bit is set to zero, then the device server may write any value into each LOGICAL BLOCK APPLICATION TAG field.

i If multiple errors occur while the device server is processing the command, then the selection by the device server of which error to report is not defined by this standard.

j If type 1 protection is enabled, then the device server shall check the logical block reference tag by comparing it to the lower four bytes of the LBA associated with the logical block. If type 2 protection is enabled and the device server has knowledge of the contents of the LOGICAL BLOCK REFERENCE TAG field, then the device server shall check each logical block reference tag. If type 2 protection is enabled, then this knowledge may be acquired through the EXPECTED INITIAL LOGICAL BLOCK REFERENCE TAG field in a WRITE (32) command, a WRITE ATOMIC (32) command, a WRITE SAME (32) command, WRITE SCATTERED (32) command, or a WRITE STREAM (32) command. If type 3 protection is enabled, the ATO bit is set to one in the Control mode page (see SPC-6), and the device server has knowledge of the contents of the LOGICAL BLOCK REFERENCE TAG field, then the device server may check each logical block reference tag. If type 3 protection is enabled, then the method for acquiring this knowledge is not defined by this standard.

k If the NO_PI_CHK bit is set to one in the Extended INQUIRY Data VPD page (see SPC-6) and the device server detects:
   a) a LOGICAL BLOCK APPLICATION TAG field set to FFFFh and type 1 protection (see 4.21.2.3) or type 2 protection (see 4.21.2.4) is enabled; or
   b) a LOGICAL BLOCK APPLICATION TAG field set to FFFFh, LOGICAL BLOCK REFERENCE TAG field set to FFFF_FFFFh, and type 3 protection (see 4.21.2.5) is enabled,

   then the device server shall not check any protection information in the associated protection information interval.

See the READ (10) command (see 5.16) for the definition of the DPO bit.

A force unit access (FUA) bit set to one specifies that the device server shall write the logical blocks to:

a) the non-volatile cache, if any; or
b) the medium.

An FUA bit set to zero specifies that the device server shall write the logical blocks to:

a) volatile cache, if any;
b) non-volatile cache, if any; or
c) the medium.

See the PRE-FETCH (10) command (see 5.13) for the definition of the LOGICAL BLOCK ADDRESS field.

See the PRE-FETCH (10) command and 4.22 for the definition of the GROUP NUMBER field.

The TRANSFER LENGTH field specifies the number of contiguous logical blocks of data that shall be transferred from the Data-Out Buffer and written, starting with the logical block referenced by the LBA specified by the LOGICAL BLOCK ADDRESS field. A TRANSFER LENGTH field set to zero specifies that no logical blocks shall be transferred or written. This condition shall not be considered an error. Any other value specifies the number of logical blocks that shall be transferred and written. If the specified LBA and the specified transfer length exceed the capacity of the medium (see 4.5), then the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to LOGICAL BLOCK ADDRESS OUT OF RANGE. The TRANSFER LENGTH field is constrained by the MAXIMUM TRANSFER LENGTH field (see 6.6.4).

The CONTROL byte is defined in SAM-6.

5.40.2 RWWP interaction

If the device server has knowledge of the contents of the LOGICAL BLOCK APPLICATION TAG field and the ATO bit is set to one in the Control mode page (see SPC-6), then the device server:

a) may check each logical block application tag if the RWWP bit is set to zero in the Control mode page (see SPC-6); and
b) shall check each logical block application tag if the RWWP bit is set to one in the Control mode page.

If the ATO bit in the Control mode page (see SPC-6) is set to one, then this knowledge is acquired from:

a) the EXPECTED LOGICAL BLOCK APPLICATION TAG field and the LOGICAL BLOCK APPLICATION TAG MASK field in the CDB, if a WRITE (32) command (see 5.43), a WRITE ATOMIC (32) command (see 5.49), a WRITE SAME (32) command (see 5.54), or a WRITE STREAM (32) command (see 5.58) is received by the device server;
b) the EXPECTED LOGICAL BLOCK APPLICATION TAG field and the LOGICAL BLOCK APPLICATION TAG MASK field in each LBA range descriptor, if a WRITE SCATTERED (32) command (see 5.56), is received by the device server;
c) the Application Tag mode page (see 6.5.3), if a command other than WRITE (32), WRITE ATOMIC (32), WRITE SAME (32), WRITE SCATTERED (32), or WRITE STREAM (32) is received by the device server and the ATMPE bit in the Control mode page (see SPC-6) is set to one; or
d) a method not defined by this standard, if a command other than WRITE (32), WRITE ATOMIC (32), WRITE SAME (32), WRITE SCATTERED (32), or WRITE STREAM (32) is received by the device server, and the ATMPE bit is set to zero.

5.41 WRITE (12) command

The WRITE (12) command (see table 136) requests that the device server perform the actions defined for the WRITE (10) command (see 5.40).

NOTE 17 - Migration from the WRITE (12) command to the WRITE (16) command is recommended for all implementations.

Table 136 — WRITE (12) command

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

The OPERATION CODE field is defined in SPC-6 and shall be set to the value shown in table 136 for the WRITE (12) command.

See the WRITE (10) command (see 5.40) for the definitions of the other fields in this command.

5.42 WRITE (16) command

The WRITE (16) command (see table 137) requests that the device server perform the actions defined for the WRITE (10) command (see 5.40).

Table 137 — WRITE (16) command

Byte \ Bit	7	6	5	4	3	2	1	0
0	OPERATION CODE (8Ah)							
1	WRPROTECT			DPO	FUA	Reserved	Obsolete	DLD2
2	(MSB)	LOGICAL BLOCK ADDRESS						
•••								
9								(LSB)
10	(MSB)	TRANSFER LENGTH						
•••								
13								(LSB)
14	DLD1	DLD0	GROUP NUMBER					
15	CONTROL							

The OPERATION CODE field is defined in SPC-6 and shall be set to the value shown in table 137 for the WRITE (16) command.

See the READ (16) command (see 5.18) for the definitions of the DLD2 bit, the DLD1 bit, and the DLD0 bit.

See the WRITE (10) command (see 5.40) for the definitions of the other fields in this command.

5.43 WRITE (32) command

The WRITE (32) command (see table 138) requests that the device server perform the actions defined for the WRITE (10) command (see 5.40).

The device server shall process a WRITE (32) command only if type 2 protection is enabled (see 4.21.2.4).

Table 138 — WRITE (32) command

Byte \ Bit	7	6	5	4	3	2	1	0
0	OPERATION CODE (7Fh)							
1	CONTROL							
2 ••• 5	Reserved							
6	Reserved		GROUP NUMBER					
7	ADDITIONAL CDB LENGTH (18h)							
8	(MSB)	SERVICE ACTION (000Bh)						
9								(LSB)
10	WRPROTECT			DPO	FUA	Reserved	Obsolete	Reserved
11	Reserved					DLD2	DLD1	DLD0
12	(MSB)	LOGICAL BLOCK ADDRESS						
••• 19								(LSB)
20	(MSB)	EXPECTED INITIAL LOGICAL BLOCK REFERENCE TAG						
••• 23								(LSB)
24	(MSB)	EXPECTED LOGICAL BLOCK APPLICATION TAG						
25								(LSB)
26	(MSB)	LOGICAL BLOCK APPLICATION TAG MASK						
27								(LSB)
28	(MSB)	TRANSFER LENGTH						
••• 31								(LSB)

The OPERATION CODE field, the ADDITIONAL CDB LENGTH field, and the SERVICE ACTION field are defined in SPC-6 and shall be set to the values shown in table 138 for the WRITE (32) command.

See the WRITE (10) command (see 5.40) for the definitions of the CONTROL byte, the GROUP NUMBER field, the WRPROTECT field, the DPO bit, the FUA bit, the LOGICAL BLOCK ADDRESS field, and the TRANSFER LENGTH field.

See the READ (16) command (see 5.18) for the definitions of the DLD2 bit, DLD1 bit, and DLD0 bit.

If checking of the LOGICAL BLOCK REFERENCE TAG field is enabled (see table 135), then the EXPECTED INITIAL LOGICAL BLOCK REFERENCE TAG field contains the value of the LOGICAL BLOCK REFERENCE TAG field expected in the protection information of the first logical block accessed by the command instead of a value based on the LBA (see 4.21.3).

If the ATO bit is set to one in the Control mode page (see SPC-6) and checking of the LOGICAL BLOCK APPLICATION TAG field is enabled (see table 135), then the LOGICAL BLOCK APPLICATION TAG MASK field contains a value that is a bit mask for enabling the checking of the LOGICAL BLOCK APPLICATION TAG field in every instance of protection information for each logical block accessed by the command. A LOGICAL BLOCK APPLICATION TAG MASK bit set to one enables the checking of the corresponding bit of the EXPECTED LOGICAL BLOCK APPLICATION TAG field with the corresponding bit of the LOGICAL BLOCK APPLICATION TAG field in every instance of protection information. A LOGICAL BLOCK APPLICATION TAG MASK field bit set to zero disables the checking of the corresponding bit of the EXPECTED LOGICAL BLOCK APPLICATION TAG field with the corresponding bit of the LOGICAL BLOCK APPLICATION TAG field in every instance of protection information.

If the ATO bit is set to:

a) zero; or
b) one in the Control mode page (see SPC-6) and checking of the LOGICAL BLOCK APPLICATION TAG field is disabled (see table 135),

then the LOGICAL BLOCK APPLICATION TAG MASK field and the EXPECTED LOGICAL BLOCK APPLICATION TAG field shall be ignored

5.44 WRITE AND VERIFY (10) command

The WRITE AND VERIFY (10) command (see table 139) requests that the device server:

1) transfer the specified the logical block data for the command from the Data-Out Buffer;
2) perform write medium operations to the specified LBAs;
3) perform verify operations from the specified LBAs; and
4) if specified, perform a compare operation on:
   A) the logical block data transferred from the Data-Out Buffer; and
   B) the logical block data from the verify operations.

The device server may process the LBAs in any order but shall perform this sequence in the specified order for a given LBA.

NOTE 18 - Migration from the WRITE AND VERIFY (10) command to the WRITE AND VERIFY (16) command is recommended for all implementations.

Table 139 — 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)							
•••	LOGICAL BLOCK ADDRESS							
5								(LSB)
6	Reserved		GROUP NUMBER					
7	(MSB)							
	TRANSFER LENGTH							
8								(LSB)
9	CONTROL							

The OPERATION CODE field is defined in SPC-6 and shall be set to the value shown in table 139 for the WRITE AND VERIFY (10) command.

See the PRE-FETCH (10) command (see 5.13) for the definition of the LOGICAL BLOCK ADDRESS field. See the PRE-FETCH (10) command and 4.22 for the definition of the GROUP NUMBER field. See the WRITE (10) command (see 5.40) for the definitions of the CONTROL byte, and the TRANSFER LENGTH field. See the READ (10) command (see 5.16) for the definition of the DPO bit.

See the WRITE (10) command for the definition of the WRPROTECT field for the write operation.

If the verify operation performs protection information checking, then the WRPROTECT field shall be processed using the specification for the VRPROTECT field in the VERIFY (10) command (see table 127) for that verify operation.

See the VERIFY (10) command (see 5.36) for definition of the byte check (BYTCHK) field when set to 00b, 01b, and 10b. For a WRITE AND VERIFY (10) command, a BYTCHK field set to 11b is reserved.

5.45 WRITE AND VERIFY (12) command

The WRITE AND VERIFY (12) command (see table 140) requests that the device server perform the actions defined for the WRITE AND VERIFY (10) command (see 5.44).

NOTE 19 - Migration from the WRITE AND VERIFY (12) command to the WRITE AND VERIFY (16) command is recommended for all implementations.

Table 140 — 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)	LOGICAL BLOCK ADDRESS						
•••								
5								(LSB)
6	(MSB)	TRANSFER LENGTH						
•••								
9								(LSB)
10	Reserved		GROUP NUMBER					
11	CONTROL							

The OPERATION CODE field is defined in SPC-6 and shall be set to the value shown in table 140 for the WRITE AND VERIFY (12) command.

See the WRITE AND VERIFY (10) command (see 5.44) for the definitions of the other fields in this command.

5.46 WRITE AND VERIFY (16) command

The WRITE AND VERIFY (16) command (see table 141) requests that the device server perform the actions defined for the WRITE AND VERIFY (10) command (see 5.44).

Table 141 — 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)							
•••	LOGICAL BLOCK ADDRESS							
9								(LSB)
10	(MSB)							
•••	TRANSFER LENGTH							
13								(LSB)
14	Reserved		GROUP NUMBER					
15	CONTROL							

The OPERATION CODE field is defined in SPC-6 and shall be set to the value shown in table 141 for the WRITE AND VERIFY (16) command.

See the WRITE AND VERIFY (10) command (see 5.44) for the definitions of the other fields in this command.

5.47 WRITE AND VERIFY (32) command

The WRITE AND VERIFY (32) command (see table 142) requests that the device server perform the actions defined for the WRITE AND VERIFY (10) command (see 5.44).

The device server shall process a WRITE AND VERIFY (32) command only if type 2 protection is enabled (see 4.21.2.4).

Table 142 — WRITE AND VERIFY (32) command

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

The OPERATION CODE field, the ADDITIONAL CDB LENGTH field, and the SERVICE ACTION field are defined in SPC-6 and shall be set to the values shown in table 142 for the WRITE AND VERIFY (32) command.

See the WRITE AND VERIFY (10) command (see 5.44) for the definitions of the CONTROL byte, the GROUP NUMBER field, the WRPROTECT field, the DPO bit, the BYTCHK field, the LOGICAL BLOCK ADDRESS field, and the TRANSFER LENGTH field.

If checking of the LOGICAL BLOCK REFERENCE TAG field is enabled (see table 135), then the EXPECTED INITIAL LOGICAL BLOCK REFERENCE TAG field contains the value of the LOGICAL BLOCK REFERENCE TAG field expected in the protection information of the first logical block accessed by the command instead of a value based on the LBA (see 4.21.3).

If the ATO bit is set to one in the Control mode page (see SPC-6) and checking of the LOGICAL BLOCK APPLICATION TAG field is enabled (see table 135 in 5.40), then the LOGICAL BLOCK APPLICATION TAG MASK field contains a value that is a bit mask for enabling the checking of the LOGICAL BLOCK APPLICATION TAG field in every instance of protection information for each logical block accessed by the command. A LOGICAL BLOCK APPLICATION TAG MASK bit set to one enables the checking of the corresponding bit of the EXPECTED LOGICAL BLOCK APPLICATION TAG field with the corresponding bit of the LOGICAL BLOCK APPLICATION TAG field in every instance of protection information. A LOGICAL BLOCK APPLICATION TAG MASK field bit set to zero disables the checking of the corresponding bit of the EXPECTED LOGICAL BLOCK APPLICATION TAG field with the corresponding bit of the LOGICAL BLOCK APPLICATION TAG field in every instance of protection information.

If the ATO bit is set to one in the Control mode page (see SPC-6) and checking of the LOGICAL BLOCK APPLICATION TAG field is disabled (see table 135), or if the ATO bit is set to zero, then the LOGICAL BLOCK APPLICATION TAG MASK field and the EXPECTED LOGICAL BLOCK APPLICATION TAG field shall be ignored.

5.48 WRITE ATOMIC (16) command

The WRITE ATOMIC (16) command (see table 143) requests that the device server:

a) transfer logical block data from the Data-Out Buffer; and
b) perform one or more atomic write operations (see 4.29) of the LBAs specified by this command.

Table 143 — WRITE ATOMIC (16) command

Byte \ Bit	7	6	5	4	3	2	1	0
0	OPERATION CODE (9Ch)							
1	WRPROTECT			DPO	FUA	Reserved		
2	(MSB)							
•••	LOGICAL BLOCK ADDRESS							
9								(LSB)
10	ATOMIC BOUNDARY							
11								
12	(MSB)							
13	TRANSFER LENGTH							(LSB)
14	Reserved		GROUP NUMBER					
15	CONTROL							

The OPERATION CODE field is defined in SPC-6 and shall be set to the value shown in table 143 for the WRITE ATOMIC (16) command.

The ATOMIC BOUNDARY field specifies whether multiple atomic write operations may be performed. If the ATOMIC BOUNDARY field is set to zero, then a single atomic write operation of the length specified in the TRANSFER LENGTH field shall be performed. If the ATOMIC BOUNDARY field is set to a non-zero value then multiple atomic write operations may be performed as described in 4.29.

See the WRITE (10) command (see 5.40) for the definitions of the other fields in this command.

5.49 WRITE ATOMIC (32) command

The WRITE ATOMIC (32) command (see table 144) requests that the device server perform the actions defined for the WRITE ATOMIC (16) command (see 5.48).

The device server shall process a WRITE ATOMIC (32) command only if type 2 protection is enabled (see 4.21.2.4).

Table 144 — WRITE ATOMIC (32) command

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

The OPERATION CODE field, the ADDITIONAL CDB LENGTH field, and the SERVICE ACTION field are defined in SPC-6 and shall be set to the values shown in table 144 for the WRITE ATOMIC (32) command.

See the WRITE ATOMIC (16) command (see 5.48) for the definition of the ATOMIC BOUNDARY field.

See the WRITE (32) command (see 5.43) for the definitions of the other fields in this command.

5.50 WRITE LONG (10) command

The WRITE LONG (10) command (see table 145) requests that the device server mark a logical block as containing a pseudo unrecovered error. If a cache contains the specified logical block, then the device server shall invalidate that logical block in the cache.

NOTE 20 - Migration from the WRITE LONG (10) command to the WRITE LONG (16) command is recommended for all implementations.

Table 145 — WRITE LONG (10) command

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

The OPERATION CODE field is defined in SPC-6 and shall be set to the value shown in table 145 for the WRITE LONG (10) command.

The write uncorrectable error (WR_UNCOR) bit is shown in table 146.

Table 146 — WR_UNCOR bit

WR_UNCOR	Description
0	Obsolete
1	Mark the specified logical block as containing a pseudo unrecovered error with correction disabled (see 4.18.2). No data is transferred.

If the WRITE LONG command is supported, then the WU_SUP bit in the Extended INQUIRY Data VPD page (see SPC-6) shall be set to one to indicate that the logical unit supports setting the WR_UNCOR bit to one.

The LOGICAL BLOCK ADDRESS field specifies an LBA. If the specified LBA exceeds the capacity of the medium (see 4.5), then the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to LOGICAL BLOCK ADDRESS OUT OF RANGE.

The CONTROL byte is defined in SAM-6.

5.51 WRITE LONG (16) command

The WRITE LONG (16) command (see table 147) requests that the device server mark a logical block as containing an error. If a cache contains the specified logical block, then the device server shall invalidate that logical block in the cache.

This command uses the SERVICE ACTION OUT (16) CDB format (see clause B.2).

Table 147 — WRITE LONG (16) command

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

The OPERATION CODE field and SERVICE ACTION field are defined in SPC-6 and shall be set to the values shown in table 147 for the WRITE LONG (16) command.

See the WRITE LONG (10) command (see 5.50) for the definitions of the fields in this command.

5.52 WRITE SAME (10) command

The WRITE SAME (10) command (see table 148) requests that the device server transfer a single logical block from the Data-Out Buffer and for each LBA in the specified range of LBAs:

a) perform a write operation using the contents of that logical block; or
b) perform an unmap operation.

The device server writes (i.e., subsequent read operations behave as if the device server wrote) the single block of user data received from the Data-Out Buffer to each logical block without modification (see 4.7.3.4.4).

If the medium is formatted with protection information and the WRPROTECT field is set to 000b, then the device server shall write the LOGICAL BLOCK GUARD field, APPLICATION TAG field, and LOGICAL BLOCK REFERENCE TAG field (see 4.21) for each logical block as described in table 135 (i.e., code equal to 000b row of table 135).

If:

a) the medium is formatted with protection information;
b) the WRPROTECT field is not set to 000b or a reserved value (see table 135); and
c) the protection information from the Data-Out Buffer is set to FFFF_FFFF_FFFF_FFFFh,

then the device server shall write FFFF_FFFF_FFFF_FFFFh to the protection information for each logical block.

If:

a) the medium is formatted with type 1 or type 2 protection information;
b) the WRPROTECT field is not set to 000b or a reserved value (see table 135); and
c) the protection information from the Data-Out Buffer is not set to FFFF_FFFF_FFFF_FFFFh,

then:

a) the device server shall write the value from the LOGICAL BLOCK REFERENCE TAG field (see 4.21) received in the logical block from the Data-Out Buffer into the corresponding LOGICAL BLOCK REFERENCE TAG field of the first logical block written. The device server shall write the value of the previous LOGICAL BLOCK REFERENCE TAG field plus one into each of the subsequent LOGICAL BLOCK REFERENCE TAG fields;
b) if the ATO bit is set to one in the Control mode page (see SPC-6) and the ATMPE bit is set to zero in the Control mode page, then the device server shall write the logical block application tag received in the logical block from the Data-Out Buffer into the corresponding LOGICAL BLOCK APPLICATION TAG field (see 4.21) of each logical block;
c) if the ATO bit is set to one in the Control mode page and the ATMPE bit is set to zero in the Control mode page, then the device server shall write the value defined in the Application Tag mode page (see 6.5.3) into the corresponding LOGICAL BLOCK APPLICATION TAG field of each logical block;
d) if the ATO bit is set to zero in the Control mode page, then the device server may write any value into the LOGICAL BLOCK APPLICATION TAG field of each logical block; and
e) the device server shall write the value from the LOGICAL BLOCK GUARD field (see 4.21) received in the logical block from the Data-Out Buffer into the corresponding LOGICAL BLOCK GUARD field of each logical block.

If:

a) the medium is formatted with type 3 protection information;
b) the WRPROTECT field is not set to 000b or a reserved value (see table 135); and
c) the protection information from the Data-Out Buffer is not set to FFFF_FFFF_FFFF_FFFFh,

then:

a) if the ATO bit is set to one in the Control mode page (see SPC-6), then the device server shall write the value from the LOGICAL BLOCK REFERENCE TAG field and the LOGICAL BLOCK APPLICATION TAG field received in the logical block from the Data-Out Buffer into the corresponding LOGICAL BLOCK REFERENCE TAG field of each logical block;
b) if the ATO bit is set to zero in the Control mode page, then the device server may write any value into the LOGICAL BLOCK REFERENCE TAG field of each logical block; and
c) the device server shall write the value from the LOGICAL BLOCK GUARD field and the LOGICAL BLOCK APPLICATION TAG field received in the logical block from the Data-Out Buffer into the corresponding LOGICAL BLOCK GUARD field of each logical block.

NOTE 21 - Migration from the WRITE SAME (10) command to the WRITE SAME (16) command is recommended for all implementations.

Table 148 — WRITE SAME (10) command

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

The OPERATION CODE field is defined in SPC-6 and shall be set to the value shown in table 148 for the WRITE SAME (10) command.

See the WRITE (10) command (see 5.40) for the definition of the WRPROTECT field.

If the logical unit supports logical block provisioning management (see 4.7.3), then the ANCHOR bit in the CDB, the UNMAP bit in the CDB, and the ANC_SUP bit (see 6.6.9) determine how the device server processes the command as described in table 149.

Table 149 — UNMAP bit, ANCHOR bit, and ANC_SUP bit relationships

UNMAP bit [a]	ANCHOR bit	ANC_SUP bit [b]	Action
0	0	0 or 1	Write [c]
	1	0 or 1	Error [d]
1	0	0 or 1	Deallocate request (see 4.7.3.4.4)
	1	0	Error [d]
		1	Anchor request (see 4.7.3.4.4)

[a] The device server in a logical unit that supports logical block provisioning management (see 4.7.3) may implement the UNMAP bit.
[b] See the Logical Block Provisioning VPD page (see 6.6.9).
[c] The device server shall perform the specified write operation to each LBA specified by the command.
[d] 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.

The device server shall ignore the UNMAP bit and the ANCHOR bit, or the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB if:

a) the logical unit is fully provisioned (i.e., the LBPME bit is set to zero in the READ CAPACITY (16) parameter data (see 5.21.2)); and
b) the UNMAP bit is set to one or the ANCHOR bit is set to one.

See the PRE-FETCH (10) command (see 5.13) for the definition of the LOGICAL BLOCK ADDRESS field.

See the PRE-FETCH (10) command (see 5.13) and 4.22 for the definition of the GROUP NUMBER field.

The NUMBER OF LOGICAL BLOCKS field specifies the number of contiguous logical blocks that are requested be unmapped or written, starting with the logical block referenced by the LBA specified by the LOGICAL BLOCK ADDRESS field.

If the NUMBER OF LOGICAL BLOCKS field is set to zero and:

a) the WSNZ bit is set to zero (see 6.6.4), then the number of contiguous logical blocks that are requested to be unmapped or written includes all of the logical blocks starting with the LBA specified in the LOGICAL BLOCK ADDRESS field and ending with the last logical block on the medium; or
b) the WSNZ bit is set to one, then the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.

If the number of contiguous logical blocks requested to be unmapped or written by the NUMBER OF LOGICAL BLOCKS field exceeds the value indicated in the MAXIMUM WRITE SAME LENGTH field (see 6.6.4), 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.

See the WRITE (10) command (see 5.40) for the definition of the CONTROL byte.

5.53 WRITE SAME (16) command

The WRITE SAME (16) command (see table 150) requests that the device server perform the actions defined for the WRITE SAME (10) command (see 5.52).

Table 150 — WRITE SAME (16) command

Bit / Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (93h)							
1	WRPROTECT			ANCHOR	UNMAP	Obsolete		NDOB
2	(MSB)							
•••	LOGICAL BLOCK ADDRESS							
9								(LSB)
10	(MSB)							
•••	NUMBER OF LOGICAL BLOCKS							
13								(LSB)
14	Reserved		GROUP NUMBER					
15	CONTROL							

The OPERATION CODE field is defined in SPC-6 and shall be set to the value shown in table 150 for the WRITE SAME (16) command.

A NDOB bit set to zero specifies that the device server shall process the command using logical block data from the Data-Out Buffer. A NDOB bit set to one specifies that:

a) the device server shall not transfer data from the Data-Out Buffer;
b) if the Logical Block Provisioning VPD page (see 6.6.9) is not supported or the LBPRZ field (see 6.6.9) is set to 000b or xx1b, then the device server shall process the command as if the Data-Out Buffer contained user data set to all zeroes and protection information, if any, containing:
   A) the LOGICAL BLOCK GUARD field set to FFFFh;
   B) the LOGICAL BLOCK REFERENCE TAG field set to FFFF_FFFFh; and
   C) the LOGICAL BLOCK APPLICATION TAG field set to FFFFh;

   and
c) if the LBPRZ field is set to 010b, then the device server shall process the command as if the Data-Out Buffer contained user data set to the provisioning initialization pattern and protection information, if any, containing:
   A) the LOGICAL BLOCK GUARD field set to FFFFh;
   B) the LOGICAL BLOCK REFERENCE TAG field set to FFFF_FFFFh; and
   C) the LOGICAL BLOCK APPLICATION TAG field set to FFFFh.

See the WRITE SAME (10) command (see 5.52) for the definitions of the other fields in this command.

5.54 WRITE SAME (32) command

The WRITE SAME (32) command (see table 151) requests that the device server perform the actions defined for the WRITE SAME (10) command (see 5.52).

The device server shall process a WRITE SAME (32) command only if type 2 protection is enabled (see 4.21.2.4).

Table 151 — WRITE SAME (32) command

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

The OPERATION CODE field, the ADDITIONAL CDB LENGTH field, and the SERVICE ACTION field are defined in SPC-6 and shall be set to the values shown in table 151 for the WRITE SAME (32) command.

See the WRITE SAME (10) command (see 5.52) for the definitions of the CONTROL byte, the GROUP NUMBER field, the WRPROTECT field, the LOGICAL BLOCK ADDRESS field, the NUMBER OF LOGICAL BLOCKS field, the UNMAP bit, and the ANCHOR bit.

See the WRITE SAME (16) command (see 5.53) for the definition of the NDOB bit.

If checking of the LOGICAL BLOCK REFERENCE TAG field is enabled (see table 135 in 5.40), then the EXPECTED INITIAL LOGICAL BLOCK REFERENCE TAG field contains the value of the LOGICAL BLOCK REFERENCE TAG field expected in the protection information of the first logical block accessed by the command instead of a value based on the LBA (see 4.21.3).

If the ATO bit is set to one in the Control mode page (see SPC-6) and checking of the LOGICAL BLOCK APPLICATION TAG field is enabled (see table 135 in 5.40), then the LOGICAL BLOCK APPLICATION TAG MASK field contains a value that is a bit mask for enabling the checking of the LOGICAL BLOCK APPLICATION TAG field in every instance of protection information for each logical block accessed by the command. A LOGICAL BLOCK

APPLICATION TAG MASK bit set to one enables the checking of the corresponding bit of the EXPECTED LOGICAL BLOCK APPLICATION TAG field with the corresponding bit of the LOGICAL BLOCK APPLICATION TAG field in every instance of protection information. A LOGICAL BLOCK APPLICATION TAG MASK field bit set to zero disables the checking of the corresponding bit of the EXPECTED LOGICAL BLOCK APPLICATION TAG field with the corresponding bit of the LOGICAL BLOCK APPLICATION TAG field in every instance of protection information.

If the ATO bit is set to one in the Control mode page (see SPC-6) and checking of the LOGICAL BLOCK APPLICATION TAG field is disabled (see table 135 in 5.40), or if the ATO bit is set to zero, then the LOGICAL BLOCK APPLICATION TAG MASK field and the EXPECTED LOGICAL BLOCK APPLICATION TAG field shall be ignored.

5.55 WRITE SCATTERED (16) command

5.55.1 WRITE SCATTERED (16) command overview

The WRITE SCATTERED (16) command (see table 152) requests that the device server perform scattered writes (see 4.35).

This command uses the SERVICE ACTION OUT (16) CDB format (see SPC-6).

Table 152 — WRITE SCATTERED (16) command

Byte \ Bit	7	6	5	4	3	2	1	0
0	OPERATION CODE (9Fh)							
1	Reserved			SERVICE ACTION (12h)				
2	WRPROTECT			DPO	FUA	Reserved		DLD2
3	Reserved							
4	(MSB)	LOGICAL BLOCK DATA OFFSET						
5								(LSB)
6	Reserved							
7								
8	(MSB)	NUMBER OF LBA RANGE DESCRIPTORS						
9								(LSB)
10	(MSB)	BUFFER TRANSFER LENGTH						
•••								
13								(LSB)
14	DLD1	DLD0	GROUP NUMBER					
15	CONTROL							

The OPERATION CODE field and the SERVICE ACTION field are defined in SPC-6 and shall be set to the values shown in table 152 for the WRITE SCATTERED (16) command.

See the WRITE (10) command (see 5.40) for the definition of the WRPROTECT field and the FUA bit. See the READ (10) command (see 5.16) for the definition of the DPO bit. See the READ (16) command (see 5.18) for the definition of the DLD2 bit.

The LOGICAL BLOCK DATA OFFSET field specifies the offset of the scattered write logical block data in the Data-Out Buffer (see table 153) in multiples of the length of a logical block (i.e., the length of the user data and protection information, if any, in one logical block). The value in the LOGICAL BLOCK DATA OFFSET field is calculated by dividing (m+1) (see table 153) by the length of a logical block (i.e., the length of the user data and protection information, if any, in one logical block).

The NUMBER OF LBA RANGE DESCRIPTORS field specifies the number of LBA range descriptors (see table 154) in the LBA range descriptor list that shall be transferred from the Data-Out Buffer. If the NUMBER OF LBA RANGE DESCRIPTORS field is greater than the number indicated by the MAXIMUM SCATTERED LBA RANGE DESCRIPTOR COUNT field in the Block Limits Extension VPD page (see 6.6.5), 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. A NUMBER OF LBA RANGE DESCRIPTORS field set to 0000h specifies that no data shall be transferred or written. This condition shall not be considered an error. If the value in the LOGICAL BLOCK DATA OFFSET field is not large enough to contain the specified number of LBA range descriptors, 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.

The BUFFER TRANSFER LENGTH field specifies the number of logical blocks that shall be transferred from the Data-Out Buffer. If the BUFFER TRANSFER LENGTH field is greater than the MAXIMUM SCATTERED TRANSFER LENGTH field in the Block Limits Extension VPD page (see 6.6.5), 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. A BUFFER TRANSFER LENGTH field set to zero specifies that no logical blocks shall be transferred. This condition shall not be considered an error.

See the READ (16) command for the definitions of the DLD1 bit and the DLD0 bit.

The CONTROL byte is defined in SAM-6.

5.55.2 WRITE SCATTERED (16) command Data-Out Buffer contents

Table 153 defines the Data-Out Buffer contents for the WRITE SCATTERED (16) command

Table 153 — Data-Out Buffer contents for the WRITE SCATTERED (16) command

Byte \ Bit	7	6	5	4	3	2	1	0
	WRITE SCATTERED (16) parameter list header							
0 ••• 31	Reserved							
	LBA range descriptor list							
32 ••• 63	LBA range descriptor [first] (see table 154)							
	⋮							
k-31 ••• k	LBA range descriptor [last] (see table 154)							
k+1 ••• m	PAD (if any)							
	Scattered write logical block data							
m+1 •••	Logical block data for first LBA range							
	⋮							
••• k	Logical block data for last LBA range							

The LBA range descriptor list specifies one or more LBA range descriptors (see table 154). The LBA range descriptors may be in any order. The device server may process the LBA range descriptors in any order. The application client should not specify overlapping LBA ranges. If overlapping LBA ranges are specified, then the logical block data in the overlapping LBAs may be any of the logical block data specified for those LBAs. The device server shall not consider an LBA range with the NUMBER OF LOGICAL BLOCKS field set to 0000_0000h as overlapping with any other LBA range.

The PAD field shall contain zero or more bytes set to zero such that the total length of the LBA range descriptor list and the PAD field is a multiple of the length of a logical block (i.e., the length of the user data and protection information, if any, in one logical block). Device servers shall ignore the PAD field.

Table 154 defines the LBA range descriptor.

Table 154 — LBA range descriptor

Byte \ Bit	7	6	5	4	3	2	1	0
0	(MSB)							
•••	LOGICAL BLOCK ADDRESS							
7								(LSB)
8	(MSB)							
•••	NUMBER OF LOGICAL BLOCKS							
11								(LSB)
12								
•••	Reserved							
31								

The LOGICAL BLOCK ADDRESS field specifies the LBA of the first logical block (see 4.5) in this LBA range.

The NUMBER OF LOGICAL BLOCKS field specifies the number of contiguous logical blocks of logical block data from the scattered write logical block data (see table 153) that are associated with this LBA range descriptor. If the NUMBER OF LOGICAL BLOCKS field is greater than the MAXIMUM SCATTERED LBA RANGE TRANSFER LENGTH field (see 6.6.5), then the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN PARAMETER LIST. A NUMBER OF LOGICAL BLOCKS field set to zero specifies that no logical blocks shall be written. This condition shall not be considered an error. Any other value specifies the number of logical blocks that shall be written to the medium as specified in 4.35

If the specified LBA and the specified number of logical blocks exceed the capacity of the medium (see 4.5), then the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to LOGICAL BLOCK ADDRESS OUT OF RANGE.

If:

a) the device server processes an LBA range descriptor and the value in the NUMBER OF LOGICAL BLOCKS field added to the sum of the values from the NUMBER OF LOGICAL BLOCKS fields in the LBA range descriptors from this command that have already been processed exceeds the BUFFER TRANSFER LENGTH field in the CDB minus the LOGICAL BLOCK DATA OFFSET field in the CDB (i.e., the number of logical blocks requested to be written by all of the processed LBA range descriptors exceeds the number of logical blocks in the scattered write logical block data (see table 153)); or
b) the device server processes all of the LBA range descriptors and the sum of the values in the NUMBER OF LOGICAL BLOCKS fields from all of the LBA range descriptors does not equal the BUFFER TRANSFER LENGTH field in the CDB minus the LOGICAL BLOCK DATA OFFSET field in the CDB,

then the device server shall terminate the command with CHECK CONDITION status, with the sense key set to ABORTED COMMAND, the additional sense code set to PARAMETER LIST LENGTH ERROR, and the field pointer pointing to the BUFFER TRANSFER LENGTH field. The contents of the logical blocks referenced by any LBA range descriptors already processed may be old data, new data, or any combination thereof.

5.56 WRITE SCATTERED (32) command

5.56.1 WRITE SCATTERED (32) command overview

The WRITE SCATTERED (32) command (see table 155) requests that the device server perform the actions defined for the WRITE SCATTERED (16) command (see 5.55).

Table 155 — WRITE SCATTERED (32) command

Byte \ Bit	7	6	5	4	3	2	1	0
0	OPERATION CODE (7Fh)							
1	CONTROL							
2 ••• 5	Reserved							
6	Reserved		GROUP NUMBER					
7	ADDITIONAL CDB LENGTH (18h)							
8	(MSB)	SERVICE ACTION (0011h)						
9								(LSB)
10	WRPROTECT			DPO	FUA	Reserved		
11	Reserved							
12	(MSB)	LOGICAL BLOCK DATA OFFSET						
13								(LSB)
14	Reserved							
15								
16	(MSB)	NUMBER OF LBA RANGE DESCRIPTORS						
17								(LSB)
18 ••• 27	Reserved							
28	(MSB)	BUFFER TRANSFER LENGTH						
•••								
31								(LSB)

The OPERATION CODE field, the ADDITIONAL CDB LENGTH field, and the SERVICE ACTION field are defined in SPC-6 and shall be set to the values shown in table 155 for the WRITE SCATTERED (32) command.

See the WRITE SCATTERED (16) command (see 5.55) for the definitions of the other fields in this command.

5.56.2 WRITE SCATTERED (32) command Data-Out Buffer contents

Table 156 defines the Data-Out Buffer contents for the WRITE SCATTERED (32) command

Table 156 — Data-Out Buffer contents for the WRITE SCATTERED (32) command

Byte \ Bit	7	6	5	4	3	2	1	0
	WRITE SCATTERED (32) parameter list header							
0 ••• 31	Reserved							
	LBA range descriptor list							
32 ••• 63	LBA range descriptor [first] (see table 157)							
	⋮							
k-31 ••• k	LBA range descriptor [last] (see table 157)							
k+1 ••• m	PAD (if any)							
	Scattered write logical block data							
m+1 •••	Logical block data for first LBA range							
	⋮							
••• k	Logical block data for last LBA range							

The LBA range descriptor list specifies one or more LBA range descriptors (see table 157). The LBA range descriptors may be in any order. The device server may process the LBA range descriptors in any order. The application client should not specify overlapping LBA ranges. If overlapping LBA ranges are specified, then the logical block data in the overlapping LBAs may be any of the logical block data specified for those LBAs. The device server shall not consider an LBA range with the NUMBER OF LOGICAL BLOCKS field set to 0000_0000h as overlapping with any other LBA range.

The PAD field shall contain zero or more bytes set to zero such that the total length of the LBA range descriptor list and the PAD field is a multiple of the length of a logical block (i.e., the length of the user data and protection information, if any, in one logical block). Device servers shall ignore the PAD field.

Table 157 defines the LBA range descriptor.

Table 157 — LBA range descriptor

Byte \ Bit	7	6	5	4	3	2	1	0
0	(MSB)							
•••	LOGICAL BLOCK ADDRESS							
7								(LSB)
8	(MSB)							
•••	NUMBER OF LOGICAL BLOCKS							
11								(LSB)
12	(MSB)							
•••	EXPECTED INITIAL LOGICAL BLOCK REFERENCE TAG							
15								(LSB)
16	(MSB)	EXPECTED LOGICAL BLOCK APPLICATION TAG						
17								(LSB)
18	(MSB)	LOGICAL BLOCK APPLICATION TAG MASK						
19								(LSB)
20								
•••	Reserved							
31								

See the WRITE SCATTERED (16) command (see 5.55) for the definitions of the LOGICAL BLOCK ADDRESS field and the NUMBER OF LOGICAL BLOCKS field.

See the WRITE (32) command (see 5.43) for the definitions of the EXPECTED INITIAL LOGICAL BLOCK REFERENCE TAG field, the EXPECTED LOGICAL BLOCK APPLICATION TAG field, and the LOGICAL BLOCK APPLICATION TAG MASK field.

5.57 WRITE STREAM (16) command

The WRITE STREAM (16) command (see table 158) requests that the device server perform the actions defined for the WRITE (10) command (see 5.40).

Table 158 — WRITE STREAM (16) command

Byte \ Bit	7	6	5	4	3	2	1	0
0	OPERATION CODE (9Ah)							
1	WRPROTECT			DPO	FUA	Reserved		
2	(MSB)	LOGICAL BLOCK ADDRESS						
•••								
9								(LSB)
10	(MSB)	STR_ID						
11								(LSB)
12	(MSB)	TRANSFER LENGTH						
13								(LSB)
14	Reserved		GROUP NUMBER					
15	CONTROL							

The OPERATION CODE field is defined in SPC-6 and shall be set to the value shown in table 158 for the WRITE STREAM (16) command.

The stream identifier (STR_ID) field specifies the stream identifier associated with this command as described in 4.32. If:

a) the GROUP NUMBER field specifies a group for which the ST_ENBLE bit is set to one in the associated IO advice hints group descriptor (see 6.5.7), if any; and
b) the value in the STR_ID field is different from the value in the group number field,

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.

See the WRITE (10) command (see 5.33) for the definitions of the other fields in this command.

5.58 WRITE STREAM (32) command

The WRITE STREAM (32) command (see table 159) requests that the device server perform the actions defined for the WRITE (32) command (see 5.43).

The device server shall process a WRITE STREAM (32) command only if type 2 protection is enabled (see 4.21.2.4).

Table 159 — WRITE STREAM (32) command

Byte \ Bit	7	6	5	4	3	2	1	0
0	OPERATION CODE (7Fh)							
1	CONTROL							
2	Reserved							
3								
4	STR_ID							
5								
6	Reserved		GROUP NUMBER					
7	ADDITIONAL CDB LENGTH (18h)							
8	(MSB)	SERVICE ACTION (0010h)						
9								(LSB)
10	WRPROTECT			DPO	FUA	Reserved		
11	Reserved							
12	(MSB)	LOGICAL BLOCK ADDRESS						
•••								
19								(LSB)
20	(MSB)	EXPECTED INITIAL LOGICAL BLOCK REFERENCE TAG						
•••								
23								(LSB)
24	(MSB)	EXPECTED LOGICAL BLOCK APPLICATION TAG						
25								(LSB)
26	(MSB)	LOGICAL BLOCK APPLICATION TAG MASK						
27								(LSB)
28	(MSB)	TRANSFER LENGTH						
•••								
31								(LSB)

The OPERATION CODE field, the ADDITIONAL CDB LENGTH field, and the SERVICE ACTION field are defined in SPC-6 and shall be set to the values shown in table 159 for the WRITE STREAM (32) command.

See the WRITE STREAM (16) command (see 5.57) for the definition of the STR_ID field.

See the WRITE (32) command (see 5.43) for the definitions of the other fields in this command.

5.59 WRITE USING TOKEN command

5.59.1 WRITE USING TOKEN command overview

The WRITE USING TOKEN command (see table 160) requests that the copy manager (see SPC-6) write logical block data represented by the specified ROD token to the specified LBAs.

Table 160 — WRITE USING TOKEN command

Byte \ Bit	7	6	5	4	3	2	1	0
0	OPERATION CODE (83h)							
1	Reserved			SERVICE ACTION (11h)				
2 ••• 5	Reserved							
6 ••• 9	(MSB) LIST IDENTIFIER (LSB)							
10 ••• 13	(MSB) PARAMETER LIST LENGTH (LSB)							
14	Reserved		GROUP NUMBER					
15	CONTROL							

The OPERATION CODE field and the SERVICE ACTION field are defined in SPC-6 and shall be set to the values shown in table 160 for the WRITE USING TOKEN command.

The LIST IDENTIFIER field is defined in SPC-6. The list identifier shall be processed as if the LIST ID USAGE field is set to 00b in the parameter data for an EXTENDED COPY(LID4) command (see SPC-6).

The PARAMETER LIST LENGTH field specifies the length in bytes of the parameter data that is available to be transferred from the Data-Out Buffer. A PARAMETER LIST LENGTH set to zero specifies that no data shall be transferred. This shall not be considered an error.

See the PRE-FETCH (10) command (see 5.13 and 4.22) for the definition of the GROUP NUMBER field.

The CONTROL byte is defined in SAM-6.

5.59.2 WRITE USING TOKEN parameter list

The parameter list for the WRITE USING TOKEN command is shown in table 161.

Table 161 — WRITE USING TOKEN parameter list

Byte \ Bit	7	6	5	4	3	2	1	0
0	(MSB)	WRITE USING TOKEN DATA LENGTH (n - 1)						
1								(LSB)
2	Reserved						DEL_TKN	IMMED
3 ••• 7	Reserved							
8	(MSB)	OFFSET INTO ROD						
••• 15								(LSB)
16 ••• 527	ROD TOKEN							
528 ••• 533	Reserved							
534 535	BLOCK DEVICE RANGE DESCRIPTOR LENGTH (n - 535)							
Block device range descriptor list (if any)								
536 ••• 551	Block device range descriptor [first] (see 5.12.3)							
⋮								
n - 15 ••• n	Block device range descriptor [last] (see 5.12.3)							

The WRITE USING TOKEN DATA LENGTH field specifies the length in bytes of the data that is available to be transferred from the Data-Out Buffer. The write using token data length does not include the number of bytes in the WRITE USING TOKEN DATA LENGTH field.

If the WRITE USING TOKEN DATA LENGTH field is less than 0226h (i.e., 550), then the copy manager shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

If the contents of the WRITE USING TOKEN DATA LENGTH field is not equal to the contents of the BLOCK DEVICE RANGE DESCRIPTOR LENGTH field plus 534 then the device server should terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

The immediate (IMMED) bit specifies when the copy manager shall return status for the WRITE USING TOKEN command. If the IMMED bit is set to zero, then the copy manager shall process the WRITE USING TOKEN command until all specified operations are complete or an error is detected. If the IMMED bit is set to one, then the copy manager:

1) shall validate the CDB (i.e., detect and report all errors in the CDB);
2) shall transfer all the parameter data to the copy manager;
3) may validate the parameter data;
4) shall complete the WRITE USING TOKEN command with GOOD status; and
5) shall complete processing of all specified operations as a background operation (see SPC-6).

If the operations specified by a WRITE USING TOKEN command are processed as a background operation (i.e., the IMMED bit is set to one) (see SPC-6), then the copy manager shall not generate deferred errors (see SAM-6) to report the errors encountered, if any, during this processing. The copy manager shall make error information available to an application client using a RECEIVE ROD TOKEN INFORMATION command (see 5.25).

A delete token (DEL_TKN) bit set to one specifies that the ROD token specified in the ROD TOKEN field should be deleted when processing of the WRITE USING TOKEN command is complete. A DEL_TKN bit set to zero specifies that the ROD token lifetime for the ROD token specified in the ROD TOKEN field shall be as described in SPC-6.

The OFFSET INTO ROD field specifies the offset into the data represented by the ROD token from the first byte represented by the ROD token to the first byte to be transferred. The offset is specified in number of blocks based on the logical block length of the logical unit to which the WRITE USING TOKEN command is to write data. The copy manager that processes the WRITE USING TOKEN command shall compute the byte offset into the ROD by multiplying the contents of the OFFSET INTO ROD field by the logical block length of the logical unit to which the WRITE USING TOKEN command is to write data.

EXAMPLE - To calculate an offset, a ROD token is created from LBAs 15 to 20 followed by LBAs 40 to 100 by a copy manager associated with a logical unit with a logical block length of 512 bytes per logical block. That ROD token is specified in a WRITE USING TOKEN command that transfers one logical block to a logical unit with a logical block length of 4 096 bytes per logical block. The subsequent RECEIVE ROD TOKEN INFORMATION command indicates the successful transfer of 4 096 bytes by setting the TRANSFER COUNT field to one. To create a WRITE USING TOKEN command that transfers bytes from the ROD token starting at the point where the previous WRITE USING TOKEN command stopped, the OFFSET INTO ROD field is set to one (i.e., the contents of the TRANSFER COUNT field) plus the value in the OFFSET INTO ROD field in the previous WRITE USING TOKEN command (i.e., zero). The copy manager multiplies one (i.e., the value in the OFFSET INTO ROD field) by 4 096 (i.e., the logical block length for the logical unit to which the data is being written) and the result is 4 096. As a result, the ROD token logical block that is the start of the transfer is LBA 8 (i.e., LBA 42 from the logical unit whose logical block length is 512 bytes per logical block that was used to create the ROD token).

If the computed byte offset into the ROD is greater than or equal to the number of bytes represented by the ROD token, then the copy manager shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

If the ROD TOKEN LENGTH field (see SPC-6) in the ROD TOKEN field is not set to 01F8h, then the copy manager shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense key set to INVALID TOKEN OPERATION, INVALID TOKEN LENGTH.

The ROD TOKEN field specifies the ROD token that represents the data from which logical block data is written. The ROD token is defined as follows:

a) a ROD token returned by a RECEIVE ROD TOKEN INFORMATION command; or
b) a block device zero ROD token (see 4.28.4).

If the ROD token does not match any known to the copy manager, then the copy manager shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID TOKEN OPERATION, TOKEN UNKNOWN.

The BLOCK DEVICE RANGE DESCRIPTOR LENGTH field specifies the length in bytes of the block device range descriptor list. The block device range descriptor list length should be a multiple of 16. If the block device range descriptor list length is not a multiple of 16, then the last block device range (see 5.12.3) descriptor is incomplete and shall be ignored. If the block device range descriptor list length is less than 16, then the copy manager shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

If the number of complete block device range descriptors is larger than the MAXIMUM RANGE DESCRIPTORS field (see 6.6.11.3), then the copy manager shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to TOO MANY SEGMENT DESCRIPTORS.

If the same LBA is included in more than one block device range descriptor, then the copy manager shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

If the number of bytes of user data represented by the sum of the contents of the NUMBER OF LOGICAL BLOCKS fields in all of the complete block device range descriptors is larger than:

a) the MAXIMUM BYTES IN BLOCK ROD field in the block ROD device type specific features descriptor in the ROD token features third-party copy descriptor in the Third-party Copy VPD page (see SPC-6) and that field is set to a non-zero value; or
b) the MAXIMUM TOKEN TRANSFER SIZE field (see 6.6.11.3) and that field is set to a non-zero value,

then the copy manager shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

If the number of bytes of user data represented by the sum of the contents of the NUMBER OF LOGICAL BLOCKS fields in all of the complete block device range descriptors is larger than the number of bytes in the data represented by the ROD token minus the computed byte offset into the ROD (i.e., the total requested length of the transfer exceeds the length of the data available in the data represented by the ROD token), then the copy manager shall:

a) transfer as many whole logical blocks as possible; and
b) if any portion of a logical block that is written by the copy manager corresponds to offsets into the ROD at or beyond the length of the data represented by the ROD token, then write that portion of the logical block with user data with all bits set to zero.

The copy manager may perform this check during the processing of each block device range descriptor.

6 Parameters for direct access block devices

6.1 Parameters for direct access block devices introduction

Table 162 shows the parameters for direct access block devices defined in clause 6 and a reference to the subclause where each parameter type is defined.

Table 162 — Parameters for direct access block devices

Parameter type	Reference
Address descriptors	6.2
Diagnostic parameters	6.3
Log parameters	6.4
Mode parameters	6.5
Vital product data (VPD) parameters	6.6
Copy manager parameters	6.7
Logical block markup descriptors	6.8

6.2 Address descriptors

6.2.1 Address descriptor overview

This subclause describes the address descriptors (see table 163) used for:

a) the FORMAT UNIT command (see 5.4);
b) the READ DEFECT DATA commands (see 5.22 and 5.23);
c) the Translate Address Input diagnostic page (see 6.3.4) for the SEND DIAGNOSTIC command (see SPC-6); and
d) the Translate Address Output diagnostic page (see 6.3.5) for the RECEIVE DIAGNOSTIC RESULTS command (see SPC-6).

The format type of an address descriptor is:

a) specified in the DEFECT LIST FORMAT field (see 5.4.1);
b) indicated in the DEFECT LIST FORMAT field (see 5.22.2 and 5.23.2);
c) specified in the SUPPLIED FORMAT field and the TRANSLATE FORMAT field for the Translate Address Output diagnostic page; or
d) indicated in the SUPPLIED FORMAT field and the TRANSLATE FORMAT field for the Translate Address Input diagnostic page.

Table 163 defines the types of address descriptors.

Table 163 — Address descriptors

Type	Description	Reference
000b	Short block format address descriptor	6.2.2
001b	Extended bytes from index format address descriptor [a]	6.2.3
010b	Extended physical sector format address descriptor [a]	6.2.4
011b	Long block format address descriptor	6.2.5
100b	Bytes from index format address descriptor [a]	6.2.6
101b	Physical sector format address descriptor [a]	6.2.7
110b	Vendor specific	
111b	Reserved	

[a] This address descriptor format type is defined for direct access block devices using rotating media (see 4.3.2).

6.2.2 Short block format address descriptor

A format type of 000b specifies the short block format address descriptor (see table 164).

Table 164 — 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 parameter list, the SHORT BLOCK ADDRESS field specifies a four-byte LBA. If the physical block containing the logical block referenced by the specified LBA contains additional logical blocks, then the device server may consider the LBAs of those additional logical blocks to also have been specified.

For the READ DEFECT DATA parameter data, the SHORT BLOCK ADDRESS field indicates a vendor specific four-byte value.

For the Translate Address diagnostic pages, the SHORT BLOCK ADDRESS field contains:

a) a four-byte LBA, if the value is less than or equal to the capacity of the medium; or
b) a vendor specific four-byte value, if the value is greater than the capacity of the medium.

6.2.3 Extended bytes from index address descriptor

A format type of 001b specifies the extended bytes from index format address descriptor (see table 165). For the FORMAT UNIT parameter list and the READ DEFECT DATA parameter data, this address descriptor contains the location of a defect that:

a) is the length of one track (see 4.3.2);
b) is less than the length of a physical block; or
c) starts from one address descriptor and extends to the next address descriptor.

For the Translate Address diagnostic pages, this address descriptor contains the location of an LBA. For the Translate Address Output diagnostic page (see 6.2.5), if the SUPPLIED FORMAT field is set to 001b and the MADS bit in the ADDRESS TO TRANSLATE field is set to one, then the device server shall terminate the SEND DIAGNOSTIC command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

Table 165 — Extended bytes from index format address descriptor (001b)

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

The CYLINDER NUMBER field contains the cylinder number (see 4.3.2).

The HEAD NUMBER field contains the head number (see 4.3.2).

A multi-address descriptor start (MADS) bit set to one specifies that this address descriptor defines the beginning of a defect that spans multiple addresses. The defect may be a number of sequential physical blocks on the same cylinder and head (i.e., a track) or may span a number of sequential tracks on the same head. A MADS bit set to zero specifies that:

a) this address descriptor defines the end of a defect if the previous address descriptor has the MADS bit set to one; or
b) this address descriptor defines a single track that contains one or more defects (i.e., the BYTES FROM INDEX field contains FFF_FFFFh) or a single defect (i.e., the BYTES FROM INDEX field does not contain FFF_FFFFh).

See 4.13.2 for valid combinations of two address descriptors that describe a defect.

The BYTES FROM INDEX field:

a) if not set to FFF_FFFFh, contains the number of bytes from the index (e.g., from the start of the track) to the location being described; or
b) if set to FFF_FFFFh, specifies or indicates that the entire track is being described.

More than one logical block may be described by this address descriptor.

Table 166 defines the order of the fields used for sorting extended bytes from index format address descriptors if the command using the address descriptors specifies sorting.

Table 166 — Sorting order for extended bytes from index format address descriptors

Bit	(MSB) 59	•••	36	35	•••	28	27	•••	(LSB) 0
	CYLINDER NUMBER field			HEAD NUMBER field			BYTES FROM INDEX field		

6.2.4 Extended physical sector format address descriptor

A format type of 010b specifies the extended physical sector format address descriptor (see table 167). For the FORMAT UNIT parameter list and the READ DEFECT DATA parameter data, this address descriptor contains the location of a defect that:

a) is the length of one track (see 4.3.2);
b) is less than the length of a physical block; or
c) starts from one address descriptor and extends to the next address descriptor.

For the Translate Address diagnostic pages, this address descriptor specifies the location of an LBA. For the Translate Address Output diagnostic page (see 6.2.5), if the SUPPLIED FORMAT field is set to 010b and the MADS bit in the ADDRESS TO TRANSLATE field is set to one, then the device server shall terminate the SEND DIAGNOSTIC command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

Table 167 — Extended physical sector format address descriptor (010b)

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

The CYLINDER NUMBER field contains the cylinder number (see 4.3.2).

The HEAD NUMBER field contains the head number (see 4.3.2).

A multi-address descriptor start (MADS) bit set to one specifies that this address descriptor defines the beginning of a defect that spans multiple addresses. The defect may span a number of sequential physical blocks on the same cylinder and head (i.e., a track) or may span a number of sequential tracks on the same head. A MADS bit set to zero specifies that:

a) this address descriptor defines the end of a defect if the previous address descriptor has the MADS bit set to one; or
b) this address descriptor defines a single track that contains one or more defects (i.e., the SECTOR NUMBER field contains FFF_FFFFh) or a single defect (i.e., the SECTOR NUMBER field does not contain FFF_FFFFh).

See 4.13.2 for valid combinations of two address descriptors that describe a defect.

The SECTOR NUMBER field:

a) if not set to FFF_FFFFh, contains the sector number (see 4.3.2) of the location being described; or
b) if set to FFF_FFFFh, specifies or indicates that the entire track is being described.

More than one logical block may be described by this address descriptor.

Table 168 defines the order of the fields used for sorting extended physical sector format address descriptors if the command using the address descriptors specifies sorting.

Table 168 — Sorting order for extended physical sector format address descriptors

Bit	(MSB) 59	•••	36	35	•••	28	27	•••	(LSB) 0
	CYLINDER NUMBER field			HEAD NUMBER field			SECTOR NUMBER field		

6.2.5 Long block format address descriptor

A format type of 011b specifies the long block format address descriptor (see table 169).

Table 169 — 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 parameter list, the LONG BLOCK ADDRESS field specifies an eight-byte LBA. If the physical block containing the logical block referenced by the specified LBA contains additional logical blocks, then the device server may consider the LBAs of those additional logical blocks to also have been specified.

For the READ DEFECT DATA parameter data, the LONG BLOCK ADDRESS field indicates a vendor specific eight-byte value.

For the Translate Address diagnostic pages, the LONG BLOCK ADDRESS field contains:

a) an eight-byte LBA, if the value is less than or equal to the capacity of the medium; or
b) a vendor specific eight-byte, if the value is greater than the capacity of the medium.

6.2.6 Bytes from index format address descriptor

A format type of 100b specifies the bytes from index format address descriptor (see table 170). This address descriptor contains the location of a track or an offset from the start of a track.

Table 170 — 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 (see 4.3.2).

The HEAD NUMBER field contains the head number (see 4.3.2).

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 FFFF_FFFFh specifies or indicates that the entire track is being described.

More than one logical block may be described by this address descriptor.

Table 171 defines the order of the fields used for sorting bytes from index format address descriptors if the command using the address descriptors specifies sorting.

Table 171 — Sorting order for bytes from index format address descriptors

Bit	(MSB) 63	...	40	39	...	32	31	...	(LSB) 0
	CYLINDER NUMBER field			HEAD NUMBER field			BYTES FROM INDEX field		

6.2.7 Physical sector format address descriptor

A format type of 101b specifies the physical sector format address descriptor (see table 172). This address descriptor contains the location of a track or a sector (see 4.3.2).

Table 172 — 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 (see 4.3.2).

The HEAD NUMBER field contains the head number (see 4.3.2).

The SECTOR NUMBER field contains the sector number (see 4.3.2). A SECTOR NUMBER field set to FFFF_FFFFh specifies or indicates that the entire track is being described.

More than one logical block may be described by this address descriptor.

Table 173 defines the order of the fields used for sorting physical sector format address descriptors if the command using the address descriptors specifies sorting.

Table 173 — Sorting order for physical sector format address descriptors

Bit	(MSB) 63	...	40	39	...	32	31	...	(LSB) 0
	CYLINDER NUMBER field			HEAD NUMBER field			SECTOR NUMBER field		

6.3 Diagnostic parameters

6.3.1 Diagnostic parameters overview

See table 174 for references to the pages and descriptors for diagnostic parameters used by direct access block devices.

The diagnostic pages and their corresponding page codes for direct access block devices are shown in table 174.

Table 174 — Diagnostic page codes for direct access block devices

Diagnostic page name	Page code	Reference
Diagnostic pages assigned by SPC-6	30h to 3Fh	SPC-6
Rebuild Assist Input diagnostic page	42h	6.3.2
Rebuild Assist Output diagnostic page		6.3.3
SCSI Enclosure Services diagnostic pages	01h to 2Fh	SES-3
Supported Diagnostic Page diagnostic page	00h	SPC-6
Translate Address Input diagnostic page	40h	6.3.4
Translate Address Output diagnostic page		6.3.5
Obsolete	41h	
Vendor specific diagnostic pages	80h to FFh	
Reserved for this standard	43h to 7Fh	

6.3.2 Rebuild Assist Input diagnostic page

An application client sends a RECEIVE DIAGNOSTIC RESULTS command to retrieve a Rebuild Assist Input diagnostic page (see table 175), which provides information about whether the rebuild assist mode (see 4.19) is enabled or not and a device server's rebuild assist mode capabilities.

Table 175 — Rebuild Assist Input diagnostic page

Byte \ Bit	7	6	5	4	3	2	1	0
0	PAGE CODE (42h)							
1	Reserved							
2	(MSB)	PAGE LENGTH (4 + (2 × n))						
3								(LSB)
4	Reserved							ENABLED
5	Reserved							
6								
7	PHYSICAL ELEMENT LENGTH (n)							
8	DISABLED PHYSICAL ELEMENT MASK (if any)							
•••								
7 + n								
8 + n	DISABLED PHYSICAL ELEMENT (if any)							
•••								
7 + (2 × n)								

The PAGE CODE field and the PAGE LENGTH field are defined in SPC-6 and shall be set to the values shown in table 175.

An ENABLED bit set to one indicates that the rebuild assist mode is enabled. An ENABLED bit set to zero indicates that the rebuild assist mode is disabled.

The PHYSICAL ELEMENT LENGTH field indicates the length in bytes of the DISABLED PHYSICAL ELEMENT MASK field and the length in bytes of the DISABLED PHYSICAL ELEMENT field.

The bits in the DISABLED PHYSICAL ELEMENT MASK field indicate the bits in the DISABLED PHYSICAL ELEMENT field that are supported. Each bit set to one in the DISABLED PHYSICAL ELEMENT MASK field indicates that the corresponding bit in the DISABLED PHYSICAL ELEMENT field is supported and may be set to one in a Rebuild Assist Output diagnostic page sent with a SEND DIAGNOSTIC command.

The bits in the DISABLED PHYSICAL ELEMENT field indicate the physical elements that are disabled in this logical unit. Each bit set to one indicates that a physical element is disabled, and the device server shall report predicted read errors and predicted write errors for the associated group of LBAs.

6.3.3 Rebuild Assist Output diagnostic page

The Rebuild Assist Output diagnostic page (see table 176) provides a method for an application client to manage rebuild assist mode (see 4.19).

Table 176 — Rebuild Assist Output diagnostic page

Byte \ Bit	7	6	5	4	3	2	1	0
0	PAGE CODE (42h)							
1	Reserved							
2	(MSB)	PAGE LENGTH (4 + (2 × n))						
3								(LSB)
4	Reserved							ENABLE
5	Reserved							
6								
7	PHYSICAL ELEMENT LENGTH (n)							
8	DISABLED PHYSICAL ELEMENT MASK (if any)							
•••								
7 + n								
8 + n	DISABLE PHYSICAL ELEMENT (if any)							
•••								
7 + (2 × n)								

The PAGE CODE field and the PAGE LENGTH field are defined in SPC-6 and shall be set to the values shown in table 176.

An ENABLE bit set to one specifies that, after all fields in this diagnostic page have been validated:

a) a self-test of the physical elements in the logical unit may be performed; and
b) rebuild assist mode is enabled.

An ENABLE bit set to zero specifies that:

a) rebuild assist mode shall be disabled;
b) the other fields in this page shall be ignored; and
c) all physical elements shall be enabled.

The PHYSICAL ELEMENT LENGTH field shall be set to the same value that is returned in the PHYSICAL ELEMENT LENGTH field (see 6.3.2).

If the PHYSICAL ELEMENT LENGTH field is not set to the same value that is returned in the PHYSICAL ELEMENT LENGTH field , then the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

The device server shall ignore the DISABLED PHYSICAL ELEMENT MASK field.

Each bit in the DISABLE PHYSICAL ELEMENT field specifies a physical element that shall be disabled. A bit set to one in the DISABLE PHYSICAL ELEMENT field specifies that the device server shall respond to read commands and write commands specifying LBAs associated with that physical element as if the associated LBAs have predicted errors. A bit set to zero in the DISABLE PHYSICAL ELEMENT field specifies that the device server shall respond to read commands and write commands specifying LBAs associated with that physical element as if the associated LBAs do not have predicted errors. If the ENABLE bit is set to one, and the DISABLE PHYSICAL ELEMENT field specifies:

a) any bits set to one that are not supported by the logical unit;
b) all bits that are supported by the logical unit are set to one; or
c) setting to zero any bits that are set to one,

then the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

6.3.4 Translate Address Input diagnostic page

Table 177 defines the Translate Address Input diagnostic page sent by a device server in response to a RECEIVE DIAGNOSTIC RESULTS command after a Translate Address Output diagnostic page (see 6.3.4) has been sent by an application client with the SEND DIAGNOSTIC command. If a Translate Address Output diagnostic page has not yet been processed by the device server, the results of a RECEIVE DIAGNOSTIC RESULTS command requesting this diagnostic page are vendor specific.

Table 177 — 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 list							
6	(MSB)	TRANSLATED ADDRESS 1 (if any)						
•••								
13								(LSB)
	⋮							
n - 7	(MSB)	TRANSLATED ADDRESS x (if any)						
•••								
n								(LSB)

The PAGE CODE field is defined in SPC-6 and shall be set to the value shown in table 177 for the Translate Address Input diagnostic page.

The PAGE LENGTH field is defined in SPC-6.

The SUPPLIED FORMAT field contains the value from the SUPPLIED FORMAT field in the previous Translate Address Output diagnostic page (see 6.3.5).

A reserved area (RAREA) bit set to zero indicates that no part of the translated address falls within a reserved area of the medium (e.g., speed tolerance gap, alternate sector, or vendor reserved area). A RAREA bit set to one indicates that all or part of the translated address falls within a reserved area of the medium. If the entire translated address falls within a reserved area, then 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 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, then the device server shall set ~~this~~the ALTSEC 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 6.3.4).

Each TRANSLATED ADDRESS field contains an address descriptor (see 6.2) that the device server translated from the address descriptor supplied by the application client in the previous Translate Address Output diagnostic page (see 6.3.5). Each field shall be in the format (see 6.2) specified in the TRANSLATED FORMAT field. If the short block format address descriptor (see 6.2.2) is specified, then the first four bytes of the TRANSLATED ADDRESS field shall contain the short block format address descriptor and the last four bytes shall contain 0000_0000h.

If the returned data is in short block format (see 6.2.2), long block format (see 6.2.5), or physical sector format (see 6.2.7) 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), then 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 (see 6.2.6), 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.

6.3.5 Translate Address Output diagnostic page

The Translate Address diagnostic pages provides a method for an application client to have a device server translate an address descriptor (see 6.2) from one format to another. The address descriptor to be translated is sent to the device server in the Translate Address Output diagnostic page with a SEND DIAGNOSTIC command and the results are returned by the device server in the Translate Address Input diagnostic page sent in response to a RECEIVE DIAGNOSTIC RESULTS command.

Table 178 defines the format of the Translate Address Output diagnostic page sent with the SEND DIAGNOSTIC command. The translated address returned in the Translate Address Input diagnostic page is defined in 6.3.4.

Table 178 — Translate Address Output diagnostic page

Byte \ Bit	7	6	5	4	3	2	1	0
0	PAGE CODE (40h)							
1	Reserved							
2	(MSB)	PAGE LENGTH (000Ah)						
3								(LSB)
4	Reserved					SUPPLIED FORMAT		
5	Reserved					TRANSLATE FORMAT		
6	(MSB)	ADDRESS TO TRANSLATE						
•••								
13								(LSB)

The PAGE CODE field and PAGE LENGTH field are defined in SPC-6 and shall be set to the values shown in table 178 for the Translate Address Output diagnostic page.

The SUPPLIED FORMAT field specifies the format (see 6.2) of the ADDRESS TO TRANSLATE field. If the device server does not support the requested format, then the device server shall terminate the SEND DIAGNOSTIC command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

The TRANSLATE FORMAT field specifies the format (see 6.2) the device server shall use for the result of the address translation. If the device server does not support the specified format, then the device server shall terminate the SEND DIAGNOSTIC command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

The ADDRESS TO TRANSLATE field contains a single address descriptor that 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 6.2. If the short block format address descriptor is specified, then the first four bytes of the ADDRESS TO TRANSLATE field shall contain the short block format address descriptor and the last four bytes shall contain 0000_0000h.

6.4 Log parameters

6.4.1 Log parameters overview

6.4.1.1 Summary of log pages

See table 179 for references to the log pages and their corresponding page codes and subpage codes for direct access block devices. See SPC-6 for a detailed description of logging operations.

Table 179 — Log page codes and subpage codes for direct access block devices (part 1 of 2)

Log page name	Page code [a]	Subpage code [a]	Reference
Application Client	0Fh	00h	SPC-6
ATA PASS-THROUGH Results	16h	00h	SAT-4
Background Scan Results	15h	00h	6.4.2
Background Operation	15h	02h	6.4.3
Buffer Over-Run/Under-Run	01h	00h	SPC-6
Cache Memory Statistics	19h	20h	SPC-6
Command Duration Limits Statistics	19h	21h	SPC-6
Environmental Limits	0Dh	02h	SPC-6
Environmental Reporting	0Dh	01h	SPC-6
Format Status	08h	00h	6.4.4
General Statistics and Performance	19h	00h	SPC-6
Group Statistics and Performance (1 to 31)	19h	01h to 1Fh	SPC-6
Informational Exceptions	2Fh	00h	SPC-6
Last n Deferred Errors or Asynchronous Events	0Bh	00h	SPC-6
Last n Error Events	07h	00h	SPC-6
Logical Block Provisioning	0Ch	00h	6.4.5

[a] All page code and subpage code combinations not shown in this table are reserved.

Table 179 — Log page codes and subpage codes for direct access block devices (part 2 of 2)

Log page name	Page code [a]	Subpage code [a]	Reference
LPS Misalignment	15h	03h	6.4.6
Non-Medium Error	06h	00h	SPC-6
Non-volatile Cache	17h	00h	6.4.7
Pending Defects	15h	01h	6.4.8
Power Condition Transitions	1Ah	00h	SPC-6
Protocol-Specific Ports	18h	00h to FEh	SPC-6
Read Error Counters	03h	00h	SPC-6
Self-Test Results	10h	00h	SPC-6
Solid State Media	11h	00h	6.4.9
Start-Stop Cycle Counter	0Eh	00h	SPC-6
Supported Log Pages	00h	00h	SPC-6
Supported Log Pages and Subpages	00h	FFh	SPC-6
Supported Subpages	01h to 3Fh	FFh	SPC-6
Temperature	0Dh	00h	SPC-6
Utilization	0Eh	01h	6.4.10
Verify Error Counters	05h	00h	SPC-6
Write Error Counters	02h	00h	SPC-6
Zoned Block Device Statistics	14h	01h	ZBC-2
Vendor specific	30h to 3Eh	00h to FEh	n/a

[a] All page code and subpage code combinations not shown in this table are reserved.

6.4.1.2 Setting and resetting log parameters

In a LOG SELECT command (see SPC-6), an application client may specify that:

a) all the parameters in a log page or pages are to be reset (i.e., the PCR bit set to one and the PARAMETER LIST LENGTH field is set to zero); or
b) individual parameters in log page are to be changed to specified new values (i.e., the PCR bit is set to zero and the PARAMETER LIST LENGTH field is not set to zero).

The device server processing of LOG SELECT commands (see SPC-6) that request changes to individual log parameters or reset all log parameters depend on the log parameter that is being changed or reset, and is specified in the table that defines the log parameter using the keywords shown in table 180 (also see SPC-6).

Table 180 — Keywords for resetting or changing log parameters

<table>
<tr><th rowspan="2">Keyword</th><th colspan="2">Device server processing when:</th></tr>
<tr><th>PCR bit is set to one [a]</th><th>PCR bit is set to zero [b]</th></tr>
<tr><td>Always</td><td>Reset the log parameter.</td><td>Change the log parameter.</td></tr>
<tr><td>Reset Only</td><td>Reset the log parameter.</td><td rowspan="2">If any changes are requested in the PARAMETER VALUE field of the log parameter, then:
a) terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST; and
b) do not make any requested changes in any field in any log parameter in any log page</td></tr>
<tr><td>Never</td><td>Do not reset the log parameter; see the LOG SELECT command in SPC-6 for description of possible error conditions.</td></tr>
<tr><td colspan="3">[a] If the PCR bit is set to one, and the PARAMETER LIST LENGTH field is not set to zero, then the device server shall terminate the LOG SELECT command (see SPC-6).
[b] If the PCR bit is set to zero, and the PARAMETER LIST LENGTH field is set to zero. then no log parameters are changed (see SPC-6).</td></tr>
</table>

6.4.2 Background Scan log page

6.4.2.1 Background Scan log page overview

Using the format shown in table 182, the Background Scan log page reports information about:

a) background pre-scan operations (see 4.23.2) and background medium scan operations (see 4.23.3); and
b) any logical blocks where an error was detected during a background scan operation.

The parameter codes for the Background Scan log page are listed in table 181.

Table 181 — Background Scan log page parameter codes

Parameter code	Description	Resettable or Changeable [a]	Reference	Support
0000h	Background Scan Status	Never	6.4.2.2	Mandatory
0001h to 0800h	Background Scan Results	Reset Only	6.4.2.3	Optional [b]
8000h to AFFFh	Vendor specific		n/a	Optional
All others	Reserved			

[a] The keywords in this column – Always, Reset Only, and Never – are defined in 6.4.1.2.
[b] If the Background Scan log page is supported, then at least one Background Scan Results log parameter shall be supported.

The Background Scan log page has the format shown in table 182.

Table 182 — Background Scan log page

Byte \ Bit	7	6	5	4	3	2	1	0
0	DS (1b)	SPF (0b)	PAGE CODE (15h)					
1	SUBPAGE CODE (00h)							
2	(MSB)	PAGE LENGTH (n - 3)						
3								(LSB)
	Background scan parameters							
4 •••	Background scan parameter [first] (if any)							
	⋮							
••• n	Background scan parameter [last] (if any)							

The disable save (DS) bit, the subpage format (SPF) bit, the PAGE CODE field, the SUBPAGE CODE field, and the PAGE LENGTH field are described in SPC-6.

The DS bit, the SPF bit, the PAGE CODE field, and the SUBPAGE CODE field shall be set to the values shown in table 182 for the Background Scan log page.

If the device server processes a LOG SELECT command with the PCR bit set to one (see SPC-6), then the device server shall:

a) not change the values in the Background Scan Status log parameter; and
b) delete all Background Scan Results log parameters.

6.4.2.2 Background Scan Status log parameter

The Background Scan Status log parameter for the Background Scan log page has the format shown in table 183.

Table 183 — Background Scan Status log parameter format

Byte \ Bit	7	6	5	4	3	2	1	0
0	(MSB)	PARAMETER CODE (0000h)						
1								(LSB)
2	Parameter control byte – binary format list log parameter (see SPC-6)							
	DU	Obsolete	TSD	Obsolete			FORMAT AND LINKING	
3	PARAMETER LENGTH (0Ch)							
4	(MSB)	ACCUMULATED POWER ON MINUTES						
•••								
7								(LSB)
8	Reserved							
9	BACKGROUND SCAN STATUS							
10	(MSB)	NUMBER OF BACKGROUND SCANS PERFORMED						
11								(LSB)
12	(MSB)	BACKGROUND SCAN PROGRESS						
13								(LSB)
14	(MSB)	NUMBER OF BACKGROUND MEDIUM SCANS PERFORMED						
15								(LSB)

The PARAMETER CODE field is described in SPC-6 and shall be set to the value shown in table 183 for the Background Scan Status log parameter.

The DU bit, the TSD bit, and the FORMAT AND LINKING field for the Background Scan Status log parameter shall be set for a binary format list log parameter as described in SPC-6.

The PARAMETER LENGTH field is described in SPC-6 and shall be set to the value shown in table 183 for the Background Scan Status log parameter.

The ACCUMULATED POWER ON MINUTES field indicates the number of minutes the device server has been powered on since manufacturing.

Table 184 defines the BACKGROUND SCAN STATUS field.

Table 184 — BACKGROUND SCAN STATUS field

Code	Description
00h	No background scan operation is active.
01h	A background medium scan operation is active.
02h	A background pre-scan operation is active.
03h	A background scan operation was halted as a result of a fatal error.
04h	A background scan operation was halted as a result of a vendor specific pattern of errors.
05h	A background scan operation was halted as a result of the medium being formatted without the PLIST.
06h	A background scan operation was halted as a result of a vendor specific cause.
07h	A background scan operation was halted as a result of the temperature being out of the allowed range.
08h	Background medium scan operations are enabled (i.e., the EN_BMS bit is set to one in the Background Control mode page (see 6.5.4)), and no background medium scan operation is active (i.e., the device server is waiting for Background Medium Scan Interval timer expiration before starting the next background medium scan operation).
09h	A background scan operation was halted as a result of the S_L_FULL bit being set to one in the Background Control mode page (see 6.5.4) and the background scan results list being full.
0Ah	A background pre-scan operation was halted as a result of the Background Pre-scan Time Limit timer expiring.
0Bh to FFh	Reserved

The NUMBER OF BACKGROUND SCANS PERFORMED field indicates the number of background scan operations (i.e., the total number of background pre-scan operations plus the number of background medium scan operations) that have been performed since the SCSI target device was shipped by the manufacturer.

The BACKGROUND SCAN PROGRESS field indicates the percent complete of a background scan operation in progress. The returned value is a numerator that has 65 536 (i.e., 1_0000h) as its denominator. If there is no background scan operation in progress (i.e., no background scan operation has been initiated since power on or the most recent background scan operation has completed), then the device server shall set the BACKGROUND SCAN PROGRESS field to 0000h.

The NUMBER OF BACKGROUND MEDIUM SCANS PERFORMED field indicates the number of background medium scan operations that have been performed since the SCSI target device was shipped by the manufacturer. If the NUMBER OF BACKGROUND MEDIUM SCANS PERFORMED field contains 0000h, then the number of background medium scan operations is unknown.

The total number of background pre-scan operations that have been performed is the value in the NUMBER OF BACKGROUND SCANS PERFORMED field minus the value in the NUMBER OF BACKGROUND MEDIUM SCANS PERFORMED field.

6.4.2.3 Background Scan Results log parameter

The Background Scan Results log parameter for the Background Scan log page has the format shown in table 185. If the Background Scan log page is reset, then all Background Scan Results log parameters are discarded. If no errors have occurred during any background scan since the most recent reset of the Background Scan log page, then no Background Scan Results log parameters shall be present.

Table 185 — Background Scan Results log parameter format

Byte \ Bit	7	6	5	4	3	2	1	0
0	(MSB)	PARAMETER CODE (0001h to 0800h)						
1								(LSB)
2	Parameter control byte – binary format list log parameter (see SPC-6)							
	DU	Obsolete	TSD	Obsolete			FORMAT AND LINKING	
3	PARAMETER LENGTH (14h)							
4	(MSB)	ACCUMULATED POWER ON MINUTES						
•••								
7								(LSB)
8	REASSIGN STATUS				SENSE KEY			
9	ADDITIONAL SENSE CODE							
10	ADDITIONAL SENSE CODE QUALIFIER							
11	Vendor specific							
•••								
15								
16	(MSB)	LOGICAL BLOCK ADDRESS						
•••								
23								(LSB)

The PARAMETER CODE field is described in SPC-6 and shall be set to a value from 0001h through 0800h in sequence as errors are discovered during a background scan operation. When all of the supported parameter code values have been used, and a new error is discovered during a background scan operation, the oldest Background Scan Results log parameter in the list (i.e., the Background Scan Results log parameter with the smallest value in the ACCUMULATED POWER ON MINUTES field) shall be discarded, and the PARAMETER CODE field (see 6.4.2.3) for the new defect shall be set to the parameter code value of the discarded Background Scan Results log parameter.

The DU bit, the TSD bit, and the FORMAT AND LINKING field for a Background Scan Results log parameter shall be set for a binary format list log parameter as described in SPC-6.

The PARAMETER LENGTH field is defined in SPC-6 and shall be set to the value shown in table 185 for the Background Scan Results log parameter.

The ACCUMULATED POWER ON MINUTES field indicates the number of minutes that the device server has been powered on since manufacturing at the time the background scan error reported in the Background Scan Results log parameter occurred.

Table 186 defines the REASSIGN STATUS field.

Table 186 — REASSIGN STATUS field

Code	Reason			
	LOWIR bit [a]		Original error [b]	Additional conditions
	0	1		
1h	Yes	Yes	Recovered or unrecovered	The LBA has not yet been reassigned. [c]
2h	Yes	No	Recovered	The device server performed automatic read reassignment for the LBA (i.e., performed a reassign operation for the LBA and a write operation with recovered logical block data). [d]
4h	Yes	Yes	Recovered	The device server's attempt to perform automatic read reassignment failed. The logical block may or may not now have an uncorrectable error. [c]
5h	Yes	No	Recovered	The error was corrected by the device server rewriting the logical block without performing a reassign operation.
6h	Yes	Yes	Recovered or unrecovered	Either: a) an application client caused automatic write reassignment for the LBA with a command performing a write operation; or b) the LBPRZ field is set to xx1b (see 6.6.9), and an application client caused an unmap operation for the LBA. [c]
7h	Yes	Yes	Recovered or unrecovered	Either: a) an application client caused a reassign operation for the LBA with a REASSIGN BLOCKS command; or b) the LBPRZ field is set to 000b or is set to 010b (see 6.6.9), and an application client caused an unmap operation for the LBA. [c]
8h	Yes	Yes	Recovered or unrecovered	An application client's request for a reassign operation for the LBA with a REASSIGN BLOCKS command failed. The logical block referenced by the LBA may or may not still have an uncorrectable error.
All others	Reserved			

Key:

Yes = specifies that a Background Scan Results log parameter shall be generated for the error.
No = specifies that a Background Scan Results log parameter shall not be generated for the error

[a] The LOWIR bit (see 6.5.4).
[b] Type of error detected while reading the logical block referenced by the LBA specified by the LOGICAL BLOCK ADDRESS field(see 6.4.2.3) during a background scan operation.
[c] The REASSIGN STATUS field in a given log parameter changes from 1h or 4h to 6h, 7h, or 8h when a reassign operation, write operation, or unmap operation on the LBA succeeds or when a reassign operation on the LBA fails. After the LBA is reassigned, any subsequent medium error occurring for the LBA is reported in a new log parameter with the same value in the LOGICAL BLOCK ADDRESS field as the value in the LOGICAL BLOCK ADDRESS field in the log parameter for the previous medium error for the LBA.
[d] The ARRE bit (see 6.5.10) controls automatic read reassignment based on errors detected during all read medium operations, including those that are part of background scan operations.

If sense data is available, then the device server shall set the SENSE KEY field, the ADDITIONAL SENSE CODE field, and the ADDITIONAL SENSE CODE QUALIFIER field to a hierarchy of additional information relating to error conditions that occurred during the background scan operation. The content of these fields is represented in the same format used by the sense data (see SPC-6).

The LOGICAL BLOCK ADDRESS field indicates the LBA associated with the medium error.

6.4.3 Background Operation log page

6.4.3.1 Background Operation log page overview

Using the format shown in table 187, the Background Operation log page reports parameters that are specific to background operations.

Table 187 — Background Operation log page

Byte \ Bit	7	6	5	4	3	2	1	0
0	DS	SPF (1b)	PAGE CODE (15h)					
1	SUBPAGE CODE (02h)							
2	(MSB)	PAGE LENGTH (n - 3)						
3								(LSB)
	Background operation parameters							
4 •••	Background operation parameter [first] (if any)							
	⋮							
••• n	Background operation parameter [last] (if any)							

The disable save (DS) bit, the subpage format (SPF) bit, the PAGE CODE field, the SUBPAGE CODE field, and the PAGE LENGTH field are described in SPC-6.

The SPF bit, the PAGE CODE field, and the SUBPAGE CODE field shall be set to the values shown in table 187 for the Background Operation log page.

The parameter codes for the Background Operation log page are listed in table 188.

Table 188 — Background Operation log page parameter codes

Parameter code	Description	Resettable or Changeable [a]	Reference	Support
0000h	Background Operation	Never	6.4.3.2	Mandatory
All others	Reserved			
[a] The keywords in this column – Always, Reset Only, and Never – are defined in 6.4.1.2.				

6.4.3.2 Background Operation log parameter

The Background Operation log parameter of the Background Operation log page has the format shown in table 189.

Table 189 — Background Operation log parameter format

Byte \ Bit	7	6	5	4	3	2	1	0
0	(MSB)	PARAMETER CODE (0000h)						
1								(LSB)
2	Parameter control byte – binary format list log parameter (see SPC-6)							
	DU	Reserved	TSD	Reserved			FORMAT AND LINKING	
3	PARAMETER LENGTH (4h)							
4	BO_STATUS							
5	Reserved							
•••								
n								

The PARAMETER CODE field is described in SPC-6 and shall be set to the value shown in table 189 for the Background Operation log parameter.

The DU bit, the TSD bit, and the FORMAT AND LINKING field for the Background Operation log parameter shall be set for a binary format list log parameter as described in SPC-6.

The PARAMETER LENGTH field is described in SPC-6 and shall be set to the value shown in table 189 for the Background Operation log parameter.

The background operation status (BO_STATUS) field indicates the type of background operation, if any, that is being performed by the device server as shown in table 190.

TABLE 190 — BO_STATUS DEFINITIONS

Code	Description
00h	No indication
01h	No advanced background operation being performed
02h	Host initiated advanced background operation being performed
03h	Device initiated advanced background operation being performed
All others	Reserved

6.4.4 Format Status log page

6.4.4.1 Format Status log page overview

Using the format shown table 192, the Format Status log page reports information about the most recent successful format operation and the state of the direct access block device since that operation was performed. The parameter codes for the Format Status log page are listed in table 191.

Table 191 — Format Status log page parameter codes

Parameter code	Description	Resettable or Changeable [a]	Reference	Support
0000h	Format Data Out	Never	6.4.4.2	Mandatory
0001h	Grown Defects During Certification	Never	6.4.4.3	Mandatory
0002h	Total Blocks Reassigned During Format	Never	6.4.4.4	Mandatory
0003h	Total New Blocks Reassigned	Never	6.4.4.5	Mandatory
0004h	Power On Minutes Since Format	Never	6.4.4.6	Mandatory
0005h to 7FFFh	Reserved			
8000h to FFFFh	Vendor specific			Optional

[a] The keywords in this column – Always, Reset Only, and Never – are defined in 6.4.1.2.

The Format Status log page has the format shown in table 192.

Table 192 — Format Status log page

Byte \ Bit	7	6	5	4	3	2	1	0
0	DS (1b)	SPF (0b)	PAGE CODE (08h)					
1	SUBPAGE CODE (00h)							
2	(MSB)	PAGE LENGTH (n - 3)						
3								(LSB)
	Format status log parameters							
4 •••	Format status log parameter [first]							
⋮								
••• n	Format status log parameter [last]							

The disable save (DS) bit, the subpage format (SPF) bit, the PAGE CODE field, the SUBPAGE CODE field, and the PAGE LENGTH field are described in SPC-6.

The DS bit, the SPF bit, the PAGE CODE field, and the SUBPAGE CODE field shall be set to the values shown in table 192 for the Format Status log page.

If a format operation has never been performed by the logical unit, then the log parameter for each Format Status log parameter listed in table 191 is not defined by this standard. If a device server begins a format operation, then the device server shall set each byte of the log parameter data (i.e., bytes four to n of the log parameter), if any, to FFh for each Format Status log parameter (e.g., if the PARAMETER LENGTH field is set to 02h, then the log parameter data is set to FFFFh).

If the most recent format operation failed or the information for a Format Status log parameter is not available, then the device server shall return FFh in each byte of the log parameter data (i.e., bytes four to n of the log parameter), if any, for the Format Status log parameter (e.g., if the PARAMETER LENGTH field is set to 04h, then the log parameter data shall be set to FFFF_FFFFh). The device server shall set each Format Status log parameter to be a multiple of four bytes.

6.4.4.2 Format Data Out log parameter

The Format Data Out log parameter of the Format Status log page has the format shown in table 193.

Table 193 — Format Data Out log parameter format

Byte \ Bit	7	6	5	4	3	2	1	0
0	(MSB)	PARAMETER CODE (0000h)						
1								(LSB)
2	Parameter control byte – binary format list log parameter (see SPC-6)							
	DU	Obsolete	TSD	Obsolete			FORMAT AND LINKING	
3	PARAMETER LENGTH (n - 3)							
4	(MSB)	FORMAT DATA OUT						
•••								
n								(LSB)

The PARAMETER CODE field is described in SPC-6 and shall be set to the value shown in table 193 for the Format Data Out log parameter.

The DU bit, and the FORMAT AND LINKING field for the Format Data Out log parameter shall be set for a binary format list log parameter as described in SPC-6.

The target save disable (TSD) bit (see SPC-6) shall be set to zero for the Format Data Out log parameter, indicating that the logical unit saves the Format Data Out log parameter at vendor specific intervals without any request from an application client.

The PARAMETER LENGTH field is described in SPC-6.

After a successful format operation, the FORMAT DATA OUT field contains the FORMAT UNIT parameter list (see 5.4.2).

6.4.4.3 Grown Defects During Certification log parameter

The Grown Defects During Certification log parameter for the Format Status log page has the format shown in table 194.

Table 194 — Grown Defects During Certification log parameter format

Byte \ Bit	7	6	5	4	3	2	1	0
0	(MSB)	PARAMETER CODE (0001h)						
1								(LSB)
2	Parameter control byte – binary format list log parameter (see SPC-6)							
	DU	Obsolete	TSD	Obsolete			FORMAT AND LINKING	
3	PARAMETER LENGTH (08h)							
4	(MSB)	GROWN DEFECTS DURING CERTIFICATION						
•••								
11								(LSB)

The PARAMETER CODE field is described in SPC-6 and shall be set to the value shown in table 194 for the Grown Defects During Certification log parameter.

The DU bit, and the FORMAT AND LINKING field for the Grown Defects During Certification log parameter shall be set for a binary format list log parameters as described in SPC-6.

The target save disable (TSD) bit (see SPC-6) shall be set to zero for the Grown Defects During Certification log parameter, indicating that the logical unit saves the Grown Defects During Certification log parameter at vendor specific intervals without any request from an application client.

The PARAMETER LENGTH field is described in SPC-6 and shall be set to the value shown in table 194 for the Grown Defects During Certification log parameter.

After a successful format operation during which certification was performed, the GROWN DEFECTS DURING CERTIFICATION field shall indicate the number of defects detected as a result of performing the certification. The value in the GROWN DEFECTS DURING CERTIFICATION field count reflects only those defects detected and replaced during the successful format operation that were not already part of the PLIST or GLIST.

After a successful format operation during which certification was not performed, the GROWN DEFECTS DURING CERTIFICATION field shall be set to zero.

6.4.4.4 Total Blocks Reassigned During Format log parameter

The Total Blocks Reassigned During Format log parameter for the Format Status log page has the format shown in table 195.

Table 195 — Total Blocks Reassigned During Format log parameter format

Byte \ Bit	7	6	5	4	3	2	1	0
0	(MSB)	PARAMETER CODE (0002h)						
1								(LSB)
2	Parameter control byte – binary format list log parameter (see SPC-6)							
	DU	Obsolete	TSD	Obsolete			FORMAT AND LINKING	
3	PARAMETER LENGTH (08h)							
4	(MSB)	TOTAL BLOCKS REASSIGNED DURING FORMAT						
•••								
11								(LSB)

The PARAMETER CODE field is described in SPC-6 and shall be set to the value shown in table 195 for the Total Blocks Reassigned During Format log parameter.

The DU bit, and the FORMAT AND LINKING field for the Total Blocks Reassigned During Format log parameter shall be set for a binary format list log parameters described in SPC-6.

The target save disable (TSD) bit (see SPC-6) shall be set to zero for the Total Blocks Reassigned During Format log parameter, indicating that the logical unit saves the Total Blocks Reassigned During Format log parameter at vendor specific intervals without any request from an application client.

The PARAMETER LENGTH field is described in SPC-6 and shall be set to the value shown in table 195 for the Total Blocks Reassigned During Format log parameter.

The TOTAL BLOCKS REASSIGNED DURING FORMAT field contains the count of the total number of logical blocks that were reassigned during the most recent successful format operation.

6.4.4.5 Total New Blocks Reassigned log parameter

The Total New Blocks Reassigned log parameter for the Format Status log page has the format shown in table 196.

Table 196 — Total New Blocks Reassigned log parameter format

Byte \ Bit	7	6	5	4	3	2	1	0
0	(MSB)	PARAMETER CODE (0003h)						
1								(LSB)
2	Parameter control byte – binary format list log parameter (see SPC-6)							
	DU	Obsolete	TSD	Obsolete			FORMAT AND LINKING	
3	PARAMETER LENGTH (08h)							
4	(MSB)	TOTAL NEW BLOCKS REASSIGNED						
•••								
11								(LSB)

The PARAMETER CODE field is described in SPC-6 and shall be set to the value shown in table 196 for the Total New Blocks Reassigned log parameter.

The DU bit, and the FORMAT AND LINKING field for the Total New Blocks Reassigned log parameter shall be set for a binary format list log parameters described in SPC-6.

The target save disable (TSD) bit (see SPC-6) shall be set to zero for the Total New Blocks Reassigned log parameter, indicating that the logical unit saves the Total New Blocks Reassigned log parameter at vendor specific intervals without any request from an application client.

The PARAMETER LENGTH field is described in SPC-6 and shall be set to the value shown in table 196 for the Total New Blocks Reassigned log parameter.

The TOTAL NEW BLOCKS REASSIGNED field contains a count of the total number of logical blocks that have been reassigned since the completion of the most recent successful format operation.

6.4.4.6 Power On Minutes Since Format log parameter

The Power On Minutes Since Format log parameter for the Format Status log page has the format shown in table 197.

Table 197 — Power On Minutes Since Format log parameter format

Byte \ Bit	7	6	5	4	3	2	1	0
0	(MSB)	PARAMETER CODE (0004h)						
1								(LSB)
2	Parameter control byte – binary format list log parameter (see SPC-6)							
	DU	Obsolete	TSD	Obsolete			FORMAT AND LINKING	
3	PARAMETER LENGTH (04h)							
4	(MSB)	POWER ON MINUTES SINCE FORMAT						
•••								
7								(LSB)

The PARAMETER CODE field is described in SPC-6 and shall be set to the value shown in table 197 for the Power On Minutes Since Format log parameter.

The DU bit, and the FORMAT AND LINKING field for the Power On Minutes Since Format log parameter shall be set for a binary format list log parameter as described in SPC-6.

The target save disable (TSD) bit (see SPC-6) shall be set to zero for the Power On Minutes Since Format log parameter, indicating that the logical unit saves the Power On Minutes Since Format log parameter at vendor specific intervals without any request from an application client.

The PARAMETER LENGTH field is described in SPC-6 and shall be set to the value shown in table 197 for the Power On Minutes Since Format log parameter.

The POWER ON MINUTES SINCE FORMAT field contains the unsigned number of usage minutes (i.e., minutes with power applied regardless of power state) that have elapsed since the most recent successful format operation.

6.4.5 Logical Block Provisioning log page

6.4.5.1 Logical Block Provisioning log page overview

Using the format shown in table 199, the Logical Block Provisioning log page reports the logical block provisioning status of the logical unit. The parameter codes for the Logical Block Provisioning log page are listed in table 198.

Table 198 — Logical Block Provisioning log parameters

Parameter code [a]	Description	Resettable or Changeable [b]	Reference	Support
Resources that are associated with thresholds (0000h to 00FFh)				
0000h	Reserved			
0001h	Available LBA Mapping Resource Count	Never	6.4.5.2	Optional [c]
0002h	Used LBA Mapping Resource Count	Never	6.4.5.3	
0003h	Available Provisioning Resource Percentage	Never	6.4.5.4	
0004h to 00FFh	Reserved			
Resources that are not associated with thresholds (0000h to 00FFh)				
0100h	De-duplicated LBA Resource Count	Never	6.4.5.5	Optional
0101h	Compressed LBA Resource Count	Never	6.4.5.6	
0102h	Total Efficiency LBA Resource Count	Never	6.4.5.7	
0103h to FFEFh	Reserved			
FFF0h to FFFFh	Vendor specific			

[a] Parameter codes 0000h to 00FFh are coordinated with the THRESHOLD RESOURCE field (see 6.5.9).
[b] The keywords in this column – Always, Reset Only, and Never – are defined in 6.4.1.2.
[c] If this log page is supported, then at least one Logical Block Provisioning log parameter shall be supported. A Logical Block Provisioning log parameter in the range 0001h to 00FFh should be provided to report resource usage for each threshold resource for which a threshold descriptor in the Logical Block Provisioning mode page (see 6.5.9) is available.

The Logical Block Provisioning log page has the format shown in table 199.

Table 199 — Logical Block Provisioning log page

Byte \ Bit	7	6	5	4	3	2	1	0
0	DS (1b)	SPF (0b)	PAGE CODE (0Ch)					
1	SUBPAGE CODE (00h)							
2	(MSB)	PAGE LENGTH (n - 3)						
3								(LSB)
Logical block provisioning parameter list								
4	Logical block provisioning log parameter [first]							
•••								
⋮								
•••	Logical block provisioning log parameter [last]							
n								

The disable save (DS) bit, the subpage format (SPF) bit, the PAGE CODE field, the SUBPAGE CODE field, and the PAGE LENGTH field are described in SPC-6.

The DS bit, the SPF bit, the PAGE CODE field, and the SUBPAGE CODE field shall be set to the values shown in table 199 for the Logical Block Provisioning log page.

6.4.5.2 Available LBA Mapping Resource Count log parameter

6.4.5.2.1 Available LBA Mapping Resource Count log parameter overview

The Available LBA Mapping Resource Count log parameter of the Logical Block Provisioning log page has the format shown in table 200.

Table 200 — Available LBA Mapping Resource Count log parameter format

Byte \ Bit	7	6	5	4	3	2	1	0
0	(MSB)	PARAMETER CODE (0001h)						
1								(LSB)
2	Parameter control byte – binary format list log parameter (see SPC-6)							
	DU	Obsolete	TSD	Obsolete			FORMAT AND LINKING	
3	PARAMETER LENGTH (08h)							
4	(MSB)	RESOURCE COUNT						
•••								
7								(LSB)
8	Reserved						SCOPE	
9	Reserved							
•••								
11								

The PARAMETER CODE field is described in SPC-6 and shall be set to the value shown in table 200 for the Available LBA Mapping Resource Count log parameter.

The DU bit, the TSD bit, and the FORMAT AND LINKING field for the Available LBA Mapping Resource Count shall be set for a binary format list log parameter as described in SPC-6.

The PARAMETER LENGTH field is described in SPC-6 and shall be set to the value shown in table 200 for the Available LBA Mapping Resource Count.

The RESOURCE COUNT field indicates an estimate of the number of available LBA mapping resources and is defined in 6.4.5.2.2.

The SCOPE field indicates the scope to which the RESOURCE COUNT field applies and is shown in table 201.

Table 201 — SCOPE field

Code	Description
00b	The scope of the resource count is not reported.
01b	The RESOURCE COUNT field indicates a resource that is dedicated to the logical unit. Usage of resources on other logical units does not impact the resource count.
10b	The RESOURCE COUNT field indicates resources that may or may not be dedicated to any logical unit including the addressed logical unit. Usage of resources on other logical units may impact the resource count.
11b	Reserved

6.4.5.2.2 RESOURCE COUNT field

The RESOURCE COUNT field indicates an estimate of the number of LBA resources expressed as a number of threshold sets for the threshold resource indicated by the parameter code value. The nominal number of LBA resources is calculated as follows:

$$\text{LBA resources} = \text{resource count} \times \text{threshold set size}$$

where:

resource count — is the value in the RESOURCE COUNT field; and

threshold set size — is the number of LBAs in each threshold set (i.e., $2^{(\text{threshold exponent})}$ LBAs, where the threshold exponent is indicated in the Logical Block Provisioning VPD page (see 6.6.9)).

6.4.5.3 Used LBA Mapping Resource Count log parameter

The Used LBA Mapping Resource Count log parameter of the Logical Block Provisioning log page has the format shown in table 202.

Table 202 — Used LBA Mapping Resource Count log parameter format

Byte \ Bit	7	6	5	4	3	2	1	0
0	(MSB)	PARAMETER CODE (0002h)						
1								(LSB)
2	Parameter control byte – binary format list log parameter (see SPC-6)							
	DU	Obsolete	TSD	Obsolete			FORMAT AND LINKING	
3	PARAMETER LENGTH (08h)							
4	(MSB)	RESOURCE COUNT						
•••								
7								(LSB)
8	Reserved						SCOPE	
9	Reserved							
•••								
11								

The PARAMETER CODE field is described in SPC-6 and shall be set to the value shown in table 202 for the Used LBA Mapping Resource Count log parameter.

The DU bit, the TSD bit, and the FORMAT AND LINKING field for the Used LBA Mapping Resource Count log parameter shall be set for a binary format list log parameter as described in SPC-6.

The PARAMETER LENGTH field is described in SPC-6 and shall be set to the value shown in table 202 for the Used LBA Mapping Resource Count log parameter.

The RESOURCE COUNT field indicates an estimate of the number of used LBA mapping resources and is defined in 6.4.5.2.2.

The SCOPE field indicates the scope to which the RESOURCE COUNT field applies and is shown in table 201.

6.4.5.4 Available Provisioning Resource Percentage log parameter

6.4.5.4.1 Available Provisioning Resource Percentage log parameter overview

The Available Provisioning Resource Percentage log parameter of the Logical Block Provisioning log page has the format shown in table 203

Table 203 — Available Provisioning Resource Percentage log parameter format

Byte \ Bit	7	6	5	4	3	2	1	0
0	(MSB)	PARAMETER CODE (0003h)						
1								(LSB)
2	Parameter control byte – binary format list log parameter (see SPC-6)							
	DU	Obsolete	TSD	Obsolete			FORMAT AND LINKING	
3	PARAMETER LENGTH (08h)							
4	(MSB)	RESOURCE COUNT						
5								(LSB)
6	Reserved							
7								
8	Reserved						SCOPE	
9	Reserved							
•••								
11								

The PARAMETER CODE field is described in SPC-6 and shall be set to the value shown in table 203 for the Available Provisioning Resource Percentage log parameter.

The DU bit, the TSD bit, and the FORMAT AND LINKING field for the Available Provisioning Resource Percentage log parameter shall be set for a binary format list log parameter as described in SPC-6.

The PARAMETER LENGTH field is described in SPC-6 and shall be set to the value shown in table 203 for the Available Provisioning Resource Percentage.

The RESOURCE COUNT field indicates an estimate of the percentage of available provisioning resources used for this logical block provisioning threshold and is defined in 6.4.5.4.2.

The SCOPE field indicates the scope to which the RESOURCE COUNT field applies and is shown in table 201.

6.4.5.4.2 RESOURCE COUNT field

The RESOURCE COUNT field (see table 204) contains an estimate of the percentage of resources available for allocation to LBAs as a percentage of the manufacturer's total resources available for allocation. The units for the reported values are percent and range from 0% to 100%.

Table 204 — RESOURCE COUNT field

Code	Description
0 to 100	0% to 100% of the provisioning resources of the logical unit are available
All others	Reserved

6.4.5.5 De-duplicated LBA Resource Count log parameter

The De-duplicated LBA Resource Count log parameter of the Logical Block Provisioning log page (see table 205) contains information about de-duplicated LBA resources.

Table 205 — De-duplicated LBA Resource Count log parameter format

Byte \ Bit	7	6	5	4	3	2	1	0
0	(MSB)	PARAMETER CODE (0100h)						
1								(LSB)
2	Parameter control byte – binary format list log parameter (see SPC-6)							
	DU	Obsolete	TSD	Obsolete			FORMAT AND LINKING	
3	PARAMETER LENGTH (08h)							
4	(MSB)	RESOURCE COUNT						
•••								
7								(LSB)
8	Reserved						SCOPE	
9	Reserved							
•••								
11								

The PARAMETER CODE field is described in SPC-6 and shall be set to the value shown in table 205 for the De-duplicated LBA Resource Count log parameter.

The DU bit, the TSD bit, and the FORMAT AND LINKING field for the De-duplicated LBA Resource Count log parameter shall be set for a binary format list log parameter as described in SPC-6.

The PARAMETER LENGTH field is described in SPC-6 and shall be set to the value shown in table 205 for the De-duplicated LBA Resource Count log parameter.

The RESOURCE COUNT field indicates an estimate of the number of LBA resources made available as a result of de-duplication and is defined in 6.4.5.2.2.

The SCOPE field indicates the scope to which the RESOURCE COUNT field applies and is shown in table 201.

6.4.5.6 Compressed LBA Resource Count log parameter

The Compressed LBA Resource Count log parameter of the Logical Block Provisioning log page (see table 206) contains information about compressed LBA resources.

Table 206 — Compressed LBA Resource Count log parameter format

Byte \ Bit	7	6	5	4	3	2	1	0
0	(MSB)	PARAMETER CODE (0101h)						
1								(LSB)
2	Parameter control byte – binary format list log parameter (see SPC-6)							
	DU	Obsolete	TSD	Obsolete			FORMAT AND LINKING	
3	PARAMETER LENGTH (08h)							
4	(MSB)	RESOURCE COUNT						
•••								
7								(LSB)
8	Reserved						SCOPE	
9	Reserved							
•••								
11								

The PARAMETER CODE field is described in SPC-6 and shall be set to the value shown in table 206 for the Compressed LBA Resource Count log parameter.

The DU bit, the TSD bit, and the FORMAT AND LINKING field for the Compressed LBA Resource Count log parameter shall be set for a binary format list log parameter as described in SPC-6.

The PARAMETER LENGTH field is described in SPC-6 and shall be set to the value shown in table 206 for the Compressed LBA Resource Count log parameter.

The RESOURCE COUNT field indicates an estimate of the number of LBA resources made available as a result of compression and is defined in 6.4.5.2.2.

The SCOPE field indicates the scope to which the RESOURCE COUNT field applies and is shown in table 201.

6.4.5.7 Total Efficiency LBA Resource Count log parameter

The Total Efficiency LBA Resource Count log parameter of the Logical Block Provisioning log page (see table 207) contains information about the combined effects of all LBA resource efficiencies (e.g., the result of the combination of de-duplicated LBA resources and compressed LBA resources).

Table 207 — Total Efficiency LBA Resource Count log parameter format

Byte \ Bit	7	6	5	4	3	2	1	0
0	(MSB)	PARAMETER CODE (0102h)						
1								(LSB)
2	Parameter control byte – binary format list log parameter (see SPC-6)							
	DU	Obsolete	TSD	Obsolete			FORMAT AND LINKING	
3	PARAMETER LENGTH (08h)							
4	(MSB)	RESOURCE COUNT						
•••								
7								(LSB)
8	Reserved						SCOPE	
9	Reserved							
•••								
11								

The PARAMETER CODE field is described in SPC-6 and shall be set to the value shown in table 207 for the Total Efficiency LBA Resource Count log parameter.

The DU bit, the TSD bit, and the FORMAT AND LINKING field for the Total Efficiency LBA Resource Count log parameter shall be set for a binary format list log parameter as described in SPC-6.

The PARAMETER LENGTH field is described in SPC-6 and shall be set to the value shown in table 207 for the Total Efficiency LBA Resource Count log parameter.

The RESOURCE COUNT field indicates an estimate of the number of LBA resources made available by the combined effects of all LBA resource efficiency methods (e.g., de-duplication and compression) and is defined in 6.4.5.2.2. The algorithm used to calculate this value is not defined by this standard.

The SCOPE field indicates the scope to which the RESOURCE COUNT field applies and is shown in table 201.

6.4.6 LPS Misalignment log page

6.4.6.1 Overview

Using the format shown in table 209, the LPS Misalignment log page reports misaligned write command information (see 4.6.2). The parameter codes for the LPS Misalignment log page are listed in table 208.

Table 208 — LPS Misalignment log page parameter codes

Parameter code	Description	Resettable or Changeable [a]	Reference	Support
0000h	LPS Misalignment Count	Reset Only	6.4.6.2	Mandatory
0001h to F000h	LPS Misalignment	Reset Only	6.4.6.3	Optional [b]
All others	Reserved			

[a] The keywords in this column – Always, Reset Only, and Never – are defined in 6.4.1.2.
[b] If the LPS Misalignment log page is supported, then at least one LPS Misalignment log parameter shall be supported.

The LPS Misalignment log page has the format shown in table 209

Table 209 — LPS Misalignment log page

Byte \ Bit	7	6	5	4	3	2	1	0
0	DS	SPF (1b)	PAGE CODE (15h)					
1	SUBPAGE CODE (03h)							
2	(MSB)	PAGE LENGTH (n - 3)						
3								(LSB)
	LPS misalignment log parameters							
4 •••	LPS misalignment log parameter [first] (see table 210 and table 211)							
⋮								
••• n	LPS misalignment log parameter [last] (see table 210 and table 211)							

The disable save (DS) bit, the subpage format (SPF) bit, the PAGE CODE field, and the SUBPAGE CODE field are described in SPC-6 and shall be set to the values shown in table 209 for the LPS Misalignment log page.

The PAGE LENGTH field is defined in SPC-6.

The contents of each LPS misalignment log parameter depends on the value in the PARAMETER CODE field (see table 208).

6.4.6.2 LPS Misalignment Count log parameter

The LPS Misalignment Count log parameter has the format shown in table 210 and indicates the number of LPS Misalignment log parameters that are available.

Table 210 — LPS Misalignment Count log parameter format

Byte \ Bit	7	6	5	4	3	2	1	0
0	(MSB)	PARAMETER CODE (0000h)						
1								(LSB)
2	Parameter control byte – binary format list log parameter (see SPC-6)							
	DU	Obsolete	TSD	Obsolete			FORMAT AND LINKING	
3	PARAMETER LENGTH (04h)							
4	(MSB)	MAX_LPSM						
5								(LSB)
6	(MSB)	LPS MISALIGNMENT COUNT						
7								(LSB)

The PARAMETER CODE field is described in SPC-6 and shall be set as shown in table 210 for the LPS Misalignment Count log parameter.

The DU bit, the TSD bit, and the FORMAT AND LINKING field for the LPS Misalignment Count log parameter shall be set for a binary format list log parameter as described in SPC-6.

The PARAMETER LENGTH field is described in SPC-6 and shall be set to the value shown in table 210 for the LPS Misalignment Count log parameter.

The MAX_LPSM field indicates the maximum number of LPS Misalignment log parameters (see 6.4.6.3) supported by the device server. The device server may support any number of LPS Misalignment log parameters from 0001h to F000h inclusive.

The LPS MISALIGNMENT COUNT field indicates the number of LPS Misalignment log parameters that are available.

6.4.6.3 LPS Misalignment log parameter

An LPS Misalignment log parameter has the format shown in table 211. If no misaligned write commands have been processed since the most recent reset of the LPS Misalignment log page then no LPS

Misalignment log parameters shall be present. LPS Misalignment log parameters are added as described in 4.6.2.

Table 211 — LPS Misalignment log parameter format

Byte \ Bit	7	6	5	4	3	2	1	0
0	(MSB)	PARAMETER CODE (0001h to F000h)						
1								(LSB)
2	Parameter control byte – binary format list log parameter (see SPC-6)							
	DU	Obsolete	TSD	Obsolete			FORMAT AND LINKING	
3	PARAMETER LENGTH (08h)							
4	(MSB)	LBA OF MISALIGNED BLOCK						
•••								
11								(LSB)

The PARAMETER CODE field is described in SPC-6. A PARAMETER CODE field set to 0001h indicates the oldest misaligned write command reported by the log, and successive values indicate successive misaligned writes.

The DU bit, the TSD bit, and the FORMAT AND LINKING field for a LPS Misalignment log parameter shall be set for a binary format list log parameter as described in SPC-6.

The PARAMETER LENGTH field is defined in SPC-6 and shall be set to the value shown in table 211 for a LPS Misalignment log parameter.

The LBA OF MISALIGNED BLOCK field indicates the starting LBA associated with the misaligned write command.

6.4.7 Non-volatile Cache log page

6.4.7.1 Non-volatile Cache log page overview

Using the format shown in table 213, the Nonvolatile Cache log page reports the status of battery backup for a nonvolatile cache. The parameter codes for the Nonvolatile Cache log page are listed in table 212.

Table 212 — Nonvolatile Cache log parameters

Parameter code	Description	Resettable or Changeable [a]	Reference	Support
0000h	Remaining Nonvolatile Time	Never	6.4.7.2	Mandatory
0001h	Maximum Nonvolatile Time	Never	6.4.7.3	Mandatory
All others	Reserved			

[a] The keywords in this column – Always, Reset Only, and Never – are defined in 6.4.1.2.

The Nonvolatile Cache log page has the format shown in table 213.

Table 213 — Nonvolatile Cache log page

Byte \ Bit	7	6	5	4	3	2	1	0
0	DS	SPF (0b)	PAGE CODE (17h)					
1	SUBPAGE CODE (00h)							
2	(MSB)	PAGE LENGTH (n - 3)						
3								(LSB)
	Nonvolatile cache log parameters							
4 •••	Non-volatile cache log parameter [first] (see table 212)							
⋮								
••• n	Nonvolatile cache log parameter [last] (see table 212)							

The disable save (DS) bit, the subpage format (SPF) bit, the PAGE CODE field, the SUBPAGE CODE field, and the PAGE LENGTH field are described in SPC-6.

The SPF bit, the PAGE CODE field, and the SUBPAGE CODE field shall be set to the values shown in table 213 for the Nonvolatile Cache log page.

6.4.7.2 Remaining Nonvolatile Time log parameter

The Remaining Nonvolatile Time log parameter of the Nonvolatile Cache log page has the format shown in table 214.

Table 214 — Remaining Nonvolatile Time log parameter format

Byte \ Bit	7	6	5	4	3	2	1	0
0	(MSB)	PARAMETER CODE (0000h)						
1								(LSB)
2	Parameter control byte – binary format list log parameter (see SPC-6)							
	DU	Obsolete	TSD	Obsolete			FORMAT AND LINKING	
3	PARAMETER LENGTH (04h)							
4	Obsolete							
5	(MSB)	REMAINING NONVOLATILE TIME						
•••								
7								(LSB)

The PARAMETER CODE field is described in SPC-6 and shall be set to the value shown in table 214 for the Remaining Nonvolatile Time log parameter.

The DU bit, the TSD bit, and the FORMAT AND LINKING field for the Remaining Nonvolatile Time log parameter shall be set for a binary format list log parameter as described in SPC-6.

The PARAMETER LENGTH field is described in SPC-6 and shall be set to the value shown in table 214 for the Remaining Nonvolatile Time log parameter.

The REMAINING NONVOLATILE TIME field is shown in table 215.

Table 215 — REMAINING NONVOLATILE TIME field

Code	Description
00_0000h	Nonvolatile cache is volatile, either permanently or temporarily (e.g., if batteries require recharging).
00_0001h	Nonvolatile cache is expected to remain nonvolatile for an unknown amount of time (e.g., if battery status is unknown)
00_0002h to FF_FFFEh	Nonvolatile cache is expected to remain nonvolatile for the number of minutes indicated (e.g., for the life of the battery supplying power to random access memory).
FF_FFFFh	Nonvolatile cache is indefinitely nonvolatile.

6.4.7.3 Maximum Nonvolatile Time log parameter

The Maximum Nonvolatile Time log parameter of the Nonvolatile Cache log page has the format shown in table 216.

Table 216 — Maximum Nonvolatile Time log parameter format

Byte \ Bit	7	6	5	4	3	2	1	0
0	(MSB)	PARAMETER CODE (0001h)						
1								(LSB)
2	Parameter control byte – binary format list log parameter (see SPC-6)							
	DU	Obsolete	TSD	Obsolete			FORMAT AND LINKING	
3	PARAMETER LENGTH (04h)							
4	Obsolete							
5	(MSB)	MAXIMUM NONVOLATILE TIME						
•••								
7								(LSB)

The PARAMETER CODE field is described in SPC-6 and shall be set to the value shown in table 216 for the Maximum Nonvolatile Time log parameter.

The DU bit, the TSD bit, and the FORMAT AND LINKING field for the Maximum Nonvolatile Time log parameter shall be set for a binary format list log parameter as described in SPC-6.

The PARAMETER LENGTH field is described in SPC-6 and shall be set to the value shown in table 216 for the Maximum Nonvolatile Time log parameter.

The MAXIMUM NONVOLATILE TIME field is shown in table 217.

Table 217 — MAXIMUM NONVOLATILE TIME field

Code	Description
00_0000h	Nonvolatile cache is volatile
00_0001h	Reserved
00_0002h to FF_FFFEh	Nonvolatile cache is capable of being nonvolatile for the estimated number of minutes indicated. If the time is based on batteries, then the time shall be based on the last full charge capacity rather than the design capacity of the batteries.
FF_FFFFh	Nonvolatile cache is indefinitely nonvolatile.

6.4.8 Pending Defects log page

6.4.8.1 Overview

Using the format shown in table 219, the Pending Defects log page reports an unsorted list of logical blocks for which the device server has detected an unrecovered medium error. The parameter codes for the Pending Defects log page are listed in table 218.

Table 218 — Pending Defects log page parameter codes

Parameter code	Description	Resettable or Changeable [a]	Reference	Support
0000h	Pending Defect Count	Never	6.4.8.2	Mandatory
0001h to F000h	Pending Defect	Never	6.4.8.3	Optional [b]
All others	Reserved			

[a] The keywords in this column – Always, Reset Only, and Never – are defined in 6.4.1.2.
[b] If the Pending Defects log page is supported, then at least one Pending Defect log parameter shall be supported.

The Pending Defects log page has the format shown in table 219

Table 219 — Pending Defects log page

Byte \ Bit	7	6	5	4	3	2	1	0
0	DS	SPF (1b)	PAGE CODE (15h)					
1	SUBPAGE CODE (01h)							
2	(MSB)	PAGE LENGTH (n-3)						
3								(LSB)
	Pending defect parameters							
4	Pending defect parameter [first]							
•••								
	⋮							
	Pending defect parameter [last]							
•••								
n								

The subpage format (SPF) bit, the PAGE CODE field, and the SUBPAGE CODE field are described in SPC-6 and shall be set to the values shown in table 219 for the Pending Defects log page.

The disable save (DS) bit, and the PAGE LENGTH field are described in SPC-6.

The contents of each pending defect parameter depends on the value in its PARAMETER CODE field (see table 218).

6.4.8.2 Pending Defect Count log parameter

The Pending Defect Count log parameter has the format shown in table 220 and indicates the number of Pending Defect log parameters that are available.

Table 220 — Pending Defect Count log parameter format

Byte \ Bit	7	6	5	4	3	2	1	0
0	(MSB)	PARAMETER CODE (0000h)						
1								(LSB)
2	Parameter control byte – binary format list log parameter (see SPC-6)							
	DU	Obsolete	TSD	Obsolete			FORMAT AND LINKING	
3	PARAMETER LENGTH (04h)							
4	(MSB)	PENDING DEFECT COUNT						
•••								
7								(LSB)

The PARAMETER CODE field is described in SPC-6 and shall be set as shown in table 220 for the Pending Defect Count log parameter.

The DU bit, the TSD bit, and the FORMAT AND LINKING field for the Pending Defect Count log parameter shall be set for a binary format list log parameter as described in SPC-6.

The PARAMETER LENGTH field is described in SPC-6 and shall be set to the value shown in table 220 for the Pending Defect Count log parameter.

The PENDING DEFECT COUNT field indicates the number of Pending Defect log parameters that are available.

6.4.8.3 Pending Defect log parameter

A Pending Defect log parameter has the format shown in table 221. If no unrecovered medium errors have occurred then no Pending Defect log parameters shall be present. A Pending Defect log parameter shall be added for each LBA for which the device server has detected an unrecovered medium error that is not:

a) a pseudo unrecovered read error (see 4.18.2);
b) a predicted unrecovered read error (see 4.19.3.3); or
c) a predicted unrecovered write error (see 4.19.3.5).

If all of the supported parameter code values have been used and a new defect is discovered, then the device server shall not add more Pending Defect log parameters and the PENDING ERROR COUNT field shall not be changed.

Pending Defect log parameters may duplicate information that is in Background Scan Results log parameters in the Background Scan log page (see 6.4.2).

A Pending Defect log parameter shall be removed if the indicated LBA:

a) is reassigned without error;
b) is written without error; or
c) is read without error.

A Pending Defect log parameter may be removed if the indicated LBA is unmapped without error.

A sanitize overwrite operation (see 5.30.2.2) and a format operation (see 5.4.1) shall cause all Pending Defect log parameters to be removed..

Table 221 — Pending Defect log parameter format

Byte \ Bit	7	6	5	4	3	2	1	0
0	(MSB)	PARAMETER CODE (0001h to F000h)						
1								(LSB)
2	Parameter control byte – binary format list log parameter (see SPC-6)							
	DU	Obsolete	TSD	Obsolete			FORMAT AND LINKING	
3	PARAMETER LENGTH (0Ch)							
4	(MSB)	ACCUMULATED POWER ON HOURS						
•••								
7								(LSB)
8	(MSB)	LOGICAL BLOCK ADDRESS						
•••								
15								(LSB)

The PARAMETER CODE field is described in SPC-6 and shall be set as shown in table 221 for a Pending Defect log parameter.

The DU bit, the TSD bit, and the FORMAT AND LINKING field for a Pending Defect log parameter shall be set for a binary format list log parameter as described in SPC-6.

The PARAMETER LENGTH field is defined in SPC-6 and shall be set to the value shown in table 221 for a Pending Defect log parameter.

The ACCUMULATED POWER ON HOURS field indicates the number of hours that the device server has been powered on since manufacturing at the time the Pending Defect log parameter was created. A value of FFFF_FFFFh indicates that the accumulated power on hours value is unknown.

The LOGICAL BLOCK ADDRESS field indicates the LBA associated with the unrecovered medium error.

6.4.9 Solid State Media log page

6.4.9.1 Solid State Media log page overview

Using the format shown in table 222, the Solid State media log page reports parameters that are specific to SCSI target devices that contain solid state media.

Table 222 — Solid State Media log page

Byte \ Bit	7	6	5	4	3	2	1	0
0	DS	SPF (0b)	PAGE CODE (11h)					
1	SUBPAGE CODE (00h)							
2	(MSB)	PAGE LENGTH (n - 3)						
3								(LSB)
	Solid state media log parameters							
4 ••• 	Solid state media parameter [first]							
⋮								
 ••• n	Solid state media parameter [last]							

The disable save (DS) bit, the subpage format (SPF) bit, the PAGE CODE field, the SUBPAGE CODE field, and the PAGE LENGTH field are described in SPC-6.

The SPF bit, the PAGE CODE field, and the SUBPAGE CODE field shall be set to the values shown in table 222 for the Solid State Media log page.

The parameter codes for the Solid State Media log page are listed in table 223.

Table 223 — Solid State Media log parameters

Parameter code	Description	Resettable or Changeable [a]	Reference	Support
0001h	Percentage Used Endurance Indicator	Never	6.4.9.2	Mandatory
All others	Reserved			

[a] The keywords in this column – Always, Reset Only, and Never – are defined in 6.4.1.2.

6.4.9.2 Percentage Used Endurance Indicator log parameter

The Percentage Used Endurance Indicator log parameter of the Solid Sate Media log page has the format shown in table 224.

Table 224 — Percentage Used Endurance Indicator log parameter format

Byte \ Bit	7	6	5	4	3	2	1	0
0	(MSB)	PARAMETER CODE (0001h)						
1								(LSB)
2	Parameter control byte – binary format list log parameter (see SPC-6)							
	DU	Obsolete	TSD	Obsolete			FORMAT AND LINKING	
3	PARAMETER LENGTH (04h)							
4	Reserved							
•••								
6								
7	PERCENTAGE USED ENDURANCE INDICATOR							

The PARAMETER CODE field is described in SPC-6 and shall be set to the value shown in table 224 for the Percentage Used Endurance Indicator log parameter.

The DU bit, the TSD bit, and the FORMAT AND LINKING field for the Percentage Used Endurance Indicator log parameter shall be set for a binary format list log parameter as described in SPC-6.

The PARAMETER LENGTH field is described in SPC-6 and shall be set to the value shown in table 224 for the Percentage Used Endurance Indicator log parameter.

The PERCENTAGE USED ENDURANCE INDICATOR field indicates an estimate of the percentage of a SCSI target device that contains solid state media life that has been used. The value in the field shall be set to zero at the time of manufacture. A value of 100 indicates that the estimated endurance of the SCSI target device that contains solid state media has been consumed, but may not indicate the presence of a solid state media failure in that SCSI target device (e.g., the minimum power-off data retention capability has been reached for a SCSI target devices that contains solid state media while the media is still functional). The value is allowed to exceed 100. Values greater than 254 shall be reported as 255. The device server shall update the value at least once per power on hour.

6.4.10 Utilization log page

6.4.10.1 Utilization log page overview

Using the format shown in table 225, the Utilization log page reports estimates of the rate at which device wear factors (e.g., damage to the recording medium) are being used.

Table 225 — Utilization log page

Byte \ Bit	7	6	5	4	3	2	1	0
0	DS (1b)	SPF (1b)	PAGE CODE (0Eh)					
1	SUBPAGE CODE (01h)							
2	(MSB)	PAGE LENGTH (n - 3)						
3								(LSB)
	Utilization log parameters							
4 •••	Utilization log parameter [first]							
⋮	⋮							
••• n	Utilization log parameter [last]							

The disable save (DS) bit, the subpage format (SPF) bit, the PAGE CODE field, and the SUBPAGE CODE field are described in SPC-6 and shall be set to the values shown in table 225 for the Utilization log page.

The PAGE LENGTH field is described in SPC-6.

The parameter codes for the Utilization log page are shown in table 226.

Table 226 — Utilization log page parameter codes

Parameter code	Description	Resettable or Changeable [a]	Reference	Support
0000h	Workload Utilization	Never	6.4.10.2	Mandatory
0001h	Utilization Usage Rate Based on Date and Time	Never	6.4.10.3	Optional
All others	Reserved			

[a] The keywords in this column – Always, Reset Only, and Never – are defined in 6.4.1.2.

6.4.10.2 Workload Utilization log parameter

The Workload Utilization log parameter for the Utilization log page has the format shown in table 227.

Table 227 — Workload Utilization log parameter format

Byte \ Bit	7	6	5	4	3	2	1	0
0	(MSB)	PARAMETER CODE (0000h)						
1								(LSB)
2	Parameter control byte – bounded data counter log parameter (see SPC-6)							
	DU	Obsolete	TSD	Obsolete			FORMAT AND LINKING	
3	PARAMETER LENGTH (02h)							
4	(MSB)	WORKLOAD UTILIZATION						
5								(LSB)

The PARAMETER CODE field is described in SPC-6 and shall be set to the value shown in table 227 for the Workload Utilization log parameter.

The DU bit, the TSD bit, and the FORMAT AND LINKING field for the Workload Utilization log parameter shall be set for a bounded data counter log parameter as described in SPC-6.

The PARAMETER LENGTH field is described in SPC-6 and shall be set to the value shown in table 227 for the Workload Utilization log parameter.

The WORKLOAD UTILIZATION field (see table 228) contains an estimate of the utilization associated with the logical unit as a percentage of the manufacturer's designs for various wear factors (e.g., wear of the medium, head load events), if any. The units for the reported values are percent times 100 and range from 0.00% to 655.35%.

Table 228 — WORKLOAD UTILIZATION field

Code	Description
0 to 9 999	Less than (i.e., 0.00% to 99.99% of) the designed workload has been utilized.
10 000	Exactly the designed workload for the device has been utilized.
10 001 to 65 534	Greater than (i.e., 100.01% to 655.34% of) the designed workload has been utilized.
65 535	Greater than 655.34% of the designed workload has been utilized.

6.4.10.3 Utilization Usage Rate Based on Date and Time

The Utilization Rate Based on Date and Time log parameter for the Utilization log page has the format shown in table 229. If the interval that begins at the date and time of manufacture and ends at the timestamp that is reported by a REPORT TIMESTAMP command is not able to be determined (e.g.,the current date and time has not been initialized by a SET TIMESTAMP command (see SPC-6) or the current date and time is prior to the date and time of manufacture), then the Utilization Rate Based on Date and Time log parameter shall not be returned in the Utilization log page.

Table 229 — Utilization Rate Based on Date and Time log parameter format

Byte \ Bit	7	6	5	4	3	2	1	0
0	(MSB)	PARAMETER CODE (0001h)						
1								(LSB)
2	Parameter control byte – bounded data counter log parameter (see SPC-6)							
	DU	Obsolete	TSD	Obsolete			FORMAT AND LINKING	
3	PARAMETER LENGTH (02h)							
4	DATE AND TIME BASED UTILIZATION RATE							
5	Reserved							

The PARAMETER CODE field is described in SPC-6 and shall be set to the value shown in table 229 for the Utilization Rate Based on Date and Time log parameter.

The DU bit, the TSD bit, and the FORMAT AND LINKING field for the Utilization Rate Based on Date and Time log parameter shall be set for a bounded data counter log parameter as described in SPC-6.

The PARAMETER LENGTH field is described in SPC-6 and shall be set to the value shown in table 229 for the Utilization Rate Based on Date and Time log parameter.

The DATE AND TIME BASED UTILIZATION RATE field (see table 230) contains an estimate of the rate at which device wear factors (e.g., damage to the recording medium) associated with the logical unit have been used during the interval that begins at the date and time of manufacture and ends at the timestamp that is reported by a REPORT TIMESTAMP command (i.e., the current value of a device clock) (see SPC-6).

Table 230 — DATE AND TIME BASED UTILIZATION RATE field

Code	Description
0 to 99	The Workload Utilization usage rate has been less than (i.e., 0% to 99% of) the designed usage rate during the interval that begins at the date and time of manufacture and ends at the timestamp.
100	The Workload Utilization usage rate has been the exact designed usage rate during the interval that begins at the date and time of manufacture and ends at the timestamp.
101 to 254	The Workload Utilization usage rate has been greater than (i.e., 101% to 254% of) the designed usage rate during the interval that begins at the date and time of manufacture and ends at the timestamp.
255	The Workload Utilization usage rate has been greater than 254% of designed usage rate during the interval that begins at the date and time of manufacture and ends at the timestamp.

6.5 Mode parameters

6.5.1 Mode pages overview

The mode pages and their corresponding page codes and subpage codes for direct access block devices are shown in table 231. See 6.5.2 for a description of block descriptors.

Table 231 — Mode page codes and subpage codes for direct access block devices (part 1 of 2)

Mode page name	Page code	Subpage code	Reference
Application Tag	0Ah	02h	6.5.3
ATA Feature Control	0Ah	F2H	SAT-4
ATA Power Condition	1Ah	F1h	SAT-4
Background Control	1Ch	01h	6.5.4
Background Operation Control	0Ah	06h	6.5.5
Caching	08h	00h	6.5.6
Command Duration Limit A	0Ah	03h	SPC-6
Command Duration Limit B	0Ah	04h	SPC-6
Command Duration Limit T2A	0Ah	07h	SPC-6
Command Duration Limit T2B	0Ah	08h	SPC-6
Control	0Ah	00h	SPC-6
Control Extension	0Ah	01h	SPC-6
Disconnect-Reconnect	02h	00h	SPC-6
Enclosure Services Management [a]	14h	00h	SES-3
IO Advice Hints Grouping	0Ah	05h	6.5.7
Informational Exceptions Control	1Ch	00h	6.5.8
Logical Block Provisioning	1Ch	02h	6.5.9
PATA Control	0Ah	F1h	SAT-4
Power Condition	1Ah	00h	SPC-6
Power Consumption	1Ah	01h	SPC-6
Protocol-Specific Logical Unit	18h	00h	SPC-6
Protocol-Specific Port	19h	00h	SPC-6
Read-Write Error Recovery	01h	00h	6.5.10
Return all mode pages and subpages [b]	3Fh	FFh	SPC-6
Return all mode pages not including subpages [b]	3Fh	00h	SPC-6
Note: SPC-6 contains a listing of mode page and subpage codes in numeric order.			

[a] Valid only if the ENCSERV bit is set to one in the standard INQUIRY data (see SPC-6).
[b] Valid only for the MODE SENSE command.
[c] All subpage codes of the following mode page codes are obsolete: 03h, 04h, 05h, 09h, 0Bh, 0Ch, 0Dh, and 10h.
[d] The following mode page code and subpage code combinations are vendor specific and do not require a page format:
a) mode page code 00h with subpage code 00h; and
b) mode page codes 20h to 3Eh with all subpage codes.

Table 231 — Mode page codes and subpage codes for direct access block devices (part 2 of 2)

Mode page name	Page code	Subpage code	Reference
Return all subpages for the specified mode page code [b]	00h to 3Eh	FFh	SPC-6
Verify Error Recovery	07h	00h	6.5.11
Zoned Block Device Control	0Ah	0Fh	ZBC-2
Obsolete [c]			
Vendor specific [d]			
Reserved	all other page and subpage code combinations for direct access block devices		
Note: SPC-6 contains a listing of mode page and subpage codes in numeric order.			

[a] Valid only if the ENCSERV bit is set to one in the standard INQUIRY data (see SPC-6).
[b] Valid only for the MODE SENSE command.
[c] All subpage codes of the following mode page codes are obsolete: 03h, 04h, 05h, 09h, 0Bh, 0Ch, 0Dh, and 10h.
[d] The following mode page code and subpage code combinations are vendor specific and do not require a page format:
a) mode page code 00h with subpage code 00h; and
b) mode page codes 20h to 3Eh with all subpage codes.

The mode parameter list, including the mode parameter header, is described in SPC-6. Direct access block devices support zero or one mode parameter block descriptors (i.e., the block descriptor is shared by all the logical blocks on the medium) (see 6.5.2).

The MEDIUM TYPE field in the mode parameter header (see SPC-6) shall be set to 00h.

The DEVICE-SPECIFIC PARAMETER field in the mode parameter header (see SPC-6) for direct access block devices is shown in table 232.

Table 232 — DEVICE-SPECIFIC PARAMETER field for direct access block devices

Bit	7	6	5	4	3	2	1	0
	WP	Reserved	CAPPID	DPOFUA	Reserved			

If the medium is write-protected (i.e., as a result of mechanisms outside the scope of this standard), or the software write protect (SWP) bit in the Control mode page (see SPC-6) is set to one, then the device server shall set the WP bit to one when returning a DEVICE-SPECIFIC PARAMETER field in response to a MODE SENSE command. If the medium is not write-protected and the SWP bit is set to zero, then the device server shall set the WP bit to zero when returning a DEVICE-SPECIFIC PARAMETER field in response to a MODE SENSE command.

The write protect (WP) bit for mode data sent with a MODE SELECT command shall be ignored by the device server.

For a MODE SENSE command, the device server shall set the capacity/product identification (CAPPID) bit to zero in the returned parameter data.

The device server shall terminate a MODE SELECT command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN PARAMETER LIST, if the mode parameter header sent with that MODE SELECT command specifies a CAPPID bit that is set to one and:

a) the Capacity/Product Identification Mapping VPD page (see 6.6.6) is not supported;
b) the BLOCK DESCRIPTOR LENGTH field (see SPC-6) is set to zero; or

c) the mode parameter block descriptor contains a LOGICAL BLOCK LENGTH field (see 6.5.2.2 and 6.5.2.3) that specifies a value that is different from the current logical block length.

If the device server processes a MODE SELECT command with a mode parameter list that includes one mode parameter block descriptor and the CAPPID bit is set to:

a) zero, then the device server shall process that mode parameter block descriptor as described in 6.5.2.2 and 6.5.2.3; and
b) one, then the device server shall process that mode parameter block descriptor as described in 6.5.2.2, 6.5.2.3, and this subclause (i.e., 6.5.1).

If:

a) the CAPPID bit is set to one;
b) the Capacity/Product Identification Mapping VPD page is supported; and
c) a mode parameter block descriptor:
   A) causes a candidate number of logical blocks to be determined as described in 6.5.2.2 and 6.5.2.3; and
   B) contains a LOGICAL BLOCK LENGTH field value that is the same as the current logical block length in the logical unit,

then:

1) the device server shall search the capacity/product identification descriptors in the Capacity/Product Identification Mapping VPD page for an ALLOWED NUMBER OF LOGICAL BLOCKS field that equals the candidate number of logical blocks; and
2) if an equal value is:
   A) found, then the device server shall:
      a) cause the candidate number of logical blocks to take effect;
      b) change the PRODUCT IDENTIFICATION field in the standard INQUIRY data to the value in the PRODUCT IDENTIFICATION field of the found capacity/product identification descriptor; and
      c) establish a unit attention condition for the SCSI initiator port associated with every I_T nexus with the additional sense code set to INQUIRY DATA HAS CHANGED;

      and
   B) not found, then the device server shall:
      a) not accept the candidate number of logical blocks value by:
         A) preventing the candidate number of logical blocks from taking effect;
         B) not changing the PRODUCT IDENTIFICATION field in the standard INQUIRY data; and
         C) terminating the MODE SELECT command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN PARAMETER LIST;

         or
      b) accept the candidate number of logical blocks value by:
         A) causing the candidate number of logical blocks to take effect;
         B) changing the PRODUCT IDENTIFICATION field in the standard INQUIRY data to a vendor specific value; and
         C) establishing a unit attention condition for the SCSI initiator port associated with every I_T nexus with the additional sense code set to INQUIRY DATA HAS CHANGED.

The DPOFUA bit is reserved for mode data sent with a MODE SELECT command.

If the device server does not support the DPO bit and the FUA bit (see 4.15) being set to one, then the device server shall set the DPO and FUA supported (DPOFUA) bit to zero when returning a DEVICE-SPECIFIC PARAMETER field in response to a MODE SENSE command. If the device server supports the DPO bit and the FUA bit being set to one, then the device server shall set the DPOFUA bit to one when returning a DEVICE-SPECIFIC PARAMETER field in response to a MODE SENSE command.

6.5.2 Mode parameter block descriptors

6.5.2.1 Mode parameter block descriptors overview

If a device server returns a mode parameter block descriptor, then the device server shall return a short LBA mode parameter block descriptor (see 6.5.2.2) in the mode parameter data in response to:a MODE SENSE (10) command with the LLBAA bit set to zero.

A device server should return a long LBA mode parameter block descriptor (see 6.5.2.3) in the mode parameter data in response to a MODE SENSE (10) command with the LLBAA bit set to one.

If an application client sends a mode parameter block descriptor in the mode parameter list, then the application client should send a long LBA mode parameter block descriptor (see 6.5.2.3) for a MODE SELECT (10) command.

Support for the mode parameter block descriptors is optional.

If the device server supports changing the block descriptor parameters by a MODE SELECT command, the number of logical blocks is changed, and the Application Tag mode page is supported, then the device server shall set:

a) the current value of the ATMPE bit to zero in the Control mode page (see SPC-6); and
b) the saved value, if saving is implemented, of the ATMPE bit to zero in the Control mode page.

If the device server supports changing the block descriptor parameters by a MODE SELECT command and the number of logical blocks or the logical block length is changed, then the device server establishes a unit attention condition of:

a) CAPACITY DATA HAS CHANGED as described in 4.10; and
b) MODE PARAMETERS CHANGED as described in SPC-6.

6.5.2.2 Short LBA mode parameter block descriptor

Table 233 defines the short LBA mode parameter block descriptor for direct access block 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-6).

Table 233 — Short LBA mode parameter block descriptor

Byte \ Bit	7	6	5	4	3	2	1	0
0	(MSB)							
•••	NUMBER OF LOGICAL BLOCKS							
3								(LSB)
4	Reserved							
5	(MSB)							
•••	LOGICAL BLOCK LENGTH							
7								(LSB)

A device server shall respond to a MODE SENSE command (see SPC-6) by reporting the number of logical blocks specified in the NUMBER OF LOGICAL 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 logical blocks shall be returned. To determine the number of logical blocks at which the logical unit is currently formatted, the application client shall use a READ CAPACITY command (see 5.20 and 5.21) rather than the MODE SENSE command.

In response to a MODE SENSE command, the device server may return a value of zero indicating that it does not report the number of logical blocks in the short LBA mode parameter block descriptor.

In response to a MODE SENSE command, if the number of logical blocks on the medium exceeds the maximum value that is able to be specified in the NUMBER OF LOGICAL BLOCKS field, then the device server shall return a value of FFFF_FFFFh.

The logical unit's maximum capacity is a vendor specific value that represents the maximum number of addressable logical blocks. If the Capacity/Product Identification Mapping VPD page (see 6.6.6) is supported, then that vendor specific value shall be less than (e.g., a reduced value due to depopulation) or equal to the value in the ALLOWED NUMBER OF LOGICAL BLOCKS field in the capacity/product identification descriptor with a value in the PRODUCT IDENTIFICATION field that equals the value of the PRODUCT IDENTIFICATION field in the standard Inquiry data.

If the logical unit does not support changing its capacity by changing the NUMBER OF LOGICAL BLOCKS field using the MODE SELECT command ~~(see SPC-6)~~, then the value in the NUMBER OF LOGICAL BLOCKS field is ignored. If the logical unit supports changing its capacity by changing the NUMBER OF LOGICAL BLOCKS field and the CAPPID bit (see 6.5.1) is set to zero, then the NUMBER OF LOGICAL BLOCKS field is interpreted as follows:

a) if the NUMBER OF LOGICAL BLOCKS field is set to zero, then the logical unit shall retain the logical unit's current capacity if the logical block length has not changed. If the NUMBER OF LOGICAL BLOCKS field is set to zero and the value in the LOGICAL BLOCK LENGTH field (i.e., new logical block length) is different than the current logical block length, then the logical unit shall be set to ~~its~~the logical unit's maximum capacity when the new logical block length takes effect (i.e., after a successful FORMAT UNIT command);
b) if the NUMBER OF LOGICAL BLOCKS field is greater than zero and less than or equal to the logical unit's maximum capacity, then the device server shall determine a candidate number of logical blocks that is equal to the value in the NUMBER OF LOGICAL BLOCKS field. If the value in the LOGICAL BLOCK LENGTH field is the same as the current logical block length, then the logical unit shall not become format corrupt. If the value in the LOGICAL BLOCK LENGTH field (i.e., new logical block length) is different than the current logical block length, then the candidate number of logical blocks shall take effect when the new logical block length takes effect (i.e., after a successful FORMAT UNIT command);
c) if the NUMBER OF LOGICAL BLOCKS field is set to a value greater than the logical unit's maximum capacity ~~of the device~~ and less than FFFF_FFFFh, then the device server shall terminate the MODE SELECT command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN PARAMETER LIST. The logical unit shall retain its previous logical block descriptor settings; or
d) if the NUMBER OF LOGICAL BLOCKS field is set to FFFF_FFFFh, then the device server shall determine a candidate number of logical blocks that is equal to the logical unit's ~~manufactured~~ maximum capacity ~~(e.g., the value in the ALLOWED NUMBER OF LOGICAL BLOCKS field of the first capacity/product identification descriptor (see 6.6.6), if any)~~. If the value in the LOGICAL BLOCK LENGTH field is the same as the current logical block length, then the logical unit shall not become format corrupt. If the value in the LOGICAL BLOCK LENGTH field (i.e., new logical block length) is different than the current logical block length, then the candidate number of logical blocks shall take effect when the new logical block length takes effect (i.e., after a successful FORMAT UNIT command).

If the logical unit supports changing its capacity by changing the NUMBER OF LOGICAL BLOCKS field and the CAPPID bit (see 6.5.1) is set to one, then the NUMBER OF LOGICAL BLOCKS field is interpreted as follows:

a) if the NUMBER OF LOGICAL BLOCKS field is set to zero, then no action shall be taken by the device server (i.e., the logical unit retains the logical unit's current capacity and retains the value in the PRODUCT IDENTIFICATION field in the standard Inquiry data);
b) if the NUMBER OF LOGICAL BLOCKS field is greater than zero and less than FFFF_FFFFh, then the device server shall determine a candidate number of logical blocks that is equal to the value in the NUMBER OF LOGICAL BLOCKS field; and
c) if the NUMBER OF LOGICAL BLOCKS field is set to FFFF_FFFFh, then the device server shall determine a candidate number of logical blocks that is equal to the value in the ALLOWED NUMBER OF LOGICAL BLOCKS field in the capacity/product identification descriptor with the largest value in the ALLOWED NUMBER OF LOGICAL BLOCKS field.

If a candidate number of logical blocks has been determined and the CAPPID bit (see 6.5.1) is set to:

a) zero, and the value in the LOGICAL BLOCK LENGTH field is the same as the current logical block length, then the candidate number of logical blocks shall take effect on successful completion of the MODE SELECT command; and
b) one, then the candidate number of logical blocks shall be processed as described in 6.5.1 and the candidate number of logical blocks shall take effect as described in 6.5.1.

After the candidate number of logical blocks, if any, takes effect, the new value shall be retained through power cycles, hard resets, logical unit resets, and I_T nexus losses.

If the device server supports changing its logical unit's capacity by changing the NUMBER OF LOGICAL BLOCKS field, is in a logical unit that supports logical block provisioning management, and the capacity is increased, then the additional LBAs shall be in the initial provisioning management condition as specified in 4.7.3.2 or 4.7.3.3.

The LOGICAL BLOCK LENGTH field specifies the length in bytes of user data in 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-6) by reporting the length of the logical blocks as specified in the LOGICAL BLOCK LENGTH field sent in the most recent MODE SELECT command that contained a mode parameter block descriptor (e.g., if the logical block length is 512 bytes and a MODE SELECT command is processed with the LOGICAL BLOCK LENGTH field set to 520 bytes, then subsequent MODE SENSE commands return 0000_0208h in the LOGICAL BLOCK LENGTH field). If no MODE SELECT command with a block descriptor has been processed, then the current logical block length shall be returned. To determine the logical block length at which the logical unit is currently formatted, the application client shall use a READ CAPACITY command (see 5.20 and 5.21) rather than a MODE SENSE command.

6.5.2.3 Long LBA mode parameter block descriptor

Table 234 defines the long LBA mode parameter block descriptor for direct access block devices used with the MODE SELECT (10) command and MODE SENSE (10) command when the LONGLBA bit is set to one in the mode parameter header (see SPC-6).

Table 234 — Long LBA mode parameter block descriptor

Byte \ Bit	7	6	5	4	3	2	1	0
0	(MSB)							
•••	NUMBER OF LOGICAL BLOCKS							
7								(LSB)
8								
•••	Reserved							
11								
12	(MSB)							
•••	LOGICAL BLOCK LENGTH							
15								(LSB)

A device server shall respond to a MODE SENSE command (see SPC-6) by reporting the number of logical blocks specified in the NUMBER OF LOGICAL 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 logical blocks shall be returned. To determine the number of logical blocks at which the logical unit is currently formatted, the application client shall use a READ CAPACITY command (see 5.20 and 5.21) rather than a MODE SENSE command.

In response to a MODE SENSE command, the device server may return a value of zero indicating that it does not report the number of logical blocks in the long LBA mode parameter block descriptor.

The logical unit's maximum capacity is a vendor specific value that represents the maximum number of addressable logical blocks. If the Capacity/Product Identification Mapping VPD page (see 6.6.6) is supported, then that vendor specific value shall be less than (e.g., a reduced value due to depopulation) or equal to the value in the ALLOWED NUMBER OF LOGICAL BLOCKS field in the capacity/product identification descriptor with a value in the PRODUCT IDENTIFICATION field that equals the value of the PRODUCT IDENTIFICATION field in the standard Inquiry data.

If the logical unit does not support changing its capacity by changing the NUMBER OF LOGICAL BLOCKS field using the MODE SELECT command (see SPC-6), then the value in the NUMBER OF LOGICAL BLOCKS field is ignored. If the logical unit supports changing its capacity by changing the NUMBER OF LOGICAL BLOCKS field and the CAPPID bit (see 6.5.1) is set to zero, then the NUMBER OF LOGICAL BLOCKS field is interpreted as follows:

a) if the NUMBER OF LOGICAL BLOCKS field is set to zero, then the logical unit shall retain the logical unit's current capacity if the logical block length has not changed. If the NUMBER OF LOGICAL BLOCKS field is set to zero and the value in the LOGICAL BLOCK LENGTH field (i.e., new logical block length) is different than the current logical block length, then the logical unit shall be set to ~~its~~ the logical unit's maximum capacity when the new logical block length takes effect (i.e., after a successful FORMAT UNIT command);
b) if the NUMBER OF LOGICAL BLOCKS field is greater than zero and less than or equal to the logical unit's maximum capacity, then the device server shall determine a candidate number of logical blocks that is equal to the value in the NUMBER OF LOGICAL BLOCKS field. If the value in the LOGICAL BLOCK LENGTH field is the same as the current logical block length, then the logical unit shall not become format corrupt. If the value in the LOGICAL BLOCK LENGTH field (i.e., new logical block length) is different than the current logical block length, then the candidate number of logical blocks setting shall take effect when the new logical block length takes effect (i.e., after a successful FORMAT UNIT command);
c) if the NUMBER OF LOGICAL BLOCKS field is set to a value greater than the logical unit's maximum capacity ~~of the device~~ and less than FFFF_FFFF_FFFF_FFFFh, then the device server shall terminate the MODE SELECT command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN PARAMETER LIST. The logical unit shall retain its previous block descriptor settings; or
d) if the NUMBER OF LOGICAL BLOCKS field is set to FFFF_FFFF_FFFF_FFFFh, then the device server shall determine a candidate number of logical blocks that is equal to the logical unit's ~~manufactured~~ maximum capacity ~~(e.g., the value in the ALLOWED NUMBER OF LOGICAL BLOCKS field of the first capacity/product identification descriptor (see 6.6.6), if any)~~. If the value in the LOGICAL BLOCK LENGTH field is the same as the current logical block length, then the logical unit shall not become format corrupt. If the value in the LOGICAL BLOCK LENGTH field (i.e., new logical block length) is different than the current logical block length, then the candidate number of logical blocks shall take effect when the new logical block length takes effect (i.e., after a successful FORMAT UNIT command).

If the logical unit supports changing its capacity by changing the NUMBER OF LOGICAL BLOCKS field and the CAPPID bit (see 6.5.1) is set to one, then the NUMBER OF LOGICAL BLOCKS field is interpreted as follows:

a) if the NUMBER OF LOGICAL BLOCKS field is set to zero, then no action shall be taken by the device server (i.e., the logical unit retains the logical unit's current capacity and retains the value in the PRODUCT IDENTIFICATION field in the standard Inquiry data);
b) if the NUMBER OF LOGICAL BLOCKS field is greater than zero and less than FFFF_FFFF_FFFF_FFFFh, then the device server shall determine a candidate number of logical blocks that is equal to the value in the NUMBER OF LOGICAL BLOCKS field; and
c) if the NUMBER OF LOGICAL BLOCKS field is set to FFFF_FFFF_FFFF_FFFFh, then the device server shall determine a candidate number of logical blocks that is equal to the value in the ALLOWED NUMBER OF LOGICAL BLOCKS field in the capacity/product identification descriptor with the largest value in the ALLOWED NUMBER OF LOGICAL BLOCKS field.

If a candidate number of logical blocks has been determined and the CAPPID bit (see 6.5.1) is set to:

a) zero, and the value in the LOGICAL BLOCK LENGTH field is the same as the current logical block length, then the candidate number of logical blocks shall take effect on successful completion of the MODE SELECT command; and
b) one, then the candidate number of logical blocks shall be processed as described in 6.5.1 and the candidate number of logical blocks shall take effect as described in 6.5.1.

After the candidate number of logical blocks, if any, takes effect, the new value shall be retained through power cycles, hard resets, logical unit resets, and I_T nexus losses.

If the device server supports changing its logical unit's capacity by changing the NUMBER OF LOGICAL BLOCKS field, supports logical block provisioning management, and the capacity is increased, then the additional LBAs shall be in the initial provisioning management condition as specified in 4.7.3.2 or 4.7.3.3.

The LOGICAL BLOCK LENGTH field specifies the length in bytes of user data in each logical block. No change shall be made to any logical blocks on the medium until a format operation (see 5.3) is initiated by an application client.

A device server shall respond to a MODE SENSE command (see SPC-6) by reporting the length of the logical blocks as specified in the LOGICAL BLOCK LENGTH field sent in the most recent MODE SELECT command that contained a mode parameter block descriptor (e.g., if the logical block length is 512 bytes and a MODE SELECT command is processed with the LOGICAL BLOCK LENGTH field set to 520 bytes, then subsequent MODE SENSE commands return 0000_0208h in the LOGICAL BLOCK LENGTH field). If no MODE SELECT command with a block descriptor has been processed, then the current logical block length shall be returned. To determine the logical block length at which the logical unit is currently formatted, the application client shall use a READ CAPACITY command (see 5.20 and 5.21) rather than a MODE SENSE command.

6.5.3 Application Tag mode page

6.5.3.1 Overview

The Application Tag mode page (see table 235) specifies the logical block application tag that a device server configured for protection information (see 4.21.2) shall use for each LBA range if:

a) the ATO bit in the Control mode page (see SPC-6) is set to one;
b) the ATMPE bit in the Control mode page (see SPC-6) is set to one; and
c) the WRPROTECT field requirements (see table 135) specify use of the Application tag mode page.

The mode page policy (see SPC-6) for this page shall be shared.

If a method not defined by this standard changes the parameter data to be returned by the device server in the Application Tag mode page, then the device server shall establish a unit attention condition for the SCSI initiator port associated with every I_T nexus with the additional sense code set to MODE PARAMETERS CHANGED.

Table 235 — Application Tag mode page

Byte \ Bit	7	6	5	4	3	2	1	0
0	PS	SPF (1b)	PAGE CODE (0Ah)					
1	SUBPAGE CODE (02h)							
2	(MSB)	PAGE LENGTH (n - 3)						
3								(LSB)
4 ••• 15	Reserved							
	Application tag descriptors							
16 ••• 39	Application tag descriptor [first] (see 6.5.3.2)							
	⋮							
n - 24 ••• n	Application tag descriptor [last] (see 6.5.3.2)							

The parameters saveable (PS) bit, the subpage format (SPF) bit, the PAGE CODE field, the SUBPAGE CODE field, and the PAGE LENGTH field are defined in SPC-6.

The SPF bit, the PAGE CODE field, and the SUBPAGE CODE field shall be set to the values shown in table 235 for the Application Tag mode page.

The application tag descriptor is defined in 6.5.3.2.

6.5.3.2 Application tag descriptor

The application tag descriptor format is shown in table 236.

Table 236 — Application tag descriptor format

Byte \ Bit	7	6	5	4	3	2	1	0
0	LAST	Reserved						
1	Reserved							
•••								
5								
6	(MSB)	LOGICAL BLOCK APPLICATION TAG						
7								(LSB)
8	(MSB)	LOGICAL BLOCK ADDRESS						
•••								
15								(LSB)
16	(MSB)	LOGICAL BLOCK COUNT						
•••								
23								(LSB)

A LAST bit set to one specifies that this application tag descriptor is the last valid application tag descriptor in the Application Tag mode page. A LAST bit set to zero specifies that the application tag descriptor is not the last valid application tag descriptor in the Application Tag mode page.

The LOGICAL BLOCK APPLICATION TAG field specifies the value to be compared with the LOGICAL BLOCK APPLICATION TAG field associated with an LBA within the range specified by the LOGICAL BLOCK ADDRESS field and the LOGICAL BLOCK COUNT field within this descriptor.

The LOGICAL BLOCK ADDRESS field contains the starting LBA for this application tag descriptor. The LOGICAL BLOCK ADDRESS field in the first Application tag descriptor shall be set to 0000_0000_0000_0000h. For subsequent application tag descriptors in which the LOGICAL BLOCK COUNT field is not set to zero, the contents of the LOGICAL BLOCK ADDRESS field shall contain the sum of the values in:

a) the LOGICAL BLOCK ADDRESS field in the previous application tag descriptor; and
b) the LOGICAL BLOCK COUNT field in the previous application tag descriptor.

For the application tag descriptor with the LAST bit set to one, the sum of the LOGICAL BLOCK ADDRESS field and the LOGICAL BLOCK COUNT field shall equal the RETURNED LOGICAL BLOCK ADDRESS field (see 5.21.2).

If an invalid combination of the LAST bit, LOGICAL BLOCK APPLICATION TAG field, and LOGICAL BLOCK ADDRESS field are sent by the application client, then the device server shall terminate the MODE SELECT command (see SPC-6) with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

The LOGICAL BLOCK COUNT field specifies the number of logical blocks to which this application tag descriptor applies.

A LOGICAL BLOCK COUNT field set to 0000_0000_0000_0000h specifies that this application tag descriptor shall be ignored.

6.5.4 Background Control mode page

The Background Control mode page (see table 237) provides controls over background scan operations (see 4.23). The mode page policy (see SPC-6) for this subpage shall be shared.

Table 237 — Background Control mode page

Byte \ Bit	7	6	5	4	3	2	1	0
0	PS	SPF (1b)	PAGE CODE (1Ch)					
1	SUBPAGE CODE (01h)							
2	(MSB)	PAGE LENGTH (000Ch)						
3								(LSB)
4	Reserved					S_L_FULL	LOWIR	EN_BMS
5	Reserved							EN_PS
6	(MSB)	BACKGROUND MEDIUM SCAN INTERVAL TIME						
7								(LSB)
8	(MSB)	BACKGROUND PRE-SCAN TIME LIMIT						
9								(LSB)
10	(MSB)	MINIMUM IDLE TIME BEFORE BACKGROUND SCAN						
11								(LSB)
12	(MSB)	MAXIMUM TIME TO SUSPEND BACKGROUND SCAN						
13								(LSB)
14	Reserved							
15								

The parameters saveable (PS) bit, the subpage format (SPF) bit, the PAGE CODE field, the SUBPAGE CODE field, and the PAGE LENGTH field are defined in SPC-6.

The SPF bit, the PAGE CODE field, the SUBPAGE CODE field, and the PAGE LENGTH field shall be set to the values shown in table 237 for the Background Control mode page.

A suspend on log full (S_L_FULL) bit set to zero specifies that the device server shall continue running a background scan operation (see 4.23.1) even if the Background Scan Results log page (see 6.4.2) contains the maximum number of Background Scan log parameters (see 6.4.2.3) supported by the logical unit. A S_L_FULL bit set to one specifies that the device server shall suspend a background scan operation if the Background Scan Results log page contains the maximum number of Background scan log parameters supported by the logical unit.

A log only when intervention required (LOWIR) bit set to zero specifies that the device server shall log all suspected recoverable medium errors or unrecoverable medium errors that are identified during background scan operations in the Background Scan Results log page. A LOWIR bit set to one specifies that the device server shall only log medium errors identified during background scan operations in the Background Scan Results log page that require application client intervention.

An enable background medium scan (EN_BMS) bit set to zero specifies that the device server shall disable background medium scan operations (see 4.23.3). An EN_BMS bit set to one specifies that the device server shall enable background medium scan operations. If the EN_PS bit is also set to one, and a background pre-scan operation is in progress, then the logical unit shall not start a background medium scan operation until after the background pre-scan operation is halted or completed. If a background medium scan operation is in progress when the EN_BMS bit is changed from one to zero, then the logical unit shall suspend the background medium scan operation before the device server completes the MODE SELECT command, and the background medium scan shall remain suspended until the EN_BMS bit is set to one, at which time the logical unit shall resume the background medium scan operation beginning with the logical block being tested when the background medium scan operation was suspended.

An enable pre-scan (EN_PS) bit set to zero specifies that the device server shall disable background pre-scan operations (see 4.23.2). If a background pre-scan operation is in progress when the EN_PS bit is changed from a one to a zero, then the logical unit shall halt the background pre-scan operation before the device server completes the MODE SELECT command. An EN_PS bit set to one specifies that the logical unit shall start a background pre-scan operation after the next power on. Once a logical unit has completed a background pre-scan operation, the logical unit shall not perform another background pre-scan operation unless the EN_PS bit is set to zero, then set to one, and another power on occurs.

The BACKGROUND MEDIUM SCAN INTERVAL TIME field specifies the minimum time, in hours, between the start of one background scan operation and the start of the next background medium scan operation. If the current background scan operation takes longer than the value specified in the BACKGROUND MEDIUM SCAN INTERVAL TIME field, then the logical unit shall:

a) continue the current background scan operation until that background scan operation is complete; and
b) start the next background medium scan operation upon completion of the current background scan operation.

The BACKGROUND PRE-SCAN TIME LIMIT field specifies the maximum time, in hours, for a background pre-scan operation to complete. If the background pre-scan operation does not complete within the specified time then the device server shall halt the background pre-scan operation. A value of zero specifies an unlimited timeout value.

The MINIMUM IDLE TIME BEFORE BACKGROUND SCAN field specifies the time, in milliseconds, that the logical unit shall be idle before resuming a background scan operation (e.g., after the device server has completed all of the commands in all task sets).

The MAXIMUM TIME TO SUSPEND BACKGROUND SCAN field specifies the time, in milliseconds, that the device server should take to start processing a command received while a logical unit is performing a background scan operation.

6.5.5 Background Operation Control mode page

The Background Operation Control mode page (see table 238) provides controls of device server background operation.

Table 238 — Background Operation Control mode page

Byte \ Bit	7	6	5	4	3	2	1	0
0	PS	SPF (1b)	PAGE CODE (0Ah)					
1	SUBPAGE CODE (06h)							
2	PAGE LENGTH (01FDh)							
3								
4	BO_MODE		Reserved					
5	Reserved							
•••								
511								

The PS bit is described in SPC-6.

The SPF bit, PAGE CODE field, SUBPAGE CODE field, and PAGE LENGTH field are described in SPC-6 and shall be set as shown in table 238 for the Background Operation Control mode page.

The background operation mode (BO_MODE) field specifies how host initiated advanced background operations shall operate during read operations or write operations as shown in table 239.

Table 239 — BO_MODE field

Code	Description
00b	Host initiated advanced background operation shall be suspended during read operations and write operations and resume advanced background operation when read operations and write operations are complete.
01b	Host initiated advanced background operation shall continue during read operations and write operations.
All others	Reserved

6.5.6 Caching mode page

The Caching mode page (see table 240) defines the parameters that affect the use of the cache.

Table 240 — Caching mode page

Byte \ Bit	7	6	5	4	3	2	1	0
0	PS	SPF (0b)	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		SYNC_PROG		NV_DIS
13	NUMBER OF CACHE SEGMENTS							
14	(MSB)	CACHE SEGMENT SIZE						
15								(LSB)
16	Reserved							
17	Obsolete							
•••								
19								

The parameters saveable (PS) bit, the subpage format (SPF) bit, the PAGE CODE field, and the PAGE LENGTH field are defined in SPC-6.

The SPF bit, the PAGE CODE field, and the PAGE LENGTH field shall be set to the values shown in table 240 for the Caching mode page.

An initiator control (IC) enable bit set to one specifies that the device server use one of the following fields to control the caching algorithm rather than the device server's own adaptive algorithm:

a) the NUMBER OF CACHE SEGMENTS field, if the SIZE bit is set to zero; or
b) the CACHE SEGMENT SIZE field, if the SIZE bit is set to one.

An abort pre-fetch (ABPF) bit set to one and a DRA bit is set to zero specify 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 and a DRA bit set to zero specify that the termination of any active pre-fetch is dependent upon Caching mode page bytes 4 through 11 and is 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 (e.g., 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 one specifies that the device server shall perform write operations by using write cache operations to volatile write cache, write cache operations to non-volatile cache, or write medium operations (e.g., a write command is able to complete without error after logical block data has been written to volatile cache but has not necessarily been written to the medium) as described in 4.15. A WCE bit set to zero specifies that the device server shall complete a write command without error only after all logical block data has been written without error by performing write cache operations to non-volatile cache or by performing write medium operations to non-volatile medium. If an application client changes the WCE bit from one to zero via a MODE SELECT command, then the device server shall perform write medium operations to any LBAs in cache containing logical block data that is not the same as the logical block data referenced by the corresponding LBAs on the medium before completing the MODE SELECT command.

A multiplication factor (MF) bit set to zero specifies that the device server shall interpret the MINIMUM PRE-FETCH field and the MAXIMUM PRE-FETCH field 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 PRE-FETCH field and the MAXIMUM PRE-FETCH field 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 shall perform read operations by using read cache operations or read medium operations as described in 4.15. An RCD bit set to one specifies that the device server shall perform read operations by only using read medium operations.

The DEMAND READ RETENTION PRIORITY field (see table 241) specifies the retention priority the device server should assign for data read into the cache that has also been transferred to the Data-In Buffer.

Table 241 — DEMAND READ RETENTION PRIORITY field

Code	Description
0h	The device server should not distinguish between retaining the indicated data and data placed into the cache by other means (e.g., pre-fetch).
1h	The device server should replace data put into the cache via a READ command sooner (i.e., read data has lower priority) than data placed into the cache by other means (e.g., pre-fetch).
2h to Eh	Reserved
Fh	The device server should replace data placed into the cache by other means (e.g., pre-fetch) sooner than data put into the cache via a READ command (i.e., read data has higher priority).

The WRITE RETENTION PRIORITY field (see table 242) specifies the retention priority the device server should assign for data written into the cache that has also been transferred from the cache to the medium.

Table 242 — WRITE RETENTION PRIORITY field

Code	Description
0h	The device server should not distinguish between retaining the indicated data and data placed into the cache by other means (e.g., pre-fetch).
1h	The device server should replace data put into the cache during a WRITE command or a WRITE AND VERIFY command sooner (i.e., has lower priority) than data placed into the cache by other means (e.g., pre-fetch).
2h to Eh	Reserved
Fh	The device server should replace data placed into the cache by other means (e.g., pre-fetch) sooner than data put into the cache during a WRITE command or a WRITE AND VERIFY command (i.e., has higher priority).

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. The DISABLE PRE-FETCH TRANSFER LENGTH field, the MINIMUM PRE-FETCH field, the MAXIMUM PRE-FETCH field, and the MAXIMUM PRE-FETCH CEILING field give an indication to the device server how it should manage the cache based on the most recent READ command. An anticipatory pre-fetch may occur based on other information. These fields 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 transfer length requested by a READ command. If the transfer length 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 DISABLE PRE-FETCH TRANSFER LENGTH field is set to zero, then all anticipatory pre-fetching is disabled for any request for data, including those with a transfer length of zero.

The MINIMUM PRE-FETCH field specifies the number of logical blocks to pre-fetch regardless of the delays that may be incurred in processing subsequent commands. The field contains either:

a) a number of logical blocks, if the MF bit is set to zero; or
b) a scalar multiplier of the value in the TRANSFER LENGTH field, if the MF bit is set to one.

The pre-fetching operation begins at the logical block after the last logical block of a READ command. Pre-fetching shall always halt when it reaches the last logical block on 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 process subsequent commands correctly as a result of the error. In this case the error may be reported either as:

a) an error for that subsequent command; or
b) a deferred error,

at the discretion of the device server and according to the rules for reporting deferred errors (see SPC-6).

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 enters the enabled state (see SAM-6). This requirement is ignored if the MINIMUM PRE-FETCH field value is equal to the MAXIMUM PRE-FETCH field value.

The MAXIMUM PRE-FETCH field specifies the number of logical blocks to pre-fetch if the pre-fetch does not delay processing of subsequent commands. The field contains either:

a) a number of logical blocks, if the MF bit is set to zero; or
b) a scalar multiplier of the value in the TRANSFER LENGTH field, if the MF bit is set to one.

The MAXIMUM PRE-FETCH field contains the maximum amount of data to pre-fetch as a result of one READ command. The MAXIMUM PRE-FETCH field is used in conjunction with the DISABLE PRE-FETCH TRANSFER LENGTH field and MAXIMUM PRE- FETCH CEILING field 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 logical blocks is greater than the value in the MAXIMUM PRE-FETCH field, then the number of logical blocks to pre-fetch shall be truncated to the value stored in the MAXIMUM PRE-FETCH CEILING field.

NOTE 22 - If the MF bit is set to one, then 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, for commands requesting write operations to more than one logical block, the device server shall write the logical blocks to the medium in ascending sequential order. An FSW bit set to zero specifies that the device server may reorder the sequence of writing logical blocks to the medium (e.g., in order to achieve 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 bit shall not impact the units of other fields.

A disable read-ahead (DRA) bit set to one specifies that the device server shall not read into the pre-fetch 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 pre-fetch buffer beyond the addressed logical block(s).

The synchronize cache progress indication support (SYNC_PROG) field (see table 243) specifies device server progress indication reporting while a SYNCHRONIZE CACHE command (see 5.33 and 5.34) is being processed.

Table 243 — SYNC_PROG field

Code	Description
00b	The device server shall not terminate commands as a result of the synchronize cache operation and shall not provide pollable sense data.
01b	The device server: a) shall not terminate commands as a result of the synchronize cache operation; and b) shall provide pollable sense data with the sense key set to NO SENSE, the additional sense code set to SYNCHRONIZE CACHE OPERATION IN PROGRESS, and the PROGRESS INDICATION field set to indicate the progress of the synchronize cache operation.
10b	The device server: a) shall process INQUIRY commands, REPORT LUNS commands, REPORT TARGET PORT GROUPS commands, and REQUEST SENSE commands; b) may process commands that do not require resources used for the synchronize cache operation; c) shall terminate commands that require resources used for the synchronize cache operation with CHECK CONDITION status with the sense key set to NOT READY, the additional sense code set to LOGICAL UNIT NOT READY, SYNCHRONIZE CACHE OPERATION IN PROGRESS, and the PROGRESS INDICATION field set to indicate the progress of the synchronize cache operation; and d) shall provide pollable sense data with the sense key set to NOT READY, the additional sense code set to LOGICAL UNIT NOT READY, SYNCHRONIZE CACHE OPERATION IN PROGRESS, and the PROGRESS INDICATION field set to indicate the progress of the synchronize cache operation.
11b	Reserved

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.

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 if the LBCSS bit is set to zero or in logical blocks if the LBCSS bit is set to one. The CACHE SEGMENT SIZE field is valid only if the SIZE bit is set to one.

6.5.7 IO Advice Hints Grouping mode page

The IO Advice Hints Grouping mode page (see table 244) provides the application client with the means to obtain or modify the IO advice hints of the logical unit and the group number associated with those IO advice hints.

The mode page policy (see SPC-6) for this page shall be shared.

Table 244 — IO Advice Hints Grouping mode page

Byte \ Bit	7	6	5	4	3	2	1	0
0	PS	SPF (1b)	PAGE CODE (0Ah)					
1	SUBPAGE CODE (05h)							
2	(MSB)	PAGE LENGTH (40Ch)						
3								(LSB)
4 ••• 15	Reserved							
IO advice hints group descriptor list								
16 ••• 31	IO advice hints group descriptor (group 0) (see table 245)							
	⋮							
1024 ••• 1039	IO advice hints group descriptor (group 63) (see table 245)							

The parameters saveable (PS) bit, the subpage format (SPF) bit, the PAGE CODE field, the SUBPAGE CODE field, and the PAGE LENGTH field are defined in SPC-6.

The SPF bit, the PAGE CODE field, the SUBPAGE CODE field, and the PAGE LENGTH field shall be set to the values shown in table 244 for the IO Advice Hints Grouping mode page.

An IO advice hints group descriptor (see table 245) is provided for each group number. The logical block markup descriptor (see 6.8) in each IO advice hints group descriptor affects the processing of commands as described in 4.22.2.

Table 245 — IO advice hints group descriptor

Byte \ Bit	7	6	5	4	3	2	1	0
0	IO ADVICE HINTS MODE		Reserved			ST_ENBLE	CS_ENBLE	IC_ENABLE
1 ••• 3	Reserved							
4 ••• p-1	(MSB) Logical block markup descriptor (see 6.8)							(LSB)
p ••• 15	Pad (if any)							

The IO ADVICE HINTS MODE field specifies the mode of the logical block markup descriptor and is described in table 246.

Table 246 — IO ADVICE HINTS MODE field

Code	Description
00b	The logical block markup descriptor is valid (see 4.22.2).
01b	The logical block markup descriptor is invalid (see 4.22.2).
all others	Reserved

The stream identifier enable (ST_ENBLE) bit specifies whether the stream identifier for write stream commands (see 4.32.2) is the group number associated with this IO advice hints group descriptor. If the ST_ENBLE bit is set to zero, then this IO advice hints group descriptor does not affect the stream processing of a write stream command. If the ST_ENBLE bit is set to one, then the stream identifier for write stream commands is the group number associated with this IO advice hints group descriptor.

The cache segment enable (CS_ENBLE) bit specifies whether cache segments (see 4.15.2) are associated with the cache ID that is the group number associated with this IO advice hints group descriptor. If the CS_ENBLE bit is set to zero, then no independent cache segments are associated with the cache ID for the group number associated with this IO advice hints group descriptor. If the CS_ENBLE bit is set to one, then independent cache segments are associated with the cache ID for the group number associated with this IO advice hints group descriptor.

The information collection enable (IC_ENABLE) bit specifies whether the information collection function (see 4.22.1) for the group associated with this IO advice hints group descriptor is enabled. If the IC_ENABLE bit is set to zero, then the information collection function for the group associated with this IO advice hints group descriptor is not enabled. If the IC_ENABLE bit is set to one, then the information collection function for the group associated with this IO advice hints group descriptor is enabled.

If the Group Statistics and Performance (n) log pages are not supported, then in the IO advice hints group descriptors for group 0 to group 31, the IC_ENABLE bit shall be set to zero and shall not be changeable.

In the IO advice hints group descriptors for group 32 to group 63, the IC_ENABLE bit shall be set to zero, and shall not be changeable.

The logical block markup descriptor is described in 6.8.

6.5.8 Informational Exceptions Control mode page

The Informational Exceptions Control mode page (see table 247) defines the methods used by the device server to control the processing and reporting of informational exception conditions. Informational exception conditions are defined as any event that the device server reports or logs as failure predictions (i.e., with the ADDITIONAL SENSE CODE field set to 5Dh (e.g., FAILURE PREDICTION THRESHOLD EXCEEDED)) or warnings (i.e., with the ADDITIONAL SENSE CODE field set to 0Bh (e.g., WARNING)).

Informational exception conditions may occur while a logical unit is processing:

a) a background self-test (see SPC-6);
b) device specific background functions (see SPC-6);
c) a command; or
d) other device specific events.

An informational exception condition may occur at any time (e.g., the condition may be asynchronous to any commands issued by an application client).

The mode page policy for this mode page shall be shared or per I_T nexus (see SPC-6).

Storage devices that support SMART (Self-Monitoring Analysis and Reporting Technology) for predictive failure software should use informational exception conditions.

Table 247 — Informational Exceptions Control mode page

Byte \ Bit	7	6	5	4	3	2	1	0
0	PS	SPF (0b)	PAGE CODE (1Ch)					
1	PAGE LENGTH (0Ah)							
2	PERF	Reserved	EBF	EWASC	DEXCPT	TEST	EBACKERR	LOGERR
3	Reserved				MRIE			
4	(MSB)							
•••	INTERVAL TIMER							
7								(LSB)
8	(MSB)							
•••	REPORT COUNT							
11								(LSB)

The PS bit, the SPF bit, the PAGE CODE field, and the PAGE LENGTH field are described in SPC-6.

The SPF bit, the PAGE CODE field, and the PAGE LENGTH field shall be set to the values shown in table 247 for the Informational Exceptions Control mode page.

If the performance (PERF) bit is set to zero, then the device server may process informational exception conditions that cause delays in processing other operations (e.g., processing a command). If the PERF bit is set to one, then the device server shall not process informational exception conditions that cause delays in processing other operations. A PERF bit set to one may cause the device server to disable some or all of the processing of informational exception conditions, thereby limiting the reporting of informational exception conditions.

If device specific background functions (see SPC-6) are implemented by the logical unit, and the enable background function (EBF) bit is set to one, then the device server shall enable device specific background functions. If the EBF bit is set to zero, then the device server shall disable device specific background functions. Background functions with separate enable control bits (e.g., the background medium scan (see 4.23)) are not controlled by the EBF bit.

The enable warning (EWASC) bit specifies if the device server enables reporting of warnings (see table 248).

The disable exception control (DEXCPT) bit specifies if the device server disables reporting of failure predictions (see table 248).

The TEST bit specifies if the device server creates a test device failure prediction (see table 248).

If an informational exception condition occurs that is not the result of the logical unit processing a background self-test (see SPC-6) or device specific background function (see SPC-6), then the device server:

a) shall use the definitions for the combination of the values in the EWASC bit, the DEXCPT bit, and the TEST bit shown in table 248 for processing informational exception conditions if the MRIE field is set from 2h to 6h;
b) may use the definitions for the combination of the values in the EWASC bit, the DEXCPT bit, and the TEST bit shown table 248 for processing informational exception conditions if the MRIE field is set from Ch to Fh; and
c) shall ignore the EWASC bit, the DEXCPT bit, and the TEST bit if the MRIE field is set to any value other than 2h to 6h or Ch to Fh.

Table 248 — Definitions for the combinations of values in EWASC, DEXCPT, and TEST

EWASC	DEXCPT	TEST	Description
0	0	0	The device server shall process informational exception conditions as follows: a) failure prediction processing shall be enabled [a]; and b) warning processing shall be disabled.
1	0	0	The device server shall process informational exception conditions as follows: a) failure prediction processing shall be enabled [a]; and b) warning processing shall be enabled [a].
0	1	0	The device server shall process informational exception conditions as follows: a) failure prediction processing shall be disabled; and b) warning processing shall be disabled.
1	1	0	The device server shall process informational exception conditions as follows: a) failure prediction processing shall be disabled; and b) warning processing shall be enabled [a].
0	0	1	The device server shall set the additional sense code to FAILURE PREDICTION THRESHOLD EXCEEDED (FALSE) [a].
1	0	1	
0	1	1	The device server shall terminate the MODE SELECT command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN PARAMETER LIST.
1	1	1	

[a] If applicable based on the value in the MRIE field (e.g., 2h to 6h), then the values in the LOGERR bit, the INTERVAL TIMER field, and the REPORT COUNT field determine how the informational exception condition is processed.

If an informational exception condition occurs while the logical unit is processing a background self-test (see SPC-6) or background function (see SPC-6), then the enable background error (EBACKERR) bit determines how the device server processes the informational exception as defined in the following:

a) if the EBACKERR bit is set to zero, then the device server shall disable reporting of informational exception conditions that occur during the processing of background self-tests and background functions;

b) if the EBACKERR bit is set to one, then, for informational exception conditions that occur during the processing of background self-tests and background functions, the device server shall:
   A) enable reporting of the informational exception conditions;
   B) use the method for reporting the informational exception conditions as determined by contents of the MRIE field; and
   C) report the informational exception conditions as soon as the method specified in the MRIE field occurs (i.e., the INTERVAL TIMER field and REPORT COUNT field do not apply for background self-test errors and errors that occur during background functions);

   and

c) logging by the device server of informational exception conditions is determined by the value in the LOGERR bit.

A LOGERR bit set to zero specifies that the device server may log any informational exception conditions in the Informational Exceptions log page (see SPC-6). A LOGERR bit set to one specifies that the device server shall log informational exception conditions in the Informational Exceptions log page.

The method of reporting informational exceptions (MRIE) field (see table 249) specifies the method that shall be used by the device server to report:

a) informational exception conditions if the specified code value is supported by the device server; and
b) background self-test errors and device specific background function errors with the ADDITIONAL SENSE CODE field set to 0Bh or 5Dh if the EBACKERR bit is set to one and the specified code value is supported by the device server.

A device server that supports the Informational Exceptions Control mode page shall support at least one code value other than zero in the MRIE field.

The priority of reporting multiple informational exceptions is vendor specific.

Table 249 — Method of reporting informational exceptions (MRIE) field (part 1 of 2)

Code	Description
0h	**No reporting of informational exception condition:** The device server shall not report information exception conditions.
1h	Obsolete
2h	**Establish unit attention condition:** The device server shall report informational exception conditions by establishing a unit attention condition (see SAM-6) for the SCSI initiator port associated with every I_T nexus, with the additional sense code set to indicate the cause of the informational exception condition.[a]
3h	**Conditionally generate recovered error:** The device server shall report informational exception conditions, if the reporting of recovered errors is allowed [b], by modifying the completion of the next command processed without encountering any errors, regardless of the I_T nexus on which the command was received. The modification shall be to terminate the command with CHECK CONDITION status with the sense key set to RECOVERED ERROR and the additional sense code set to indicate the cause of the informational exception condition.

[a] The device server terminates the command to report the unit attention condition for the informational exception condition (i.e., the device server does not process the command except to report the unit attention condition) (see SAM-6).
[b] This is controlled by the PER bit (see 6.5.10) or the PER bit (see 6.5.11).

Table 249 — Method of reporting informational exceptions (MRIE) field (part 2 of 2)

Code	Description
4h	**Unconditionally generate recovered error:** The device server shall report informational exception conditions, regardless of whether the reporting of recovered errors is allowed [b], by modifying the completion of the next command processed without encountering any errors, regardless of the I_T nexus on which the command was received. The modification shall be to terminate the command with CHECK CONDITION status with the sense key set to RECOVERED ERROR and the additional sense code set to indicate the cause of the informational exception condition.
5h	**Generate no sense:** The device server shall report informational exception conditions by modifying the completion of the next command processed without encountering any errors, regardless of the I_T nexus on which the command was received. The modification shall be to terminate the command with CHECK CONDITION status with the sense key set to NO SENSE and the additional sense code set to indicate the cause of the informational exception condition.
6h	**Only report informational exception condition on request:** The device server shall provide pollable sense data (see SPC-6) with the sense key set to NO SENSE and the additional sense code set to indicate the cause of the informational exception condition. To find out about information exception conditions, the application client polls the device server by issuing a REQUEST SENSE command.
7h to Bh	Reserved
Ch to Fh	Vendor specific

[a] The device server terminates the command to report the unit attention condition for the informational exception condition (i.e., the device server does not process the command except to report the unit attention condition) (see SAM-6).
[b] This is controlled by the PER bit (see 6.5.10) or the PER bit (see 6.5.11).

The INTERVAL TIMER field specifies the period in 100 millisecond increments that the device server shall use for reporting that an informational exception condition has occurred (see table 250). After an informational exception condition has been reported, the interval timer shall be started. An INTERVAL TIMER field set to zero or FFFF_FFFFh specifies that the period for reporting an informational exception condition is vendor specific.

The REPORT COUNT field specifies the maximum number of times the device server may report an informational exception condition to the application client. A REPORT COUNT field set to zero specifies that there is no limit on the number of times the device server may report an informational exception condition.

The device server shall use the values in the INTERVAL TIMER field and the REPORT COUNT field based on the value in the MRIE field as shown in table 250.

Table 250 — Use of the INTERVAL TIMER field and the REPORT COUNT field based on the MRIE field

MRIE [a]	Description

2h to 6h	If reporting of an informational exception condition is enabled (see table 249), then the device server shall:
	1) report an informational exception condition when the condition is first detected; and
	2) if the value in the REPORT COUNT field is not equal to one, then:
	    1) if the INTERVAL TIMER field is not set to zero or FFFF_FFFFh, then wait the time specified in the INTERVAL TIMER field, and, if that informational exception condition still exists, report the informational exception again; and
	    2) while the informational exception condition exists, continue to report the informational exception condition after waiting the time specified in the INTERVAL TIMER field until the condition has been reported the number of times specified by the REPORT COUNT field.

Ch to Fh	The device server may use or may ignore the values in the INTERVAL TIMER field and the REPORT COUNT field to report the informational exception condition based on the device specific implementation.

[a] For values in the MRIE field (see table 249) not shown in this table, the INTERVAL TIMER field and the REPORT COUNT field shall be ignored.

Maintaining the interval timer and the report counter across power cycles, hard resets, logical unit resets, and I_T nexus losses by the device server is vendor specific.

6.5.9 Logical Block Provisioning mode page

6.5.9.1 Overview

The Logical Block Provisioning mode page (see table 251) specifies the parameters that a device server that supports logical block provisioning threshold values (see 4.7.3.7) shall use to report logical block provisioning threshold notifications (see 4.7.3.7.6). The mode page policy (see SPC-6) for this page shall be shared.

If a method not defined by this standard changes the parameter data to be returned by a device server in the Logical Block Provisioning mode page, then the device server shall establish a unit attention condition for the SCSI initiator port associated with every I_T nexus with the additional sense code set to MODE PARAMETERS CHANGED.

Table 251 — Logical Block Provisioning mode page

Byte \ Bit	7	6	5	4	3	2	1	0
0	PS	SPF (1b)	PAGE CODE (1Ch)					
1	SUBPAGE CODE (02h)							
2	(MSB)	PAGE LENGTH (n - 3)						
3								(LSB)
4	Reserved							SITUA
5 ••• 15	Reserved							
	Threshold descriptors							
16 ••• 23	Threshold descriptor [first] (see 6.5.9.2)							
	⋮							
n - 7 ••• n	Threshold descriptor [last] (see 6.5.9.2)							

The parameters saveable (PS) bit, the subpage format (SPF) bit, the PAGE CODE field, the SUBPAGE CODE field, and the PAGE LENGTH field are defined in SPC-6.

The SPF bit, the PAGE CODE field, and the SUBPAGE CODE field shall be set to the values shown in table 251 for the Logical Block Provisioning mode page.

A single initiator threshold unit attention (SITUA) bit set to one specifies that the logical block provisioning threshold notification unit attention condition is established on a single I_T nexus as described in 4.7.3.7.6. A SITUA bit set to zero specifies that the logical block provisioning threshold notification unit attention condition is established on multiple I_T nexuses as described in 4.7.3.7.6.

The threshold descriptors are defined in 6.5.9.2.

6.5.9.2 Threshold descriptor format

The threshold descriptor format is shown in table 252.

Table 252 — Threshold descriptor format

Byte \ Bit	7	6	5	4	3	2	1	0
0	ENABLED	Reserved	THRESHOLD TYPE			THRESHOLD ARMING		
1	THRESHOLD RESOURCE							
2	Reserved							
3								
4	(MSB)	THRESHOLD COUNT						
•••								
7								(LSB)

An ENABLED bit set to one specifies that the threshold is enabled. An ENABLED bit set to zero specifies that the threshold is disabled.

The THRESHOLD TYPE field (see table 253) specifies the type of this threshold.

Table 253 — THRESHOLD TYPE field

Code	Description
000b	If the THRESHOLD COUNT field specifies a soft threshold, the threshold is enabled, and that threshold is reached, then the device server shall establish a unit attention condition as described in 4.7.3.7.6.
001b	If the THRESHOLD COUNT field specifies a percentage threshold, the threshold is enabled, and that threshold is reached, then the device server shall establish a unit attention condition as described in 4.7.3.7.6
All others	Reserved

The THRESHOLD ARMING field (see table 254) specifies the arming method used for operation of this threshold.

Table 254 — THRESHOLD ARMING field

Code	Description	Reference
000b	The threshold operates as an armed decreasing threshold.	4.7.3.7.4
001b	The threshold operates as an armed increasing threshold.	4.7.3.7.5
All others	Reserved	

The THRESHOLD RESOURCE field specifies the resource of this threshold. The contents of this field are as defined for parameters codes 0000h to 00FFh (see table 198) in the Logical Block Provisioning log page (see 6.4.5).

The valid combinations of the THRESHOLD TYPE field, the THRESHOLD ARMING field, and the THRESHOLD RESOURCE field are shown in table 7 and table 8.

The THRESHOLD COUNT field specifies the center of the threshold range for this threshold expressed as:

a) a number of threshold sets (i.e., the number of LBA mapping resources expressed as a number of threshold sets), if the value in the THRESHOLD TYPE field is set to 000b; or

b) a percentage value, if the value in the THRESHOLD TYPE field is set to 001b.

6.5.10 Read-Write Error Recovery mode page

The Read-Write Error Recovery mode page (see table 255) specifies the error recovery parameters the device server shall use during:

a) read medium operations; or
b) write medium operations.

Table 255 — Read-Write Error Recovery mode page

Byte \ Bit	7	6	5	4	3	2	1	0
0	PS	SPF (0b)	PAGE CODE (01h)					
1	PAGE LENGTH (0Ah)							
2	AWRE	ARRE	TB	RC	Obsolete	Error recovery bits: PER	Error recovery bits: DTE	Obsolete
3	READ RETRY COUNT							
4	Obsolete							
5	Obsolete							
6	Obsolete							
7	LBPERE	MWR		Reserved			Restricted for MMC-6	
8	WRITE RETRY COUNT							
9	Reserved							
10	(MSB)	RECOVERY TIME LIMIT						
11								(LSB)

The parameters saveable (PS) bit, the subpage format (SPF) bit, the PAGE CODE field, and the PAGE LENGTH field are defined in SPC-6.

The SPF bit, the PAGE CODE field, and the PAGE LENGTH field shall be set to the values shown in table 255 for the Read-Write Error Recovery mode page.

An automatic write reassignment enabled (AWRE) bit set to zero specifies that the device server shall not perform automatic write reassignment.

An AWRE bit set to one specifies that the device server shall enable automatic write reassignment for LBAs referencing logical blocks for which a recovered error or unrecovered error occurs during a write medium operation. Automatic write reassignment shall be performed only if the device server has the valid data (e.g., original data in a buffer or recovered from the medium). The valid data shall be placed in the logical block referenced by the reassigned LBA. The device server shall report any failures that occur during the reassign operation. Error reporting as specified by the error recovery bits (i.e., the PER bit, and the DTE bit) shall be performed only after completion of the reassign operation.

An automatic read reassignment enabled (ARRE) bit set to zero specifies that the device server shall not perform automatic read reassignment.

An ARRE bit set to one specifies that the device server shall enable automatic read reassignment for LBAs referencing logical blocks for which a recovered error occurs during a read medium operation. All error recovery actions required by the error recovery bits shall be processed. Automatic read reassignment shall then be performed only if the device server recovers the data without error. The recovered data shall be placed in the logical block referenced by the reassigned LBA. The device server shall report any failures that occur during the reassign operation. Error reporting as specified by the error recovery bits shall be performed only after completion of the reassign operation.

A transfer block (TB) bit set to zero specifies that if an unrecovered read error occurs during a read medium operation, then the device server shall not transfer any data for that logical block to the Data-In Buffer. A TB bit set to one specifies that if an unrecovered read error occurs during a read medium operation, then the device server shall transfer pseudo read data (e.g., data already in a buffer or any other vendor specific data) for that logical block before returning CHECK CONDITION status. The data returned in this case is vendor specific. The value of the TB bit does not specify any action for recovered read errors.

A read continuous (RC) bit set to zero specifies that error recovery operations that cause delays during the data transfer are acceptable. 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 during the data transfer to perform error recovery procedures. The device server may transfer pseudo read data in order to maintain a continuous flow of data. The device server shall assign priority to the RC bit over conflicting bits within this byte.

NOTE 23 - The RC bit set to one is useful for image processing, audio, or video applications.

A post error (PER) bit set to one specifies that if a recovered read error occurs while processing a read command or a write command, then the device server shall terminate the command with CHECK CONDITION status with the sense key set to RECOVERED ERROR. A PER bit set to zero specifies that if a recovered read error occurs while processing a read command or a write command, then the device server shall perform error recovery procedures within the limits established by the error recovery parameters and not terminate the specified command with CHECK CONDITION status with the sense key set to RECOVERED ERROR (e.g., the device server may terminate the specified command with CHECK CONDITION status with the sense key set to MEDIUM ERROR if an uncorrectable error is detected based on the established limits during the error recovery process). If the DTE bit is set to one, then the PER bit shall be set to one.

A data terminate on error (DTE) bit set to one specifies that, upon detection of a recovered error, the device server shall terminate the data transfer to the Data-In Buffer for a read command or the data transfer to the Data-Out Buffer for a write command upon detection of a recovered error. A DTE bit set to zero specifies that, upon detection of a recovered error, the device server shall not terminate the data transfer to the Data-In Buffer for a read command or the data transfer to the Data-Out Buffer for a write command.

The combinations of the error recovery bits (i.e., the PER bit, and the DTE bit) are shown in table 256.

Table 256 — Error recovery bit combinations (part 1 of 2)

PER	DTE	Description [b]
0	0	The device server shall perform the full number of retries as specified in the READ RETRY COUNT field for read medium operations, the WRITE RETRY COUNT field for write medium operations, and the VERIFY RETRY COUNT field (see 6.5.11) for verify medium operations and shall perform error correction in an attempt to recover the data. The device server shall not report recovered read errors or write errors. The device server shall terminate a command performing a read medium operation or a write medium operation with CHECK CONDITION status before the transfer count is exhausted only if an unrecovered error is detected. If an unrecovered read error occurs during a read medium operation, then the transfer block (TB) bit determines whether the data for the logical block with the unrecovered read error is transferred to the Data-In Buffer.
0	1	Invalid mode. The PER bit is set to one if the DTE bit is set to one. [a]
1	0	The device server shall perform the full number of retries as specified in the READ RETRY COUNT field for read medium operations, the WRITE RETRY COUNT field for write medium operations, the VERIFY RETRY COUNT field (see 6.5.11) for verify medium operations, and shall perform error correction in an attempt to recover the data. The device server shall terminate a command performing a read medium operation or write medium operation with CHECK CONDITION status before the transfer count is exhausted only if an unrecovered error is detected. If an unrecovered read error occurs during a read medium operation, the transfer block (TB) bit determines whether the data for the logical block with the unrecovered read error is transferred to the Data-In Buffer. If a recovered error occurs while the device server is performing a read medium operation or write medium operation, then, after the operation is complete, the device server shall terminate the command with CHECK CONDITION status with the sense key set to RECOVERED ERROR. The INFORMATION field in the sense data contains the LBA of the last recovered error that occurred during the command (see 4.18.1).

[a] If an invalid combination of the error recovery bits is sent by an application client, then the device server shall terminate the MODE SELECT command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

[b] This table is used by both the Read Write Error Recovery mode page and the Verify Error Recovery mode page (see 6.5.11). When used for the Read Write Error Recovery mode page, rules about the VERIFY RETRY COUNT field are not applicable. When used for the Verify Error Recovery mode page, rules about the READ RETRY COUNT field, the WRITE RETRY COUNT field, and write medium operations are not applicable and rules about read medium operations are applicable to verify medium operations.

Table 256 — Error recovery bit combinations (part 2 of 2)

PER	DTE	Description [b]
1	1	The device server shall perform the full number of retries as specified in the READ RETRY COUNT field for read medium operations, the WRITE RETRY COUNT field for write medium operations, the VERIFY RETRY COUNT field (see 6.5.11) for verify medium operations, and shall perform error correction in an attempt to recover the data. The device server shall terminate a command performing a read medium operation or write medium operation with CHECK CONDITION status before the transfer count is exhausted if any error, either recovered or unrecovered, is detected. The INFORMATION field in the sense data contains the LBA of the error (see 4.18.1). If a recovered read error occurs during a read medium operation, then the device server shall transfer the recovered read data for the logical block with the recovered read error before returning CHECK CONDITION status. If an unrecovered read error occurs during a read medium operation, the transfer block (TB) bit determines whether the data for the logical block with the unrecovered read error is transferred to the Data-In Buffer.

[a] If an invalid combination of the error recovery bits is sent by an application client, then the device server shall terminate the MODE SELECT command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

[b] This table is used by both the Read Write Error Recovery mode page and the Verify Error Recovery mode page (see 6.5.11). When used for the Read Write Error Recovery mode page, rules about the VERIFY RETRY COUNT field are not applicable. When used for the Verify Error Recovery mode page, rules about the READ RETRY COUNT field, the WRITE RETRY COUNT field, and write medium operations are not applicable and rules about read medium operations are applicable to verify medium operations.

The READ RETRY COUNT field specifies the number of times that the device server shall attempt its recovery algorithm during read medium operations.

The WRITE RETRY COUNT field specifies the number of times that the device server shall attempt its recovery algorithm during write medium operations.

A logical block provisioning error reporting enabled (LBPERE) bit set to one specifies that logical block provisioning threshold notification is enabled. A LBPERE bit set to zero specifies that logical block provisioning threshold notification is disabled (see 4.7.3.7.6).

A misaligned write reporting (MWR) field (see table 257) specifies the behavior of the device server with respect to physical block misalignment reporting (see 4.6.2).

Table 257 — MWR field

Code	Name	Description
00b	DISABLED	Complete and do not report misaligned write operations
01b	ENABLED	Complete and report misaligned write operations
10b	TERMINATE	Terminate and report misaligned write operations
11b	Reserved	

A RECOVERY TIME LIMIT field set to a non-zero value specifies in milliseconds the maximum time duration that the device server shall use for data error recovery procedures during read medium operations and during write medium operations. The device server may round this value as described in SPC-6. 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 a retry count and a recovery time limit are specified, then the field that specifies the recovery action with the shortest recovery time shall have precedence.

To disable all types of correction and retries, the application client should set:

a) the PER bit to one;
b) the DTE bit to one;
c) the READ RETRY COUNT field to 00h;
d) the WRITE RETRY COUNT field to 00h; and
e) the RECOVERY TIME LIMIT field to 0000h.

6.5.11 Verify Error Recovery mode page

The Verify Error Recovery mode page (see table 258) specifies the error recovery parameters the device server shall use during verify medium operations (e.g., from VERIFY commands and the verify medium operations of the WRITE AND VERIFY commands). Verify medium operations do not trigger automatic read reassignment.

Table 258 — Verify Error Recovery mode page

<table>
<tr><th>Byte \ Bit</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td>0</td><td>PS</td><td>SPF (0b)</td><td colspan="6">PAGE CODE (07h)</td></tr>
<tr><td>1</td><td colspan="8">PAGE LENGTH (0Ah)</td></tr>
<tr><td rowspan="2">2</td><td colspan="4" rowspan="2">Reserved</td><td rowspan="2">Obsolete</td><td colspan="2">Error recovery bits</td><td rowspan="2">Obsolete</td></tr>
<tr><td>PER</td><td>DTE</td></tr>
<tr><td>3</td><td colspan="8">VERIFY RETRY COUNT</td></tr>
<tr><td>4</td><td colspan="8">Obsolete</td></tr>
<tr><td>5
•••
9</td><td colspan="8">Reserved</td></tr>
<tr><td>10</td><td>(MSB)</td><td colspan="6" rowspan="2">VERIFY RECOVERY TIME LIMIT</td><td></td></tr>
<tr><td>11</td><td></td><td>(LSB)</td></tr>
</table>

The parameters saveable (PS) bit, the subpage format (SPF) bit, the PAGE CODE field, and the PAGE LENGTH field are defined in SPC-6.

The SPF bit, the PAGE CODE field, and the PAGE LENGTH field shall be set to the values shown in table 258 for the Verify Error Recovery mode page.

The PER bit and the DTE bit (i.e., the error recovery bits) are defined in 6.5.10. The combinations of these bits are shown in table 256.

The VERIFY RETRY COUNT field specifies the number of times that the device server shall attempt its recovery algorithm during a verify medium operation.

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 during a verify medium operation. The device server may round this value as described in SPC-6. A VERIFY RECOVERY TIME LIMIT field set to zero specifies that the device server shall use its default value.

If both a verify retry count and a verify recovery time limit are specified, then the one that requires the least time for data error recovery actions shall have priority.

To disable all types of correction and retries, the application client should set:

a) the PER bit to one;
b) the DTE bit to one;
c) the VERIFY RETRY COUNT field to 00h; and
d) the VERIFY RECOVERY TIME LIMIT field to 0000h.

6.6 Vital product data (VPD) parameters

6.6.1 VPD parameters overview

See table 259 for references to the VPD pages used with direct access block devices.

Table 259 — VPD page codes for direct access block devices

VPD page name	Page code [a]	Reference	Support
ASCII Information	01h to 7Fh	SPC-6	See SPC-6
ATA Information	89h	SAT-4	See SAT-4
Block Device Characteristics	B1h	6.6.2	Optional
Block Device Characteristics Extension	B5h	6.6.3	Optional
Block Limits	B0h	6.6.4	Optional
Block Limits Extension	B7h	6.6.5	Optional
Capacity/Product Identification Mapping	BAh	6.6.6	Optional
CFA Profile Information	8Ch	SPC-6	See SPC-6
Concurrent Positioning Ranges	B9h	6.6.7	Optional
Device Constituents	8Bh	SPC-6	See SPC-6
Device Identification	83h	SPC-6	See SPC-6
Extended INQUIRY Data	86h	SPC-6	See SPC-6
Format Presets	B8h	6.6.8	Optional
Logical Block Provisioning	B2h	6.6.9	Optional
Management Network Addresses	85h	SPC-6	See SPC-6
Mode Page Policy	87h	SPC-6	See SPC-6
Power Condition	8Ah	SPC-6	See SPC-6
Power Consumption	8Dh	SPC-6	See SPC-6
Protocol Specific Logical Unit Information	90h	SPC-6	See SPC-6
Protocol Specific Port Information	91h	SPC-6	See SPC-6
Referrals	B3h	6.6.10	Optional
SCSI Feature Sets	92h	SPC-6	See SPC-6
SCSI Ports	88h	SPC-6	See SPC-6
Software Interface Identification	84h	SPC-6	See SPC-6
Supported VPD Pages	00h	SPC-6	See SPC-6
Third-party Copy	8Fh	SPC-6 and 6.6.11	Optional
Supported Block Lengths and Protection Types	B4h	6.6.12	Optional
Unit Serial Number	80h	SPC-6	See SPC-6
Zoned Block Device Characteristics	B6h	ZBC-2	See ZBC-2
Reserved for this standard	BBh to BFh		

[a] All page codes for direct access block devices not shown in this table are reserved.

6.6.2 Block Device Characteristics VPD page

The Block Device Characteristics VPD page (see table 260) contains parameters indicating characteristics of the logical unit.

Table 260 — Block Device Characteristics VPD page

Byte \ Bit	7	6	5	4	3	2	1	0
0	PERIPHERAL QUALIFIER			PERIPHERAL DEVICE TYPE				
1	PAGE CODE (B1h)							
2	(MSB)	PAGE LENGTH (003Ch)						
3								(LSB)
4	(MSB)	MEDIUM ROTATION RATE						
5								(LSB)
6	PRODUCT TYPE							
7	WABEREQ		WACEREQ		NOMINAL FORM FACTOR			
8	Reserved	MACT	Obsolete		RBWZ	BOCS	FUAB	VBULS
9 ••• 11	Reserved							
12	(MSB)	DEPOPULATION TIME						
••• 15								(LSB)
16 ••• 63	Reserved							

The PERIPHERAL QUALIFIER field and PERIPHERAL DEVICE TYPE field are defined in SPC-6.

The PAGE CODE field and PAGE LENGTH field are defined in SPC-6 and shall be set to the values shown in table 260 for the Block Device Characteristics VPD page.

The MEDIUM ROTATION RATE field is shown in table 261.

Table 261 — MEDIUM ROTATION RATE field

Code	Description
0000h	Medium rotation rate is not reported
0001h	Non-rotating medium (e.g., solid state)
0002h to 0400h	Reserved
0401h to FFFEh	Nominal medium rotation rate in revolutions per minute (e.g., 7 200 rpm = 1C20h,10 000 rpm = 2710h, and 15 000 rpm = 3A98h)
FFFFh	Reserved

The PRODUCT TYPE field (see table 262) defines the product type of the storage device.

Table 262 — PRODUCT TYPE field

Code	Description
00h	Not indicated
01h	CFast™ (see CFast) [a] [d]
02h	CompactFlash® (see CF)
03h	Memory Stick™ (see MS) [b] [d]
04h	MultiMediaCard (see *e*•MMC)
05h	Secure Digital Card (see SD Card)
06h	XQD™ (see XQD) [c] [d]
07h	Universal Flash Storage (see UFS)
08h to EFh	Reserved
F0h to FFh	Vendor specific

[a] CFast is the trademark of a product supplied by the CompactFlash Association.
[b] Memory Stick is the trademark of a product supplied by the One Stop Site for Formats.
[c] XQD is the trademark of a product supplied by the CompactFlash Association.
[d] This information is given for the convenience of users of this standard and does not constitute an endorsement of the product named. Equivalent products may be used if they lead to the same results.

The write after block erase required (WABEREQ) field indicates the device server behavior (see table 263), if a write operation has not been performed to a mapped LBA since a sanitize block erase operation was performed and no other error occurs during the processing of a read command specifying that LBA.

Table 263 — WABEREQ field

Code	Description
00b	Not specified.
01b	The device server completes the read command specifying that LBA with GOOD status and any data transferred to the Data-In Buffer is indeterminate.
10b	The device server terminates the read command specifying that LBA with CHECK CONDITION status with sense key set to MEDIUM ERROR and the additional sense code set to an appropriate value other than WRITE AFTER SANITIZE REQUIRED (e.g., ID CRC OR ECC ERROR).
11b	The device server terminates the read command specifying that LBA with CHECK CONDITION status with sense key set to MEDIUM ERROR and the additional sense code set to WRITE AFTER SANITIZE REQUIRED.

The write after cryptographic erase required (WACEREQ) field indicates the device server behavior (see table 264), if a write operation has not been performed to a mapped LBA since a sanitize cryptographic erase operation was performed and no other error occurs during the processing of a read command specifying that LBA.

Table 264 — WACEREQ field

Code	Description
00b	Not specified.
01b	The device server completes the read command specifying that LBA with GOOD status and any data transferred to the Data-In Buffer is indeterminate.
10b	The device server terminates the read command specifying that LBA with CHECK CONDITION status with sense key set to MEDIUM ERROR and the additional sense code set to an appropriate value other than WRITE AFTER SANITIZE REQUIRED (e.g., ID CRC OR ECC ERROR).
11b	The device server terminates the read command specifying that LBA with CHECK CONDITION status with sense key set to MEDIUM ERROR and the additional sense code set to WRITE AFTER SANITIZE REQUIRED.

The NOMINAL FORM FACTOR field indicates the nominal form factor of the device containing the logical unit and is shown in table 265.

Table 265 — NOMINAL FORM FACTOR field

Code	Description
0h	Nominal form factor is not reported
1h	5.25 inch
2h	3.5 inch
3h	2.5 inch
4h	1.8 inch
5h	Less than 1.8 inch
All others	Reserved

A multiple actuator (MACT) bit set to one indicates that the device server is contained in a SCSI target device with multiple actuators. An MACT bit set to zero indicates that the device server does not report whether it is contained in a SCSI target device with multiple actuators.

A reassign blocks write zero (RBWZ) bit set to one indicates that during the processing of a REASSIGN BLOCKS command that does not recover logical block data, the device server writes zeros as described in 5.24. A RBWZ bit set to zero indicates that during the processing of a REASSIGN BLOCKS command that does not recover the logical block data, the device server writes vendor specific data as described in 5.24.

A background operation control supported (BOCS) bit set to one indicates that background operation control is supported as described in 4.31. A BOCS bit set to zero indicates that background operation control is not supported.

A force unit access behavior (FUAB) bit set to one indicates that the device server interprets the SYNCHRONIZE CACHE command and the FUA bit in read commands and write commands in compliance with this standard. An FUAB bit set to zero indicates that the device server interprets the SYNCHRONIZE CACHE command and the FUA bit in read commands and write commands in compliance with SBC-2.

A verify byte check unmapped LBA supported (VBULS) bit set to one indicates that the device server supports unmapped LBAs while processing VERIFY commands (see 5.36, 5.37, 5.38, and 5.39) and WRITE AND VERIFY commands (see 5.44, 5.45, 5.46, and 5.47) with the BYTCHK field set to 01b. A VBULS bit set to zero indicates that the device server does not support unmapped LBAs while processing VERIFY commands and WRITE AND VERIFY commands with the BYTCHK field set to 01b. The device server should set the VBULS bit to one.

The DEPOPULATION TIME field indicates the maximum time in seconds for the device server to perform the operations associated with a REMOVE ELEMENT AND TRUNCATE command.

6.6.3 Block Device Characteristics Extension VPD page

The Block Device Characteristics Extension VPD page (see table 266) contains parameters indicating characteristics of the logical unit.

Table 266 — Block Device Characteristics Extension VPD page

Byte \ Bit	7	6	5	4	3	2	1	0
0	PERIPHERAL QUALIFIER			PERIPHERAL DEVICE TYPE				
1	PAGE CODE (B5h)							
2	(MSB)	PAGE LENGTH (007Ch)						
3								(LSB)
4	Reserved							
5	UTILIZATION TYPE							
6	UTILIZATION UNITS							
7	UTILIZATION INTERVAL							
8	(MSB)	UTILIZATION B						
•••								
11								(LSB)
12	(MSB)	UTILIZATION A						
•••								
15								(LSB)
16	Reserved							
•••								
127								

The PERIPHERAL QUALIFIER field and PERIPHERAL DEVICE TYPE field are defined in SPC-6.

The PAGE CODE field and PAGE LENGTH field are defined in SPC-6 and shall be set to the values shown in table 266 for the Block Device Characteristics Extension VPD page.

The UTILIZATION TYPE field (see table 267) indicates the designed utilization characteristics for the direct access block device based on the contents of the UTILIZATION A field and the UTILIZATION B field evaluated using the units indicated by the UTILIZATION UNITS field over the time interval indicated by the UTILIZATION INTERVAL field.

Table 267 — UTILIZATION TYPE field

Code	Description
01h	**Combined writes and reads:** the UTILIZATION A field contains designed number of host requested bytes transferred by write operations and host requested bytes transferred by read operations. The UTILIZATION B field is reserved.
02h	**Writes only:** the UTILIZATION A field contains designed number of host requested bytes transferred by write operations. The UTILIZATION B field is reserved.
03h	**Separate writes and reads:** the UTILIZATION A field contains designed number of host requested bytes transferred by write operations. The UTILIZATION B field contains designed number of host requested bytes transferred by read operations.
all others	Reserved

The UTILIZATION UNITS field (see table 268) indicates the units of measure for the values, if any, in the UTILIZATION A field and the UTILIZATION B field.

Table 268 — UTILIZATION UNITS field

Code	Description
02h	megabytes
03h	gigabytes
04h	terabytes
05h	petabytes
06h	exabytes
all others	Reserved

The UTILIZATION INTERVAL field (see table 269) indicates a nominal calendar time reference interval over which the values, if any, in the UTILIZATION A field and the UTILIZATION B field may be applied.

Table 269 — UTILIZATION INTERVAL field

Code	Description
0Ah	per day
0Eh	per year
all others	Reserved

The UTILIZATION B field and the UTILIZATION A field indicate the designed utilization characteristics for the direct access block device as:

a) defined by the UTILIZATION TYPE field;
b) expressed in the units defined by the UTILIZATION UNITS field; and
c) over the time interval defined by the UTILIZATION INTERVAL field.

6.6.4 Block Limits VPD page

The Block Limits VPD page (see table 270) provides the application client with the means to obtain certain operating parameters of the logical unit.

Table 270 — Block Limits VPD page (part 1 of 2)

Byte \ Bit	7	6	5	4	3	2	1	0
0	PERIPHERAL QUALIFIER			PERIPHERAL DEVICE TYPE				
1	PAGE CODE (B0h)							
2	(MSB)	PAGE LENGTH (003Ch)						
3								(LSB)
4	Reserved							WSNZ
5	MAXIMUM COMPARE AND WRITE LENGTH							
6	(MSB)	OPTIMAL TRANSFER LENGTH GRANULARITY						
7								(LSB)
8	(MSB)	MAXIMUM TRANSFER LENGTH						
•••								
11								(LSB)
12	(MSB)	OPTIMAL TRANSFER LENGTH						
•••								
15								(LSB)
16	(MSB)	MAXIMUM PREFETCH LENGTH						
•••								
19								(LSB)
20	(MSB)	MAXIMUM UNMAP LBA COUNT						
•••								
23								(LSB)
24	(MSB)	MAXIMUM UNMAP BLOCK DESCRIPTOR COUNT						
•••								
27								(LSB)
28	(MSB)	OPTIMAL UNMAP GRANULARITY						
•••								
31								(LSB)
32	UGAVALID	(MSB)	UNMAP GRANULARITY ALIGNMENT					
•••								
35								(LSB)
36	(MSB)	MAXIMUM WRITE SAME LENGTH						
•••								
43								(LSB)
44	(MSB)	MAXIMUM ATOMIC TRANSFER LENGTH						
•••								
47								(LSB)

Table 270 — Block Limits VPD page (part 2 of 2)

Byte \ Bit	7	6	5	4	3	2	1	0
48	(MSB)							
•••	ATOMIC ALIGNMENT							
51								(LSB)
52	(MSB)							
•••	ATOMIC TRANSFER LENGTH GRANULARITY							
55								(LSB)
56	(MSB)							
•••	MAXIMUM ATOMIC TRANSFER LENGTH WITH ATOMIC BOUNDARY							
59								(LSB)
60	(MSB)							
•••	MAXIMUM ATOMIC BOUNDARY SIZE							
63								(LSB)

The PERIPHERAL QUALIFIER field and PERIPHERAL DEVICE TYPE field are defined in SPC-6.

The PAGE CODE field and PAGE LENGTH field are defined in SPC-6 and shall be set to the values shown in table 270 for the Block Limits VPD page.

A write same non-zero (WSNZ) bit set to one indicates that the device server does not support a value of zero in the NUMBER OF LOGICAL BLOCKS field in the WRITE SAME command CDBs (see 5.52, 5.53, and 5.54). A WSNZ bit set to zero indicates that the device server may or may not support a value of zero in the NUMBER OF LOGICAL BLOCKS field of the WRITE SAME commands.

A MAXIMUM COMPARE AND WRITE LENGTH field set to a non-zero value indicates the maximum value that the device server accepts in the NUMBER OF LOGICAL BLOCKS field in the COMPARE AND WRITE command (see 5.3). A MAXIMUM COMPARE AND WRITE LENGTH field set to 00h indicates that the device server does not support the COMPARE AND WRITE command. If the MAXIMUM TRANSFER LENGTH field is not set to zero, then the device server shall set the MAXIMUM COMPARE AND WRITE LENGTH field to a value less than or equal to the value in the MAXIMUM TRANSFER LENGTH field.

An OPTIMAL TRANSFER LENGTH GRANULARITY field set to a non-zero value indicates the optimal transfer length granularity size in logical blocks for a single command shown in the command column of table 33. If a device server receives one of these commands with a transfer size that is not equal to a multiple of this value, then the device server may incur delays in processing the command. An OPTIMAL TRANSFER LENGTH GRANULARITY field set to 0000h indicates that the device server does not report optimal transfer length granularity.

A MAXIMUM TRANSFER LENGTH field set to a non-zero value indicates the maximum transfer length in logical blocks that the device server accepts for a single command shown in table 33. If a device server receives one of these commands with a transfer size greater than this value, then the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to the value shown in table 33. A MAXIMUM TRANSFER LENGTH field set to 0000_0000h indicates that the device server does not report a limit on the transfer length.

An OPTIMAL TRANSFER LENGTH field set to a non-zero value indicates the optimal transfer size in logical blocks for a single command shown in table 33. If a device server receives one of these commands with a transfer size greater than this value, then the device server may incur delays in processing the command. An OPTIMAL TRANSFER LENGTH field set to 0000_0000h indicates that the device server does not report an optimal transfer size.

The MAXIMUM PREFETCH LENGTH field indicates the maximum prefetch length in logical blocks that the device server accepts for a single PRE-FETCH command. If the MAXIMUM TRANSFER LENGTH field is not set to zero, then the device server should set the MAXIMUM PREFETCH LENGTH field to a value less than or equal to the value in the MAXIMUM TRANSFER LENGTH field.

A MAXIMUM UNMAP LBA COUNT field set to a non-zero value indicates the maximum number of LBAs that may be unmapped by an UNMAP command (see 5.35). If the number of LBAs that may be unmapped by an UNMAP command is constrained only by the amount of data that may be contained in the UNMAP parameter list (see 5.35.2), then the device server shall set the MAXIMUM UNMAP LBA COUNT field to FFFF_FFFFh. If the device server implements the UNMAP command, then the value in this field shall be greater than or equal to one. A MAXIMUM UNMAP LBA COUNT field set to 0000_0000h indicates that the device server does not implement the UNMAP command.

A MAXIMUM UNMAP BLOCK DESCRIPTOR COUNT field set to a non-zero value indicates the maximum number of UNMAP block descriptors (see 5.35.2) that shall be contained in the parameter data transferred to the device server for an UNMAP command (see 5.35). If there is no limit on the number of UNMAP block descriptors contained in the parameter data, then the device server shall set the MAXIMUM UNMAP BLOCK DESCRIPTOR COUNT field to FFFF_FFFFh. If the device server implements the UNMAP command, then the value in the MAXIMUM UNMAP BLOCK DESCRIPTOR COUNT field shall be greater than or equal to one. A MAXIMUM UNMAP BLOCK DESCRIPTOR COUNT field set to 0000_0000h indicates that the device server does not implement the UNMAP command.

An OPTIMAL UNMAP GRANULARITY field set to a non-zero value indicates the optimal granularity in logical blocks for unmap requests (e.g., an UNMAP command or a WRITE SAME (16) command with the UNMAP bit set to one). An unmap request with a number of logical blocks that is not a multiple of this value may result in unmap operations on fewer LBAs than requested. An OPTIMAL UNMAP GRANULARITY field set to 0000_0000h indicates that the device server does not report an optimal unmap granularity.

An unmap granularity alignment valid (UGAVALID) bit set to one indicates that the UNMAP GRANULARITY ALIGNMENT field is valid. A UGAVALID bit set to zero indicates that the UNMAP GRANULARITY ALIGNMENT field is not valid.

The UNMAP GRANULARITY ALIGNMENT field indicates the LBA of the first logical block to which the OPTIMAL UNMAP GRANULARITY field applies. The unmap granularity alignment is used to calculate an optimal unmap request starting LBA as follows:

optimal unmap request starting LBA = (n × optimal unmap granularity) + unmap granularity alignment

where:

n	is zero or any positive integer value;
optimal unmap granularity	is the value in the OPTIMAL UNMAP GRANULARITY field; and
unmap granularity alignment	is the value in the UNMAP GRANULARITY ALIGNMENT field.

An unmap request with a starting LBA that is not optimal may result in unmap operations on fewer LBAs than requested.

A MAXIMUM WRITE SAME LENGTH field set to a non-zero value indicates the maximum number of contiguous logical blocks that the device server allows to be unmapped or written in a single WRITE SAME command. A MAXIMUM WRITE SAME LENGTH field set to 0000_0000h indicates that the device server does not report a limit on the number of logical blocks that it allows to be unmapped or written in a single WRITE SAME command.

If the ATOMIC BOUNDARY field in the the WRITE ATOMIC (16) command (see 5.48) or WRITE ATOMIC (32) command (see 5.49) is set to 0000h, then a MAXIMUM ATOMIC TRANSFER LENGTH field set to a non-zero value indicates the maximum atomic transfer length in logical blocks that the device server supports for a single atomic write command (see 4.29). A MAXIMUM ATOMIC TRANSFER LENGTH field set to 0000_0000h indicates that the device server does not indicate a maximum atomic transfer length. The maximum atomic transfer length indicated by the MAXIMUM ATOMIC TRANSFER LENGTH field shall be less than or equal to the maximum transfer length indicated by the MAXIMUM TRANSFER LENGTH field. The maximum atomic transfer length indicated by the MAXIMUM ATOMIC TRANSFER LENGTH field shall be a multiple of the value in the ATOMIC TRANSFER LENGTH GRANULARITY field. If the ATOMIC BOUNDARY field is set to a non-zero value, then the MAXIMUM ATOMIC TRANSFER LENGTH field is ignored.

The ATOMIC ALIGNMENT field indicates the required alignment of the starting LBA in an atomic write command. If the ATOMIC ALIGNMENT field is set to 0000_0000h, then there is no alignment requirement for atomic write commands.

If the ATOMIC ALIGNMENT field is non-zero, then the starting LBA of an atomic write request shall meet the following:

atomic request starting LBA = n × atomic alignment

where:

n	is zero or any positive integer value; and
atomic alignment	is the value in the ATOMIC ALIGNMENT field.

The ATOMIC TRANSFER LENGTH GRANULARITY field indicates the minimum transfer length for an atomic write command. Atomic write operations are required to have a transfer length that is a multiple of the atomic transfer length granularity. An ATOMIC TRANSFER LENGTH GRANULARITY field set to 0000_0000h indicates that there is no atomic transfer length granularity requirement.

If the ATOMIC BOUNDARY field in the WRITE ATOMIC (16) command (see 5.48) or WRITE ATOMIC (32) command (see 5.49) is set to a non-zero value, then a MAXIMUM ATOMIC TRANSFER LENGTH WITH ATOMIC BOUNDARY field set to a non-zero value indicates the maximum transfer length in logical blocks that the device server supports for a single atomic write command (see 4.29). A MAXIMUM ATOMIC TRANSFER LENGTH WITH ATOMIC BOUNDARY field set to 0000_0000h indicates that the device server does not indicate a maximum atomic transfer length with atomic boundary. The maximum atomic transfer length with atomic boundary indicated by the MAXIMUM ATOMIC TRANSFER LENGTH WITH ATOMIC BOUNDARY field shall be less than or equal to the maximum transfer length indicated by the MAXIMUM TRANSFER LENGTH field. The maximum atomic transfer length with atomic boundary indicated by the MAXIMUM ATOMIC TRANSFER LENGTH WITH BOUNDARY field shall be a multiple of the value in the ATOMIC TRANSFER LENGTH GRANULARITY field. If the ATOMIC BOUNDARY field is set to 0000h, then the MAXIMUM ATOMIC TRANSFER LENGTH WITH ATOMIC BOUNDARY field is ignored.

A MAXIMUM ATOMIC BOUNDARY SIZE field set to a non-zero value indicates that the device server supports atomic write commands performing more than one atomic write operation. The maximum atomic boundary size indicates the maximum number of logical blocks on which the device server is able to perform atomicwrite operations (see 4.29.1). A MAXIMUM ATOMIC BOUNDARY SIZE field set to 0000h indicates that the device server does not support atomic write commands performing more than one atomic write operation. The MAXIMUM ATOMIC BOUNDARY SIZE field shall be a multiple of the value in the ATOMIC TRANSFER LENGTH GRANULARITY field.

6.6.5 Block Limits Extension VPD page

The Block Limits Extension VPD page (see table 271) provides the application client with the means to obtain certain operating parameters of the logical unit.

Table 271 — Block Limits Extension VPD page (part 1 of 2)

Byte \ Bit	7	6	5	4	3	2	1	0
0	PERIPHERAL QUALIFIER			PERIPHERAL DEVICE TYPE				
1	PAGE CODE (B7h)							
2	(MSB)	PAGE LENGTH (n-3)						
3								(LSB)
4	Reserved							
5	Reserved							RSCS
6	(MSB)	MAXIMUM NUMBER OF STREAMS						
7								(LSB)
8	(MSB)	OPTIMAL STREAM WRITE SIZE						
9								(LSB)

Table 271 — Block Limits Extension VPD page (part 2 of 2)

Byte \ Bit	7	6	5	4	3	2	1	0
10	(MSB)							
•••	STREAM GRANULARITY SIZE							
13								(LSB)
14	Reserved							
15								
16	(MSB)							
•••	MAXIMUM SCATTERED LBA RANGE TRANSFER LENGTH							
19								(LSB)
20	Reserved							
21								
22	(MSB)	MAXIMUM SCATTERED LBA RANGE DESCRIPTOR COUNT						
23								(LSB)
24	(MSB)							
•••	MAXIMUM SCATTERED TRANSFER LENGTH							
27								(LSB)
28	Reserved							
•••								
n								

The PERIPHERAL QUALIFIER field, PERIPHERAL DEVICE TYPE field, and PAGE LENGTH field are defined in SPC-6.

The PAGE CODE field is defined in SPC-6 and shall be set to the values shown in table 271 for the Block Limits Extension VPD page.

A reduced stream control supported (RSCS) bit set to one indicates that the device server supports reduced stream control (see 4.32.3). A RSCS bit set to zero indicates that the device server does not support reduced stream control.

The MAXIMUM NUMBER OF STREAMS field indicates the maximum number of streams that the device server supports and the maximum value for the stream identifier. A MAXIMUM NUMBER OF STREAMS field set to 0000h indicates that the device server does not support stream control.

The OPTIMAL STREAM WRITE SIZE field indicates the alignment and size of the optimal stream write as a number of logical blocks. The optimal stream write size is the same for all streams in the device server. An OPTIMAL STREAM WRITE SIZE field set to 0000h indicates that no alignment and no size is applied to the optimal stream writes.

The STREAM GRANULARITY SIZE field indicates the stream granularity size in number of optimal stream write size blocks as described in 4.32.

A MAXIMUM SCATTERED LBA RANGE TRANSFER LENGTH field set to a non-zero value indicates the maximum value that the device server supports in the NUMBER OF LOGICAL BLOCKS field for a single LBA range descriptor in a WRITE SCATTERED command (see 5.55 and 5.56). A MAXIMUM SCATTERED LBA RANGE TRANSFER LENGTH field set to 0000_0000h indicates that the device server does not report a limit.

A MAXIMUM SCATTERED LBA RANGE DESCRIPTOR COUNT field set to a non-zero value indicates the maximum number of LBA range descriptors that the device server supports for a single WRITE SCATTERED command (see 5.55 and 5.56). A MAXIMUM SCATTERED LBA RANGE DESCRIPTOR COUNT field set to 0000h indicates that the device server does not report a limit.

A MAXIMUM SCATTERED TRANSFER LENGTH field set to a non-zero value indicates the maximum transfer length in logical blocks that the device server accepts for a single WRITE SCATTERED command (see 5.55 and 5.56). A MAXIMUM SCATTERED TRANSFER LENGTH field set to 0000_0000h indicates that the device server does not report a limit.

6.6.6 Capacity/Product Identification Mapping VPD page

The Capacity/Product Identification Mapping VPD page (see table 272) contains parameters that affect the processing of the CAPPID bit (see 6.5.1).

Table 272 — Capacity/Product Identification Mapping VPD page

Byte \ Bit	7	6	5	4	3	2	1	0
0	PERIPHERAL QUALIFIER			PERIPHERAL DEVICE TYPE				
1	PAGE CODE (BAh)							
2	(MSB)	PAGE LENGTH (n - 3)						
3								(LSB)
	Capacity/product identification descriptors list							
4 ••• 51	Capacity/product identification descriptor [first] (see table 273)							
	⋮							
n-47 ••• n	Capacity/product identification descriptor [last] (see table 273)							

The PERIPHERAL QUALIFIER field, PERIPHERAL DEVICE TYPE field, PAGE CODE field, and PAGE LENGTH field are defined in SPC-6.

The PAGE CODE field shall be set as shown in table 272 for the Capacity/Product Identification Mapping VPD page.

Each capacity/product identification descriptor (see table 273) indicates:

a) one possible value that may result from a change in the NUMBER OF LOGICAL BLOCKS field in a mode parameter block descriptor (see 6.5.2.2, 6.5.2.3, and 6.5.1); and
b) the product identification information associated with that number of logical blocks.

The first capacity/product identification descriptor shall indicate the number of logical blocks and product identification for which the device server was manufactured.

The second through last capacity/product identification descriptors shall indicate equivalent information for other products from the same manufacturer~~ that are limited to smaller capacities based on maximum number of logical blocks supported~~.

Downloading and activating microcode (see SPC-6):

a) may require inputs that are specific to the product identification indicated in the first capacity/product identification descriptor; and
b) may result in the device server terminating with CHECK CONDITION status any attempt to download or activate microcode that is not compatible with the product identification indicated in the first capacity/product identification descriptor.

Table 273 — Capacity/product identification descriptor

Byte \ Bit	7	6	5	4	3	2	1	0
0	(MSB)							
•••	ALLOWED NUMBER OF LOGICAL BLOCKS							
7								(LSB)
8	(MSB)							
•••	PRODUCT IDENTIFICATION							
23								(LSB)
24								
•••	Restricted (see SAT-6)							
47								

The ALLOWED NUMBER OF LOGICAL BLOCKS field indicates a value to which the processing of a mode parameter block descriptor (see 6.5.2.2 and 6.5.2.3) sent by a MODE SELECT command may cause the NUMBER OF LOGICAL BLOCKS field to change as described in 6.5.1.

The PRODUCT IDENTIFICATION field contains 16 bytes of left-aligned ASCII data (see SPC-6) that indicates the product identification that is associated with the value in the ALLOWED NUMBER OF LOGICAL BLOCKS field.

6.6.7 Concurrent Positioning Ranges VPD page

The Concurrent Positioning Ranges VPD page (see table 274) contains parameters indicating the number of sets of contiguous LBAs in which concurrent positioning is possible. This VPD page is applicable to logical units that are capable of positioning for more than one data transfer (e.g., seeking) at a time.

If one read command or one write command specifies the transferring of data using more than one of the indicated sets of contiguous LBAs (i.e., the LBA range descriptors in table 274), the device may require additional time to complete the command (e.g., as much time as the sum of the times for individual commands that transfer the same data without using more than one set of contiguous LBAs per command).

Table 274 — Concurrent Positioning Ranges VPD page

Byte \ Bit	7	6	5	4	3	2	1	0
0	PERIPHERAL QUALIFIER			PERIPHERAL DEVICE TYPE				
1	PAGE CODE (B9h)							
2	(MSB)	PAGE LENGTH (n - 3)						
3								(LSB)
4 ••• 63	Reserved							
	LBA range descriptor list							
64 ••• 95	LBA range descriptor [first] (see table 275)							
	⋮							
n-31 ••• n	LBA range descriptor [last] (see table 275)							

The PERIPHERAL QUALIFIER field, PERIPHERAL DEVICE TYPE field, and PAGE LENGTH field are defined in SPC-6.

The PAGE CODE field is defined in SPC-6 and shall be set as shown in table 274 for the Concurrent Positioning Ranges VPD page.

Each LBA range descriptor (see table 275) contains information about one LBA range configured in the logical unit.

Table 275 — LBA range descriptor

Byte \ Bit	7	6	5	4	3	2	1	0
0	LBA RANGE NUMBER							
1	NUMBER OF STORAGE ELEMENTS							
2 ••• 7	Reserved							
8	(MSB)							
•••	STARTING LBA							
15								(LSB)
16	(MSB)							
•••	NUMBER OF LBAS							
23								(LSB)
24 ••• 31	Reserved							

The LBA RANGE NUMBER field indicates the number of the LBA range described by this LBA range descriptor. LBA range numbers shall start at zero and are incremented by one for each successive LBA range.

The NUMBER OF STORAGE ELEMENTS field indicates the number of storage elements described by this LBA range descriptor. A NUMBER OF STORAGE ELEMENTS field set to zero indicates that the number of storage elements described by this LBA range descriptor is not reported.

The STARTING LBA field indicates the starting LBA described by this LBA range descriptor.

The NUMBER OF LBAS field indicates the number of logical blocks described by this LBA range descriptor. The NUMBER OF LBAS field shall not be set to zero.

6.6.8 Format Presets VPD page

6.6.8.1 Format Presets VPD page overview

The Format Presets VPD page (see table 276) provides a means to retrieve the descriptions for the presets that are specified by the PRESET IDENTIFIER field in the FORMAT WITH PRESET command (see 5.5).

Table 276 — Format Presets VPD page

Byte \ Bit	7	6	5	4	3	2	1	0
0	PERIPHERAL QUALIFIER			PERIPHERAL DEVICE TYPE				
1	PAGE CODE (B8h)							
2	(MSB)	PAGE LENGTH (n - 3)						
3								(LSB)
	Format preset descriptor list							
4 ••• 67	Format preset descriptor [first] (see table 277)							
	⋮							
n-63 ••• n	Format preset descriptor [last] (see table 277)							

The PERIPHERAL QUALIFIER field, PERIPHERAL DEVICE TYPE field, and PAGE LENGTH field are defined in SPC-6.

The PAGE CODE field is defined in SPC-6 and shall be set as shown in table 276 for the Format Presets VPD page.

Each format preset descriptor (see table 277) contains information about one preset used by the FORMAT WITH PRESET command.

Table 277 — Format preset descriptor

Byte \ Bit	7	6	5	4	3	2	1	0
0	(MSB)							
•••	PRESET IDENTIFIER							
3								(LSB)
4	SCHEMA TYPE							
5	Reserved							
6								
7	Reserved				LOGICAL BLOCKS PER PHYSICAL BLOCK EXPONENT			
8	(MSB)							
•••	LOGICAL BLOCK LENGTH							
11								(LSB)
12	Reserved							
•••								
15								
16	(MSB)							
•••	DESIGNED LAST LOGICAL BLOCK ADDRESS							
23								(LSB)
24	Reserved							
•••								
37								
38	FMTPINFO		Reserved			PROTECTION FIELD USAGE		
39	Reserved				PROTECTION INTERVAL EXPONENT			
40	(MSB)							
•••	Schema type specific information							
63								(LSB)

If the contents of the PRESET IDENTIFIER field (see table 278) is equal to the value in the PRESET IDENTIFIER field in a FORMAT WITH PRESET command (see 5.5), then this format preset descriptor is selected for use in the processing of that FORMAT WITH PRESET command.

Table 278 — PRESET IDENTIFIER field

Code	Description	SCHEMA TYPE field (see table 279)
0000_0000h	Default non-zoned with 512-bytes of user data in each logical block [a]	01h
0000_0001h	Obsolete	
0000_0002h	Default host managed zoned block device model (see ZBC-2) with 512-bytes of user data in each logical block [a]	03h
0000_0003h	Default zone domains and realms zoned block device model (see ZBC-2) with 512-bytes of user data in each logical block [a]	04h
0000_0004h to 0000_00FFh	Reserved	
0000_0100h	Default non-zoned with 4 096-bytes of user data in each logical block [b]	01h
0000_0101h	Obsolete	
0000_0102h	Default host managed zoned block device model (see ZBC-2) with 4 096-bytes of user data in each logical block [b]	03h
0000_0103h	Default zone domains and realms zoned block device model (see ZBC-2) with 4 096-bytes of user data in each logical block [b]	04h
0000_0104h to 0000_FFFFh	Reserved	
all others	Vendor specific	

[a] In this format preset descriptor, the LOGICAL BLOCK LENGTH field shall be set to 0000_0200h and the LOGICAL BLOCKS PER PHYSICAL BLOCK EXPONENT field shall be set to 0h or 3h.

[b] In this format preset descriptor, the LOGICAL BLOCK LENGTH field shall be set to 0000_1000h and the LOGICAL BLOCKS PER PHYSICAL BLOCK EXPONENT field shall be set to 0h.

The SCHEMA TYPE field (see table 279) indicates the overall format described by this format preset descriptor.

Table 279 — SCHEMA TYPE field

Code	Description	Schema type specific information (see table 277)
01h	This format preset descriptor does not define any zones (i.e., none of the requirements in ZBC-2 apply).	Reserved
02h	Obsolete	
03h	This format preset descriptor defines the zones in the host managed zoned block device model (see ZBC-2).	See 6.6.8.2
04h	This format preset descriptor defines zones in the zone domains and realms zoned block device model (see ZBC-2).	See 6.6.8.3
all others	Reserved	

The LOGICAL BLOCKS PER PHYSICAL BLOCK EXPONENT field indicates the value of x where the number of logical blocks in one physical block is 2^x for this format preset descriptor.

The LOGICAL BLOCK LENGTH field indicates the number of bytes of user data in a logical block for this format preset descriptor.

The DESIGNED LAST LOGICAL BLOCK ADDRESS field indicates the value that this format preset descriptor is designed to return in the RETURNED LOGICAL BLOCK ADDRESS field of a subsequent READ CAPACITY (16) command (see 5.21). The value actually returned in the RETURNED LOGICAL BLOCK ADDRESS field of a subsequent READ CAPACITY (16) command is affected by the value of the FMTMAXLBA bit (see 5.5) and the condition of the media being formatted.

The FMTPINFO field, PROTECTION FIELD USAGE field, and PROTECTION INTERVAL EXPONENT field are defined in 5.4.

The contents of the schema type specific information depends on the schema type (see table 279).

6.6.8.2 Host managed zones schema type specific information

If the SCHEMA TYPE field is set to 03h, then the schema type specific information is shown in table 280.

Table 280 — Host managed zones schema type specific information

Byte \ Bit	7	6	5	4	3	2	1	0
0	LOW LBA CONVENTIONAL ZONES PERCENTAGE							
1	HIGH LBA CONVENTIONAL ZONES PERCENTAGE							
2	Reserved							
3	Reserved				DESIGNED ZONE ALIGNMENT METHOD			
4	(MSB)							
•••	DESIGNED ZONE STARTING LBA GRANULARITY							
11								(LSB)
12	(MSB)							
•••	LOGICAL BLOCKS PER ZONE							
15								(LSB)
16								
•••	Reserved							
23								

The LOW LBA CONVENTIONAL ZONES PERCENTAGE field indicates the approximate percentage of the zones formatted as conventional zones (see ZBC-2) starting with LBA 0.

The HIGH LBA CONVENTIONAL ZONES PERCENTAGE field indicates the approximate percentage of the zones formatted as conventional zones ending with the largest valued LBA formatted.

The units for the conventional zones percentages are tenths of a percent (e.g., 0 indicates 0%, 1 indicates 0.1%, 10 indicates 1.0%, 255 indicates 25.5%).

The actual number of zones formatted as conventional zones depends on several factors (e.g., the number of logical blocks in a zone). Zones that are not formatted as conventional zones are formatted as sequential write required write pointer zones (see ZBC-2).

The DESIGNED ZONE ALIGNMENT METHOD field indicates the value that this format preset descriptor is designed to return in the ZONE ALIGNMENT METHOD field in the Zoned Block Device Characteristics VPD page (see ZBC-2).

The DESIGNED ZONE STARTING LBA GRANULARITY field indicates the value that this format preset descriptor is designed to return in the ZONE STARTING LBA GRANULARITY field in the Zoned Block Device Characteristics VPD page.

The LOGICAL BLOCKS PER ZONE field indicates the number of logical blocks in each zone. A LOGICAL BLOCKS PER ZONE field set to zero indicates that the number of logical blocks in a zone is not reported.

6.6.8.3 Zone domains and realms schema type specific information

If the SCHEMA TYPE field is set to 04h, the schema type specific information is shown in table 281.

Table 281 — Zone domains and realms schema type specific information

Byte \ Bit	7	6	5	4	3	2	1	0
0	ZONE TYPE FOR ZONE DOMAIN 0				ZONE TYPE FOR ZONE DOMAIN 1			
1	ZONE TYPE FOR ZONE DOMAIN 2				ZONE TYPE FOR ZONE DOMAIN 3			
2	Reserved							
3	Reserved				DESIGNED ZONE ALIGNMENT METHOD			
4 ••• 11	DESIGNED ZONE STARTING LBA GRANULARITY							
12 ••• 15	(MSB) LOGICAL BLOCKS PER ZONE (LSB)							
16 ••• 23	(MSB) DESIGNED ZONED MAXIMUM ADDRESS (LSB)							

The DESIGNED ZONE ALIGNMENT METHOD field indicates the value that this format preset descriptor is designed to return in the ZONE ALIGNMENT METHOD field in the Zoned Block Device Characteristics VPD page (see ZBC-2).

The DESIGNED ZONE STARTING LBA GRANULARITY field indicates the value that this format preset descriptor is designed to return in the ZONE STARTING LBA GRANULARITY field in the Zoned Block Device Characteristics VPD page.

Each ZONE TYPE FOR ZONE DOMAIN field indicates the zone type (see ZBC-2) for zones in the indicated zone domain. A ZONE TYPE FOR ZONE DOMAIN field set to zero indicates that:

a) the indicated zone domain is not included in the format;
b) more than one zone type is part of the format for the indicated zone domain; or
c) the zone type is not reported for the indicated zone domain.

The LOGICAL BLOCKS PER ZONE field indicates the number of logical blocks in each zone. A LOGICAL BLOCKS PER ZONE field set to zero indicates that the number of logical blocks in a zone is not reported.

The DESIGNED ZONED MAXIMUM ADDRESS field indicates the value that this format preset descriptor is designed to return in the MAXIMUM LBA field returned by a subsequent REPORT ZONES command (see ZBC-2).

6.6.9 Logical Block Provisioning VPD page

The Logical Block Provisioning VPD page (see table 282) provides the application client with logical block provisioning related operating parameters of the logical unit.

Table 282 — Logical Block Provisioning VPD page

Byte \ Bit	7	6	5	4	3	2	1	0
0	PERIPHERAL QUALIFIER			PERIPHERAL DEVICE TYPE (00000b)				
1	PAGE CODE (B2h)							
2	(MSB)	PAGE LENGTH (0004h or (n - 3))						
3								(LSB)
4	THRESHOLD EXPONENT							
5	LBPU	LBPWS	LBPWS10	LBPRZ			ANC_SUP	DP
6	MINIMUM PERCENTAGE					PROVISIONING TYPE		
7	THRESHOLD PERCENTAGE							
8	PROVISIONING GROUP DESCRIPTOR (if any)							
•••								
n								

The PERIPHERAL QUALIFIER field is defined in SPC-6.

The PAGE CODE field and PERIPHERAL DEVICE TYPE field are defined in SPC-6 and shall be set to the value shown in table 282 for the Logical Block Provisioning VPD page.

The PAGE LENGTH field is defined in SPC-6. If the DP bit is set to zero, then the PAGE LENGTH field shall be set to 0004h. If the DP bit is set to one, then the PAGE LENGTH field shall be set to n-3 as shown in table 282.

If the Logical Block Provisioning log page (see 6.4.5.1) is supported, then the logical unit shall support:

a) logical block provisioning threshold sets; or
b) logical block provisioning percentages.

The THRESHOLD EXPONENT field indicates the threshold set size as described in 4.7.3.7.2. A THRESHOLD EXPONENT field set to zero indicates that the logical unit does not support logical block provisioning threshold sets.

If logical block provisioning threshold sets are supported, then the threshold exponent shall be a non-zero value selected such that:

$$(\text{capacity} \div 2^{(\text{threshold exponent})}) < 2^{(32)}$$

where:

capacity	is 1 + the LBA of the last logical block as returned in the READ CAPACITY (16) parameter data (see 5.21.2) (i.e., the number of logical blocks on the direct access block device);
threshold exponent	is the contents of the THRESHOLD EXPONENT field; and
$2^{(32)}$	is the constant value 1_0000_0000h (i.e., 4 294 967 296).

A THRESHOLD PERCENTAGE field set to zero indicates that the logical unit does not support logical block provisioning percentages. If logical block provisioning percentages are supported, then the threshold percentage shall be set to a non-zero value selected from the values in table 283. The units for the threshold percentage is tenths of a percent. This percentage represents the range over which logical block provisioning threshold percentages operates as described in 4.7.3.7.3.

Table 283 — THRESHOLD PERCENTAGE field

Code	Description
0	The logical unit does not support logical block provisioning percentages
1 to 255 (i.e., 01h to FFh)	0.1% to 25.5% of the total allocation resources

A MINIMUM PERCENTAGE field set to zero indicates that the logical unit does not report a minimum percentage of resources required by the device. A MINIMUM PERCENTAGE field set to a non-zero value indicates the minimum percentage of resources required by the device server as described in table 284. This value indicates the point where a device server may begin device initiated advanced background operations.

Table 284 — MINIMUM PERCENTAGE field

Code	Description
0	The logical unit does not report a minimum percentage of resources required
1 to 30 (i.e., 01h to 1Eh)	1% to 30% of the total allocation resources
All others	Reserved

A logical block provisioning UNMAP command (LBPU) bit set to one indicates that the device server supports the UNMAP command (see 5.35). An LBPU bit set to zero indicates that the device server does not support the UNMAP command.

A logical block provisioning WRITE SAME (16) command (LBPWS) bit set to one indicates that the device server supports the use of the WRITE SAME (16) command (see 5.53) to unmap LBAs. An LBPWS bit set to zero indicates that the device server does not support the use of the WRITE SAME (16) command to unmap LBAs.

A logical block provisioning WRITE SAME (10) command (LBPWS10) bit set to one indicates that the device server supports the use of the WRITE SAME (10) command (see 5.52) to unmap LBAs. An LBPWS10 bit set to zero indicates that the device server does not support the use of the WRITE SAME (10) command to unmap LBAs.

The logical block provisioning read zeros (LBPRZ) field is described in table 285. See table 10 for the definition of the logical block data returned by a read operation from an unmapped LBA for the different values of the LBPRZ field.

Table 285 — LBPRZ field

Code	Description
000b	The logical block data represented by unmapped LBAs (see 4.7.4.4) is vendor specific
xx1b	The logical block data represented by unmapped LBAs is set to zeros
010b	The logical block data represented by unmapped LBAs is set to the provisioning initialization pattern
all others	Reserved

An anchor supported (ANC_SUP) bit set to one indicates that the device server supports anchored LBAs (see 4.7.1). An ANC_SUP bit set to zero indicates that the device server does not support anchored LBAs.

A descriptor present (DP) bit set to one indicates a PROVISIONING GROUP DESCRIPTOR field is present. A DP bit set to zero indicates a PROVISIONING GROUP DESCRIPTOR field is not present.

The PROVISIONING TYPE field is shown in table 286.

Table 286 — PROVISIONING TYPE field

Code	Description
000b	The device server does not report a provisioning type or may be fully provisioned.
001b	The logical unit is resource provisioned (see 4.7.3.2).
010b	The logical unit is thin provisioned (see 4.7.3.3).
All others	Reserved

The PROVISIONING GROUP DESCRIPTOR field, if any, contains a designation descriptor (see SPC-6) for the LBA mapping resources used by this logical unit.

If a PROVISIONING GROUP DESCRIPTOR field is present:

a) the ASSOCIATION field shall be set to 00b (i.e. logical unit); and
b) the DESIGNATOR TYPE field shall be set to:
   A) 1h (i.e., T10 vendor ID based);
   B) 3h (i.e., NAA); or
   C) Ah (i.e., UUID identifier).

6.6.10 Referrals VPD page

The Referrals VPD page (see table 287) contains parameters indicating characteristics of the user data segments contained within this logical unit.

Table 287 — Referrals VPD page

Byte \ Bit	7	6	5	4	3	2	1	0
0	PERIPHERAL QUALIFIER			PERIPHERAL DEVICE TYPE (00000b)				
1	PAGE CODE (B3h)							
2	(MSB)	PAGE LENGTH (000Ch)						
3								(LSB)
4 ••• 7	Reserved							
8	(MSB)	USER DATA SEGMENT SIZE						
•••								
11								(LSB)
12	(MSB)	USER DATA SEGMENT MULTIPLIER						
•••								
15								(LSB)

The PERIPHERAL QUALIFIER field is defined in SPC-6.

The PAGE CODE field, PERIPHERAL DEVICE TYPE field, and PAGE LENGTH field are defined in SPC-6 and shall be set to the values shown in table 287 for the Referrals VPD page.

A USER DATA SEGMENT SIZE field set to a non-zero value indicates the number of contiguous logical blocks in a user data segment (see 4.26.2). A USER DATA SEGMENT SIZE field set to zero indicates the user data segment size information (i.e., the first user data segment LBA to the last user data segment LBA) is as indicated in the user data segment referral descriptor (see table 18).

The USER DATA SEGMENT MULTIPLIER field is used by an application client to calculate the beginning LBA of each user data segment as described in 4.26.2.

6.6.11 Third-party Copy VPD page

6.6.11.1 Third-party Copy VPD page overview

The Third-party Copy VPD page (see SPC-6) provides a means to retrieve third-party copy descriptors including a descriptor that describes operating parameters for the POPULATE TOKEN command (see 5.12) and the WRITE USING TOKEN command (see 5.59).

6.6.11.2 Block device third-party copy descriptor type codes

Block device third-party copy descriptor type codes (see table 288) indicate which third-party copy descriptor is being returned.

Table 288 — Block device third-party copy descriptor type codes

Descriptor code	Third-party copy descriptor name	Reference	Support requirements
0000h	Block Device ROD Limits	6.6.11.3	See [a]
All other codes	See SPC-6	See SPC-6	See SPC-6

[a] Mandatory if the POPULATE TOKEN command and the WRITE USING TOKEN command are supported.

6.6.11.3 Block Device ROD Limits descriptor

The Block Device ROD Limits descriptor (see table 289) is a third-party copy descriptor in the Third-party Copy VPD page (see SPC-6) that provides the application client with a method to obtain operating parameters for direct access block device ROD token operations (see 4.28).

Table 289 — Block Device ROD Limits descriptor

Byte \ Bit	7	6	5	4	3	2	1	0
0	(MSB)	THIRD-PARTY COPY DESCRIPTOR TYPE (0000h)						
1								(LSB)
2	(MSB)	THIRD-PARTY COPY DESCRIPTOR LENGTH (0020h)						
3								(LSB)
4 ••• 9	Vendor specific							
10	(MSB)	MAXIMUM RANGE DESCRIPTORS						
11								(LSB)
12	(MSB)	MAXIMUM INACTIVITY TIMEOUT						
•••								
15								(LSB)
16	(MSB)	DEFAULT INACTIVITY TIMEOUT						
•••								
19								(LSB)
20	(MSB)	MAXIMUM TOKEN TRANSFER SIZE						
•••								
27								(LSB)
28	(MSB)	OPTIMAL TRANSFER COUNT						
•••								
35								(LSB)

The THIRD-PARTY COPY DESCRIPTOR TYPE field and the THIRD-PARTY COPY DESCRIPTOR LENGTH field are defined in SPC-6 and shall be set to the values shown in table 289 for the Block Device ROD Limits descriptor.

The MAXIMUM RANGE DESCRIPTORS field indicates the maximum number of block device range descriptors that may be specified in the parameter data of a POPULATE TOKEN command (see 5.12) and the parameter data of a WRITE USING TOKEN command (see 5.59). If the MAXIMUM RANGE DESCRIPTORS field is set to zero, then the copy manager does not report a maximum number of block device range descriptors.

The MAXIMUM INACTIVITY TIMEOUT field indicates the maximum value in the INACTIVITY TIMEOUT field of the parameter data of a POPULATE TOKEN command that is accepted by the copy manager. If the MAXIMUM INACTIVITY TIMEOUT field is set to zero, then the device server does not report a maximum inactivity timeout value. If the MAXIMUM INACTIVITY TIMEOUT field is set to FFFF_FFFFh then there is no maximum value that may be specified in the INACTIVITY TIMEOUT field in the parameter data of the POPULATE TOKEN command.

The DEFAULT INACTIVITY TIMEOUT field indicates the inactivity timeout value that is used if the INACTIVITY TIMEOUT field in the parameter data of a POPULATE TOKEN command is set to zero. If the DEFAULT INACTIVITY TIMEOUT field is set to zero, then the copy manager does not report a default inactivity timeout value.

The MAXIMUM TOKEN TRANSFER SIZE field indicates the maximum size in logical blocks that may be specified by the sum of the NUMBER OF LOGICAL BLOCKS fields in all block device range descriptors for the following commands:

a) POPULATE TOKEN; and
b) WRITE USING TOKEN.

If the MAXIMUM TOKEN TRANSFER SIZE field is set to zero, then the copy manager does not report a maximum token transfer size.

If the MAXIMUM BYTES IN BLOCK ROD field in the block ROD device type specific features descriptor in the ROD token features third-party copy descriptor in the Third-party Copy VPD page (see SPC-6) is reported, then the MAXIMUM TOKEN TRANSFER SIZE field shall be set to the number of logical blocks that represents the value in the MAXIMUM BYTES IN BLOCK ROD field in the block ROD device type specific features descriptor in the ROD token features third-party copy descriptor in the Third-party Copy VPD page.

The OPTIMAL TRANSFER COUNT field indicates the optimal number of logical blocks that the copy manager is able to transfer. If the sum of the NUMBER OF LOGICAL BLOCKS fields in all block device range descriptors in the parameter data of a POPULATE TOKEN command or the parameter data of a WRITE USING TOKEN command exceeds this value then, a delay in processing the request may be incurred. If the field is set to zero, then the copy manager does not report an optimal transfer count.

If the OPTIMAL BYTES IN BLOCK ROD TRANSFER field in the block ROD device type specific features descriptor in the ROD token features third-party copy descriptor in the Third-party Copy VPD page is reported, then the OPTIMAL TRANSFER COUNT field shall be set to the number of logical blocks that represents the value in the OPTIMAL BYTES IN BLOCK ROD TRANSFER field in the block ROD device type specific features descriptor in the ROD token features third-party copy descriptor in the Third-party Copy VPD page.

6.6.12 Supported Block Lengths and Protection Types VPD page

The Supported Block Lengths and Protection Types VPD page (see table 290) contains parameters indicating the specific protection types (see 4.21) supported for each supported logical block length. If the SBLP bit is set to zero in the Control mode page (See SPC-6), then the device server shall not support this VPD page.

Table 290 — Supported Block Lengths and Protection Types VPD page

Bit / Byte	7	6	5	4	3	2	1	0
0	PERIPHERAL QUALIFIER			PERIPHERAL DEVICE TYPE				
1	PAGE CODE (B4h)							
2	(MSB)	PAGE LENGTH (n-3)						
3								(LSB)
	Logical block length and protection types descriptor list							
4 ••• 11	Logical block length and protection types descriptor [first]							
	⋮							
n-7 ••• n	Logical block length and protection types descriptor [last]							

The PERIPHERAL QUALIFIER field and PERIPHERAL DEVICE TYPE field are defined in SPC-6.

The PAGE CODE field is defined in SPC-6 and shall be set to the value shown in table 290 for the Supported Logical Block Lengths and Protection Types VPD page.

The PAGE LENGTH field is defined in SPC-6.

The logical block length and protection types descriptor list shall contain one logical block length and protection types descriptor for each logical block length that the device server supports.

Each logical block length and protection types descriptor describes the protection types supported for the logical block length indicated (see table 291).

Table 291 — Logical block length and protection types descriptor format

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)	LOGICAL BLOCK LENGTH						
•••								
3								(LSB)
4	Reserved	P_I_I_SUP	Reserved		NO_PI_CHK	GRD_CHK	APP_CHK	REF_CHK
5	Reserved				T3PS	T2PS	T1PS	T0PS
6	Reserved							
7								

The LOGICAL BLOCK LENGTH field indicates the logical block length in bytes of user data that is supported by the device server for which the device server supports the protection types identified in the P_I_I_SUP bit, T3PS bit, T2PS bit, T1PS bit, T0PS bit, GRD_CHK bit, APP_CHK bit, and REF_CHK bit. If protection information is not supported, then the T0PS bit shall be set to one.

A protection information interval supported (P_I_I_SUP) bit set to one indicates that the logical unit supports protection information intervals for the indicated logical block length. A P_I_I_SUP bit set to zero indicates that the logical unit does not support protection information intervals for the indicated logical block length.

A no protection information checking (NO_PI_CHK) bit set to one indicates that the device server disables checking of all protection information for the associated protection information interval when performing a write operation if:

a) the LOGICAL BLOCK APPLICATION TAG field is set to FFFFh and type 1 protection (see 4.21) is enabled;
b) the LOGICAL BLOCK APPLICATION TAG field is set to FFFFh and type 2 protection is enabled; or
c) the LOGICAL BLOCK APPLICATION TAG field is set to FFFFh, the LOGICAL BLOCK REFERENCE TAG field is set to FFFF FFFFh, and type 3 protection is enabled.

A NO_PI_CHK bit set to zero indicates that the device server checks protection information as specified by the WRPROTECT field when performing a write operation.

A guard check (GRD_CHK) bit set to zero indicates that the device server does not check the LOGICAL BLOCK GUARD field in the protection information, if any, for the indicated logical block length. A GRD_CHK bit set to one indicates that the device server checks the LOGICAL BLOCK GUARD field in the protection information, if any, for the indicated logical block length.

An application tag check (APP_CHK) bit set to zero indicates that the device server does not check the LOGICAL BLOCK APPLICATION TAG field in the protection information, if any, for the indicated logical block length. An APP_CHK bit set to one indicates that the device server checks the LOGICAL BLOCK APPLICATION TAG field in the protection information, if any, for the indicated logical block length.

A reference tag check (REF_CHK) bit set to zero indicates that the device server does not check the LOGICAL BLOCK REFERENCE TAG field in the protection information, if any, for the indicated logical block length. A REF_CHK bit set to one indicates that the device server checks the LOGICAL BLOCK REFERENCE TAG field in the protection information, if any, for the indicated logical block length.

A type 3 protection supported (T3PS) bit set to one indicates that type 3 protection is supported for the indicated logical block length. A T3PS bit set to zero indicates that type 3 protection is not supported for the indicated logical block length.

A type 2 protection supported (T2PS) bit set to one indicates that type 2 protection is supported for the indicated logical block length. A T2PS bit set to zero indicates that type 2 protection is not supported for the indicated logical block length.

A type 1 protection supported (T1PS) bit set to one indicates that type 1 protection is supported for the indicated logical block length. A T1PS bit set to zero indicates that type 1 protection is not supported for the indicated logical block length.

A type 0 protection supported (T0PS) bit set to one indicates that type 0 protection is supported for the indicated logical block length. A T0PS bit set to zero indicates that type 0 protection is not supported for the indicated logical block length.

6.7 Copy manager parameters

The copy manager parameters are device type specific data for ROD tokens (see SPC-6) created by a copy manager for a direct access block device (see 4.28) and are shown in table 292.

Table 292 — ROD token device type specific data

<table>
<tr><th>Byte \ Bit</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td>0</td><td colspan="8">(MSB)</td></tr>
<tr><td>•••</td><td colspan="8">LOGICAL BLOCK LENGTH IN BYTES</td></tr>
<tr><td>3</td><td colspan="8">(LSB)</td></tr>
<tr><td>4</td><td colspan="4">Reserved</td><td colspan="3">P_TYPE</td><td>PROT_EN</td></tr>
<tr><td>5</td><td colspan="4">P_I_EXPONENT</td><td colspan="4">LOGICAL BLOCKS PER PHYSICAL BLOCK EXPONENT</td></tr>
<tr><td>6</td><td>LBPME</td><td>LBPRZ</td><td colspan="6">(MSB) LOWEST ALIGNED LOGICAL BLOCK ADDRESS</td></tr>
<tr><td>7</td><td colspan="8">(LSB)</td></tr>
<tr><td>8</td><td colspan="8" rowspan="3">Reserved</td></tr>
<tr><td>•••</td></tr>
<tr><td>31</td></tr>
</table>

The LOGICAL BLOCK LENGTH IN BYTES field, the P_TYPE field, the PROT_EN bit, the P_I_EXPONENT field, the LOGICAL BLOCKS PER PHYSICAL BLOCK EXPONENT field, the LBPME bit, the LBPRZ bit, and the LOWEST ALIGNED LOGICAL BLOCK ADDRESS field are defined in 5.21.2.

6.8 Logical block markup descriptors

6.8.1 Logical block markup descriptor overview

A logical block markup descriptor communicates information about anticipated usage of a logical block or range of logical blocks to the device server that manages access to that logical block. The requirements, if any, placed on a device server that receives a logical block markup descriptor are associated with the command that causes the device server to receive one or more logical block markup descriptors. Logical block markup descriptors are used in the IO Advice Hints Grouping mode page (see 6.5.7). Logical block markup descriptor formats and the meanings of logical block markup descriptor fields are described in 6.8.

6.8.2 Logical block markup descriptor formats and types

The format of an logical block markup descriptor is shown in table 293.

Table 293 — Logical block markup descriptor format

Byte \ Bit	7	6	5	4	3	2	1	0
0	Logical block markup descriptor type specific data				LBM DESCRIPTOR TYPE			
1	logical block markup descriptor type specific data							
•••								
n								

The LBM DESCRIPTOR TYPE field is shown in table 294.

Table 294 — LBM DESCRIPTOR TYPE field

Code	Logical block markup descriptor type	Reference
0h	Access patterns	6.8.3
all others	Reserved	

6.8.3 Access patterns logical block markup descriptors

6.8.3.1 Access patterns logical block markup descriptor format

The access patterns logical block markup descriptor format is shown in table 295.

Table 295 — Access patterns logical block markup descriptor format

Byte \ Bit	7	6	5	4	3	2	1	0
0	ACDLU	Reserved	RLBSR		LBM DESCRIPTOR TYPE (0h)			
1	OVERALL FREQUENCY		READ/WRITE FREQUENCY		WRITE SEQUENTIALITY		READ SEQUENTIALITY	
2	IO CLASS				SUBSEQUENT I/O		OSI PROXIMITY	
3	Reserved							

The LBM DESCRIPTOR TYPE field is defined in 6.8.2 and shall be set as shown in table 295 for the access patterns logical block markup descriptor format.

An ACDLU bit set to one specified that any logical blocks associated with this logical block markup descriptor have a high probability of being accessed during time intervals in which most logical blocks are not being accessed by any read commands or write commands. If the ACDLU bit is set to zero, then any logical blocks associated with this logical block markup descriptor have no utilization related probability of being read or written.

The RLBSR field (see table 296) specifies information about the relations of the logical blocks associated with a command that references this logical block markup descriptor. The device server may use this information to optimize resource management (e.g. reduce advanced background operations).

Table 296 — RLBSR field

Code	Description
00b	No information is provided about the relationship among the logical blocks associated with the command that references this logical block markup descriptor or the probability of subsequent reads to these logical blocks.
01b	Logical blocks associated with the command that references this logical block markup descriptor are related to each other (e.g. are a file or part of the same application client data structure). No information is provided about the probability of subsequent reads to these logical blocks.
10b	Reserved
11b	Logical blocks associated with the command that references this logical block markup descriptor are related to each other and a read to any logical block in the group of related logical blocks increases the probability of subsequent reads to other logical blocks in that group of related logical blocks.

The OVERALL FREQUENCY field (see table 297) specifies the probability that a logical block in the range associated with this logical block markup descriptor is likely to be accessed more frequently than other logical blocks stored on the same medium.

Table 297 — OVERALL FREQUENCY field

Code	Description
00b	Overall access frequency is unknown or equally probable.
01b	Overall accesses are less frequent than average.
10b	Overall accesses are more frequent than average.
11b	Reserved

The READ/WRITE FREQUENCY field (see table 298) specifies whether read operations or write operations are more probable to a logical block in the range associated with this logical block markup descriptor.

Table 298 — READ/WRITE FREQUENCY field

Code	Description
00b	Read operation frequency versus write operation frequency is unknown or equally probable.
01b	Read operations are more probable.
10b	Write operations are more probable.
11b	Reserved

Using the values shown in table 299:

a) the WRITE SEQUENTIALITY field specifies whether sequential write operations or random write operations are more probable to a logical block in the range associated with this logical block markup descriptor; and

b) the READ SEQUENTIALITY field specifies whether sequential read operations or random read operations are more probable to a logical block in the range associated with this logical block markup descriptor.

Table 299 — WRITE SEQUENTIALITY field and READ SEQUENTIALITY field

Code	Description
00b	Access sequentiality is unknown or equally probable.
01b	Random operations are more probable.
10b	Sequential operations are more probable.
11b	Reserved

The IO CLASS field (see table 300) specifies the classification of user data that is associated with this logical block markup descriptor.

Table 300 — IO CLASS field

Code	Description
0h	None specified
1h	IOs that specify this logical block markup descriptor are related to user data that describes other user data (e.g., file system metadata). [a]
4h	IOs that specify this logical block markup descriptor are related to a small collection of user data where small is defined by the application client. [a]
5h	IOs that specify this logical block markup descriptor are related to a large collection of user data where large is defined by the application client. [a]
all others	Reserved

[a] See Differentiated Storage Services

The SUBSEQUENT I/O field (see table 301) specifies the probability that the application client may be delaying the sending of additional read commands or write commands to the device server until the completion of a read command or a write command for a logical block associated with this logical block markup descriptor (e.g., commands are being delayed by the application client's file system until the completion of write commands to logical blocks in a file system allocation bit map that are associated with this logical block markup descriptor).

Table 301 — SUBSEQUENT I/O field

Code	Description
00b	The probability is unknown whether the application client is delaying the sending of commands to the device server until completion of a read command or a write command for a logical block associated with this logical block markup descriptor.
01b	The probability is low that the application client is delaying the sending of commands to the device server until completion of a read command or a write command for a logical block associated with this logical block markup descriptor.
10b	The probability is high that the application client is delaying the sending of commands to the device server until completion of a read command or a write command for a logical block associated with this logical block markup descriptor.
11b	Reserved

The OSI PROXIMITY field (see table 302) specifies the probability that any logical block associated with this logical block markup descriptor is likely to be accessed during an operating system or file system initialization operation (e.g., a boot block).

Table 302 — OSI PROXIMITY field

Code	Description
00b	Access during initialization probability is unknown or equally probable.
01b	Accesses are not probable during an initialization process.
10b	Accesses are probable during an initialization process.
11b	Reserved

Methods to determine whether an operating system or file system initialization operation is in progress are outside the scope of this standard.

EXAMPLE - Possible sources of knowledge about when an operating system initialization operation is occurring depend on the system in which the device server is participating and include prior knowledge of which LBA accesses are associated with an operating system initialization or detection of a power on event.

6.8.3.2 Access patterns logical block markup descriptor usage considerations

Device servers may ignore all or part of the information contained in an access patterns logical block markup descriptor.

EXAMPLE 1 - A device server that processes only the OVERALL FREQUENCY field.

EXAMPLE 2 - A device server that processes all logical block markup descriptor fields except the WRITE SEQUENTIALITY field and the READ SEQUENTIALITY field.

EXAMPLE 3 - In a product with a focus on sequential writes (e.g., a disk drive based on shingled magnetic recording), the device server ignores all values in all fields except a WRITE SEQUENTIALITY field that is set to 10b.

Although Annex H shows some possible combinations of the fields in an access patterns logical block markup descriptor, this standard places no requirements on how the device server interprets the interactions, if any, between the fields in an access patterns logical block markup descriptor.

Annex A
(normative)

SBC feature sets

A.1 Overview

This annex defines SCSI feature sets (see SPC-6) for block devices.

Table A.1 lists the feature sets.

Table A.1 — Feature sets

Feature set	Feature set code	Reference
Reserved	0000h	
SBC Base 2010	0102h	Clause A.2
SBC Base 2016	0101h	Clause A.3
Basic Provisioning 2016	0103h	Clause A.4
Drive Maintenance 2016	0104h	Clause A.5
Reserved	all others	

A.2 SBC Base 2010 feature set

A.2.1 SBC Base 2010 feature set overview

The SBC Base 2010 feature set includes features intended for use by operating system direct access block storage device class drivers during operating system initialization and runtime. The features defined in this feature set are intended to provide compatibility with software designed for device servers that don't support 16 byte media access commands.

Device servers that support the SBC Base 2010 feature set may also support the SBC Base 2016 feature set (see clause A.3)

Table A.2 lists the commands that are mandatory or have additional mandatory requirements for support of the SBC Base 2010 feature set.

Table A.2 — Commands mandatory for the SBC Base 2010 feature set

Command	Additional requirements reference	Reference
FORMAT UNIT	A.3.3.1 [a]	5.4
INQUIRY	n/a	SPC-6
MODE SELECT (10)	A.3.4 [a]	SPC-6
MODE SENSE (10)	A.3.4 [a]	SPC-6
READ (10)	n/a	5.16
READ CAPACITY (10)	A.2.2.1	5.20
REPORT LUNS	n/a	SPC-6
REQUEST SENSE	n/a	SPC-6
START STOP UNIT	n/a	5.31
SYNCHRONIZE CACHE (10)	A.2.2.2	5.33
TEST UNIT READY	n/a	SPC-6
WRITE (10)	n/a	5.40
WRITE SAME (10)	A.2.2.3	5.52
[a] The additional requirements for this command are the same as the SBC Base 2016 additional requirements for this command.		

Table A.3 lists the SBC Base 2010 mandatory block descriptors and mode pages.

Table A.3 — Block descriptor and mode pages mandatory for the SBC Base 2010 feature set

Mode page	Additional requirements reference	Reference
Block descriptors		
Mode parameter block descriptor	A.3.4.1 [a]	6.5.2
Mode pages		
Caching	A.3.4.2 [a]	6.5.6
Control	A.3.4.3 [a]	SPC-6
Control Extension	n/a	SPC-6
Read-Write Error Recovery	A.3.4.5 [a]	6.5.10
[a] The additional requirements for this block descriptor or mode page are the same as the SBC Base 2016 additional requirements for this block descriptor or mode page.		

Table A.4 lists the SBC Base 2010 feature set VPD pages.

Table A.4 — VPD pages mandatory for the SBC Base 2010 feature set

VPD page	Additional requirements reference	Reference
ATA Information [a]	n/a	SAT-4
Block Device Characteristics	A.3.5.1 [b]	6.6.2
Device Identification [c]	n/a	SPC-6
Extended INQUIRY Data [c]	A.3.5.3 [b]	SPC-6
Mode Page Policy [c]	n/a	SPC-6
SCSI Feature Sets [c]	n/a	SPC-6
Supported VPD Pages [c]	n/a	SPC-6
Power Condition	n/a	SPC-6

[a] Mandatory for devices that implement SAT-4, optional for all other device types
[b] The additional requirements for this command are the same as the SBC Base 2016 additional requirements for this command.
[c] VPD page shall be available without incurring any medium access delays even if the device server is not ready for other commands (i.e., the device server shall not return ASCII spaces (20h) in ASCII fields and zeros in other fields until the data is available from the media).

A.2.2 SBC Base 2010 feature set commands

A.2.2.1 READ CAPACITY (10) command

The requirements for the READ CAPACITY (10) command are the same as the requirements for the READ CAPACITY (16) command in the SBC Base 2016 feature set (see A.3.3.2).

A.2.2.2 SYNCHRONIZE CACHE (10) command

The requirements for the SYNCHRONIZE CACHE (10) command are the same as the requirements for the SYNCHRONIZE CACHE (16) command in the SBC Base 2016 feature set (see A.3.3.6).

A.2.2.3 WRITE SAME (10) command

The requirements for the WRITE SAME (10) command are the same as the requirements for the WRITE SAME (16) command in the SBC Base 2016 feature set (see A.3.3.7).

A.3 SBC Base 2016 feature set

A.3.1 SBC Base 2016 feature set overview

The SBC Base 2016 feature set includes features used by operating system direct access block storage device class drivers during operating system initialization and runtime. The features defined in this feature set provide application clients with the versions of the commands that offer the largest LBA range and most extensibility of commands in an SBC base feature set (e.g., requires support of the 16 byte versions of commands that have both 10 byte and 16 byte versions).

Device servers that support the SBC Base 2016 feature set may support the SBC Base 2010 feature set (see clause A.2).

Table A.5 lists the commands that are mandatory or have additional mandatory requirements for support of the SBC Base 2016 feature set.

Table A.5 — Commands mandatory for the SBC Base 2016 feature set

Command	Additional requirements reference	Reference
FORMAT UNIT	A.3.3.1	5.4
INQUIRY	n/a	SPC-6
MODE SELECT (10)	A.3.4	SPC-6
MODE SENSE (10)	A.3.4	SPC-6
READ (16)	n/a	5.18
READ CAPACITY (16)	A.3.3.2	5.21
REPORT LUNS	n/a	SPC-6
REPORT SUPPORTED OPERATION CODES	A.3.3.3	SPC-6
REPORT SUPPORTED TASK MANAGEMENT FUNCTIONS	A.3.3.4	SPC-6
REQUEST SENSE	A.3.3.5	SPC-6
START STOP UNIT	n/a	5.31
SYNCHRONIZE CACHE (16)	A.3.3.6	5.34
TEST UNIT READY	n/a	SPC-6
WRITE (16)	n/a	5.42
WRITE SAME (16)	A.3.3.7	5.53

Table A.6 lists the SBC Base 2016 mandatory block descriptors and mode pages.

Table A.6 — Block descriptor and mode pages mandatory for the SBC Base 2010 feature set

Mode page	Additional requirements reference	Reference
Block descriptors		
Mode parameter block descriptor	A.3.4.1	6.5.2
Mode pages		
Caching	A.3.4.2	6.5.6
Control	A.3.4.3	SPC-6
Control Extension	n/a	SPC-6
Information Exceptions Control	A.3.4.4	6.5.8
Read-Write Error Recovery	A.3.4.5	6.5.10

Table A.7 lists the SBC Base 2016 feature set VPD pages.

Table A.7 — VPD pages mandatory for the SBC Base 2016 feature set

VPD page	Additional requirements reference	Reference
ATA Information [a]	n/a	SAT-4
Block Device Characteristics	A.3.5.1	6.6.2
Block Limits	A.3.5.2	6.6.4
Device Identification [b]	n/a	SPC-6
Extended INQUIRY Data [b]	A.3.5.3	SPC-6
Mode Page Policy [b]	n/a	SPC-6
SCSI Feature Sets [b]	n/a	SPC-6
Supported VPD Pages [b]	n/a	SPC-6
Power Condition	n/a	SPC-6

[a] Mandatory for devices that implement SAT-4, optional for all other device types
[b] VPD page shall be available without incurring any medium access delays even if the device server is not ready for other commands (i.e., the device server shall not return ASCII spaces (20h) in ASCII fields and zeros in other fields until the data is available from the media).

A.3.2 SBC Base 2016 feature set model

The device server shall support the Discovery 2016 feature set (see SPC-6).

If the device server supports LBAs greater than FFFF_FFFFh, then the device server shall support descriptor format sense data (see SPC-6).

If the device server implements a volatile writeback cache (see 4.15), then the device server shall:

a) set the V_SUP bit to one in the Extended INQUIRY Data VPD page (see SPC-6); and
b) support the FUA bit set to one in commands performing write operations.

If the device server implements a non-volatile writeback cache that may become volatile (see 4.15.10), then the device server shall:

a) set the NV_SUP bit to one in the Extended INQUIRY Data VPD page (see SPC-6).

A.3.3 SBC Base 2016 feature set commands

A.3.3.1 FORMAT UNIT command

If the device server does not implement logical block provisioning, then the default initialization pattern shall be set to all zeros. If the device server implements logical block provisioning then the default initialization pattern shall be set to all zeros or the provisioning initialization pattern.

The device server shall support the following bits in the FORMAT UNIT CDB (see 5.4.1):

a) the LONGLIST bit set to one (i.e., long parameter list header is supported);
b) the FMTDATA bit set to one (i.e., parameter list is supported); and
c) if protection information is supported, then the FMTPINFO field in the CDB and the PROTECTION FIELD USAGE field in the parameter data.

The device server shall support the following bits in the FORMAT UNIT parameter list header (see 5.4.2.2):

a) the FOV bit set to one (i.e., format options are supported);

b) the IP bit set to one (i.e., an initialization pattern descriptor is supported); and
c) the IMMED bit set to one.

A.3.3.2 READ CAPACITY (16) command

The device server shall report the following values in the READ CAPACITY (16) parameter data:

a) the LOGICAL BLOCKS PER PHYSICAL BLOCK EXPONENT field set to a value that represents the number of bytes that the device is able to write to the media as a unit (e.g., based on the NAND page size); and
b) the LOWEST ALIGNED LOGICAL BLOCK ADDRESS field set to 0000h (i.e., LBA 0 is aligned).

A.3.3.3 REPORT SUPPORTED OPERATION CODES command

The device server shall support the following bits or fields in the CDB:

a) the RCTD bit set to one (i.e., returning command timeouts descriptors is supported); and
b) the REPORTING OPTIONS field set to:
   A) 000b (i.e., operation codes and service actions);
   B) 001b (i.e., command support data for an operation code); and
   C) 010b (i.e., command support data for an operation code and service action).

The device server shall process the MODE field in the READ BUFFER command, if supported, and the MODE field in the WRITE BUFFER command as specifying service actions.

The values reported in the command timeouts descriptors for each command shall be based on at least a 1 MiB transfer length.

A.3.3.4 REPORT SUPPORTED TASK MANAGEMENT FUNCTIONS command

The device server shall support the following bit in the CDB:

a) the REPD bit set to one (i.e., return extended parameter data is supported).

A.3.3.5 REQUEST SENSE command

The device server shall support the following bit in the CDB:

a) the DESC bit set to zero; (i.e., fixed format sense data is supported).

If the device server supports LBAs greater than FFFF_FFFFh, then the device server shall support the following bit in the CDB:

a) the DESC bit set to one (i.e., descriptor format sense data is supported).

A.3.3.6 SYNCHRONIZE CACHE (16) command

The device server shall support the following bits or fields in the CDB:

a) the IMMED bit set to one.

If the Extended INQUIRY Data VPD page (see SPC-6) indicates that the device server does not contain volatile cache (i.e., the V_SUP bit is set to zero) and indicates that the device server does not contain non-volatile cache (i.e., the NV_SUP bit is set to zero), then the device server may complete the command without making any changes (i.e., implement the command as a no operation command).

A.3.3.7 WRITE SAME (16) command

If the device server supports the UNMAP bit in the CDB set to one, then the device server shall support the NDOB bit set to one.

A.3.4 SBC Base 2016 feature set mode pages

A.3.4.1 Mode parameter block descriptor

If the device supports LBAs greater than FFFF_FFFFh, then the device server shall support the long LBA mode parameter block descriptor (see 6.5.2.3).

The device server is not required to support changing its capacity by changing the number of logical blocks field using the MODE SELECT command.

A.3.4.2 Caching mode page

If the Extended INQUIRY Data VPD page (see SPC-6) indicates that the device server contains volatile cache (i.e., the V_SUP bit is set to one), then the device server shall support:

a) the WCE bit set to zero; and
b) the WCE bit set to one.

A.3.4.3 Control mode page

The device server shall support the following bits or fields:

a) the D_SENSE bit set to one (i.e., descriptor format sense data is enabled) if the device server supports LBAs greater than FFFF_FFFFh;
b) the QUEUE ALGORITHM MODIFIER field set to 1h (i.e., unrestricted reordering allowed); and
c) the QERR field set to 00b (i.e., CHECK CONDITION on one command does not affect others).

A.3.4.4 Informational Exceptions Control mode page

The device server shall support the following bits or fields:

a) the MRIE field set to one or more values including 0h (i.e., no reporting);
b) the EWASC bit set to zero;
c) the EWASC bit set to one;
d) the DEXCEPT bit set to zero; and
e) the DEXCEPT bit set to one.

Devices that implement SAT-4 shall support the MRIE field set to 6h (i.e., only report on request).

Devices that do not implement SAT-4 shall support the MRIE field set to:

a) 0h (i.e., no reporting);
b) 2h (i.e., establish unit attention condition);
c) 4h (i.e., unconditionally generate recovered error); and
d) 6h (i.e., only report on request).

A.3.4.5 Read-Write Error Recovery mode page

The device server shall support the following bit:

a) the AWRE bit set to one (i.e., automatic write reallocation enabled is supported).

Devices that do not implement SAT-4 shall support the following bit:

a) the ARRE bit set to one (i.e., automatic read reallocation enabled is supported).

A.3.5 SBC Base 2016 feature set VPD pages

A.3.5.1 Block Device Characteristics VPD page

The device server shall report the MEDIUM ROTATION RATE field set to a non-zero value.

A.3.5.2 Block Limits VPD page

The device server shall report the following values:

a) the MAXIMUM TRANSFER LENGTH field set to a value that represents at least 1 MiB (e.g., for a logical block size of 512 bytes, at least 00000800h);
b) the MAXIMUM WRITE SAME LENGTH field set to a value that represents at least 1 MiB (e.g., for a logical block size of 512 bytes, at least 00008000h;
c) the OPTIMAL TRANSFER LENGTH GRANULARITY field set to the minimum number of logical blocks that the device server prefers for random I/O; and
d) the OPTIMAL TRANSFER LENGTH field set to the maximum number of logical blocks that the device server prefers for streaming I/O.

The device server is not required to support:

a) the WSNZ bit set to zero (i.e., the WRITE SAME command NUMBER OF LOGICAL BLOCKS field set to zero is supported).

A.3.5.3 Extended INQUIRY Data VPD page

If the device server implements a non-volatile writeback cache that may become volatile (see 4.15.10), then the device server shall report the NV_SUP bit set to one.

If the device server implements a volatile writeback cache, then the device server shall report the V_SUP bit set to one.

The device server is not required to support:

a) the HEADSUP bit set to one (i.e., HEAD OF QUEUE task attribute is supported); and
b) the ORDSUP bit set to one (i.e., ORDERED task attribute is supported).

A.4 Basic Provisioning 2016 feature set

A.4.1 Basic Provisioning 2016 feature set overview

The Basic Provisioning 2016 feature set includes features related to logical block provisioning.

Logical units that support the Basic Provisioning 2016 feature set:

a) shall support either the SBC Base 2016 feature set (see clause A.3) or the SBC Base 2010 feature set (see clause A.2); and
b) should support the SBC Base 2016 feature set.

Table A.8 lists the Basic Provisioning 2016 feature set commands.

Table A.8 — Commands mandatory for the Basic Provisioning 2016 feature set

Command	Additional requirements reference	Reference
GET LBA STATUS (16)	A.4.3.1	5.6
READ CAPACITY (16) [a]	A.4.3.2	5.21
UNMAP	n/a	5.35
WRITE SAME (16) [a]	A.4.3.3	5.53
[a] This command is required by the Base 2016 feature set (see clause A.3) and the Base 2010 feature set (see clause A.2). This feature set adds additional requirements.		

Table A.9 lists the Basic Provisioning 2016 feature set VPD pages.

Table A.9 — VPD pages mandatory for the Basic Provisioning 2016 feature set

VPD page	Additional requirements reference	Reference
Block Limits	A.4.4.1	6.6.4
Logical Block Provisioning	A.4.4.2	6.6.9

A.4.2 SBC Basic Provisioning 2016 feature set model additional requirements

The device server shall support logical block provisioning management (see 4.7.3.1).

If the logical unit is thin provisioned (see 4.7.3.3), then the device server shall support:

a) at least two logical block provisioning thresholds (see 4.7.3.7); and
b) the logical block provisioning log page (see 6.4.5).

A.4.3 Basic Provisioning 2016 feature set commands

A.4.3.1 GET LBA STATUS (16) command

The device server shall return only as many LBA status descriptors as it is able to return within the nominal command processing timeout reported in the command timeouts descriptor in the REPORT SUPPORTED OPERATION CODES command (see SPC-6).

A.4.3.2 READ CAPACITY (16) command

The device server shall report the LBPME bit set to one (i.e., logical block provisioning management is implemented) in the READ CAPACITY (16) parameter data.

A.4.3.3 WRITE SAME (16) command

The device server shall support the following bits in the CDB:

a) the UNMAP bit set to one; and
b) the NDOB bit set to one.

A.4.4 SBC Basic Provisioning 2016 feature set VPD pages

A.4.4.1 Block Limits VPD page

The device server shall report the following values:

a) the MAXIMUM UNMAP LBA COUNT field set to at least the value of the MAXIMUM WRITE SAME LENGTH field;
b) the MAXIMUM UNMAP BLOCK DESCRIPTOR COUNT field set to at least 0000_0040h (i.e., 64 descriptors); and
c) if the logical unit has an optimal unmap granularity, then the OPTIMAL UNMAP GRANULARITY field set to a non-zero value.

A.4.4.2 Logical Block Provisioning VPD page

The device server shall report the following values:

a) the LBPU bit set to one (i.e., UNMAP command supported);
b) the LBPWS bit set to one (i.e., WRITE SAME (16) command UNMAP bit is supported); and
c) the PROVISIONING TYPE field set to 001b (i.e., resource provisioned) or 010b (i.e., thin provisioned).

The device server shall not report the LBPRZ field set to 000b (i.e., unmapped LBA data is vendor specific).

The device server is not required to report the ANC_SUP bit set to one.

A.5 Drive Maintenance 2016 feature set

A.5.1 Drive Maintenance 2016 feature set overview

The Drive Maintenance 2016 feature set includes features intended for use by maintenance application clients.

Logical units that support the Drive Maintenance 2016 feature set:

a) shall support either the SBC Base 2016 feature set (see clause A.3) or the SBC Base 2010 feature set (see clause A.2); and
b) should support the SBC Base 2016 feature set.

Table A.10 lists the Drive Maintenance 2016 feature set commands.

Table A.10 — Commands mandatory for the Drive Maintenance 2016 feature set

Command	Additional requirements reference	Reference
FORMAT UNIT	n/a	5.4
LOG SELECT	n/a	SPC-6
LOG SENSE	n/a	SPC-6
READ BUFFER (10)	A.5.2.1	SPC-6
READ DEFECT DATA (12)	A.5.2.2	5.23
REASIGN BLOCKS	A.5.2.3	5.24
SANITIZE	A.5.2.4	5.30
SEND DIAGNOSTICS	A.5.2.5	SPC-6
RECEIVE DIAGNOSTIC RESULTS	n/a	SPC-6
WRITE BUFFER	A.5.2.6	SPC-6
WRITE LONG (16)	n/a	5.51

Table A.11 lists the Drive Maintenance 2016 feature set VPD pages.

Table A.11 — VPD pages mandatory for the Drive Maintenance 2016 feature set

VPD page	Additional requirements reference	Reference
Extended INQUIRY Data	n/a	SPC-6
Block Device Characteristics	A.5.3.1	6.6.2
Power Consumption	n/a	SPC-6

Table A.12 lists the Drive Maintenance 2016 feature set log pages.

Table A.12 — Log pages mandatory for the Drive Maintenance 2016 feature set

Log page	Additional requirements reference	Reference
Supported Log Pages n/a SPC-6	n/a	SPC-6
Supported Log Pages and Subpages	n/a	SPC-6
Background Scan	A.5.4.1	6.4.2.1
Informational Exceptions	n/a	SPC-6
Non-Medium Error	n/a	SPC-6
Non-volatile Cache	n/a	6.4.7
Read Error Counters	A.5.4.2	SPC-6
Self-Test Results	n/a	SPC-6
Solid State Media	n/a	6.4.9
Start-Stop Cycle Counter	A.5.4.3	SPC-6
Temperature	A.5.4.4	SPC-6

A.5.2 Drive Maintenance 2016 feature set commands

A.5.2.1 READ BUFFER (10) command

The device server shall support the following modes:

a) 03h (i.e., descriptor); and
b) 1Ch (i.e., error history).

A.5.2.2 READ DEFECT DATA (12) command

The device server shall support the same address descriptor format types that it supports in the FORMAT UNIT command.

A.5.2.3 REASSIGN BLOCKS command

If the device server supports LBAs greater than FFFF_FFFFh, then the device server shall support the following bits in the CDB:

a) the LONGLBA bit set to one; and
b) the LONGLIST bit set to one.

A.5.2.4 SANITIZE command

The device server shall support the following bits in the CDB:

a) the IMMED bit set to one; and
b) the AUSE bit set to one.

The logical unit shall support the following service actions:

a) EXIT FAILURE MODE.

If the logical unit is a memory media device (e.g., a solid state drive), then the logical unit shall:

a) support the BLOCK ERASE service action; and
b) unmap the entire capacity upon completion.

If the logical unit is a rotating media device (e.g., a hard disk drive), then the logical unit shall support the OVERWRITE service action with:

a) the INVERT bit set to zero;
b) the TEST field set to 00b;
c) the OVERWRITE COUNT field set to 01h;
d) the INITIALIZATION PATTERN LENGTH field set to 0004h; and
e) the INITIALIZATION PATTERN field set to 00000000h.

A.5.2.5 SEND DIAGNOSTIC command

The logical unit shall support a default self-test.

If the logical unit supports an extended self-test, then the device server shall support:

a) the EXTENDED SELF-TEST COMPLETION TIME field in the Control mode page; and
b) the EXTENDED SELF-TEST COMPLETION MINUTES field in the Extended INQUIRY Data VPD page.

The device server is not required to support:

a) the Translate Address Output diagnostic page; or
b) the Translate Address Input diagnostic page.

A.5.2.6 WRITE BUFFER command

The logical unit shall support the following modes:

a) 0Dh (i.e., download microcode with offsets, select activation events, save, and defer activate); and
b) 0Fh (i.e., activate deferred microcode).

The device server is not required to support the following modes:

a) 00h (i.e., combined header and data);
b) 02h (i.e., data);
c) 0Ah (i.e., write data to echo buffer);
d) 1Ah (i.e., enable expander communications protocol and echo buffer); and
e) 1Bh (i.e., disable expander communications protocol).

A.5.3 Drive Maintenance 2016 feature set VPD pages

A.5.3.1 Block Device characteristics VPD page

The device server shall report the following values:

a) the RBWZ bit set to one (i.e., REASSIGN BLOCKS writes zeros if the logical block data is not recovered).

A.5.4 Drive Maintenance 2016 feature set log pages

A.5.4.1 Background Scan Results log page

The device server shall report the following log parameters:

a) Background Scan Status, as required by 6.4.2.1; and
b) at least 256 Background Scan Results log parameters.

A.5.4.2 Read Error Counters log page

The device server shall report the following log parameters:

a) 0006h (i.e., total uncorrected errors).

A.5.4.3 Start-Stop Cycle Counter log page

The device server shall report the following log parameters:

a) Date of Manufacture; and
b) Accounting Date.

A.5.4.4 Temperature log page

The device server shall report the following log parameters:

a) Temperature, as required by SPC-6; and
b) Reference Temperature.

Annex B
(informative)

Numeric order codes

B.1 Variable length CDBs

Commands that use operation code 7Fh in table 34 use the variable length command format defined in SPC-6 and are differentiated by service action codes as described in table B.1.

Table B.1 — Variable length command service action code assignments

Operation code/service action code	Description
7Fh/0000h	Reserved
7Fh/0001h	Reserved
7Fh/0002h	Reserved
7Fh/0003h	Obsolete (XDREAD (32))
7Fh/0004h	Obsolete (XDWRITE (32))
7Fh/0005h	Reserved
7Fh/0006h	Obsolete (XPWRITE (32))
7Fh/0007h	Obsolete (XDWRITEREAD (32))
7Fh/0008h	Reserved
7Fh/0009h	READ (32)
7Fh/000Ah	VERIFY (32)
7Fh/000Bh	WRITE (32)
7Fh/000Ch	WRITE AND VERIFY (32)
7Fh/000Dh	WRITE SAME (32)
7Fh/000Eh	ORWRITE (32)
7Fh/000Fh	WRITE ATOMIC (32)
7Fh/0010h	WRITE STREAM (32)
7Fh/0011h	WRITE SCATTERED (32)
7F/0012h	GET LBA STATUS (32)
7Fh/0013h to 07FFh	Reserved
7Fh/0800h to FFFFh	See SPC-6

B.2 SERVICE ACTION IN commands and SERVICE ACTION OUT commands

Commands that use operation code 9Eh (i.e., SERVICE ACTION IN (16)) (see SPC-6) in table 34 are differentiated by service action codes as described in table B.2.

Table B.2 — SERVICE ACTION IN (16) service actions

Operation code/service action code	Description
9Eh/00h to 0Fh	Reserved for commands applicable to all device types (see SPC-6)
9Eh/10h	READ CAPACITY (16)
9Eh/11h	Obsolete READ LONG (16)
9Eh/12h	GET LBA STATUS (16)
9Eh/13h	REPORT REFERRALS
9Eh/14h	STREAM CONTROL
9Eh/15h	BACKGROUND CONTROL
9Eh/16h	GET STREAM STATUS
9Eh/17h	GET PHYSICAL ELEMENT STATUS
9Eh/18h	REMOVE ELEMENT AND TRUNCATE
9Eh/19h	RESTORE ELEMENTS AND REBUILD
9Eh/1Ah	REMOVE ELEMENT AND MODIFY ZONES
9Eh/1Bh to 1Fh	Reserved

Commands that use operation code 9Fh (i.e., SERVICE ACTION OUT (16)) (see SPC-6) in table 34 are differentiated by service action codes as described in table B.3.

Table B.3 — SERVICE ACTION OUT (16) service actions

Operation code/service action code	Description
9Fh/00h to 0Fh	Reserved for commands applicable to all device types (see SPC-6)
9Fh/10h	Reserved
9Fh/11h	WRITE LONG (16)
9Fh/12h	WRITE SCATTERED (16)
9Fh/13h to 1Fh	Reserved

Annex C
(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 4.21).

```
// picrc.cpp : SCSI SBC-5 Protection Information 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 */
```

Annex D
(informative)

Sense information for locked or encrypted logical units

A device server may complete some commands with CHECK CONDITION status under certain conditions while the logical unit is locked or encrypted. Table D.1 describes the conditions relative to the sense key and the additional sense code returned by the device server with the CHECK CONDITION status.

Table D.1 — Sense information for locked or encrypted logical units

Sense key	Additional sense code	Description
DATA PROTECT	ACCESS DENIED – NO ACCESS RIGHTS	The logical unit is locked. This condition may occur for read commands or write commands. This condition may occur for the entire logical unit or for a range of LBAs contained in the logical unit. To clear this condition, an application client performs a security protocol specific procedure to unlock access to the logical unit.
ABORTED COMMAND	LOGICAL BLOCK REFERENCE TAG CHECK FAILED	These conditions may occur for a read command. The additional sense codes may indicate that an encrypting logical unit has changed the encryption/decryption key, and the LBAs requested by the command have not yet been rewritten. Disabling protection information checking in a CDB may allow the command to complete successfully, but the data returned for the command may be invalid (i.e., not decrypted). To clear this condition, an application client writes the LBAs for which the condition occurred with new data.
ABORTED COMMAND	LOGICAL BLOCK APPLICATION TAG CHECK FAILED	
ABORTED COMMAND	LOGICAL BLOCK GUARD CHECK FAILED	

Annex E

(informative)

Optimizing block access characteristics

E.1 Overview

This annex describes example methods that application clients may use to achieve optimal performance for logical block access. These examples use the following information:

a) the LOWEST ALIGNED LOGICAL BLOCK ADDRESS field (see 5.21.2);
b) the LOGICAL BLOCKS PER PHYSICAL BLOCK EXPONENT field (see 5.21.2);
c) the OPTIMAL TRANSFER LENGTH GRANULARITY field(see 6.6.4);
d) the OPTIMAL TRANSFER LENGTH field (see 6.6.4);
e) the MAXIMUM TRANSFER LENGTH field (see 6.6.4);
f) the OPTIMAL STREAM WRITE SIZE field (see 6.6.5); and
g) the STREAM GRANULARITY SIZE field (see 6.6.5).

E.2 Starting logical block offset

The READ CAPACITY (16) command transfers parameter data which includes a value in the LOWEST ALIGNED LOGICAL BLOCK ADDRESS field. As shown in figure 4, the value in this field indicates the starting alignment of logical block addresses where optimal performance for logical block access begins.

E.3 Optimal granularity sizes

The READ CAPACITY (16) command transfers parameter data that includes a value in the LOGICAL BLOCKS PER PHYSICAL BLOCK EXPONENT field. As shown in figure 2 and in figure 4, the value in this field enables the application client to determine the number of logical blocks per physical block.

The Block Limits VPD page may include values in the OPTIMAL TRANSFER LENGTH GRANULARITY field, the OPTIMAL TRANSFER LENGTH field, and the MAXIMUM TRANSFER LENGTH field. These values may be used to determine optimum transfer sizes.

If the OPTIMAL TRANSFER LENGTH GRANULARITY field is valid (i.e., contains a value greater than zero), then the value in the OPTIMAL TRANSFER LENGTH GRANULARITY field is the optimal granularity size. If:

a) the Block Limits VPD page is not supported; or
b) the Block Limits VPD page is supported and the OPTIMAL TRANSFER LENGTH GRANULARITY field is set to zero,

then the value $2^{(\text{logical blocks per physical block exponent})}$ is the optimal granularity size.

E.4 Optimal stream granularity sizes

The Block Limits Extension VPD page may include values in the OPTIMAL STREAM WRITE SIZE field and the STREAM GRANULARITY SIZE field. These values may be used to determine optimum transfer sizes and optimum transfer alignment to use with the write stream commands (see 4.32.2).

If the OPTIMAL STREAM WRITE SIZE field is valid (i.e., contains a value greater than zero), then the value in the OPTIMAL STREAM WRITE SIZE field indicates the optimum transfer size and the optimum transfer alignment.

If:

a) the Block Limits Extension VPD page is not supported; or
b) the Block Limits Extension VPD page is supported and the OPTIMAL STREAM WRITE SIZE field is set to zero,

then the optimum transfer size and optimum transfer alignment is not reported for write stream commands.

E.5 Optimizing transfers

E.5.1 Optimizing transfers overview

Non-stream write commands may be optimized using the method described in E.5.2. write stream commands (see 4.32.2) may be optimized using the method described in E.5.3.

E.5.2 Optimizing non-stream transfers

If:

a) the Block Limits Extension VPD page is not supported;
b) the OPTIMAL STREAM WRITE SIZE field (see 6.6.5) is zero; or
c) a write command other than a write stream command (see 4.33.2) is sent to the device server,

then to obtain optimal performance, the application client requests transfers with a starting LBA of the form calculated by the following formula:

starting LBA = lowest aligned LBA + (optimal transfer length granularity × n)

where:

starting LBA	is the LBA of the first logical block accessed;
lowest aligned LBA	is the value in the LOWEST ALIGNED LOGICAL BLOCK ADDRESS field; and
n	is zero or a positive integer.

and using transfer lengths of the form:

transfer length = (optimal granularity size × k)

where:

transfer length	is the number of contiguous logical blocks of data being accessed;
optimal granularity size	is the value described in E.3; and
k	is a positive integer.

To obtain optimal performance, the application client requests a transfer length, in logical blocks, that is no larger than the value in the MAXIMUM TRANSFER LENGTH field, and is:

a) no larger than the optimal transfer length for logical units where the delay in processing transfers larger than the optimal transfer length is large; or
b) not limited by the value in the OPTIMAL TRANSFER LENGTH field for logical units where the delay in processing transfers larger than the optimal transfer length is small (i.e., most direct access block devices exhibit this type of operation).

NOTE 24 - There is no method available to determine if the delay in processing for various transfer lengths is large or small.

It is more important that the application client meet the logical unit’s starting and ending alignment boundary conditions than the maximum transfer length conditions. These considerations have larger impacts on write performance than read performance.

E.5.3 Optimizing stream transfers

If the Block Limits Extension VPD page is supported, the OPTIMAL STREAM WRITE SIZE field (see 6.6.5) is non-zero, and write stream commands (see 4.33.2) are sent to the device server with a given stream identifier, then to obtain optimal performance, the application client requests transfers using transfer lengths of the form:

$$\text{transfer length} = (\text{optimal stream granularity} \times k)$$

where:

transfer length	is the number of contiguous logical blocks of data being accessed;
optimal stream granularity	is the optimal stream write size (see 6.6.5); and
k	is a positive integer,

and a stream length of the form:

$$\text{stream length} = (\text{optimal stream granularity} \times \text{stream granularity size} \times k)$$

where:

stream length	is the sum of the transfer length of all transfers for that stream (i.e., from when the stream is opened to when the stream is closed);
optimal granularity size	is the optimal stream write size;
stream granularity size	is the value in the STREAM GRANULARITY SIZE field; and
k	is a positive integer.

E.6 Examples

In this first example, a logical unit reports the following information:

a) the LOWEST ALIGNED LOGICAL BLOCK ADDRESS field set to 0003h in the READ CAPACITY (16) parameter data (see 5.21.2);
b) the OPTIMAL TRANSFER LENGTH GRANULARITY field set to 0008h in the Block Limits VPD page (see 6.6.4);
c) the MAXIMUM TRANSFER LENGTH field set to 0000_0000h in the Block Limits VPD page; and
d) the OPTIMAL TRANSFER LENGTH field set to 0000_0080h (i.e., 128) in the Block Limits VPD page.

The starting LBA for optimal transfers on this logical unit should be of the form ((8 × n) + 3) where n is any integer greater than or equal to zero (e.g., starting LBAs of 3, 11, 19, 27, and 35). The transfer length for optimal transfers should be a multiple of eight logical blocks (e.g., transfer lengths of 8 blocks, 32 blocks, or 128 blocks).

A write command with the LOGICAL BLOCK ADDRESS field set to 19 and the TRANSFER LENGTH field set to 32 should exhibit improved performance over a write command with the LOGICAL BLOCK ADDRESS field set to 18 and the TRANSFER LENGTH field set to 32.

If the device has a delay in processing transfers larger than the optimal transfer length, some operations may exhibit improved performance if a single large request is broken into multiple smaller requests (e.g., rather than performing a single read of 248 logical blocks, the transfer may be optimized by setting the transfer length of one read command to 128 logical blocks and setting the transfer length of a second read command to 120 logical blocks).

In this second example, a logical unit reports the following information:

a) the OPTIMAL STREAM WRITE SIZE field set to 0010h (i.e., 16) in the Block Limits Extension VPD page (see 6.6.5); and

b) the STREAM GRANULARITY SIZE field set to 0040h (i.e., 64) in the Block Limits Extension VPD page.

The LOGICAL BLOCK ADDRESS field in a write stream command (see 4.33.2) for optimal transfers should be of the form (16 × j) where j is any integer greater than or equal to zero (e.g., logical block addresses of 16, 32, or 48). The transfer length in a write stream command for optimal transfers should be of the form (16 × k) where k is any integer greater than or equal to zero (e.g., transfer lengths of 16 blocks, 32 blocks, or 48 blocks). The stream length for optimal transfers should be of the form (16 × 64 × m) where m is any integer greater than or equal to zero (e.g., stream lengths of 1 024 blocks, 2 048 blocks, or 3 072 blocks).

The stream length is the total of all write transactions using the specified stream identifier (e.g.., from the receipt of a STREAM CONTROL command with the STR_CTL field set to 01b (i.e., open) to the receipt of a STREAM CONTROL command with the STR_CTL field set to 10b (i.e., close) and the STR_ID field set to the specified stream identifier).

A write stream command with the LOGICAL BLOCK ADDRESS field set to 32 (i.e., an integer multiple of 16) and the TRANSFER LENGTH field set to 64 (i.e., an integer multiple of 16) should exhibit improved performance over a write stream command with the LOGICAL BLOCK ADDRESS field set to 16 (i.e., an integer multiple of 16) and the TRANSFER LENGTH field set to 20 (i.e., not an integer multiple of 16).

A sequence of write stream commands to the stream that specify a total of 2 048 logical blocks (i.e., an integer multiple of 1 024) should exhibit improved performance over a sequence of write stream commands to the stream that specify a total of 1 536 logical blocks (i.e., not an integer multiple of 1 024).

Annex F
(informative)

Logical block provisioning reporting examples

F.1 Overview

Logical block provisioning reporting may be implemented using different methods. Implementations may include one or more of the following:

a) use of dedicated LBA mapping resources (e.g., resources are associated with a specific logical unit);
b) use of shared LBA mapping resources (e.g., resources are shared by multiple logical units);
c) reporting based on dedicated LBA mapping resources (e.g., resources are reported specific to the logical unit);
d) reporting based on shared LBA mapping resources (e.g., resources are reported for the resource pool as a whole);
e) LBA mapping resource tracking based on logical blocks; and
f) LBA mapping resource tracking based on threshold sets.

This annex describes examples of logical block provisioning reporting. Each example follows logical block provisioning resource usage and reporting over time as a specified set of operations occur.

F.2 Interpreting log parameter counts

As a result of the variation of the threshold set size implementations, logical block usage and resource reporting may not have a direct relationship. The second example (see clause F.4) demonstrates an implementation where logical blocks are allocated on an individual LBA basis and reported using a larger threshold set basis. The reporting is a direct calculation from a logical block based count to a threshold set based count.

In implementations where a threshold set contains a set of contiguous logical blocks, the reporting may be substantially different. LUN 1 in the first example (see clause F.3) demonstrates such an implementation. At the initial conditions, two threshold sets are reported as being used. With a threshold set size of 1 024 blocks, these two threshold sets may contain as little as one logical block of application client data in each threshold set, or as many as 1 024 contiguous logical blocks in each threshold set. Which LBAs have been written by the application client has a substantial impact on how the usage of those resources is reported.

The relationship of the physical blocks to the logical blocks (see figure 4 and figure 5) may have an impact on the logical block provisioning log parameters. Which LBAs are written by the application may impact the number of physical blocks required to be allocated and therefore impact the reporting of the LBA mapping resource parameters.

The device server may not prioritize the maintenance of the values in the Logical Block Provisioning log page (see 6.4.5) above the completion of other operations (e.g., read operations or write operations). This may result in delays in updates to these values (e.g., after a request to unmap a large number of logical blocks). The logical block provisioning log parameters may also appear inaccurate for logical units where unmap operations cause LBA mapping resources to be released using a periodic background function.

Data de-duplication (see 4.8) and compression may impact the logical block provisioning log parameters. If the device server is able to perform data de-duplication or compression, then the number of LBA mapping resources:

a) used for a successful write operation may not be the same as specified in the command that requested that operation; or
b) made available after one or more successful unmap operations may not be the same as specified in the command that requested those operations.

EXAMPLE 1 - If a write command to LBAs that are all in the deallocated state (see 4.7.4.6) results in data that is all de-duplicated, then there may be no additional LBA mapping resources used when those LBAs transition to the mapped state (see 4.7.4.5). This may result in no change in the available LBA mapping resource count (see 6.4.5.2), or the used LBA mapping resource count (see 6.4.5.3).

EXAMPLE 2 - If a write command to LBAs that are all in the mapped state results in data that is no longer de-duplicated, then additional LBA mapping resources may be required even though all the LBAs addressed by the command remain in the mapped state. This may result in a reduction in the available LBA mapping resource count, or an increase in the used LBA mapping resource count. On a thin provisioned logical unit, such a write command may also be terminated as a result of a resource exhaustion condition (see 4.7.3.6.1).

EXAMPLE 3 - If a write command to LBAs that are all in the mapped state results in data that is more compressible than the previous data in those LBAs, then fewer LBA mapping resources may be required even though all the LBAs addressed by that command remain in the mapped state. This may result in an increase in the available LBA mapping resource count, or a decrease in the used LBA mapping resource count.

EXAMPLE 4 - If a write command to LBAs that are all in the mapped state results in data that is less compressible than the previous data in those LBAs, then additional LBA mapping resources may be required even though all the LBAs addressed by that command remain in the mapped state. This may result in a decrease in the available LBA mapping resource count, or an increase in the used LBA mapping resource count. On a thin provisioned logical unit, such a write command may also be terminated as a result of a resource exhaustion condition.

EXAMPLE 5 - If an unmap command addresses LBAs that are all in the mapped state and contain data that has been de-duplicated (i.e., additional logical blocks still contain that same de-duplicated data), then on a thin provisioned logical unit, there may be no change in the LBA mapping resources even though all of the LBAs addressed by that command transition to the deallocated state. This may result in no change to the available LBA mapping resource count or the used LBA mapping resource count.

As a result, application clients using logical block provisioning thresholds and examining logical block provisioning log parameters should not expect application client determined usage values or application client determined available space values to match log parameters or threshold events as reported by the logical unit.

F.3 Dedicated resource, threshold set tracked example

F.3.1 Dedicated resource, threshold set tracked example overview

This example describes a method that reports dedicated logical block provisioning resources based on threshold sets. In this example, the values reported by the logical unit in the Logical Block Provisioning log page (see 6.4.5) reflect the usage for each logical unit and the available resources dedicated to each logical unit. Each threshold set is allocated to contain a set of contiguous logical blocks (e.g., LBAs 1 024 to 2 047 are contained in the same threshold set).

F.3.2 Dedicated resource, threshold set tracked example configuration

The configuration used for this example consists of two thin provisioned logical units, each with dedicated logical block provisioning resources. Table F.1 shows logical block provisioning related capacity values used in this example.

Table F.1 — Dedicated resource, threshold set tracked example capacity information

LUN	Capacity		THRESHOLD EXPONENT field [c]	Number of threshold sets [d]
	LBA [a]	Logical blocks [b]		
1	3FFF_FFFFh	1 Gi	0Ah (i.e., 512 KiB, 1 024 logical blocks)	0010_0000h (i.e., 1 Mi)
2	BFFF_FFFFh	3 Gi	0Ch (i.e., 2 MiB, 4 096 logical blocks)	000C_0000h (i.e., 768 Ki)

[a] RETURNED LOGICAL BLOCK ADDRESS field in READ CAPACITY parameter data (see 5.20.2 and 5.21.2).
[b] The value returned in the RETURNED LOGICAL BLOCK ADDRESS field plus one.
[c] In the Logical Block Provisioning VPD page (see 6.6.9).
[d] Number of threshold sets = capacity ÷ $2^{(\text{threshold exponent})}$.

Table F.2 shows LUN 1 with four enabled threshold descriptors and LUN 2 with two enabled threshold descriptors. The threshold descriptors in the Logical Block Provisioning mode page (see 6.5.9) for LUN 1 are configured to report a logical block provisioning threshold crossing (see 4.7.3.7) when:

a) the percentage of available LBA mapping resources reaches 30% of reported capacity;
b) the percentage of available LBA mapping resources reaches 20% of reported capacity;
c) the percentage of available LBA mapping resources reaches 10% of reported capacity; or
d) the percentage of used LBA mapping resources reaches 75% of reported capacity.

The threshold descriptors in the Logical Block Provisioning mode page for LUN 2 are configured to report a logical block provisioning threshold crossing when:

a) the percentage of available LBA mapping resources reaches 50% of reported capacity; or
b) the percentage of available LBA mapping resources reaches 10% of reported capacity.

Table F.2 — Dedicated resource, threshold set tracked example capacity information

LUN	Threshold resource [a]	Threshold count [b]	Description
1	0001h	0004_CCCCh	An available LBA mapping resource threshold set to 30% of the reported capacity (i.e., number of threshold sets from table F.1 × 0.30 = 0004_CCCCh threshold sets)
	0001h	0003_3333h	An available LBA mapping resource threshold set to 20% of the reported capacity
	0001h	0001_9999h	An available LBA mapping resource threshold set to 10% of the reported capacity
	0002h	000C_0000h	A used LBA mapping resource threshold set to 75% of the reported capacity (i.e., number of threshold sets from table F.1 (i.e., 0010_0000h) × 0.75 = 000C_0000h threshold sets)
2	0001h	0006_0000h	An available LBA mapping resource threshold set to 50% of the reported capacity (i.e., number of threshold sets from table F.1 (i.e., 000C_0000h) × 0.50 = 0006_0000h threshold sets)
	0001h	0001_3333h	An available LBA mapping resource threshold set to 10% of the reported capacity

[a] THRESHOLD RESOURCE field (see 6.5.9.2) and the PARAMETER CODE field (see 6.4.5.2).
[b] THRESHOLD COUNT field (see 6.5.9.2).

F.3.3 Dedicated resource, threshold set tracked example sequence

The sequence of events for this example are:

1) initial conditions (see F.3.4);
2) operations that occur (see F.3.5); and
3) final values in the logical block provisioning log page (see F.3.6).

F.3.4 Dedicated resource, threshold set tracked example initial conditions

Initially, LUN 1 has two threshold sets used and has 69 108 736 logical blocks available (i.e. 0001_07A1h threshold sets). The application client has written at least one logical block into each of the two logical block ranges that correspond to those two threshold sets, therefore the application client may have written from two logical blocks to 2 048 logical blocks. LUN 2 has 1 073 741 824 logical blocks available (i.e., 0004_0000h threshold sets). LUN 2 does not report a used LBA mapping resource parameter. Table F.3 shows the values in the Logical Block Provisioning log page for the initial conditions in this example.

Table F.3 — Dedicated resource, threshold set tracked example initial conditions

LUN	Log page parameter [a]	Resource count [b]	Scope [c]	Description [d]
1	0001h	0001_07A1h	01b	The available LBA mapping resource parameter indicates that 69 108 736 logical blocks (i.e., 1_07A1h threshold sets × 1 024 logical blocks per threshold set) are available for LUN 1.
	0002h	0000_0002h	01b	The used LBA mapping resource parameter indicates that 2 048 logical blocks (i.e., 2h threshold sets × 1 024 logical blocks per threshold set) have been used (i.e., allocated) by LUN 1.
2	0001h	0004_0000h	01b	The available LBA mapping resource parameter indicates that 1 073 741 824 logical blocks (i.e., 4_0000h threshold sets × 4 096 logical blocks per threshold set) are available for LUN 2.

[a] THRESHOLD RESOURCE field (see 6.5.9.2) and the PARAMETER CODE field (see 6.4.5.2).
[b] RESOURCE COUNT field (see 6.5.9.2).
[c] SCOPE field (see 6.4.5.2).
[d] LBA count = capacity × $2^{(\text{threshold exponent})}$.

F.3.5 Operations that occur

Write operations occur to LUN 1 that require one additional threshold set to be allocated when the application client writes 50 additional contiguous logical blocks. Used LBA mapping resources on LUN 1 are now 3 072 logical blocks (i.e., three threshold sets), and available LBA mapping resources are 69 107 712 logical blocks. Write operations also occur to LUN 2 that require no additional threshold sets when the application client writes an additional 100 logical blocks into a threshold set that was already allocated.

F.3.6 Dedicated resource, threshold set tracked example final log page values

Table F.4 shows the values in the Logical Block Provisioning log page after the operations described in F.3.5 have occurred.

Table F.4 — Dedicated resource, threshold set tracked example final log page values

LUN	Log page parameter [a]	Resource count [b]	Scope [c]	Description [d]
1	0001h	0001_07A0h	01b	The available LBA mapping resource parameter indicates that 69 107 712 logical blocks (i.e., 1_07A0h threshold sets × 1 024 logical blocks per threshold set) are available for LUN 1.
	0002h	0000_0003h	01b	The used LBA mapping resource parameter indicates that 3 072 logical blocks (i.e., 3h threshold sets × 1 024 logical blocks per threshold set) have been used (i.e., allocated) by LUN 1.
2	0001h	0004_0000h	01b	The available LBA mapping resource parameter indicates that 1 073 741 824 (i.e., 4_0000h threshold sets × 4 096 logical blocks per threshold set) are available for LUN 2.

[a] THRESHOLD RESOURCE field (see 6.5.9.2) and the PARAMETER CODE field (see 6.4.5.2).
[b] RESOURCE COUNT field (see 6.5.9.2).
[c] SCOPE field (see 6.4.5.2).
[d] LBA count = capacity × $2^{(\text{threshold exponent})}$.

F.4 Shared resource, logical block tracked example

F.4.1 Shared resource, logical block tracked example overview

This example describes a method that tracks shared logical block provisioning resources based on logical blocks. The logical block provisioning resources are shared by multiple logical units. In this example, the values reported by each logical unit in its Logical Block Provisioning log page (see 6.4.5) reflect the combined usage of all logical units that share the logical block provisioning resources and the resources available for use by any of the logical units that share the logical block provisioning resources. Resources are allocated one logical block at a time but reported with a larger threshold set size.

F.4.2 Shared resource, logical block tracked example configuration

The configuration used for this example consists of two thin provisioned logical units, where the logical block provisioning resources are shared between both logical units. Table F.5 shows logical block provisioning related capacity values used in this example.

Table F.5 — Shared resource, logical block tracked example capacity information

LUN	Capacity		THRESHOLD EXPONENT field [c]	Number of threshold sets [d]
	LBA [a]	Logical blocks [b]		
1	3FFF_FFFFh	1 Gi	0Bh (i.e., 1 MiB, 2 048 logical blocks)	0008_0000h (i.e., 512 Ki)
2	BFFF_FFFFh	3 Gi	0Bh (i.e., 1 MiB, 2 048 logical blocks)	0018_0000h (i.e., 1 536 Ki)

[a] RETURNED LOGICAL BLOCK ADDRESS field in READ CAPACITY parameter data (see 5.20.2 and 5.21.2).
[b] The value returned in the RETURNED LOGICAL BLOCK ADDRESS field plus one.
[c] In the Logical Block Provisioning VPD page (see 6.6.9).
[d] Number of threshold sets = capacity ÷ $2^{(\text{threshold exponent})}$.

F.4.3 Shared resource, logical block tracked example time line

The sequence of events for this example are:

1) initial conditions (see F.4.4);
2) operations that occur (see F.4.5); and
3) final values in the logical block provisioning log page (see F.4.6).

F.4.4 Shared resource, logical block tracked example initial conditions

Initially, LUN 1 and LUN 2 have used a combined total of 57 000 logical blocks. LUN1 and LUN 2 have 1 073 741 900 logical blocks available for use by either LUN 1 or LUN 2. Table F.6 shows the values in the Logical Block Provisioning log page for the initial conditions in this example.

Table F.6 — Shared resource, logical block tracked example initial conditions

LUN	Log page parameter [a]	Resource count [b]	Scope [c]	Description
1	0001h	0008_0000h	10b	The available LBA mapping resource parameter indicates that from 1 073 741 824 logical blocks (i.e., 8_0000h threshold sets × 2 048 logical block per threshold set) to 1 073 743 871 logical blocks are available for LUN 1 or LUN 2. [d]
	0002h	0000_001Ch	10b	The used LBA mapping resource parameter indicates that from 55 297 logical blocks to 57 344 logical blocks (i.e., 1Ch threshold sets × 2 048 logical blocks per threshold set) have been used (i.e., allocated) by LUN 1 and LUN 2. [e]
2	0001h	0008_0000h	10b	The available LBA mapping resource parameter indicates that 1 073 741 824 logical blocks (i.e., 8_0000h threshold sets × 2 048 logical blocks per threshold set) are available for LUN 1 or LUN 2. [d]

[a] THRESHOLD RESOURCE field (see 6.5.9.2) and the PARAMETER CODE field (see 6.4.5.2).
[b] RESOURCE COUNT field (see 6.5.9.2).
[c] SCOPE field (see 6.4.5.2).
[d] Minimum available LBA count = resource count × $2^{(\text{threshold exponent})}$.
[e] Maximum used LBA count = resource count × $2^{(\text{threshold exponent})}$.

F.4.5 Operations that occur

Write operations occur to LUN 1 that require 2 000 additional logical blocks to be used and write operations occur to LUN 2 that require 3 000 additional logical blocks to be used. Used LBA mapping resources on LUN 1 and LUN 2 are now 62 000 logical blocks, and the combined LBA mapping resources available to both LUN 1 and LUN 2 are 1 073 736 900 logical blocks (i.e., 1 073 741 900 minus 5 000).

F.4.6 Shared resource, logical block tracked example final log page values

Table F.7 shows the values in the Logical Block Provisioning log page after the operations described in F.4.5 have occurred.

Table F.7 — Shared resource, logical block tracked example final log page values

LUN	Log page parameter [a]	Resource count [b]	Scope [c]	Description
1	0001h	0007_FFFDh	10b	The available LBA mapping resource parameter indicates that from 1 073 735 680 logical blocks (i.e., 7_FFFDh threshold sets × 2 048 logical blocks per threshold set) to 1 073 737 727 logical blocks are available for LUN 1 or LUN 2. [d]
	0002h	0000_001Fh	10b	The used LBA mapping resource parameter indicates that from 61 441 logical blocks to 63 488 logical blocks (i.e., 1Fh threshold sets × 2 048 logical blocks per threshold set) have been used (i.e., allocated) by LUN 1 and LUN 2. [e]
2	0001h	0007_FFFDh	10b	The available LBA mapping resource parameter indicates that from 1 073 735 680 logical blocks (i.e., 7_FFFDh threshold sets × 2 048 logical blocks per threshold set) to 1 073 737 727 logical blocks are available for LUN 1 or LUN 2. [d]

[a] THRESHOLD RESOURCE field (see 6.5.9.2) and the PARAMETER CODE field (see 6.4.5.2).
[b] RESOURCE COUNT field (see 6.5.9.2).
[c] SCOPE field (see 6.4.5.2).
[d] Minimum available LBA count = resource count × $2^{(\text{threshold exponent})}$.
[e] Maximum used LBA count = resource count × $2^{(\text{threshold exponent})}$.

F.5 Shared available, dedicated used, logical block tracked example

F.5.1 Shared available, dedicated used, logical block tracked example overview

This example describes a method that tracks available shared logical block provisioning resources based on logical blocks and dedicated used logical block provisioning resources based on logical blocks. The available logical block provisioning resources are shared by multiple logical units. In this example, the values reported by the logical unit in the available LBA mapping resource parameter of the Logical Block Provisioning log page (see 6.4.5) reflect the resources available for use by any of the logical units that share the logical block provisioning resources. The values reported by the logical unit in the used LBA mapping resource parameter of the Logical Block Provisioning log page reflect the usage for the individual logical unit.

F.5.2 Shared available, dedicated used, logical block tracked example configuration

The configuration used for this example consists of two thin provisioned logical units, where the available logical block provisioning resources are shared between both logical units and used logical block provisioning resources are reported independently for each logical unit. Table F.8 shows logical block provisioning related capacity values used in this example.

Table F.8 — Shared available, dedicated used example capacity information

LUN	Capacity		THRESHOLD EXPONENT field [c]	Number of threshold sets [d]
	LBA [a]	Logical blocks [b]		
1	3FFF_FFFFh	1 Gi	0Bh (i.e., 1 MiB, 2 048 logical blocks)	0008_0000h (i.e., 512 Ki)
2	BFFF_FFFFh	3 Gi	0Bh (i.e., 1 MiB, 2 048 logical blocks)	0018_0000h (i.e., 1 536 Ki)

[a] RETURNED LOGICAL BLOCK ADDRESS field in READ CAPACITY parameter data (see 5.20.2 and 5.21.2).
[b] The value returned in the RETURNED LOGICAL BLOCK ADDRESS field plus one.
[c] In the Logical Block Provisioning VPD page (see 6.6.9).
[d] Number of threshold sets = capacity ÷ $2^{(\text{threshold exponent})}$.

F.5.3 Shared available, dedicated used, logical block tracked example time line

The sequence of events for this example are:

1) initial conditions (see F.5.4);
2) operations that occur (see F.5.5); and
3) final values in the logical block provisioning log page (see F.5.6).

F.5.4 Shared available, dedicated used, logical block tracked example initial conditions

Initially, LUN 1 has used 57 000 logical blocks and, LUN 2 has used 103 000 logical blocks. LUN 1 and LUN 2 have 1 073 741 900 logical blocks available for use by either LUN 1 or LUN 2. Table F.9 shows the values in the Logical Block Provisioning log page for the initial conditions in this example.

Table F.9 — Shared resource, logical block tracked example initial conditions

LUN	Log page parameter [a]	Resource count [b]	Scope [c]	Description
1	0001h	0008_0000h	10b	The available LBA mapping resource parameter indicates that from 1 073 741 824 logical blocks (i.e., 8_0000h threshold sets × 2 048 logical blocks per threshold set) to 1 073 743 871 logical blocks are available for LUN 1 or LUN 2. [d]
	0002h	0000_001Ch	01b	The used LBA mapping resource parameter indicates that from 55 297 logical blocks to 57 344 logical blocks (i.e., 1Ch threshold sets × 2 048 logical blocks per threshold set) have been used (i.e., allocated) by LUN 1. [e]
2	0001h	0008_0000h	10b	The available LBA mapping resource parameter indicates that from 1 073 741 824 logical blocks (i.e., 8_0000h threshold sets × 2 048 logical blocks per threshold set) to 1 073 743 871 logical blocks are available for LUN 1 or LUN 2. [d]
	0002h	0000_0033h	01b	The used LBA mapping resource parameter indicates that from 102 401 logical blocks to 104 448 (i.e., 33h threshold sets × 2 048 logical blocks per threshold set) have been used (i.e., allocated) by LUN 2. [e]

[a] THRESHOLD RESOURCE field (see 6.5.9.2) and the PARAMETER CODE field (see 6.4.5.2).
[b] RESOURCE COUNT field (see 6.5.9.2).
[c] SCOPE field (see 6.5.9.2).
[d] Minimum available LBA count = resource count × $2^{(\text{threshold exponent})}$.
[e] Maximum used LBA count = resource count × $2^{(\text{threshold exponent})}$.

F.5.5 Operations that occur

Write operations occur to LUN 1 that require 2 000 additional logical blocks to be used and write operations occur to LUN 2 that require 3 000 additional logical blocks to be used. Used LBA mapping resources on LUN 1 are now 59 000 logical blocks, used LBA mapping resources on LUN 2 are now 106 000 logical blocks, and the combined LBA mapping resources available to both LUN 1 and LUN 2 are 1 073 736 900 logical blocks.

F.5.6 Shared available, dedicated used, example final log page values

Table F.10 shows the values in the Logical Block Provisioning log page after the operations described in F.5.5 have occurred.

Table F.10 — Shared available, dedicated used example final log page values

LUN	Log page parameter [a]	Resource count [b]	Scope [c]	Description
1	0001h	0007_FFFDh	10b	The available LBA mapping resource parameter indicates that from 1 073 735 680 logical blocks (i.e., 7_FFFDh threshold sets × 2 048 logical blocks per threshold set) to 1 073 737 727 logical blocks are available for LUN 1 or LUN 2. [d]
	0002h	0000_001Dh	01b	The used LBA mapping resource parameter indicates that from 57 345 logical blocks to 59 392 logical blocks (i.e., 1Dh threshold sets × 2 048 logical blocks per threshold set) have been used (i.e., allocated) by LUN 1. [e]
2	0001h	0007_FFFDh	10b	The available LBA mapping resource parameter indicates that from 1 073 735 680 logical blocks (i.e., 7_FFFDh threshold sets × 2 048 logical blocks per threshold set) to 1 073 737 727 logical blocks are available for LUN 1 or LUN 2. [d]
	0002h	0000_0034h	01b	The used LBA mapping resource parameter indicates that from 104 449 logical blocks to 106 496 logical blocks (i.e., 34h threshold sets × 2_048 logical blocks per threshold set) have been used (i.e., allocated) by LUN 2. [e]

[a] THRESHOLD RESOURCE field (see 6.5.9.2) and the PARAMETER CODE field (see 6.4.5.2).
[b] RESOURCE COUNT field (see 6.5.9.2).
[c] SCOPE field (see 6.4.5.2).
[d] Minimum available LBA count = resource count × $2^{(\text{threshold exponent})}$.
[e] Maximum used LBA count = resource count × $2^{(\text{threshold exponent})}$.

Annex G
(informative)

Discovering referrals examples

G.1 Referrals example with no user data segment multiplier

This annex demonstrates a method an application client may use to determine the optimal target port group from which to access logical blocks using information sent from the device server when the user data segment multiplier is set to zero.

Figure G.1 shows an example of a SCSI device in which referrals have been implemented with a user data segment multiplier of zero.

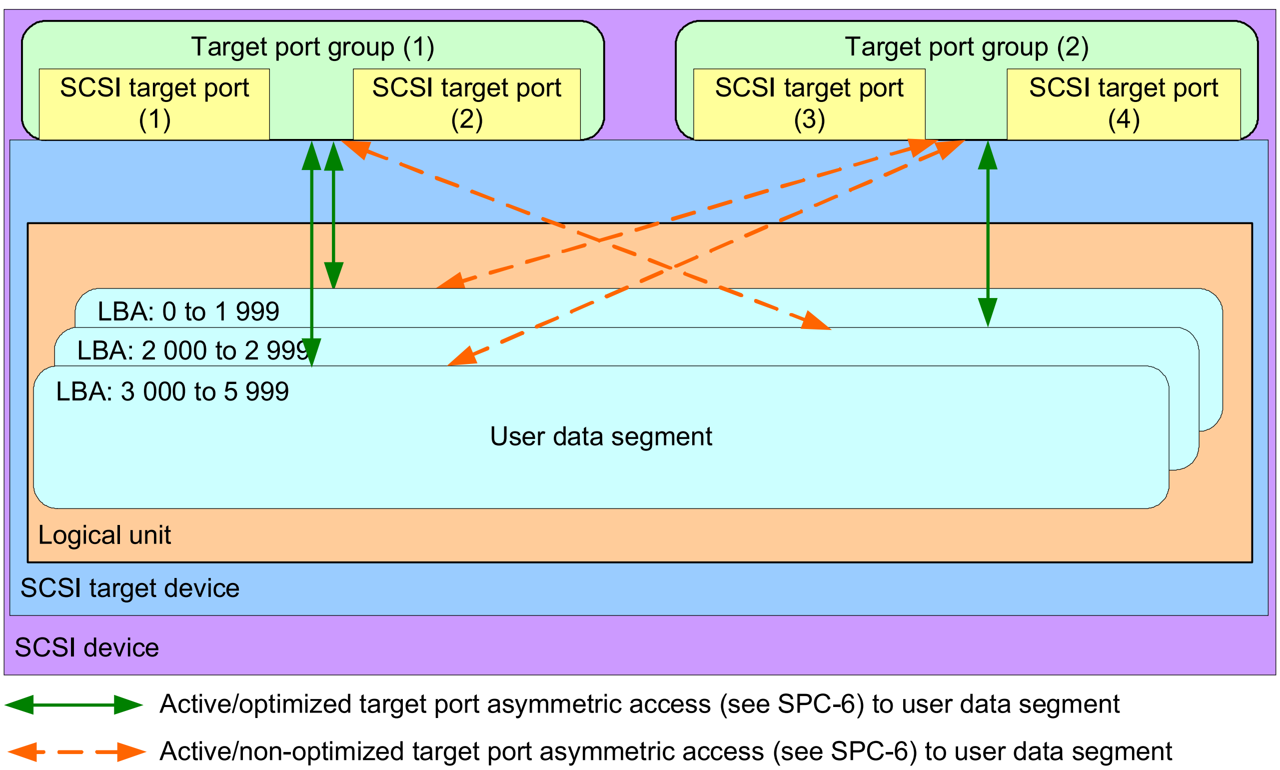


Figure G.1 — Referrals example with no user data segment multiplier

In the example shown in figure G.1, the application client acquires the information from the logical unit as shown in table G.1.

Table G.1 — Referrals application client information with no user data segment multiplier

<table>
<tr><th colspan="4">Referrals VPD page</th></tr>
<tr><th colspan="2">User data segment size</th><th colspan="2">User data segment multiplier</th></tr>
<tr><td colspan="2">ignored</td><td colspan="2">0</td></tr>
<tr><th colspan="4">REPORT TARGET PORT GROUPS command</th></tr>
<tr><th>Asymmetric access state</th><th>Target port group</th><th colspan="2">Relative target port identifier</th></tr>
<tr><td rowspan="4">4h (i.e., logical block dependent)</td><td rowspan="2">1</td><td colspan="2">1</td></tr>
<tr><td colspan="2">2</td></tr>
<tr><td rowspan="2">2</td><td colspan="2">3</td></tr>
<tr><td colspan="2">4</td></tr>
<tr><th colspan="4">REPORT REFERRALS command or user data segment referral sense data descriptors</th></tr>
<tr><th>First user data segment LBA</th><th>Last user data segment LBA</th><th>Asymmetric access state</th><th>Target port group</th></tr>
<tr><td rowspan="2">0</td><td rowspan="2">1 999</td><td>0 (i.e., active/optimized)</td><td>1</td></tr>
<tr><td>1 (i.e., active/non-optimized)</td><td>2</td></tr>
<tr><td rowspan="2">2 000</td><td rowspan="2">2 999</td><td>0 (i.e., active/optimized)</td><td>2</td></tr>
<tr><td>1 (i.e., active/non-optimized)</td><td>1</td></tr>
<tr><td rowspan="2">3 000</td><td rowspan="2">5 999</td><td>0 (i.e., active/optimized)</td><td>1</td></tr>
<tr><td>1 (i.e., active/non-optimized)</td><td>2</td></tr>
</table>

The application may determine the user data segments that are optimally accessed through the two target port groups as shown in table G.2.

Table G.2 — User data segment calculations with no user data segment multiplier

<table>
<tr><th>First LBA of user data segment</th><th>Calculation (see 4.26.2)</th><th>Last LBA of user data segment</th><th>Calculation (see 4.26.2)</th></tr>
<tr><th colspan="4">Target port group 1 user data segments in active/optimized asymmetric access state</th></tr>
<tr><td>0</td><td>0 [a]</td><td>1 999</td><td>1 999 [b]</td></tr>
<tr><td>3 000</td><td>3 000 [a]</td><td>5 999</td><td>5 999 [b]</td></tr>
<tr><th colspan="4">Target port group 2 user data segments in active/optimized asymmetric access state</th></tr>
<tr><td>2 000</td><td>2 000 [a]</td><td>2 999</td><td>2 999 [b]</td></tr>
<tr><td colspan="4">[a] The first user data segment LBA
[b] The last user data segment LBA</td></tr>
</table>

G.2 Referrals example with non-zero user data segment multiplier

This subclause demonstrates a method that an application client may use to determine the optimal target port group from which to access logical blocks using information sent from the device server when the user data segment multiplier is set to a non-zero value.

Figure G.2 shows an example of a SCSI device in which referrals have been implemented with a user data segment multiplier of two and a user data segment size of 1 000.

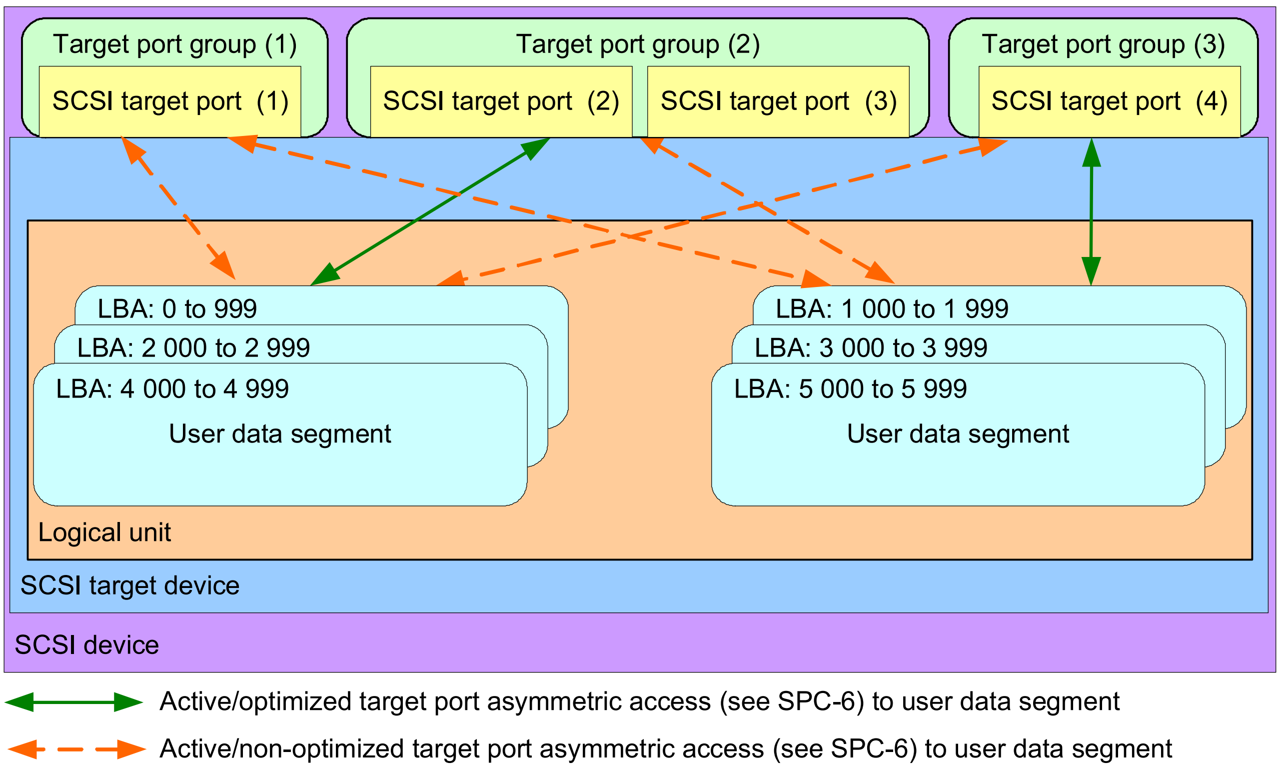


Figure G.2 — Referrals example with non-zero user data segment multiplier

In the example shown in figure G.2, the application client acquires the information from the logical unit as shown in table G.3.

Table G.3 — Referrals application client information with non-zero user data segment multiplier

<table>
<tr><th colspan="4">Referrals VPD page</th></tr>
<tr><th colspan="2">User data segment size</th><th colspan="2">User data segment multiplier</th></tr>
<tr><td colspan="2">1 000</td><td colspan="2">2</td></tr>
<tr><th colspan="4">REPORT TARGET PORT GROUPS command</th></tr>
<tr><th>Asymmetric access state</th><th>Target port group</th><th colspan="2">Relative target port identifier</th></tr>
<tr><td rowspan="4">4h (i.e., logical block dependent)</td><td>1</td><td colspan="2">1</td></tr>
<tr><td rowspan="2">2</td><td colspan="2">2</td></tr>
<tr><td colspan="2">3</td></tr>
<tr><td>3</td><td colspan="2">4</td></tr>
<tr><th colspan="4">REPORT REFERRALS command or user data segment referral sense data descriptors</th></tr>
<tr><th>First user data segment LBA</th><th>Last user data segment LBA</th><th>Asymmetric access state</th><th>Target port group</th></tr>
<tr><td rowspan="3">0</td><td rowspan="3">4 999</td><td>0 (i.e., active/optimized)</td><td>2</td></tr>
<tr><td>1 (i.e., active/non-optimized)</td><td>1</td></tr>
<tr><td>1 (i.e., active/non-optimized)</td><td>3</td></tr>
<tr><td rowspan="3">1 000</td><td rowspan="3">5 999</td><td>0 (i.e., active/optimized)</td><td>3</td></tr>
<tr><td>1 (i.e., active/non-optimized)</td><td>1</td></tr>
<tr><td>1 (i.e., active/non-optimized)</td><td>2</td></tr>
</table>

The application may determine the user data segments that are optimally accessed through the two target port groups as shown in table G.4.

Table G.4 — User data segment calculations with non-zero user data segment multiplier

<table>
<tr><th>First LBA of user data segment</th><th>Calculation (see 4.26.2)</th><th>Last LBA of user data segment</th><th>Calculation (see 4.26.2)</th></tr>
<tr><th colspan="4">Target port group 2 user data segments in active/optimized asymmetric access state</th></tr>
<tr><td>0</td><td>0 [a]</td><td>999</td><td>0 [a] + (1 000 – 1)</td></tr>
<tr><td>2 000</td><td>0 + (1 000 × 2)</td><td>2 999</td><td>2 000 + (1 000 – 1)</td></tr>
<tr><td>4 000</td><td>2 000 + (1 000 × 2)</td><td>4 999</td><td>4 000 + (1 000 – 1)</td></tr>
<tr><th colspan="4">Target port group 3 user data segments in active/optimized asymmetric access state</th></tr>
<tr><td>1 000</td><td>1 000 [a]</td><td>1 999</td><td>1 000 [a] + (1 000 – 1)</td></tr>
<tr><td>3 000</td><td>1 000 + (1 000 × 2)</td><td>3 999</td><td>3 000 + (1 000 – 1)</td></tr>
<tr><td>5 000</td><td>3 000 + (1 000 × 2)</td><td>5 999</td><td>5 000 + (1 000 – 1)</td></tr>
<tr><td colspan="4">[a] The first user data segment LBA.</td></tr>
</table>

Annex H
(informative)

IO advice hints usage

H.1 Overview

This annex describes how application clients and device servers may use IO advice hints and their associated logical block markup descriptors to provide information to a storage device to enhance access to data.

H.2 IO Advice Hints Grouping mode page

The IO Advice Hints Grouping mode page (see 6.5.7) contains the definitions of the IO advice hints (see 4.22.2) that are associated with each group number.

A device server may implement pre-defined IO advice hints that are not changeable and indicate this by making one or more IO advice hints group descriptors in the mode page not changeable.

The application client may use the MODE SENSE command to retrieve IO advice hints group descriptors to determine the IO advice hints that are available, and determine which IO advice hints are associated with each group number.

If any IO advice hints group descriptors in the mode page are changeable, the application client may use a MODE SELECT command to associate IO group numbers with IO advice hints that meet the application client needs.

A device server may implement any combination of changeable IO advice hints group descriptors or predefined IO advice hints group descriptors.

H.3 Issuing I/O commands with IO advice hints

H.3.1 Group numbers and I/O commands

An application client:

1) determines the IO advice hints group descriptor appropriate for each particular type of command using the process described in clause H.2;
2) determines the value to specify in the GROUP NUMBER field of each command as described in 4.22.2; and

places that value into the GROUP NUMBER field of the CDB.If cache segmentation (see 4.15.2) is enabled (see 6.5.7), then the use of different group numbers by components within the application client (e.g., file system, virtual memory swapping/paging, payroll application) allows the device server to associate an IO command and the associated cached data with other IO commands and other cached data from that same component.

H.3.2 Possible constraints on IO advice hints

There is no method to report how an IO advice hints has been processed by the device server. The implications of this model include the ability of the device server to ignore any, or even all, IO advice hints it receives.

A device server may constrain the complexity of its IO advice hints implementation by ignoring one or more of the values in the logical block markup descriptor (see 6.8) (e.g., to ignore the READ SEQUENTIALITY field, and the READ/WRITE FREQUENCY field).

If the complexity of specifying values or locating values in the IO Advice Hints Grouping mode page (see 6.5.7) is a concern, the device server may implement the page, but indicate that one or more IO advice hints groups descriptors (see 6.5.7) are not changeable (see SPC-6). A device server that does this should:

a) define the unchangeable IO advice hints for groups descriptors in the lowest group numbers supported for usage by IO advice hints; and
b) allow at least four IO advice hints group descriptors to be changeable.

H.4 Logical block markup descriptor usage examples

H.4.1 Example usage in tiered storage device implementations

Products (e.g., SCSI target devices and ATA devices) may use the contents of an logical block markup descriptor in ways that are not related to the contents of one field in a logical block markup descriptor in any way that is traceable to the way the field is defined. As an example, suppose a product contains the following storage tiers:

a) volatile cache;
b) non-volatile cache;
c) solid state device storage; and
d) magnetic media storage.

Table H.1 shows examples of how such a tiered product may interpret combinations of values in fields in an access patterns logical block markup descriptor (see 6.8.3.1).

Table H.1 — Tiered product access patterns logical block markup descriptor examples

Field	Code	Field	Code	Field	Code
Data that should not be written to the solid state device storage					
READ/WRITE FREQUENCY	10b	WRITE SEQUENTIALITY	see [a]	OSI PROXIMITY	see [a]
OVERALL FREQUENCY	10b	READ SEQUENTIALITY	see [a]	ACDLU	see [a]
SUBSEQUENT I/O	see [a]	IO CLASS	see [a]		
Data that should be written to non-volatile media and not retained in cache					
READ/WRITE FREQUENCY	10b	WRITE SEQUENTIALITY	see [a]	OSI PROXIMITY	see [a]
OVERALL FREQUENCY	01b	READ SEQUENTIALITY	see [a]	ACDLU	see [a]
SUBSEQUENT I/O	see [a]	IO CLASS	see [a]		
[a] The contents of this field do not contribute to this example case.					

H.4.2 Example logical block markup descriptor values for software that sends read commands and write commands

Table H.2 shows example mappings to access patterns logical block markup descriptors (see 6.8.3.1) from:

a) file types in common usage; and
b) published file system and operating system specifications for usage characteristics.

Table H.2 — Sending device access patterns logical block markup descriptor examples (part 1 of 2)

Field	Code	Field	Code	Field	Code
LBMd value: 90h, A5h, 52h, 00h **Data type:** swap/page file		Applicable published specifications Microsoft®: FILE_TYPE_NOTIFICATION_GUID_PAGE_FILE Linux: None known			
READ/WRITE FREQUENCY	10b	WRITE SEQUENTIALITY	01b	OSI PROXIMITY	10b
OVERALL FREQUENCY	10b	READ SEQUENTIALITY	01b	ACDLU	1b
SUBSEQUENT I/O	00b	IO CLASS	5h	RLBSR	01b
LBMd value: 03h, 00h, 56h, 00h **Data type:** hibernate file		Applicable published specifications Microsoft®: FILE_TYPE_NOTIFICATION_GUID_HIBERNATION_FILE Linux: None known			
READ/WRITE FREQUENCY	00b	WRITE SEQUENTIALITY	00b	OSI PROXIMITY	10b
OVERALL FREQUENCY	00b	READ SEQUENTIALITY	00b	ACDLU	0b
SUBSEQUENT I/O	01b	IO CLASS	5h	RLBSR	11b
LBMd value: 00h, 55h, 06h, 00h **Data type:** system initialization (e.g., registry)		Applicable published specifications Microsoft®: None known Linux: None known			
READ/WRITE FREQUENCY	01b	WRITE SEQUENTIALITY	01b	OSI PROXIMITY	10b
OVERALL FREQUENCY	01b	READ SEQUENTIALITY	01b	ACDLU	0b
SUBSEQUENT I/O	01b	IO CLASS	0h	RLBSR	00b
LBMd value: 01h, 05h, 14h, 00h **Data type:** general file system metadata [c] (e.g., directory files)		Applicable published specifications Microsoft®: None known Linux: None known			
READ/WRITE FREQUENCY	00b	WRITE SEQUENTIALITY	01b	OSI PROXIMITY	00b
OVERALL FREQUENCY	00b	READ SEQUENTIALITY	01b	ACDLU	0b
SUBSEQUENT I/O	01b	IO CLASS	1h	RLBSR	01b
LBMd value: 00h, 05h, x0h [a],00h **Data type:** random access		Applicable published specifications Microsoft®: None known Linux: FADV_RANDOM			

Key:
LBMd = logical block markup descriptor

Note 1 - Microsoft information obtained from http://msdn.microsoft.com/en-us/library/windows/desktop/hh404249%28v=vs.85%29.aspx.
Note 2 - Linux information obtained from http://linux.die.net/man/2/fadvise.

[a] Where x represents the io_class value.
[b] The application client should select the io_class value (e.g., 4h or 5h) appropriate to the size of the object.
[c] File system metadata accessed during normal operation of the file system (e.g. security information, time stamps, directories).
[d] File system metadata accessed during mounting of the file system (e.g., superblock, bitmaps, mapping tables).

Table H.2 — Sending device access patterns logical block markup descriptor examples (part 2 of 2)

Field	Code	Field	Code	Field	Code
READ/WRITE FREQUENCY	00b	WRITE SEQUENTIALITY	01b	OSI PROXIMITY	00b
OVERALL FREQUENCY	00b	READ SEQUENTIALITY	01b	ACDLU	0b
SUBSEQUENT I/O	00b	IO CLASS	see [b]	RLBSR	00b
LBMd value: 00h, 0Ah, x0h [a], 00h **Data type:** sequential access		Applicable published specifications Microsoft®: None known Linux: FADV_SEQUENTIAL			
READ/WRITE FREQUENCY	00b	WRITE SEQUENTIALITY	10b	OSI PROXIMITY	00b
OVERALL FREQUENCY	00b	READ SEQUENTIALITY	10b	ACDLU	0b
SUBSEQUENT I/O	00b	IO CLASS	see [b]	RLBSR	00b
LBMd value: 30h,AAh,1Ah,00h **Data type:** circular log that contains metadata		Applicable published specifications Microsoft®: None known Linux: None known			
READ/WRITE FREQUENCY	10b	WRITE SEQUENTIALITY	10b	OSI PROXIMITY	10b
OVERALL FREQUENCY	10b	READ SEQUENTIALITY	10b	ACDLU	0b
SUBSEQUENT I/O	10b	IO CLASS	1h	RLBSR	11b
LBMd value: 30h,A5h,1Ah,00h **Data type:** critical filesystem metadata [d]		Applicable published specifications Microsoft®: None known Linux: None known			
READ/WRITE FREQUENCY	10b	WRITE SEQUENTIALITY	01b	OSI PROXIMITY	10b
OVERALL FREQUENCY	10b	READ SEQUENTIALITY	01b	ACDLU	0b
SUBSEQUENT I/O	10b	IO CLASS	1h	RLBSR	11b

Key:
LBMd = logical block markup descriptor

Note 1 - Microsoft information obtained from
http://msdn.microsoft.com/en-us/library/windows/desktop/hh404249%28v=vs.85%29.aspx.
Note 2 - Linux information obtained from http://linux.die.net/man/2/fadvise.

[a] Where x represents the io_class value.
[b] The application client should select the io_class value (e.g., 4h or 5h) appropriate to the size of the object.
[c] File system metadata accessed during normal operation of the file system (e.g. security information, time stamps, directories).
[d] File system metadata accessed during mounting of the file system (e.g., superblock, bitmaps, mapping tables).

Annex I
(informative)

Using storage element depopulation

Storage element depopulation may be used when a logical unit is suboptimal.

A device server indicates a suboptimal condition of a storage element to an application client by establishing an informational exception condition with the additional sense code set to WARNING - PHYSICAL ELEMENT STATUS CHANGE (see 4.36.3).

The application client may determine which storage elements have a physical element health (see 5.8.2.2) that is outside the manufacturer's specification limit using the GET PHYSICAL ELEMENT STATUS command (see 5.8).

If the application client determines that a storage element that is outside manufacturer's specification limit and should be depopulated, then the application client may send a REMOVE ELEMENT AND TRUNCATE command that specifies:

a) the element identifier of the storage element to be depopulated; and
b) the requested capacity, if any.

The REMOVE ELEMENT AND TRUNCATE command requests storage element depopulation (see 4.36.4). If the application client requires logical block data to be initialized after a storage element depopulation has completed, then the application client should initialize all logical block data.

To determine when the operations described in 4.36.4 have completed, the application client may use a REQUEST SENSE command to retrieve pollable sense data (see SPC-6). To determine which storage element has been depopulated, the application client may use a GET PHYSICAL ELEMENT STATUS command.

A sequence of REMOVE ELEMENT AND TRUNCATE commands may be used to depopulate multiple storage elements. A device server may have a limit on the number of storage elements that may be depopulated. If the device server is requested to depopulate a storage element in excess of this limit, then the device server may terminate that request (see 5.26).

Storage elements may transition outside manufacturer's specification limit during the time other storage elements are being depopulated. If a storage element transitions outside manufacturer's specification limit during processing of a storage element depopulation for a different storage element, the device server notifies the application client of that storage element status change as described in 4.36.3.

The RESTORE ELEMENTS AND REBUILD command (see 5.29) restores storage elements that had previously been depopulated. The device server indicates which storage elements, if any, are able to be affected by a RESTORE ELEMENTS AND REBUILD command using the RALWD bit (see 5.8.2.2) in the GET PHYSICAL ELEMENT STATUS parameter data.

The RESTORE ELEMENTS AND REBUILD command requests storage element restoration (see 4.36.5). A successful storage element restoration restores at least one storage element and may result in an increase in the number of LBA resources. The storage element restoration is not required to preserve logical block data. After a storage element restoration has completed, the host may initialize all logical block data.

To determine when the operations described in 4.36.5 have completed, the application client may use a REQUEST SENSE command to retrieve pollable sense data (see SPC-6). To determine which storage elements have been restored, the application client may use a GET PHYSICAL ELEMENT STATUS command.

Annex J
(informative)

Rebuild assist using the GET LBA STATUS command

J.1 Overview

This annex describes an example of an application client rebuilding data from a SCSI device with a physical element failure. This examples uses the following commands:

a) the GET PHYSICAL ELEMENT STATUS command (see 5.8);
b) the GET LBA STATUS (32) command (see 5.7); and
c) write commands.

J.2 Discovery process

This example show how an application client is able to discover what LBAs are affected on a block storage device with a capacity of 2_147_483_648 bytes, a logical block length of 4_096 bytes, and the physical element identified by identifier 7 with a health status of 65. The discovery process for the LBAs that are affected by the failure of a this physical element consists of following sequence. The application client issues a GET PHYSICAL ELEMENT STATUS command with:

a) the STARTING ELEMENT field set to 0000h;
b) the FILTER field set to 01b (i.e., physical elements with a health outside manufacturers specification limit or depopulated); and
c) the REPORT TYPE field set to 1h (i.e., storage elements).

In this example the GET PHYSICAL ELEMENT STATUS command returns one descriptor in the parameter data. That descriptor contains:

a) an element identifier of 0007h;
b) a physical element type of 01h; and
c) a physical element health of 65h.

The application client then issues a GET LBA STATUS (32) command with:

a) the REPORT TYPE field set to 00h (i.e., return descriptors for all LBAs);
b) the STARTING LOGICAL BLOCK ADDRESS field set to 0000_0000h;
c) the SCAN LENGTH field set to FFFFh; and
d) the ELEMENT IDENTIFIER field set to 0007h.

Any LBAs from 0000_0000h to 0000_FFFFh that are mapped to physical element 0007h are included in the returned parameter data.

The application client then issues a GET LBA STATUS (32) command with:

a) the REPORT TYPE field set to 00h;
b) the STARTING LOGICAL BLOCK ADDRESS field set to 0001_0000h;
c) the SCAN LENGTH field set to FFFFh; and
d) the ELEMENT IDENTIFIER field set to 0007h.

Any LBAs from 0001_0000h to 0001_FFFFh that are mapped to physical element 0007h are included in the returned parameter data.

The application client continues issuing GET LBA STATUS (32) commands incrementing the STARTING LOGICAL BLOCK ADDRESS field by 1_0000h for each command until it issues a GET LBA STATUS (32) command with a STARTING LOGICAL BLOCK ADDRESS field set to 0079_0000h.

After issuing all necessary GET LBA STATUS (32) commands, the application client has a complete list of the LBAs affected by the physical element that is bad and can retrieve the user data from another source and write it to the SCSI device causing it to be written to a different physical element.

Annex K
(informative)

Direct access block devices with shared resources

K.1 Overview

Traditionally, most disk drives have shipped with only one logical unit supported. SAM-6 allows for multiple logical units to be contained in one SCSI target device, and many products (e.g., storage arrays) have incorporated this feature. Within the boundaries defined in SAM-6, these logical units are completely independent of each other. In practice, there may be resources shared between the logical units and such sharing may be observable to SCSI initiator devices.

This annex reviews effects that SCSI initiator devices may be able to observe if a direct access block device shares resources between logical units and describes methods by which a SCSI initiator device may be able to discover that certain resources are shared.

Designs for direct access block devices balance component costs against effects that are observable to SCSI initiator devices. Small numbers of components lower cost but increase the numbers and severity of the effects that are observable to SCSI initiator devices. This annex discusses these tradeoffs.

K.2 Downloading and activating microcode

SPC-6 allows for either separate microcode per logical unit or one microcode that is used by multiple logical units contained in a SCSI target device. An application client may detect if microcode is shared by multiple logical units by observing that the MICROCODE HAS BEEN CHANGED unit attention condition, if any, or the MICROCODE HAS BEEN CHANGED WITHOUT RESET unit attention condition, if any, occurs on logical units other than the logical unit that performed the microcode download operation and activation.

If one microcode is used by multiple logical units, then the SCSI target device that contains those logical units may support the TARGET COMMANDS well known logical unit (see SPC-6) to provide one microcode download interface for the one microcode being downloaded. The effects of using the TARGET COMMANDS well known logical unit in this way are described in K.8.6.

K.3 Caching

The direct access block device caching model (see 4.15) allows, among other choices, a separate cache per logical unit or a cache that is shared by the attached logical units. The advantages and disadvantages of these approaches are vendor specific.

The sharing characteristics of the cache in a direct access block device are the same as the sharing characteristics of the Caching mode page (see 6.5.6). The MLUS bit for the Caching mode page (i.e., mode page 08h) in the Mode Page Policy VPD page (see SPC-6) indicates whether the affected logical units share the cache or each logical unit has its own cache.

K.4 Power management

The SPC-6 requirements for the idle power condition and the standby power condition allow a logical unit to ignore any request to change a specific power condition until a vendor specific group of logical units have changed their power condition in similar ways. As a result, there is variability in the exact point at which the power condition of a logical unit in a SCSI target device changes in a way that is observable to the device server.

The START STOP UNIT command requirements for the START bit (see 5.31) define a case in which the logical unit is required to transition to the stopped power condition (e.g., the rotating medium spindle is stopped). This requirement may result in a START STOP UNIT command that is processed by one logical unit affecting more than one logical unit (e.g., in a SCSI target device that has only one rotating medium spindle).

Some SCSI target devices process the stopped power condition caused by a START STOP UNIT command in the same way that the idle power condition and the standby power condition are processed in SPC-6 (i.e., requests to change a specific power condition are ignored until a vendor specific group of logical units have changed their power condition in similar ways).

Application clients that are trying to stop the rotating medium spindle in a SCSI target device (e.g., to facilitate removal of that SCSI target device from the configuration), should send appropriate START STOP UNIT commands to each logical unit in that SCSI target device that indicates participation in a vendor specific group of logical units. Information about participation in a vendor specific group of logical units may be available from the MLU field in the descriptor associated with the START STOP UNIT command in the REPORT SUPPORTED OPERATION CODES command parameter data (see K.7).

K.5 Mode page considerations

The MLUS bit in the Mode Page Policy VPD page (see SPC-6) indicates whether or not multiple logical units in a SCSI target device share the mode page and subpage identified by each descriptor in the Mode Page Policy VPD page. If any field in a mode page is shared by multiple logical units, then the Mode Page Policy VPD page reports that mode page as being shared.

The Disconnect-Reconnect mode page and all protocol-specific modes pages are shared by all logical units in a SCSI target device (see SPC-6).

A change that is made in a mode parameter that is shared by multiple logical units results in a unit attention condition being established with the additional sense code set to MODE PARAMETERS CHANGED as described in SPC-6.

EXAMPLE - The Power Conditions mode page is an example of a mode page where some fields are shared and some fields are not. If a rotating medium spindle is shared by multiple logical units and multiple head assemblies are not shared by those logical units, then:

a) the Power Conditions mode page fields (see SPC-6) that affect whether the medium spindle is rotating are shared by those logical units; and
b) the Power Conditions mode page fields that affect whether a head assembly is retracted from the medium are not shared by those logical units.

K.6 Log page considerations

The choice of whether a log page is maintained separately for each logical unit or aggregates information for multiple logical units is vendor specific.

If a SCSI target device returns a log page that aggregates information for multiple logical units, then a LOG SELECT command that resets that log page resets the log data for all of those logical units in that SCSI target device.

The Power Condition Transitions log page (see SPC-6) has a high probability of being shared. Sharing for the Cache Memory Statistics log page (see SPC-6) has a high probability of having the same sharing as the Caching mode page (see K.3).

Separate log pages for each logical unit are recommended in cases where:

a) the information is specific to the interactions between that logical unit and the application client (e.g., the Last n Deferred Error or Asynchronous Events log page (see SPC-6), and the Application Client log page (see SPC-6)); and
b) LBA values that are specific to one logical unit are returned (e.g., Background Scan Results log page (see 6.4.2) and the Pending Defects log page (see 6.4.8)).

EXAMPLE - Some log pages may have a mix of log parameters where some parameter values are an aggregate of multiple logical units and other parameters are specific to that logical unit. If a rotating medium spindle is shared by multiple logical units and multiple head assemblies are not shared by those logical units, then:
a) the Power Conditions Transitions log page (see SPC-6) log parameters that are affected by whether the medium spindle is rotating may report the same value for each logical unit; and
b) the Power Conditions Transitions log page log parameters that are affected by whether a head assembly is retracted from the medium may report a different value for each logical unit.

K.7 Command considerations

The MLU field in the REPORT SUPPORTED OPERATION CODES command parameter data (see SPC-6) indicates whether the command indicated by the command descriptor affects multiple logical units in a SCSI target device.

Some commands (see K.8) are defined to produce adverse effects (e.g., data loss, degraded performance) on the logical unit in which they are processed. If these commands affect logical units other than the one in which they are processed, then application clients that are accessing those other logical units may observe unexpected adverse effects.

If a SCSI target device supports any of the commands described in K.8 on LUN 0 only, then an application client may detect this case by comparing the REPORT SUPPORTED OPERATION CODES parameter data (see SPC-6) from different logical units.

K.8 Commands with a high probability of affecting more than one logical unit

K.8.1 The FORMAT UNIT command

The MLU field (see K.7) may indicate whether the processing of a FORMAT UNIT command (see 5.4) affects more than one logical unit.

If the FORMAT UNIT command affects multiple logical units, then the processing of a FORMAT UNIT command in any of those logical units causes all of those logical units:

a) to process commands as described in 4.33; and
b) to be initialized as required by the FORMAT UNIT command parameters (see 5.4).

K.8.2 The REMOVE ELEMENT AND TRUNCATE command

The MLU field (see K.7) may indicate whether the processing of a REMOVE ELEMENT AND TRUNCATE command (see 5.26) affects more than one logical unit.

If one or more of the elements processed by a REMOVE ELEMENT AND TRUNCATE command is shared between logical units, then processing the command in one logical unit affects all the logical units that share that element.

If a REMOVE ELEMENT AND TRUNCATE command service action affects multiple logical units, then the processing of that REMOVE ELEMENT AND TRUNCATE command service action in any of those logical units causes all of those logical units:

a) to process commands as described in 4.36; and

b) to be affected as required by the REMOVE ELEMENT AND TRUNCATE command parameters (see 5.26).

K.8.3 The SANITIZE command

The MLU field (see K.7) may indicate whether the processing of a SANITIZE command (see 5.30) affects more than one logical unit. Each sanitize service action is reported separately by the REPORT SUPPORTED OPERATION CODES command (see SPC-6). Some service actions may affect multiple logical units and some may affect only one logical unit.

If a SANITIZE command service action affects multiple logical units, then the processing of that SANITIZE command service action in any of those logical units causes all of those logical units:

a) to process commands as described in 4.11; and
b) to be sanitized as required by the SANITIZE command parameters (see 5.30).

K.8.4 The START STOP UNIT command

The MLU field (see K.7) may indicate whether the processing of a START STOP UNIT command (see 5.31) affects more than one logical unit.

The START STOP UNIT command is able to cause a shared rotating medium spindle to stop rotating, a condition that affects all the logical units that share that rotating medium spindle (see K.4).

K.8.5 The SEND DIAGNOSTIC command and RECEIVE DIAGNOSTIC RESULTS command

The MLU field (see K.7) for the SEND DIAGNOSTIC command may indicate whether the default self test and the SELF-TEST CODE field diagnostics affects more than one logical unit. If the default self test and the SELF-TEST CODE field diagnostics affect multiple logical units, then all affected logical units return a SELF-TEST IN PROGRESS additional sense code while the test is in progress.

The RECEIVE DIAGNOSTIC RESULTS command only affects a single logical unit, but may return results that apply to multiple logical units.

The Translate Address diagnostic page (see 6.3.4 and 6.3.5) only applies to one logical unit, regardless of the value in the MLU field for the SEND DIAGNOSTIC command.

The choice of whether other diagnostic pages affect one logical unit or more than one logical unit is vendor specific.

K.8.6 The WRITE BUFFER command and READ BUFFER command

The MLU field (see K.7) may indicate whether the processing of a WRITE BUFFER command (see SPC-6) affects more than one logical unit. The MLU field (see K.7) indicates whether the processing of a READ BUFFER command (see SPC-6) may affect more than one logical unit.

The considerations associated with microcode download and activation (see K.2) may affect the value returned in the MLU field for the WRITE BUFFER command.

The WRITE BUFFER command MODE field results in there being one command that processes different operations that:

a) are not associated with microcode download and activation, including the MODE field set to:
   A) 0Ah (i.e., write data to echo buffer); and
   B) 1Ch (i.e., download application client error history);

and

b) are associated with microcode download and activation, including the MODE field set to:
   A) 04h (i.e., Download microcode and activate);
   B) 05h (i.e., Download microcode, save, and activate);
   C) 06h (i.e., Download microcode with offsets and activate);
   D) 07h (i.e., Download microcode with offsets, save, and activate);

  - E) 0Dh (i.e., Download microcode with offsets, select activation events, save, and defer activate);
  - F) 0Eh (i.e., Download microcode with offsets, save, and defer activate); and
  - G) 0Fh (i.e., Activate deferred microcode).

If the SCSI target device supports the TARGET COMMANDS well known logical unit (see SPC-6), then some MODE field values (e.g., 1Ch) are able to be processed by a device server contained in a logical unit other than the TARGET COMMANDS well known logical unit. If a device server contained in a logical unit other than the TARGET COMMANDS well known logical unit receives a WRITE BUFFER command with the MODE field set to one of these values, then that device server processes the command as if the TARGET COMMANDS well known logical unit were not supported.

In some cases, a device server contained in a logical unit other than the TARGET COMMANDS well known logical unit may be able to process a WRITE BUFFER command with a MODE field set to a value associated with microcode download and activation. For example, a device server contained in a logical unit other than the TARGET COMMANDS well known logical unit may be able to provide useful processing for a WRITE BUFFER command with:

a) the MODE field set to 0Dh; and
b) the MODE SPECIFIC field set to 100b (see SPC-6) to specify:
  - A) the PO_ACT bit set to one (i.e., activation if a power on occurs);
  - B) the HR_ACT bit set to zero (i.e., no activation if a hard reset occurs); and
  - C) the VS_ACT bit set to zero (i.e., no activation if a vendor specific event occurs).

For any logical unit in a SCSI target device that has deferred microcode (see SPC-6) present but not activated, the device server for that logical unit returns pollable sense data as described in SPC-6 to indicate whether or not that device server downloaded the deferred microcode.

The success of this example is based on one specific fact. Although the processing state of the entire SCSI target device is affected by the power on event, the processing state of the entire SCSI target device is not affected by the processing of the WRITE BUFFER command with the very constrained set of inputs, with the exception that the presence of downloaded microcode that is not activated affects how all device servers return pollable sense data.

For a SCSI target device in which microcode download and activation affects the entire SCSI target device, the logical units in a SCSI target device may restrict to the TARGET COMMANDS well known logical unit the processing of WRITE BUFFER commands with a MODE field set to one of the values associated with microcode download and activation that affects the entire SCSI target device as follows:

a) logical units other than the TARGET COMMANDS well known logical unit:
  - A) terminate WRITE BUFFER commands associated with microcode download and activation that affect the entire SCSI target device; and
  - B) processing other WRITE BUFFER commands as described in SPC-6;

  and
b) the TARGET COMMANDS well known logical unit processes WRITE BUFFER commands associated with microcode download and activation that affect the entire SCSI target device.

An application client may determine which WRITE BUFFER command MODE field values are supported by which device severs using information returned by each device server in response to:

a) REPORT SUPPORTED OPERATION CODES commands (see SPC-6); or
b) INQUIRY commands that specify the Extended INQUIRY Data VPD page (see SPC-6), if the returned VPD page has the DMS_VALID bit set to one.

K.9 Common Mandatory SCSI Commands

The REQUEST SENSE command (see SPC-6) and the TEST UNIT READY command return information that depends on the current operating conditions in one logical unit. Although some current operating conditions affect multiple logical units in a SCSI target device, sense data is interpreted as applying only to the addressed logical unit.

Although some of the information returned by the INQUIRY command (see SPC-6) is common to multiple logical units in a SCSI target device, separate processing should be provided for an INQUIRY command received by a specific logical unit.

For each of the multiple logical units in a SCSI target device, the REPORT LUNS command (see SPC-6) reports all the accessible logical units in the SCSI target device, limited only by the contents of the SELECT REPORT field in the CDB and the size of the Data-In Buffer.

Annex L
(informative)

Bibliography

INCITS 481-2011, Fibre Channel Protocol for SCSI - 4 (FCP-4)

INCITS 534-2019, Serial Attached SCSI-4 (SAS-3)

ISO/IEC 14776-154, Information technology – Small Computer System Interface (SCSI) – Part 154: Serial Attached SCSI-3 (SAS-3)

ISO/IEC 14776-321, Information technology – Small Computer System Interface (SCSI-3) – Part 321: SCSI Block Commands (SBC)

ISO/IEC 14776-322, Information technology – Small Computer System Interface (SCSI) – Part 322: SCSI Block Commands-2 (SBC-2)

CFast (CFast)™, CompactFlash Association (see http://www.compactflash.org)

CompactFlash® (CF), CompactFlash Association (see http://www.compactflash.org)

Memory Stick™ (MS). One Stop Site for Formats (see https://www.oss-formats.org)

MultiMediaCard (*e*•MMC), JEDEC® (see http://www.jedec.org)

NOTE 25 JEDEC® is a registered trademark of JEDEC Solid State Technology Association. This information is given for the convenience of users of this standard and does not constitute an endorsement by ISO, IEC, or ANSI.

Secure Digital Card (SD Card), SD Association (see http://www.sdcard.org)

XQD™ (XQD), CompactFlash Association (see http://www.compactflash.org)

Universal Flash Storage (UFS), JEDEC® (see http://www.jedec.org)

Differentiated Storage Services, ACM Special Interest Group on Operating Systems (see http://sigops.org/sosp/sosp11/current/2011-Cascais/printable/05-mesnier.pdf)